**华中科技大学计算机学院智能大数据与多媒体实验室**

**2018年3月**

**中建智能高空抛物检测系统**

**详细设计说明书**

目 录

[1 引言 7](#_Toc516651030)

[1.1 目的 7](#_Toc516651031)

[1.2 背景与意义 7](#_Toc516651032)

[2 系统需求分析 8](#_Toc516651033)

[3 系统的总体设计 9](#_Toc516651034)

[3.1 系统概述与模块划分 9](#_Toc516651035)

[3.2 系统环境需求 10](#_Toc516651036)

[3.2 系统总体架构 11](#_Toc516651037)

[4 系统设置功能模块设计说明 12](#_Toc516651038)

[4.1 程序描述 12](#_Toc516651039)

[4.2 功能 12](#_Toc516651040)

[4.3 性能 12](#_Toc516651041)

[4.4 输人项 12](#_Toc516651042)

[4.5 输出项 13](#_Toc516651043)

[4.6 流程逻辑 13](#_Toc516651044)

[5 设备管理功能模块设计说明 14](#_Toc516651045)

[5.1 程序描述 14](#_Toc516651046)

[5.2 设备启动方案设计 15](#_Toc516651047)

[5.2.1 客户端系统 15](#_Toc516651048)

[5.2.3 服务端系统（摄像头控制结点，抛物检测系统) 15](#_Toc516651049)

[5.3 设备登录 16](#_Toc516651050)

[5.3.1 设备登录功能 16](#_Toc516651051)

[5.3.2 设备登录输入项 16](#_Toc516651052)

[5.3.3 设备登录输出项 16](#_Toc516651053)

[5.3.4 设备登录流程逻辑 17](#_Toc516651054)

[5.4 设备退出 18](#_Toc516651055)

[5.4.1 设备退出功能 18](#_Toc516651056)

[5.4.2 设备退出输入项 18](#_Toc516651057)

[5.4.3 设备退出输出项 18](#_Toc516651058)

[5.4.4 设备退出流程逻辑 18](#_Toc516651059)

[5.5 设备添加 19](#_Toc516651060)

[5.5.1 设备添加功能 19](#_Toc516651061)

[5.5.2 设备添加输入项 19](#_Toc516651062)

[5.5.3 设备添加输出项 19](#_Toc516651063)

[5.5.4 设备添加流程逻辑 20](#_Toc516651064)

[6 抛物结果显示功能模块设计 20](#_Toc516651065)

[6.1 程序描述与程序功能 20](#_Toc516651066)

[6.2 性能 20](#_Toc516651067)

[6.3 输出项 21](#_Toc516651068)

[6.4 抛物信息计算方法 21](#_Toc516651069)

[6.5 流程逻辑 22](#_Toc516651070)

[6.6 接口 22](#_Toc516651071)

[6.7 限制条件 22](#_Toc516651072)

[7 抛物视频保存与回放功能模块设计说明 23](#_Toc516651073)

[7.1 抛物视频保存 23](#_Toc516651074)

[7.1.1 抛物视频保存功能与程序描述 23](#_Toc516651075)

[7.1.2 抛物视频保存性能说明 23](#_Toc516651076)

[7.1.3 抛物视频保存输入项 23](#_Toc516651077)

[7.1.4 抛物视频保存输出项 24](#_Toc516651078)

[7.1.5 流程逻辑 24](#_Toc516651079)

[7.2 抛物视频回放 24](#_Toc516651080)

[7.2.1 程序描述 24](#_Toc516651081)

[7.2.2 功能 25](#_Toc516651082)

[7.2.3 性能 25](#_Toc516651083)

[7.2.4 输入项 25](#_Toc516651084)

[7.2.5 输出项 25](#_Toc516651085)

[7.2.6 流程逻辑 25](#_Toc516651086)

[7.2.7 存储分配 26](#_Toc516651087)

[7.2.8 限制条件 26](#_Toc516651088)

[8 视频拼接显示功能模块设计说明 26](#_Toc516651089)

[8.1 程序描述 26](#_Toc516651090)

[8.2 功能 27](#_Toc516651091)

[8.3 性能 27](#_Toc516651092)

[8.4 输入项 27](#_Toc516651093)

[8.5 输出项 27](#_Toc516651094)

[8.6 算法 27](#_Toc516651095)

[8.7 流程逻辑 32](#_Toc516651096)

[8.8 接口 32](#_Toc516651097)

[8.9 限制条件 33](#_Toc516651098)

[9 图像采集功能模块设计说明 33](#_Toc516651099)

[9.1 程序描述与功能 33](#_Toc516651100)

[9.2 性能 33](#_Toc516651101)

[9.3 输入项 33](#_Toc516651102)

[9.4 输出项 34](#_Toc516651103)

[9.5 算法 34](#_Toc516651104)

[9.6 流程逻辑 34](#_Toc516651105)

[9.8 接口 35](#_Toc516651106)

[9.9 存储分配 35](#_Toc516651107)

[10 运动物体检测功能模块设计说明 35](#_Toc516651108)

[10.1 程序描述 35](#_Toc516651109)

[10.2 功能 36](#_Toc516651110)

[10.3 性能 36](#_Toc516651111)

[10.4 输入项 36](#_Toc516651112)

[10.5 输出项 36](#_Toc516651113)

[10.6 算法 36](#_Toc516651114)

[10.7 流程逻辑 39](#_Toc516651115)

[10.8 接口 39](#_Toc516651116)

[10.9 存储分配 40](#_Toc516651117)

[10.10 限制条件 40](#_Toc516651118)

[11 运动目标追踪功能模块设计说明 40](#_Toc516651119)

[11.1 程序描述 40](#_Toc516651120)

[11.2 功能 40](#_Toc516651121)

[11.3 性能 40](#_Toc516651122)

[11.4 输入项 41](#_Toc516651123)

[11.5 输出项 41](#_Toc516651124)

[11.6 算法 41](#_Toc516651125)

[11.7 流程逻辑 44](#_Toc516651126)

[11.8 接口 44](#_Toc516651127)

[12 高空抛物判定功能模块设计说明 45](#_Toc516651128)

[12.1 程序描述 45](#_Toc516651129)

[12.2 功能 45](#_Toc516651130)

[12.3 性能 45](#_Toc516651131)

[12.4 输入项 45](#_Toc516651132)

[12.5 输出项 46](#_Toc516651133)

[12.6 算法 46](#_Toc516651134)

[12.7 流程逻辑 47](#_Toc516651135)

[12.8 接口 48](#_Toc516651136)

[12.9 存储分配 48](#_Toc516651137)

[12.10 限制条件 48](#_Toc516651138)

[13 用户界面设计 48](#_Toc516651139)

[13.1 界面设计规范 48](#_Toc516651140)

[13.1.1 以用户为中心 48](#_Toc516651141)

[13.1.2 清楚一致的设计 49](#_Toc516651142)

[13.1.3 有良好的直觉特征 50](#_Toc516651143)

[13.1.4 较快的响应速度 50](#_Toc516651144)

[13.1.5 简单且美观 50](#_Toc516651145)

[13.2 界面关系和流程 51](#_Toc516651146)

[13.3 系统界面设计 52](#_Toc516651147)

[13.3.1 主界面设计 52](#_Toc516651148)

[13.3.2 系统设置界面 53](#_Toc516651149)

[13.3.3 设备管理界面 53](#_Toc516651150)

[13.3.4 抛物视频回放界面 54](#_Toc516651151)

[13.3.5 拼接视频显示及检测结果显示 55](#_Toc516651152)

[13.3.6非拼接视频显示及检测结果显示 56](#_Toc516651153)

# 1 引言

## 1.1 目的

编写本设计说明书的目的主要是详细说明高空抛物检测系统的整体设计与架构，给程序开发者一个详细的说明和详细的设计，提供系统的模块划分以及各种技术的解决方案。同时给系统开发者提供技术支持，为系统使用者提供功能概述。

## 1.2 背景与意义

高空抛物被称为“悬在城市上空的痛”，高空抛物事件的新闻时常会看到，能够产生严重的后果，严重时能够让居民当场死亡，并出现事件追责纠纷问题。通过科学计算与测试，从12米左右高的楼层抛出30克的鸡蛋砸到身上能够使起肿包，从54米左右抛下能够将人的头骨砸破，从75米左右抛下即可使人当场死亡，造成无法挽回的伤害。高空抛物行为是非常的不文明的，破坏了生活环境，导致环境脏乱，危害了人民的生命安全，侵犯了人民的权利。据统计，在近百个小区中进行问卷调查，高空抛物现象是普遍存在的。因此，对高空抛物的检测就非常的重要，否则就会出现无法确认抛物者，最终追责落到了整栋楼上，每户居民都要负一定的法律责任，而该受到处罚的抛物者只受到较轻的处罚。针对此种情况，想要防止高空抛物行为并妥善解决高空抛物纠纷事件，一方面需要加强公民道德建设，提高公民的道德素质，减少高空抛物行为，从根本上进行解决；另一方面需要完善法律制度，对高空抛物行为追责形成法律依据，对责任人加大处罚力度，通过增加视频监控点加强对高空抛物的监控，实现高空抛物监控无死角。

