基于PyQt5的分布式移动端目标检测监控平台

软件架构文档

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <26/11/2024> | <1.0> | 软件架构文档 | 第四小组 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 参考资料 4

2. 用例视图 4

3. 逻辑视图 5

4. 部署视图 7

5. 进程视图 7

5.1 主进程：Main process 7

5.2 帧子进程：Frame process 7

5.3 模型子进程：Model process 7

6. 实现视图 8

7. 技术视图 8

8. 数据视图 8

9. 算法视图 9

9.1 目标检测和跟踪 9

9.1.1 目标检测 9

9.1.2 目标跟踪 9

9.2 基于目标跟踪结果的场景分析 9

9.2.1 进入人数计数 9

9.2.2 保密区域感知 9

10. 性能视图 9

软件架构文档

# 简介

## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策，包含用例视图、逻辑视图、部署视图、进程视图、实现视图、算法视图、性能视图。

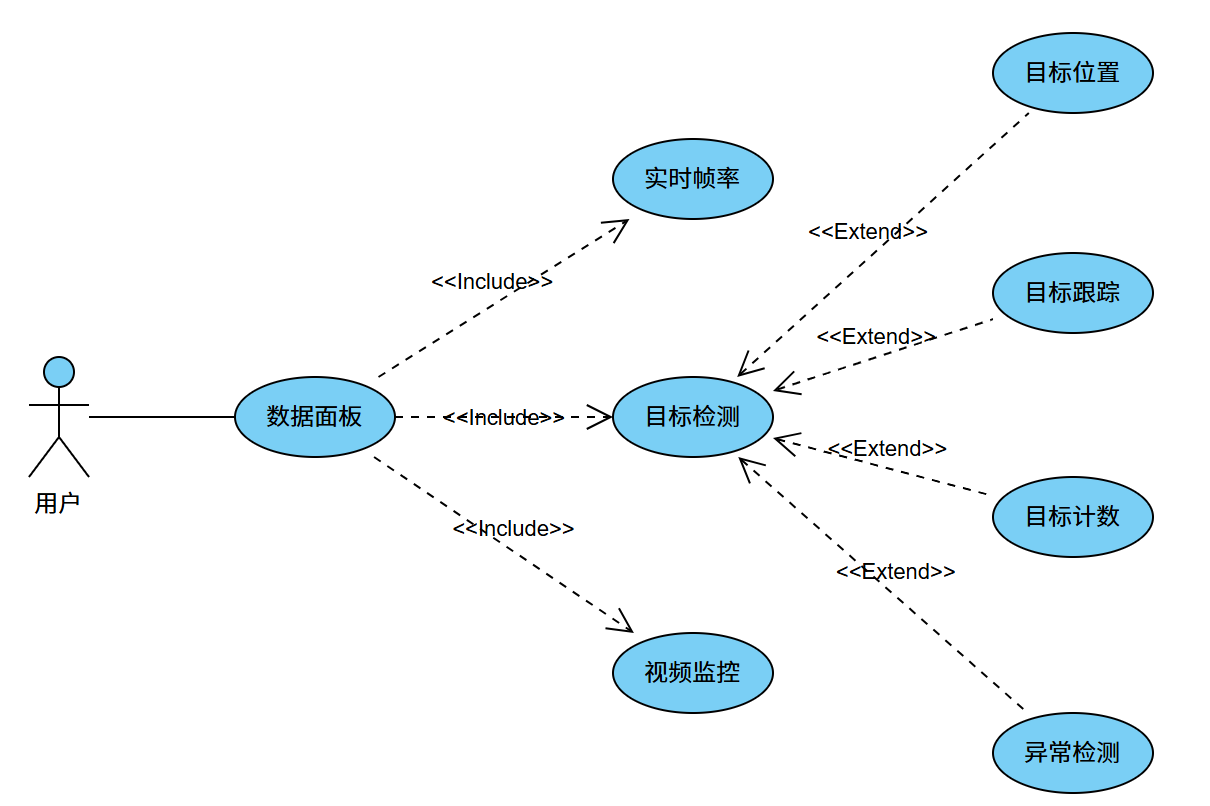