系统名称：高空抛物检测系统

本系统设计的目的是利用计算机视觉相关技术并结合视频监控系统，实现高空抛物检测，确定抛物的抛出点，对高空抛物进行报警，同时能够查看抛物的时间段的视频，获得抛物的视频的回放。本智能监控系统中检测目标是高空抛物，这一特殊的检测目标决定了系统的检测是针对小目标运动物体；同时整个系统中，抛物的检测需要以下几个步骤实现：运动目标检测、目标跟踪、行为分析、抛物判断几个环节，因此要实现并改进抛物检测流程及算法，使其运算更快，目标检测更加精确。

该项目由中建三局智慧社区提出，由华中科技大学计算机学院智能大数据与多媒体实验室负责开发，中建三局建设的小区使用。

# 2 系统需求分析

高空抛物检测主要是针对小区进行监控的智能视频监控系统，利用计算机视觉的方法实现对高空抛物的智能检测，同时推测抛出位置。整个流程智能实现，不需要人工进行处理。针对实际情况，本课题系统实现主要有以下需求：

(1) 针对100米高、50米宽的大楼进行监控，实现对大楼的全方位、无死角的监控，并能够实现对高空抛物的实时检测；

(2) 能够检测10厘米\*10厘米的高空抛物。通常高空抛物体积较小，例如抛出的物体是易拉罐、矿泉水瓶，酒瓶等等。抛出物体的大小大概在10厘米\*10厘米，因此系统要实现对10厘米\*10厘米的高空抛物的检测；

(3) 制定具体的监控方案，如何对大楼进行监控，如何安装相机，包括相机的部署，安装角度，能够监控的范围等；

(4) 系统能够实时的获得相机监控场景，并能够实时的对监控场景进行处理、检测，实现对高空抛物的检测；

(5) 能够实现对监控场景的24小时监控，发生高空抛物问题的时间晚上的频率较高，因此晚上也能够实现对高空抛物的实时监控异常的重要；

(6) 系统对于监控场景的缓慢变化具有较好的适应能力，当监控场景因为光照缓慢变化等情况也在发生着缓慢的变化时，系统依然能够进行较好的检测；

(7) 系统能够适应监控场景中发生光线突变等引起的场景突变问题，对高空抛物的检测不会产生较大的影响；

(8) 系统能够实现对高空抛物的实时检测，发生高空抛物能够实现实时报警，并将高空抛物关键视频时间段进行保存，并能够对高空抛物视频段进行查看回放。

针对高空抛物检测需求，本文对其进行研究与分析，对系统进行了设计与实现，系统主要分为了硬件系统和软件系统。

# 3 系统的总体设计

## 3.1 系统概述与模块划分

该软件系统是获得摄像机监控录像并对图像进行分析，判断是否有高空抛物，本文将软件系统分为了两个系统，分别是检测系统和显示系统。每个摄像头对应一个检测系统，检测系统对监控数据采集，并进行运动目标检测和运动目标追踪，最后将追踪的结果通过网络传输给显示系统；显示系统则结合各个检测系统的检测结果，通过多摄像头运动目标的融合分析判断运动目标是否为高空抛物。出现高空抛物，则保存此关键视频，并能够对此视频进行回放查看。

本系统包括一下几个模块：

(1) 系统设置模块

(2) 设备管理模块

(3) 视频图像采集模块

(4) 运动目标检测模块

(5) 运动目标追踪模块

(6) 抛物判定模块

(7) 抛物结果显示模块

(8) 视频拼接与显示模块

(9) 抛物视频保存与回放模块

如下图为各模块之间的结构流程，系统分为了两个子系统，分别是检测系统和控制显示系统。由于控制显示系统是供用户使用，实现对系统的控制和结果显示等功能，又称为客户端，包括系统设置模块、设备管理模块、抛物结果显示模块、视频拼接与显示模块、抛物视频保存与回放模块五个模块；检测系统实现对监控视频数据进行采集并分析处理，又称为服务端，包括视频图像采集模块、运动目标检测模块、运动目标追踪模块、抛物判定模块四个模块。服务端和控制端之间通过网络实现数据传输。



图3-1系统模块数据流程示意图

## 3.2 系统环境需求

如下表中对系统开发和运行需要的环境进行了说明。

表3-1 系统环境说明

|  |  |  |
| --- | --- | --- |
| 环境类别 | 系统需求 | 详细说明 |
| 运行环境 | Windows | Windows7操作系统及以上版本，GPU |
| 开发环境 | Windows | 服务端：VS2013、QT5.6、OpenCV3.0.0、C++、cuda  客户端：VS2013、QT5.6、OpenCV2.4.13、C++、cuda |

## 3.2 系统总体架构



图3-2 系统架构图

# 4 系统设置功能模块设计说明

## 4.1 程序描述

系统设置主要对系统进行先验的条件设置，否则系统无法进行工作。系统设置主要包括监控楼层的环境条件的设置、监控相机设备的条件设置以及系统工作中其他的一些设置等。

## 4.2 功能

对系统工作的环境进行设置，需要对系统以下参数进行输入，实现系统初始参数的设置。



图4-1 系统设置IPO图

## 4.3 性能

系统需要非常精确的对各个参数进行设置，否则将影响系统检测结果的正确性。

## 4.4 输人项

页面对应的各个输入项，以及其中要输入的变量的类型、标识符、默认值等进行说明。

表4-1 各输入项的说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 标识符 | 默认值 | 备注 |
| 楼层数 | int | FloorNum | 30 | 总的楼层数 |
| 楼高/层 | float | PerFloor | 3 | 每层楼的高度 |
| 安装距离 | double | distance | 15 | 相机安装与大楼的距离 |
| 安装高度 | double | CaHeight | 3 | 相机安装的高度 |
| 相机个数（垂直） | int | CameraR | 7 | 竖直方向相机个数 |
| 相机个数（水平） | int | CameraC | 2 | 水平方向相机个数 |

## 4.5 输出项

表4-2 设置后的输出项说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 命名 | 初始值 | 备注 |
| 每个相机监控楼层 | int | pFloors | 0 | 每个相机拍摄楼层数 |
| 设置标志 | bool | SetParamMsk | False | 标志是否进行了设置，正确设置后为true，设置失败为false |

如果设置成功，则会初始化与相机个数相同的视频保存队列，用来保存每个视频，同时对显示框进行初始化，水平和垂直方向的显示框的个数与相机水平和垂直个数相同，以显示每个摄像头监控的区域。

## 4.6 流程逻辑

如图为系统设置流程图。



图3.2 参数设置流程图

# 5 设备管理功能模块设计说明

## 5.1 程序描述

设备管理功能模块主要是对设备进行管理，包括设备登录、设备退出、设备添加三个功能。每个设备对应一个高空抛物检测子系统，即服务端程序，设备的管理包括系统之间的通信，以及高空抛物子系统的启动、每个服务端系统对应相机的登录。设备管理功能进行了客户端系统与服务端系统之间的通信、服务端系统启动、服务端相机管理、客户端相机管理等功能。

## 5.2 设备启动方案设计

### 5.2.1 客户端系统

1. 首先客户端系统启动，按一定频率发送广播，探测可用服务端系统

2. 客户端系统可以显示当前可用主机（Hosts）列表，可用服务端系统列表，及已登录相机的子系统相机信息，其信息都保存在cameraNode结构体中。

3. 当客户端系统需要启动新的服务端系统时，从已连接Hosts中选择一个（或自动选择最少服务端系统数的Host），向其Master Client发送启动命令及启动参数

4. 客户端系统可以从可用服务端系统中选择一个发送登录命令，收到登录成功消息后该服务端系统的相机的登录状态即变为可用已连接状态；

5. 客户端系统同时登陆相机，提供相机单独显示的监控界面接口。

未登录成功则相机登录状态不变；

### 5.2.3 服务端系统（摄像头控制结点，抛物检测系统)

1. 客户端系统被部署在任意Host（主机）上（可以和总系统在同一Host），当Host开机时，自动启动一个客户端系统，自启的客户端系统被赋予ID值0，并此客户端系统为该Host上的Master 客户端系统，Master客户端系统可以在同一Host上启动新的客户端系统实例，新的客户端系统实例被赋予的ID值+1，即启动的客户端系统的ID值依次递增，ID值非0的客户端系统被称为Slave 客户端系统；

2. Master 客户端系统在自启时其相机的登录状态为不可用未登录状态，收到总系统广播后变为可用未登录状态，收到总系统登录命令时进行摄像头连接，连接结果回复总系统；

3. Slave 客户端系统在启动时即连接总系统，连接成功后登录摄像头，最后回复总系统

4. 客户端系统可以手动启动

5. 若Master 客户端系统启动Slave 客户端系统失败，则应该回复总系统启动失败；

## 5.3 设备登录

### 5.3.1 设备登录功能

点击设备管理按钮，进入设备管理界面，设备管理显示已经存在的检测客户端系统实例以及已经存在的检测客户端系统实例的相机登录状态，如果未登录的检测客户端系统实例可以输入相机所需的登录信息以及对应的相机的编号进行相机的登录。



图5-1 设备登录IPO图

### 5.3.2 设备登录输入项

设备登录包含四个输入项，每个输入项要转换成相应的格式，其格式与需求如下表所示。

表5-1 设备登录输入项说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 命名 | 初始值 |
| 相机IP | QString | cameraIP | - |
| 相机用户名 | QString | cameraUsername | - |
| 相机密码 | QString | cameraPass | - |
| 相机ID | int | cameraID | - |
| 最低楼层 | int | LowFloor | - |
| 最高楼层 | int | HighFloor | - |

### 5.3.3 设备登录输出项

客户端系统登录成功后会更新设备信息，因此会有以下输出结果。

表5-2 设备登录输出项说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 登录结果 | 名称 | 类型 | 命名 | 初始值 |
| 登录成功 | 相机IP | QString | cameraIP | 无 |
| 相机用户名 | QString | cameraUsername | 无 |
| 相机密码 | QString | cameraPass | 无 |
| 相机ID | int | cameraID | 0 |
| 最低楼层 | int | LowFloor | - |
| 最高楼层 | int | HighFloor | - |
| 登录状态 | quint8 | status | 0 |
|  | | | | |
| 登录失败 | 登录状态 | quint8 | status | 0 |

### 5.3.4 设备登录流程逻辑



图5-2 设备登录流程图

系统启动，按一定频率发送广播，探测可用客户端系统，显示当前可用客户端系统列表，以及客户端系统中设备登录的状态，未登录的客户端系统可以发送登录命令，通过TCP协议发送给客户端系统进行相机登录，客户端系统接收到登录命令及登录信息，进行相机登录，登陆后将登陆结果通过TCP发送给总系统，总系统更新设备信息。客户端系统设备信息保存在cameranode结构体中。

## 5.4 设备退出

### 5.4.1 设备退出功能

已经登录相机的实例，可以退出相机，退出相机包含两种情况，如果实例是master，则停止相机获取图片，如果实例是非master实例，则直接进行进程退出，退出成功后会刷新界面，只显示目前存在的客户端系统，以及相机相关信息。

### 5.4.2 设备退出输入项

设备退出发送设备退出命令。

### 5.4.3 设备退出输出项

刷新界面，只显示目前存在的客户端系统

### 5.4.4 设备退出流程逻辑



图5-3 设备退出流程逻辑

## 5.5 设备添加

### 5.5.1 设备添加功能

添加设备可以启动实例，可以选择要增加的实例的主机，同时也可以随机的选择主机，编辑要增加的实例中需要登录的相机的信息，包括相机IP，相机账号，相机密码，相机的ID，点击添加按钮就可以增加实例同时会登录相机，若增加实例成功会在下面的提示信息中显示“客户端启动成功”，否则会提示“客户端启动失败”，若相机登录是否成功，则会弹出提醒消息，同时会刷新界面。

### 5.5.2 设备添加输入项

设备添加需要选择想要添加客户端系统的主机，因此有以下输入项。

表5-3 设备添加输入项说明

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 命名 | 初始值 |
| 主机号 | string | host | 无 |
| 相机IP | string | cameraIP | 无 |
| 相机用户名 | string | cameraUsername | 无 |
| 相机密码 | string | cameraPass | 无 |
| 相机ID | int | cameraID | 无 |
| 最低楼层 | int | LowFloor | - |
| 最高楼层 | int | HighFloor | - |

### 5.5.3 设备添加输出项

客户端系统启动成功返回、客户端系统登录相机是否成功、客户端系统设备列表更新

### 5.5.4 设备添加流程逻辑



图4.4 设备添加流程逻辑图

# 6 抛物结果显示功能模块设计

## 6.1 程序描述与程序功能

抛物视频结果显示主要是对抛物检测结果的显示，抛物检测在客户端系统中实现，通过TCP网络传输给总系统，当客户端系统接收到传输的检测结果后进行抛物结果计算算法，并进行抛物结果显示。此功能是顺序处理，只有检测到抛物结果后才会执行。

## 6.2 性能

检测结果计算方法，楼层的高度计算需要明确知道检测到抛物的相机监控楼层的最低的楼层数，能够实时的将检测结果显示。

## 6.3 输出项

抛物检测结果，包括抛物时间（精确到秒）、楼层（精确到绝对楼层），

抛物报警，当出现抛物时，进行声音报警。

## 6.4 抛物信息计算方法

客户端系统通过网络将检测结果传输给总系统，检测结果中保存了检测到抛物的相机的编号（ID），检测时间，抛物下落的轨迹，抛物的抛出点在图片中的像素位置；抛物检测时间=客户端系统传输检测抛物结果的时间，抛出点的楼层计算方法与步骤如下：

(1) 计算相机拍摄中间楼层的高度



：相机拍摄中间楼层的高度

：相机拍摄的最低楼层

：相机拍摄的最高楼层

：每层楼的高度

(2) 像素转换比例



：像素转换比例

：相机安装与大楼的距离

(3) 转换后的像素



：转换后图像的高度

：图像原始高度

(4) 每层楼所占像素



：每层楼所占像素

(5) 抛物所在楼层



：检测抛物点的垂直像素点

：抛出点所在的楼层

## 6.5 流程逻辑



图6-1 检测结果显示流

## 6.6 接口

(1) 客户端系统与检测结果发送接口

void CTCPClient::sendMsg(const CDetectionNet \* msg)

(2) 总系统接收检测结果接口

void CTCPServer::handleRespDetection(QTcpSocket \*client, CDetectionNet \*msg)

## 6.7 限制条件

1. 客户端系统与总系统必须工作在局域网中。

2. 客户端系统与总系统建立了TCP链接。

# 7 抛物视频保存与回放功能模块设计说明

## 7.1 抛物视频保存

### 7.1.1 抛物视频保存功能与程序描述

抛物视频保存是客户端系统功能模块中重要的一部分，进行保存抛物视频，当检测到高空抛物时要进行此时间段的视频保存，视频的保存时长需要设置。

抛物视频保存包含缓存视频队列初始化、视频缓存、视频保存、保存文件命名、保存文件路径判断与创建等功能，其IPO图如下所示。



图7-1 视频保存IPO图

### 7.1.2 抛物视频保存性能说明

视频保存要确定保存的视频的长度，能够包含抛物的整个过程，又不能过长，所以要计算抛物整个过程所需要最长的时间，视频保存的时间略大于此时间即可，目前设置为8s，能够满足时间要求；视频保存的时间复杂度不能过高。

### 7.1.3 抛物视频保存输入项

完成视频缓存需要以下几个输入：

缓存时间长度：整型、cache\_duration

视频的帧率：整型、frame\_rate

保存的路径：字符串、save\_path

视频图片：Mat

检测时间：long

检测结果：vector<DetectResults>

### 7.1.4 抛物视频保存输出项

抛物视频保存输出含有抛物下落的抛物视频，保存在设置的保存路径中，保存的视频帧率与原视频帧率相等，视频保存的格式是.avi，视频的命名为相机ID + 抛物检测时间。

### 7.1.5 流程逻辑



图7-2 抛物视频保存流程图

## 7.2 抛物视频回放

对于检测到的高空抛物，能够查看高空抛物的抛物信息，包括出现高空抛物的时间、高空抛物的抛出位置、抛物阶段的抛物视频等信息。

### 7.2.1 程序描述

结果查看与视频回放功能能够显示抛物视频的列表，包括抛物视频的列表，列表中是检测到抛物的相机ID 检测到抛物的时间组合的名字列表，点击列表中的一个，会播放对应的检测到抛物的视频，播放完成后会进行提示“视频播放完毕”

### 7.2.2 功能

选择已经检测的某个高空抛物的，回放高空抛物的视频，并将此高空抛物的信息显示出来（抛物时间，抛出点）。



图7-3 结果查看与视频回放IPO

### 7.2.3 性能

准确的读取抛物视频，并进行播放，抛物视频的列表按照固定的频率进行刷新，以确定出现抛物视频的时候能够及时进行刷新抛物列表。

### 7.2.4 输入项

点击某个抛物视频的按钮，按钮的命名是相机ID+检测到抛物的时间

### 7.2.5 输出项

抛物视频进行播放，抛物的检测时间以及抛物的抛出点进行计算与显示，抛出点的计算要精确到抛物的楼层，以及抛出的单元户号；按照固定的频率对检测的抛物进行刷新，当发现新的抛物是，能够及时的在抛物列表中显示出来。

### 7.2.6 流程逻辑

下图为抛物视频回放流程图，抛物视频回放中有两个子线程，一个子线程刷新抛物列表，另一个子线程抛物视频的播放与信息的显示。



图7-4 抛物视频回放流程图

### 7.2.7 存储分配

需要较大的存储空间来存储抛物视频，同时抛物视频也需要定期的进行删除。

### 7.2.8 限制条件

总系统与客户端系统在一台主机上运行。

# 8 视频拼接显示功能模块设计说明

## 8.1 程序描述

对系统中的多个监控相机的视频进行拼接显示，得到监控区域大楼的整体视频显示。 通过将所有摄像机监控视频拼接显示，避免了传统多个视频的监控查看的繁琐，使得监控人员只需查看一个整体监控视频。

本程序通过对传输的多个视频，对每一帧各个摄像机对应的图像进行拼接，得到实时拼接图像，实时显示监控大楼的整个视频。初始化时，对拼接相关数据进行记录。实时拼接时，常驻内存实时替换每一帧的图片，进行GPU运算并行映射每一像素点的值，因此速度能达到实时拼接的效果。