## 参考资料

1. YOLOv5模型：<https://github.com/ultralytics/yolov5>

2. OpenCV-Python文档：<https://docs.opencv.org/3.4/df/d9d/tutorial_py_colorspaces.html>

3. PyQt5文档：<https://doc.qt.io/qtforpython-6>

# 用例视图



数据面板：用户与系统交互的界面，展示了实时处理的分析结果

实时帧率：展示视频帧的处理速度，确保视频流的平滑性和及时性

视频监控：将处理过的实时视频流在用户界面上呈现

目标检测：对视频帧进行物体识别，实时检测监控画面中的目标物体

目标位置：通过图形化方式展示检测到的目标物体在视频中的位置

目标跟踪：在目标检测的基础上，系统支持对检测到的目标进行持续追踪

目标计数：持对某一时间段内进入或离开监控画面中的目标进行计数

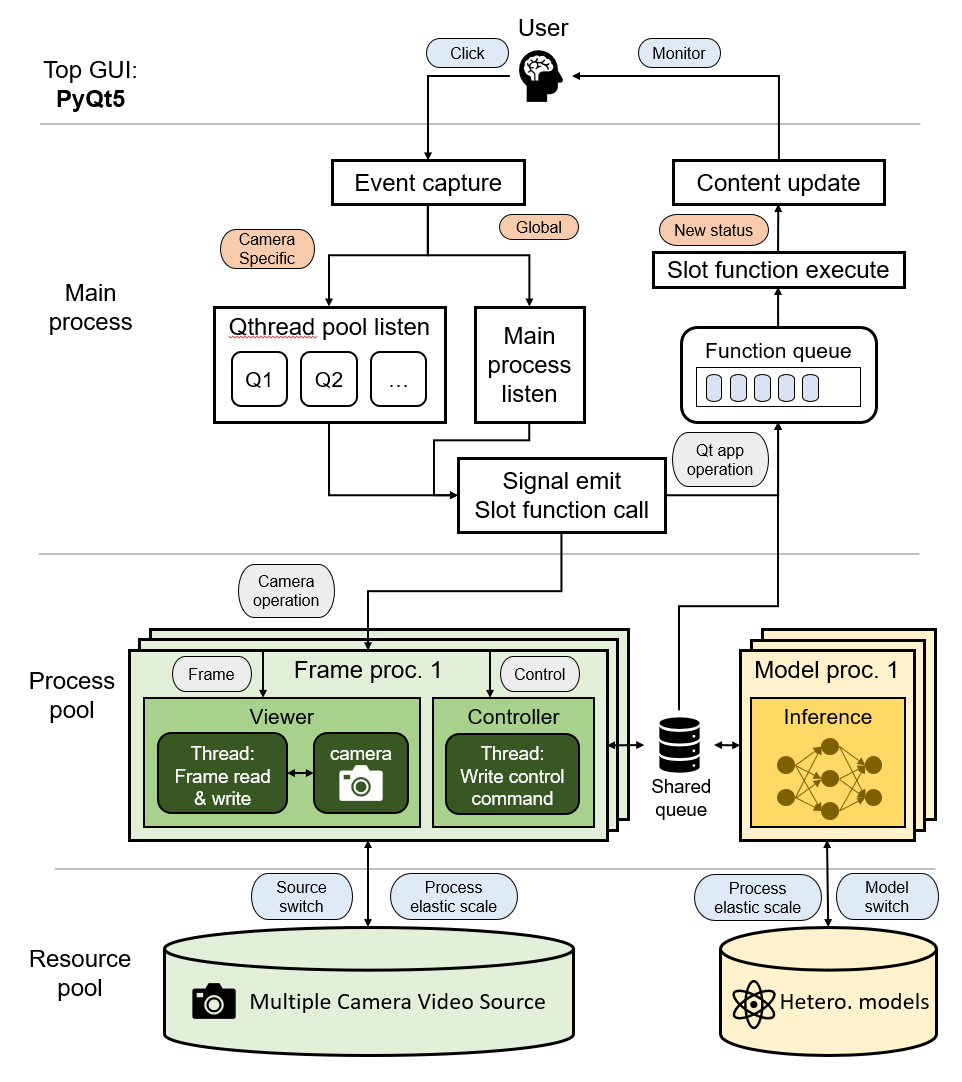
异常检测：通过分析正常环境下的目标活动模式，系统能够识别异常行为或事件

# 逻辑视图

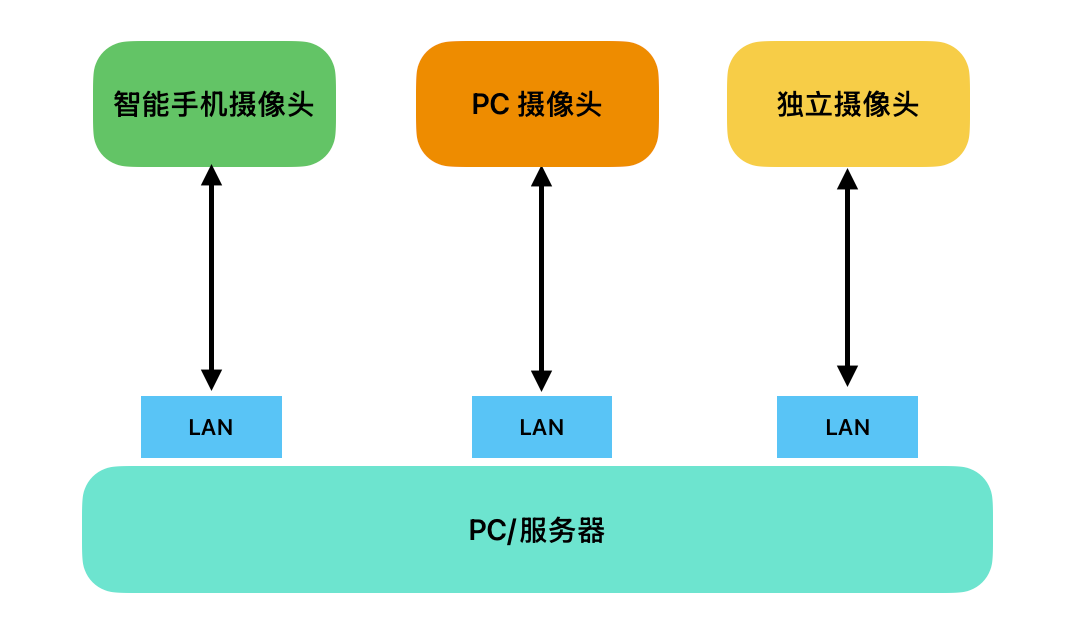