## 8.2 功能



图8-1视频流拼接IPO图

## 8.3 性能

各个摄像机准确对应的同步视频流，对拼接结果大小比例设置

## 8.4 输入项

表8-1 视频拼接输入项说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 命名 | 默认值 | 备注 |
| 帧图像 | Vector<Mat> | imgs | - | 每一帧各个视频的图像 |
| 缩小比例 | float | ResizeScale | 1 | 结果图的压缩比例 |
| 相机个数（垂直） | int | m | 2 | 竖直方向相机个数 |
| 相机个数（水平） | int | n | 1 | 水平方向相机个数 |

## 8.5 输出项

表8-2 视频拼接输出项说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 命名 | 初始值 | 备注 |
| 结果图 | Mat | result | - | 每帧拼接结果图 |

## 8.6 算法

视频拼接需要经过图像预处理、图像配准、图像融合三个步骤，其基本原理流程如下图8-2所示。

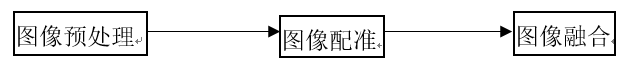


图8-2 基本原理流程

基本拼接原理：通过对两张有重复内容的图片进行特征提取，特征点匹配，计算变换矩阵，将两张图变换到同一个坐标系下，然后根据前面提取的特征点的匹配数据，将变换后的图片拼成一张大图，并对拼接缝进行融合处理。

以上是常规的图像拼接的流程。而对于本项目的视频拼接，基于视频源像素级别高，一张200W像素，2\*7张图，一秒钟处理24帧图。目前一般拼接效果比较好的拼接图片耗时都在秒级别以上，这样应用于视频流中是很不现实的。因此基于本项目特点，基于摄像机的位置不变性，通过初始化对相关参数的记录，在实时拼接视频流时，直接应用该数据进行拼接，达到实时拼接视频的效果。

**（1）初始化CPU阶段**

CPU部分主要做常规的拼接数据的获取，这部分是GPU并行的基础。主要分为拼接部分和记录参数两部分。

拼接部分：主要包括对特征点提取，匹配特征点，根据匹配对计算单应性矩阵，推算出内参矩阵和旋转矩阵，对多个相机的参数分析矫正。

特征点提取：本程序采用orb特征点提取算法。Orb是一种快速特征点提取和描述的算法。将FAST特征点的检测方法和BRIEF特征描述子结合起来，在此基础上做了改进和优化。先进行特征点的粗提取；对该像素点为圆心，半径为3pixel的圆，若圆周上有连续n个像素点的灰度值比圆心的灰度值大或小，则认为圆心处像素点为特征点。再用决策树筛选出最优特征点；对于局部较为密集的特征点，使用非极大值抑制去除，保留响应值较大的特征点；建立金字塔，实现特征点的尺度不变性；使用矩法确定特征点的方向，矩计算特征点以R为半径范围内质心，特征点坐标到质心形成的向量作为该特征点的方向。 矩定义公式如下：



特征点的匹配：使用最近邻和次近方法，相比于计算两个特征点的欧氏距离确定匹配度，可大大提高特征点匹配的正确率。

对特征点寻找欧式距离中最近的特征点与次邻近点距离的比值来对特征点进行匹配筛选，如果最近欧式距离除以次近欧式距离比值少于规定的某个比例阈值，则接受这对匹配点。降低比例阈值，匹配的特征点数目会减少，但会更加稳定。

BestMatch\_nearest/\_nextTo<ratio

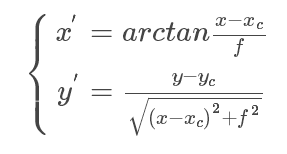
(欧氏距离针对多维空间中两点距离, A点对应n维数据：A = (x1, x2, x3….，xn);B点对应的n维数据：B = (y1, y2, y3…..,yn)。计算欧式距离的公式AB = (x1-y1)^2 + (x2-y2)^2 + (x3-y3)^2 + ……(xn-yn)^2;)

对于特征点来说，对图像矩阵中的n个像素点，这些像素值组成了该图像的特征组，就形成了n维空间）

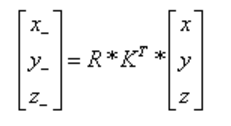
根据匹配正确的特征点，计算出单应性矩阵H，求出符合条件的焦距f，最后对f的均值或中值当成所有图形的粗略估计f, 再构建最大生成树，广度优先搜索求出根节点，并求出相机的旋转矩阵和光轴中心。

再使用Bundle Adjustment方法对所有图片进行相机参数的矫正，因为成对的单应性矩阵合成全景图时，会有全局的偏差影响，造成累积误差。对每一图像加上光束法平差值，使图像被初始化成相同的旋转和焦距长度。每一个特征点都要映射到其他图像中，计算出使误差的平方和最小的相机参数。

单应性矩阵变换：当把一个平面图像映射到一个弯曲的表面就会发生图像投影，反过来也是一样。小的视角下，得到的图像视觉上没有大的区别。如果拼接多个视角下，就会有明显的视角弯曲。圆柱投影公式：f是相机焦距，xc, yc是图像中心



原图像上点，经过旋转矩阵R,相机参数矩阵K，经过变换后的坐标，映射到投影面上。变换公式如下：反映射也需要该公式。



根据映射公式，对图像的四个边界求映射后坐标，然后确定变换后图像左上角右上角坐标，并且需要判断投影后坐标是否在有效区域内。投影像素计算使用线性插值。

记录参数：考虑到监控摄像头在一开始就是固定好的，一般情况下不需要进行大变动。因此由特征点得到的变换矩阵等信息一般不会有变动。所以可以通过记录中间参数来省去部分运算时间，达到实时拼接效果。但是图中的融合部分的数据是需要实时获取，因为基于不同时刻的光照，不同摄像头的曝光所处位置等，不同图片的相同重叠区域受光不一致。

因此，对融合部分采取渐变权重处理。通过对不同图片不同区域所占比重，来决定其在最终图片中融合权重，这样能够有效解决拼接部分的明显曝光不一致问题。同时，这种方法取代原来方法中多频段融合的复杂过程。

所以，记录参数部分主要分为三个步骤

1.对每张小图在大图区域位置记录，得到VariantMask。

2.对各个像素点在原图上位置计算，得到mapValue

3.计算重叠区域权重值，得到alphaMat

整体思想主要是利用前期的图像匹配得到的数据分析记录，隔离每个像素点之间数据的关联性，使得像素点之间数据独立。这样，才能最后应用到GPU阶段映射。

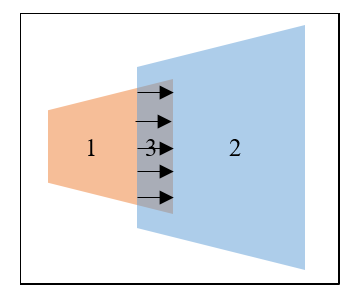


图8-3 重叠范围标记图

VariantMask: 主要根据图片来源摄像机位置确定该图片的区域值，对前面得到的变换矩阵，得到每个小图在大图上的区域范围，具体为左上顶点坐标和图片大小。遍历和结果图一样大小的mat，对每个像素值记录该处应对应于那张小图。通过对单通道图的像素值赋值来确定。这样便于GPU映射判断取值。

VariantMasl[x] = IncreaseValue\*I;

VariantMask相当于记录了结果图上某像素点的来源，告诉GPU去映射哪张小图，而mapValue则记录了该像素点的具体值，即其对应在小图的位置信息xy，告诉GPU具体去小图上哪一个像素点的值。结果图上点是由小图得到，除了重叠区域由两个小图上对应点得到，其他为单张图上点。因此，通过一个四通道浮点型mat存储。遍历大图，对每个像素点进行反变换，得到来源点在小图上位置信息。对于重叠区域，则需要记录两个小图的xy信息。取值大概思路：

int offset = mapBack(i);

MapValue[i][0] = img0[i].x; MapValue[i][1] = img0[i].y;

MapValue[i][2] = img1[i].x; MapValue[i][3] = img1[i].y;

(这里只是表明大概思路，实际代码复杂多)

以上两步是解决了对小图变换到同一张图片上的位置变换的信息记录，接下来解决边界区域比较明显问题。此处采用根据距离的权重渐变融合方法，在重叠区域，根据各个像素点间距离计算融合权重值。GPU映射时，根据该权重计算两小图的像素值之和作为结果图的值。权重计算公式：

Alpha = （x-start）/distance

**(2) GPU阶段：**

由于项目需要实时拼接视频，基于视频流像素级别高，每一帧都进行以上CPU阶段是不可行的。按照传统的对程序多线程的运行，也会涉及到图片的多次上传下载，而这部分是GPU运算中极为损耗时间的部分。因此，基于本项目特点，在初始化阶段对CPU阶段拼接相关数据进行记录实现每个像素之间的映射无关联性。这样便于在GPU映射阶段对每个像素点并行计算，极大地缩短了时间，达到实时要求。

由于前面已经记录了需要的相关数据，在GPU端主要是基于OpenCL的并行运行。初始化设置好VariantMask、mapValue、alphaMat相关值和内存，每一帧传入多个图像数组。OpenCL中对每个像素点根据VariantMask判断得到来源小图，在mapValue采样得到对应的位置取值，赋值到结果图上该像素点。

Result[idx] = img[idx];

如果处于重叠区域，则根据权重计算该点值。

Result[idx] = img0[idx1]\*alpha + img1[idx2]\*(1-alpha);

## 8.7 流程逻辑

图8-4为拼接模快的流程控制图



图8-4 流程控制图

## 8.8 接口

本模块从视频传输获取每个相机的视频帧，得到的图像容器，传入拼接模快进行拼接。得到的拼接图Mat类型给显示模块。如图8-5所示：



图8-5 关模块间关系接口

## 8.9 限制条件

前提是摄像机监控的区域，得到的图片序列彼此间有足够重叠区域，图像才能拼接成功。才能进行第一步的拼接初始化部分。

# 9 图像采集功能模块设计说明

## 9.1 程序描述与功能

本模块主要实现数据的采集，获得抛物检测的视频监控场景。

利用监控摄像头厂家提供的二次开发SDK获取监控录像视频流，视频由一系列的图片构成，将视频流按照一定的频率获取，对获得的媒体流解码，获得图片并保存在缓冲队列区，为运动目标检测模块提供数据资源。

## 9.2 性能

按照规定的频率和码率实时读取摄像头监控视频流，并对其进行无损的解码，防止解码丢失监控场景的信息。

## 9.3 输入项

客户端程序发送抛物检测功能开始命令。

## 9.4 输出项

解码获得的视频帧图像，利用CBlockingQueue<cv::Mat>结构按照队列的方式保存在内存中。

## 9.5 算法

利用CBlockingQueue<cv::Mat>结构保存采集图像使图像采集模块作为一个单独的子线程，与其他模块在不同的线程中运行，中间通过缓冲队列进行连接，缓冲队列做线程同步控制，控制方式采用生产者-消费者模型，如图9-1所示，通过给系统发送图像采集信号，唤醒图像采集线程，获得监控视频流并对视频流解码，采集视频原始图片挂入缓冲区队尾，并更改临界区相关信号量；当缓冲区中有图像数据时，便唤醒运动目标检测线程，从队头取出图片做进一步处理。利用此模式使系统图像采集与图像处理有短时间的容差。但此时间也是有限制的，缓冲区大小的设置决定了容差时间的大小。如果系统检测线程处理速度过于缓慢，导致缓冲区数据越积越多，当接近于缓冲区总大小时，系统迅速挂起数据采集线程，并回馈流水线降低速度指令， 防止缓冲区溢出系统崩溃。



图9-1 生产者-消费者模型控制方式示意图

## 9.6 流程逻辑

本模块的流程图如图9-2所示：



图5.3 检测模块流程图

## 9.8 接口

服务端接收开始命令接口：

CRawUDPClient::handleMsg(const ControlMsg &msg, const QString &fromIP)

视频图像保存在全局变量中：

CDocument::CBlockingQueue<cv::Mat> cameraImage

## 9.9 存储分配

根据设置的CBlockingQueue<cv::Mat> cameraImage的容量大小N以及读取的图片的大小决定所需内存的大小为N\*M；

# 10 运动物体检测功能模块设计说明

## 10.1 程序描述

本模块的程序是高空抛物检测追踪的核心模块之一，其功能是从输入的视频中提取出每一帧中的运动物体，并为之后的跟踪模块提供输入数据。

本程序需要常驻在内存中执行，与系统主模块（显示模块）是并行处理的。

## 10.2 功能



图10-1 运动目标检测IPO图

## 10.3 性能

本模块主要是针对输入的视频流，提取出视频流中的运动物体，其检测的最小精度大约是7\*7像素点大小的运动物体，该模块主体程序运行于GPU上，输入视频源为200万像素时，在1050ti的GPU上运行，需要5ms的时间，另外，位于CPU上的部分，需要运行7ms，故总共一帧需要12ms左右的时间。

## 10.4 输入项

输入数据时从摄像头直接读取的视频流，并在程序中解析为一帧一帧的连续图像。

## 10.5 输出项

输出是视频流中所有运动物体的大小和运动物体存在于图像中的位置，并使用vector容器来实现对多对象的存储。

## 10.6 算法

本模块选取的是Vibe算法，Vibe算法是一种像素级视频背景建模或前景检测的算法，主要利用了图像每个像素点在一段时间内的像素值以及像素点邻域的像素值，直接取它们的值作为模型参数。然后输入帧和背景样本集匹配，判断是否为前景，同时利用背景像素点及相邻像素点按照等概率获取判断对此像素点的模型更新。ViBe算法主要分为三个模块：背景模型初始化、模型匹配、背景模型更新。

首先，背景模型初始化时只需要利用输入视频序列第一帧图片。对图片中的每个像素点随机选择此点的像素值或者邻域的像素值作为模型。例如图像中在像素点，从其8邻域中随机选取个样本作为模型初始化参数。

|  |  |
| --- | --- |
|  |  |

其中，为从8邻域中随机选取像素点的值。

然后，对输入的图像进行模型匹配，通过计算输入图像像素值与背景模型的相似度检测运动目标。假设当前输入帧是，像素点处的像素值是。如图10-2所示，定义一个以为中心、为半径的球体，表示所有与距离小于的点的集合，用落在区域内样本的个数表示与背景模型的相似度，对于给定的阈值：如果，则判定的为前景；如果则判定为背景。如下式所示。

|  |  |
| --- | --- |
|  |  |

其中，表示与的距离。



图10-2 二维ViBe模型匹配示意图

最后，为了适应监控场景中模型的背景变化，使模型能够较好的描述当前时刻的背景，需要对模型进行更新。ViBe算法根据当前帧像素值来更新时刻的背景模型，每一个背景点具有相同的概率来更新模型有的概率，同时利用像素值空间传播特性对邻域点也有相同的概率进行更新模型。在连续的一段时间中，时刻模型中的某个样本没有更新的概率是，则时间后样本还是没有发生变化的概率的计算方法如下式。

|  |  |
| --- | --- |
|  |  |

算法的模型更新采用了随机选取样本值进行更新的策略保证样本值的平滑的生命周期，并达到了样本值的更新时间无关性即时间无关。因此ViBe背景建模算法具有无记忆更新、时间取样更新、空间邻域更新三个特点。

ViBe算法原理简单并巧妙的利用了图像的邻域特征，直接利用像素值没有复杂的计算，易于实现，使得模型在使用过程中达到较好的效果，并对资源占用较小，同时由于模型样本随机提取及邻域提取策略使算法抗噪能力较好，能适应大部分实际监控场景。

ViBe运动目标检测大部分是图像的像素的计算操作，为了使系统具有更快的响应速度，这里可以对图像处理ViBe算法做并行化的改进。利用OpenCL编程方法对算法并行化处理，充分利用GPU强大的并行计算能力并与CPU进行协同工作，更高效的利用硬件高效的完成ViBe算法中大规模的、高并行度的计算，实现系统加速功能。OpenCL编程利用了 GPU 的特点进行多线程并行处理实现ViBe算法，并行化后的算法在处理高分辨率的图像有明显的优势。

将获得前景图进行连通性分析，获得检测的运动物体的位置，并将检测到的运动目标从图中分割出来，计算运动目标的面积与周长，作为运动目标的特征。

## 10.7 流程逻辑

本模块的流程图如图10-3所示：



图10-3 检测模块流程图

## 10.8 接口

本模块的“上下文”关系如图10-4所示：



图10-4 检测模块“上下文”

本模块为检测模块，承接摄像头模块，获取视频数据，同时本模块的输出是跟踪模块的输入，是跟踪模块运行的基础。

## 10.9 存储分配

本模块仅存储当前帧的运动物体的坐标和大小数据。

## 10.10 限制条件

由于本模块使用了GPU，故需要在有GPU的机器上运行，才可达到预期效率。

# 11 运动目标追踪功能模块设计说明

## 11.1 程序描述

本模块是系统的另一大核心程序，本模块主要功能是讲一系列的视频帧中对应的运动物体“连”成一条轨迹，即实现对运动物体的“追踪”。

本程序需要常驻在内存中执行，与系统主模块（显示模块）是并行处理的。

## 11.2 功能



图11-1 跟踪模块IPO图

## 11.3 性能

本模块主要是针对通过检测算法得到的运动物体的坐标和大小，对视频每组相邻的两帧做匹配，进而实现运动物体的跟踪。本算法在白天和夜晚的效果有较大差距。白天算法运行效果较好，正检率可达90%以上，且误检率低于10%。晚上算法运行效果相对较差，会有部分漏检，同时误检率也会有所提升。检测算法运行于CPU上，针对200万像素的视频流得到的数据，在i7-7700hq上运行需要10ms左右。

## 11.4 输入项

输入数据是由检测模块得到的运动物体坐标及大小的数据。

## 11.5 输出项

输出数据是每一个运动物体的运动轨迹。

## 11.6 算法

经过运动目标检测过程的处理，已经获得了运动目标以及运动目标的特征，将运动目标的特征与追踪算法结合起来，对视频中每帧图片进行目标定位，获得运动目标的位置的变化并将运动轨迹标识出来。由于抛物大小条件的限制，通过第四章中对运动目标追踪算法中的介绍选择了基于卡尔曼滤波的运动目标追踪方法。