本软件与用户的交互界面基于PyQt5搭建，其工作模型是面向时间进行响应。如图所示，随着用户的点击等操作，软件监听并捕获响应事件，并对事件类型做划分，不同类型（全局、某个相机）的事件由不同的线程和进程监听。随后，监听线程或进程触发响应信号，并调用预先挂载的槽函数对事件进行响应。槽函数只能在主进程中执行，Qt内部有一个执行队列，依次运行被调用的槽函数，并更新界面状态，反馈给用户。

另一方面，对于相机特定的任务，软件使用多进程结构来处理，每个进程（图中的Frame进程）负责管理一个相机。进程中有viewer和controller两个模块，分别负责每帧的读写操作等和相机控制命令，读取到的帧会放进共享队列中，供主进程调用并更新可视化界面。当模型推理开启时，模型专用进程会从共享队列中读取待推理帧，推理结束后将结果放回共享队列中，整个过程采用异步设计，不会因为模型推理而延误实时视频显示。

不同相机资源通过资源池结构进行管理，Frame进程可以进行源切换，命令同样由用户发出，模型功能也可以进行类似切换。

**

# 部署视图



# 进程视图

## 主进程：Main process

主进程负责对界面与用户交互的事件进行响应，执行响应对应的槽函数调用；对于相机特定的事件，主进程内部维护与相机数目相同的QT线程示例，可以并发的对相机特定事件进行处理。如果要增加监控的相机数目，主进程会对帧子进程和模型子进程进行弹性扩张。