使用卡尔曼滤波器来预测视频序列中下一帧中的目标可能的位置，从而确定目标对象在X轴和Y轴的搜索区域，然后利用匈牙利匹配算法在搜索区域中找到目标的准确位置，并利用目标的准确位置来更新卡尔曼滤波器模型，缩小了运动目标可能出现的区域，从而很大的提高了追踪的效率。同时为了提升追踪的准确性，考虑抛物在下落的过程中其形变可能较严重、容易发生旋转等情况，特征提取比较困难，但目标的大小发生变化较小，因此选择目标的面积、周长特征与目标的质心特征结合在一起作为特征模板，作为追踪的输入，实现目标的追踪。因此本模块主要采用了两个算法——匈牙利匹配算法和卡尔曼滤波算法。

1. 匈牙利匹配算法

匈牙利算法是二分图匹配常见的算法之一，该算法的核心就是寻找增广路径，它是一种用增广路径求二分图最大匹配的算法，其主要分为三个步骤：

a．置匹配M为空。

b．找出一条增广路径P，通过取反操作获得更大的匹配M’代替M。

c．重复2中的操作直到找不出增广路径为止。

通过讲跟踪问题进行转化，可使用匈牙利算法实现目标跟踪：

a．将视频看作一系列的图片，首先使用检测算法检测每一帧的运动物体，并将其运动物体的中心点取出；

b．对每组相邻的两张图片，将前一张图片中的运动物体中心点视作点集A，后一张图片中的运动物体中心点视作点集B，将集合A中的每一个点与B中的每一个点连接，连接的权值为两点的距离，至此则构造了一个图结构。

c． 将2中得到的图结构输入匈牙利算法中，求其最佳匹配，即可将前后两帧的相同运动物体匹配，进而对整个视频进行处理即可得到运动物体的运动轨迹。

1. 卡尔曼滤波算法

由于会存在部分帧因为角度的问题，导致 “检测失败”，故需要运动物体在这些帧的坐标进行“预测”。卡尔曼滤波通过对已有轨迹，对部分检测不到的帧，进行预测，并将该预测值作为运动物体在该帧的位置。卡尔曼滤波器采用递推运算方式对随机信号进行计算和预测，对输入的实际系统的观测参数进行预测，输出预测的状态参数。

卡尔曼滤波主要包含状态方程式(11.1)和观测方程式(11.2)两个部分。

|  |  |
| --- | --- |
|  | (11.1) |
|  | (11.2) | |

其中，、分别是、时刻系统的实际状态向量，称为系统控制矩阵，是至时刻的状态变化矩阵，是时刻的状态预测向量，为联系矩阵，表示系统的输入，表示系统中出现的过程噪声，为系统中出现的观察噪声，并且和是相互独立的。

用表示信号的先验状态估计，表示信号后验的概率。则先验估计误差的协方差如式(3.33)和后验估计误差的协方差如式(3.34)所示。

|  |  |
| --- | --- |
|  | (3.33) |
|  | (3.34) |

其中， 表示先验估计误差的协方差，后验估计误差的协方差

则系统状态更新方程如(3.35)。

|  |  |
| --- | --- |
|  | (3.35) |

其中，反映了预计值和实际值之间的误差，为卡尔曼滤波的增益矩阵，是预测的关键。将式(3.35)带入式(3.33)中，然后对求导，令导数为0，即可解出，如(3.36)所示。

|  |  |
| --- | --- |
|  | (3.36) |

当接近于0的时候，则。当趋于零时，则。

卡卡尔曼滤波包括预测和修正两个步骤，利用系统的反馈信息预测运动目标的状态。完整的卡尔曼滤波公式主要有五个，如下所示。

卡尔曼预测方程

|  |  |
| --- | --- |
| 状态方程： | (3.37) |
| 协方差预测方程： | (3.38) |

卡尔曼修正方程

|  |  |
| --- | --- |
| 卡尔曼的滤波增益： | (3.39) |
| 修改状态向量： | (3.40) |
| 修改协方差矩阵： | (3.41) |

使用卡尔曼滤波器实现目标追踪基本分为三个步骤：首先初始化卡尔曼滤波器，将目标的位置和速度赋值给卡尔曼滤波器，并记录当前帧的时间；然后进行状态估计，利用当前帧和上一帧的时间间隔进行卡尔曼预测，预测当前帧的目标的位置和运动状态，并搜索区域中的最佳匹配，获得真实目标的位置和运动状态值，同时记录当前的时间；最后进行状态更新，用真实状态向量作为卡尔曼滤波器的输入更新卡尔曼滤波器的状态。

## 11.7 流程逻辑

本模块的流程图如图11-2所示：



图11-2 跟踪模块流程图

## 11.8 接口

本模块的“上下文”关系如图11-3所示：本模块为跟踪模块，承接检测模块，获取检测模块得到的运动物体相关数据，同时本模块的输出是判断模块的输入，判断模块通过对跟踪模块输出的轨迹进行判断，从而判别该轨迹对应的物体是否为高空抛物。



图11-3 跟踪模块“上下文”

# 12 高空抛物判定功能模块设计说明

## 12.1 程序描述

本模块是的主要功能是对跟踪模块输出的轨迹进行判别，判断哪些轨迹属于高空抛物物体的轨迹，并将此类轨迹存储显示，其它轨迹删除。

## 12.2 功能



图12-1 判断模块IPO图

## 12.3 性能

本模块主要是针对通过跟踪算法得到的运动物体的轨迹，通过判断规则，将正确的抛物轨迹提取出来，并去除剩余的非抛物轨迹。

本模块能够去除大部分明显不符合抛物规律的误检，但针对部分由噪声连成的类似于抛物轨迹的误检并不能很好的去除。针对白天的视频，可以去除90%以上的误检，但针对夜晚视频，会有相对较多的误检。

## 12.4 输入项

输入数据是由跟踪模块得到的运动轨迹。

## 12.5 输出项

输出数据是去除误检之后的抛物轨迹。

## 12.6 算法

高空抛物判定方法即是对运动目标的行为进行分析，判断运动对象是否符合高空抛物运动的运动规律，以及运动目标的轨迹是否符合高空抛物运动的限定。因此高空抛物的判定主要对多个摄像机检测与跟踪获得的运动目标的轨迹进行分析，并通过高空抛物的先验条件判别，判断哪些运动目标的轨迹属于高空抛物，并将高空抛物对象的轨迹存储显示，非高空抛物对象的轨迹删除。根据高空抛物的运动特点，主要利用了一下几个抛物特征作为抛物判断的主要条件。

(1) 高空抛物的运动轨迹始终是向下运动，因此运动目标轨迹的纵坐标的值是持续增大的。

(2) 高空抛物运动目标轨迹点的数目M要满足设置的阈值T，即M>T，运动目标在监控范围内运动，其运动定会运动一定的时间，因此检测的轨迹点的数目会有一定的限制；

(3) 高空抛物运动目标轨迹从监控范围的边界终结，抛物运动不会在某个地方停留，会从某个边界地方出监控范围，同时边界区域也不会是上边界区域，只会从左、右及下边界区域；

(4) 抛物运动在开始部分基本上在做加速运动，但受到环境等的影响，无法避免会出现检测目标没有减速和匀速的部分点，但整体上做加速运动，因此设置比例阈值以去除一直匀速或者减速的运动目标。

(5) 高空抛物运动目标下落过程中，纵坐标的差分数值不会一直太小，因此可以去除纵坐标几乎重合的噪声。

对运动轨迹的分析，需要对运动目标的运动轨迹进行曲线拟合，本文利用最小二乘法多项式拟合对运动目标的轨迹进行拟合。

最小二乘法多项式曲线拟合构造多项式曲线近似拟合给定的包含个点的曲线，使得拟合曲线的偏差最小。假设给定数据点，为次数不超过的多项式构成的函数类，求，使式(12.1)取最小值的解为最小二乘拟合多项式，当时为线性拟合，当时为多项式拟合。

|  |  |
| --- | --- |
|  | (12.1) |

从公式(12.1)可知为的多元函数，因此求解的极值问题，因此对求导得(12.2)，化简获得(12.3)。

|  |  |
| --- | --- |
|  | (12.2) |
|  | (12.3) |

因此求解方程组(12.3)可得到最小二乘法的拟合曲线。

## 12.7 流程逻辑

本模块的流程图如图12-2所示：



图12-2 高空抛物判定模块流程图

## 12.8 接口

本模块的“上下文”关系如图12-3所示：



图12-3 跟踪模块“上下文”

高空抛物检测结果进行网络传输到客户端系统，通过如下接口进行发送：

void CTCPClient::sendMsg(const CDetectionNet \* msg)

## 12.9 存储分配

本模块存储从系统开启之后所有的高空抛物物体的运动轨迹及相应数据。

## 12.10 限制条件

本模块要求输入的误检轨迹尽可能的少，即在源头上需要减少噪点。

# 13 用户界面设计

## 13.1 界面设计规范

### 13.1.1 以用户为中心

用户界面设计的一个重要原则是用户控制软件，而不是软件控制用户，也就是要以用户为中心。

 操作上假设是用户（而不是计算机或软件）开始动作。用户扮演主动角色，而不是扮演被动角色。在需要自动执行任务时，要以允许用户进行选择或控制它的方式来实现该自动任务。

 采取交互式和易于感应的窗口，尽量避免使用模态对话框，而使用"非模式"辅助窗口。 "模式"是一种状态，它排除一般的交互，或者限制用户只能进行特定的交互。当使用一个模式或该模式只是可替换的设计时（例如，用于在一个绘图程序中选定一个特定感觉），要确保该模式是显然的、可见的，是一个明确的用户选定的结果，并且容易取消。