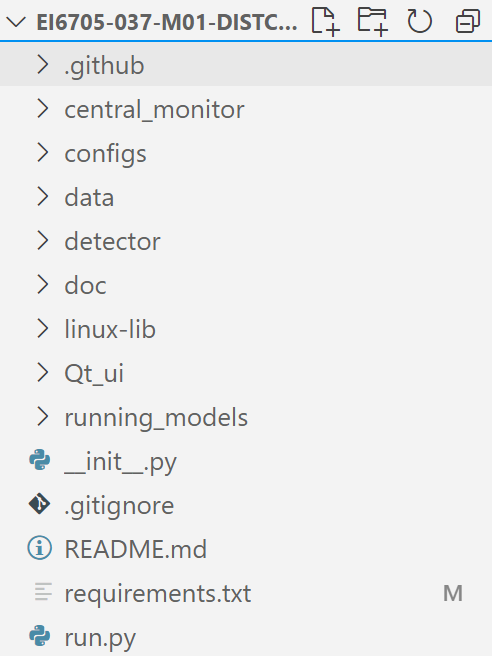
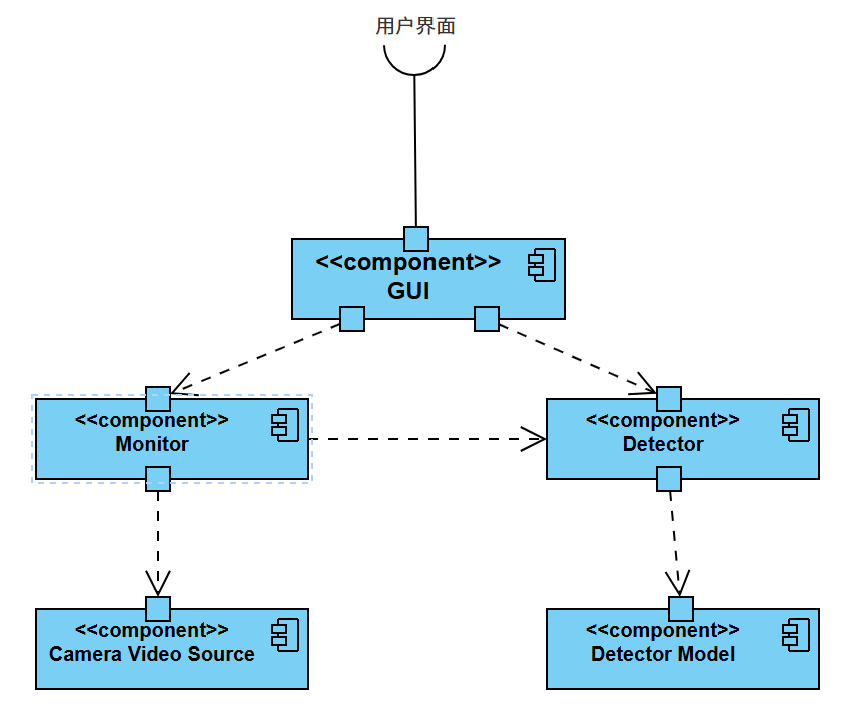
## 帧子进程：Frame process

帧子进程数目与相机数目相同，每个子进程与主进程中的一个QT线程相关联，二者通信采用Python内部的多进程队列。该子进程负责逐帧读取视频帧，并将视频帧放到共享队列中，以供主进程使用，也可以根据用户的命令，进行视频帧的截取和保存；同时，其也负责相机控制信息的维护和响应，例如画面缩放，画面移动等。每个子进程内部有两个线程，保证操作的并发性。

## 模型子进程：Model process

模型子进程负责模型推理。当推理功能开启时，其从共享队列中读取视频帧数据，执行推理后将结果返回至共享队列中。

# 实现视图



# 技术视图

编程语言：python

开发工具：vscode

框架：pyqt(用户界面),opencv(视频处理),YOLO(目标检测),torch(模型训练)

# 数据视图

本软件数据仅包括模型、本地视频和实时流，仅将推理结果以json文件形式存储，不需要数据库等模块进行管理，故不作数据视图。

# 算法视图

## 目标检测和跟踪

### 目标检测

* 使用 YOLOv11 模型对每一帧视频图像进行目标检测。
* 模型输入：经过预处理的单帧图像。
* 模型输出：每个检测到的目标的类别（Class）和边界框坐标（x, y, w, h）

### 目标跟踪

* 采用 BoT-SORT 算法，将 YOLOv11 检测到的目标进行跨帧跟踪。
* BoT-SORT 的输入包括目标边界框和类别信息。
* BoT-SORT 输出分配的目标 ID，确保同一目标在多帧中有一致的标识。

视频的跟踪处理结果以格式化输出，包含所有检测到的实例和轨迹信息。

## 基于目标跟踪结果的场景分析

根据实际场景，我们需要对目标跟踪结果进行高级分析。以办公室场景为例，我们需要对进入的人进行计数，同时，有一些保密区域需要限制人员进入。因此我们需要利用跟踪信息对上述进行感知。

### 进入人数计数

基于 BoT-SORT 跟踪的目标 ID，通过分析每个目标的路径轨迹，实现对进入人数的统计。

**计数逻辑**：

定义办公室入口的计数线（Virtual Counting Line），当目标的轨迹跨越计数线且进入方向符合预设规则时，计数器加一。

### 保密区域感知

在办公室场景中，定义保密区域的边界框（Restricted Zones）。当目标的坐标（x, y）进入保密区域范围内时，触发告警。

**感知逻辑**：

每帧检查目标的 bbox 是否与保密区域的坐标范围重叠。若重叠，记录告警事件。

# 性能视图