 当后台运行长进程时，保持前台式交互。例如，当正在打印一个文档，即使该文档不能被改变，用户也可以最小化窗口。

 谅解。用户喜欢探索一个界面，并经常从尝试和错误中学习。一个有效的界面允许交互式的发现，它只提供一组合适的选择，并在用户可能破坏系统或数据的情况时发出警告。如果可行，还应提供可逆转或可还原的操作。即使在设计得很好得界面中，用户也可能犯错误。这些错误既可以是物理上得（偶然地指向了错误的命令或数据），也可以是逻辑上的（对选定哪一个命令或哪些数据做出了错误的决定）。有效的设计避免很可能导致错误的情况。它还包容潜在的用户错误，并且使用户易于还原。

### 13.1.2 清楚一致的设计

一致允许用户将已有的知识传递到新的任务中，更快地学习新事物，并将更多的注意力集中在任务上。这是因为他们不必花时间来尝试记住交互中的不同。通过提供一种稳定的感觉，一致使得界面熟悉又可预测。一致在界面的所有方面都是很重要的，包括命令的名称、信息的可视表示，操作行为，以及元素在屏幕和窗口内部的放置。

 相同含义的词要使用统一的术语。统一约定一个称谓，且此称谓是用户熟悉的和易于理解的。

 使用一组一致的命令和界面来展示常见功能。例如，避免一个"复制"命令在一种情况下立刻执行一个操作，但在另一种情况显示一个对话框要求用户键入目标然后才执行。应该使用同样的命令来执行对用户来说相似的功能。

 操作环境内的一致。保持Windows提供的交互操作和界面约定之间的高度一致，用户将能很快熟悉软件的使用。

 使用隐喻的一致性。如果一个特定的行为更多的是一个不同的事物的特征，而不是它的隐喻的含义，那么用户可能在学习将行为和该事物相关联时遇到困难。

 建立项目保留字。通过建立保留字来明确和统一术语和操作命令。

 提供可视反馈。在后台运行长进程时（时间超过1～10秒，视具体情况而定），提供进度条等信息指示。

 除非特别必要时，不要提供声音反馈。在有严重的问题发生时，可以使用声音来提示用户，但是通常应该允许用户取消声音。

 保持文字内容清楚。信息的表达要言简意赅，易于理解而又不罗嗦；避免使用冗长的文字给用户反馈。

### 13.1.3 有良好的直觉特征

用熟悉的隐喻为用户的任务提供直接而直观的界面。通过允许用户利用他们的知识和经验，隐喻使得预测和学习基于软件的表示的行为更加容易。

 在使用隐喻时，不需要将基于计算机的实现局限在真实世界的对应物上范围之内。例如，与其基于纸张的对应物不同，Windows桌面上的文件夹可以被用来组织各种对象，例如打印机、计算器、以及其他文件夹。同样，Windows文件夹可以其真实世界对应物不可能的方式被排序。在界面中使用隐喻的目的是提供一个认知的桥梁；隐喻并不以其自身为最终目的。

 隐喻支持用户认知而不是记忆。用户记起与一个熟悉的事物相关联的意义要比他们记起一个特定命令的名称要容易得多。

 同常见软件保持一致性。出色的用户界面在程序中将实现同用户以前用过的其它成功软件一致的动作。

### 13.1.4 较快的响应速度

保持界面能很快对用户操作做出反应。

提供快捷键。特别对于有大量录入项的界面，能让用户不使用鼠标即可完成快速数据录入。在用户界面中加入一些功能，这些功能可以让熟练用户在不同的区域快速的输入数据。这些功能包括重复功能、快捷键、有意义的图标按钮等等，所有这些可以使速度快的用户可以控制界面并加快数据的输入。

 除非必要，不要刷新屏幕。

### 13.1.5 简单且美观

 1．简单

界面应该很简单（不是过分单纯化）、易于学习、并且易于使用。它还必须提供对应用程序的所有功能的访问。在界面中，扩大功能和保持简单是相互矛盾的。一个有效的设计应该平衡这些目标。

支持简单性的一种方法是将信息的表示减少到进行充分交流所需的最少信息。例如，避免命令名和消息的文字描述。不相关或冗长的句子扰乱了您的设计，使用户难以容易地提取重要信息。

另一个设计简单而有用的界面的方法是使用自然的映射和语意。界面元素的排列和表示影响它们的意义和关联。简单与熟悉相互关联。熟悉的事物通常似乎更简单。尽可能尝试建立利用用户已有的知识和经历的联系。可以使用渐进揭示来帮助用户管理复杂的事物。“渐进揭示”涉及到仔细的信息组织，以便只在恰当的时候才显示信息。通过隐藏向用户表达的信息，减少用户必须处理的信息数量。例如，可以使用菜单来显示操作或选择的列表，还可以使用对话框来显示一组选项。渐进揭示并不意味着对显示信息使用非传统的技术，例如需要一个修饰键作为访问基本功能的唯一方法，或者强迫用户通过一个更长的分级交互序列。这会使用户界面更加复杂和麻烦。

2. 美观

可视设计是应用程序界面的重要部分。可视属性提供了非常好的印象，并传达特定对象的交互行为的重要线索。同时，出现在屏幕上的每一个可视元素也是很重要的，它们可能竞争用户的注意。提供清楚地促进用户对表达的信息的理解的连贯环境。图形和可视设计的技巧对于这一方面是无价的。

## 13.2 界面关系和流程







图13-1 界面关系和流程示意图

## 13.3 系统界面设计

### 13.3.1 主界面设计

系统运行成功后会显示主界面，主要是一些操作菜单



图13-2 系统主界面

### 13.3.2 系统设置界面

系统运行成功后需要先进行系统的设置，否则系统无法进行工作。

要设置系统的工作的环境，包括系统工作监控的总的楼层数，每层楼的高度，相机安装与大楼的距离、相机安装的高度，竖直方向上相机的个数，水平方向上相机的个数，如图13-3所示。



图13-3 系统设置界面

### 13.3.3 设备管理界面

设备管理界面主要有三个功能，分别是相机登录、相机退出、添加设备。如下图13-4所示。

主机表示运行子程序的计算机的IP，标识哪一台计算机，客户端用来标识确定的子系统，一个计算机可以运行多个子系统，相机的登录状态表示此子系统登录状态的状态，若是登录相机，则表示此设备还未登陆相机，可以登录相机，否则显示退出相机，可以进行相机的退出。

(1) 添加设备：添加设备中，在主机列中可以选择要添加设备的主机，有两种方式，随机的选择添加的主机，手动的选择添加的主机。选择添加的主机，填写相机的信息，包括相机IP，相机账号，相机密码，相机的ID，此相机监控的楼层范围，即监控的最低楼层和监控的最高楼层。点击添加按钮，启动新的子系统，启动成功会在①中的提示信息中显示“客户端启动成功”，否则会提示“客户端启动失败”，同时相机登录是否成功，则会弹出提醒消息，同时会刷新整个设备管理界面。如下图为子系统启动成功，但相机登录失败。

(2) 登录相机：未登录相机的子系统正确的填写相机的登录信息，点击登录信息，登录相机是否成功会给出提示信息。

(3) 退出相机：已经登录的子系统会可以将相机登录退出。

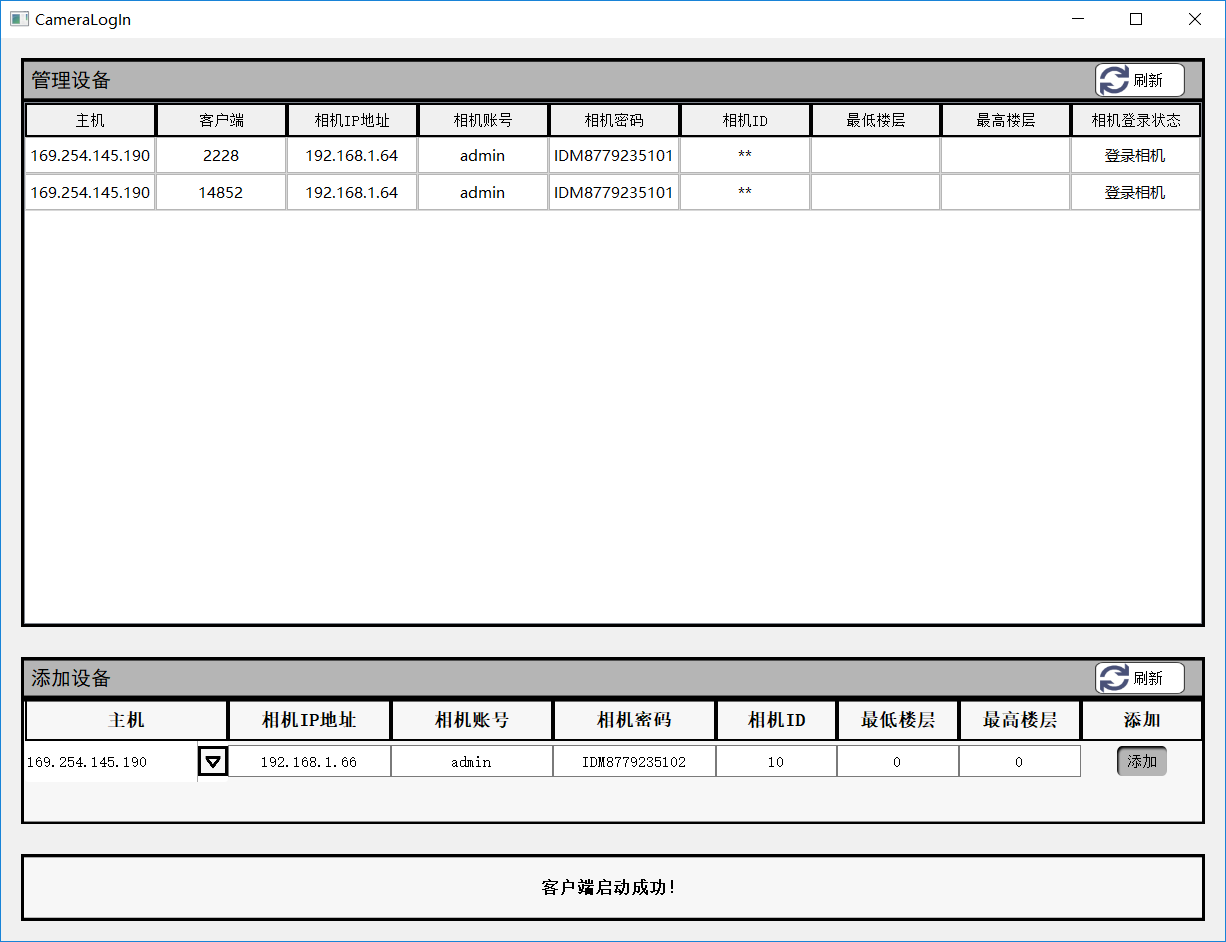


图13-4 设备管理界面

### 13.3.4 抛物视频回放界面

抛物视频回放界面主要显示抛物视频的列表，包括抛物视频的列表，列表中是检测到抛物的相机ID 检测到抛物的时间组合的名字列表，点击列表中的一个，会播放对应的检测到抛物的视频，播放完成后会进行提示“视频播放完毕”，如图13-5所示。

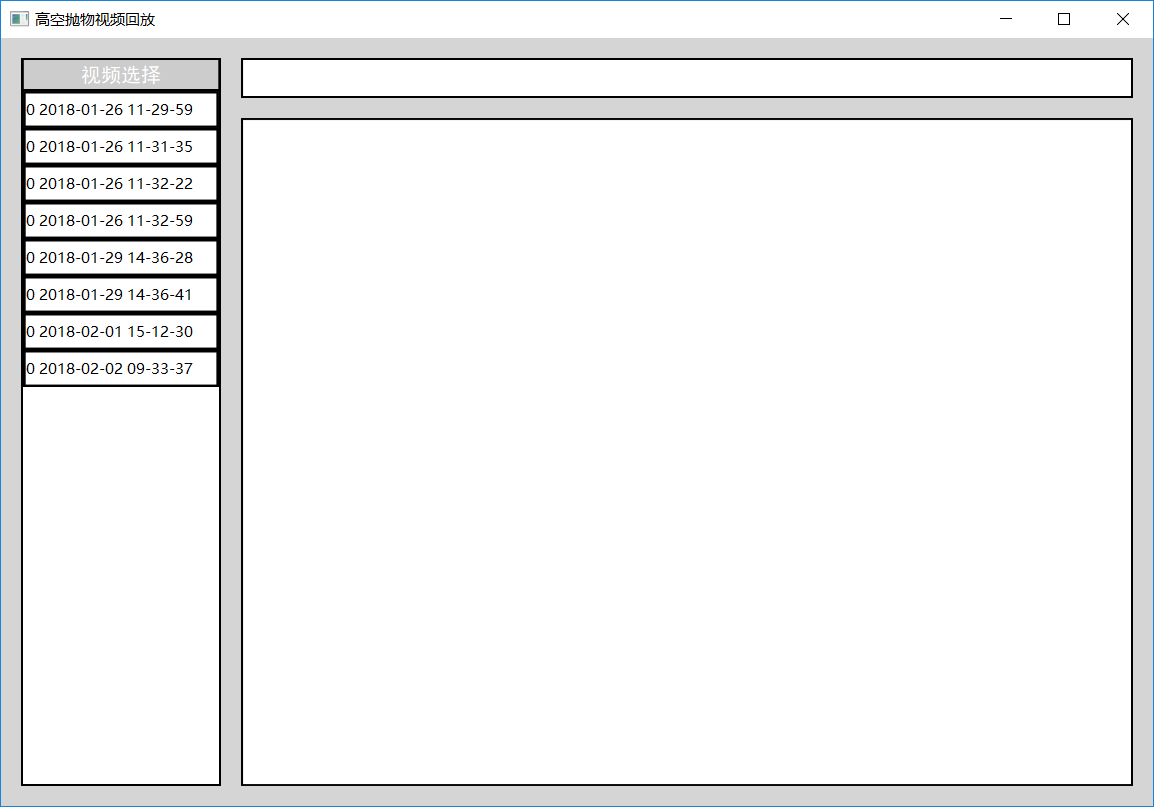


图13-5 抛物视频回放界面

### 13.3.5 拼接视频显示及检测结果显示

抛物检测结果显示

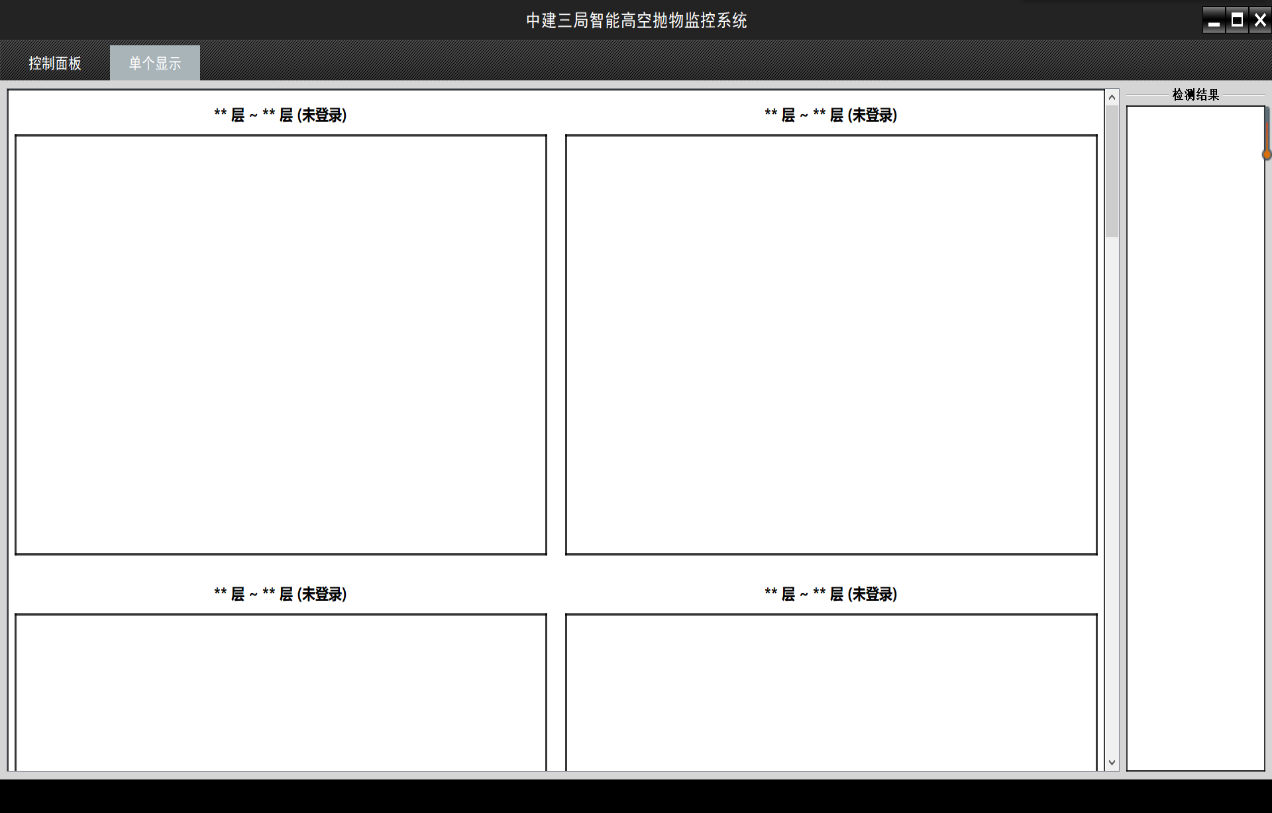
拼接视频显示



图13-6 拼接视频及抛物信息显示界面

### 13.3.6非拼接视频显示及检测结果显示

监控视频和检测结果显示界面，如下图所示。其中监控视频的显示可以显示多个相机的监控场景，每个相机的监控场景放到一个视频显示框中，同时给出监控场景中监控的楼层信息，即监控楼层时从几楼到几楼；检测结果的显示则是对检测出的抛物信息进行显示，其中包括抛物发生的时间，抛物抛出的楼层，发生抛物的监控的相机等信息。



抛物检测结果显示

非拼接视频显示

图4.7 非拼接视频显示及检测结果显示