<项目名称>

测试计划

版本 <1.0>

[注：以下提供的模板用于 Rational Unified Process。其中包括用方括号括起来并以蓝色斜体（样式=InfoBlue）显示的文本，它们用于向作者提供指导，在发布此文档之前应该将其删除。按此样式输入的段落将被自动设置为普通样式（样式=Body Text）。]

[要定制 Microsoft Word 中的自动字段（选中时显示灰色背景），请选择 File>Properties，然后将 Title、Subject 和 Company 等字段替换为此文档的相应信息。关闭该对话框后，通过选择 Edit>Select All（或 Ctrl-A）并按 F9，或只是在字段上单击并按 F9，可以在整个文档中更新自动字段。对于页眉和页脚，这一操作必须单独进行。按 Alt-F9，将在显示字段名称和字段内容之间切换。有关字段处理的详细信息，请参见 Word 帮助。]

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 10/01/2024 | 1.0 | 首次拟定 | 徐郅浩、郑心锐 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1. 简介 3](#_Toc187443817)

[1.1 目的 3](#_Toc187443818)

[1.2 背景 4](#_Toc187443819)

[1.3 范围 4](#_Toc187443820)

[1.4 项目标识 7](#_Toc187443821)

[2. 测试需求 8](#_Toc187443822)

[3. 测试策略 9](#_Toc187443823)

[3.1 测试类型 9](#_Toc187443824)

[3.1.1 数据和数据库完整性测试 9](#_Toc187443825)

[3.1.2 功能测试 10](#_Toc187443826)

[3.1.3 业务周期测试 12](#_Toc187443827)

[3.1.4 用户界面测试 13](#_Toc187443828)

[3.1.5 性能评价 15](#_Toc187443829)

[3.1.6 负载测试 16](#_Toc187443830)

[3.1.7 强度测试 17](#_Toc187443831)

[3.1.8 容量测试 18](#_Toc187443832)

[3.1.9 安全性和访问控制测试 19](#_Toc187443833)

[3.1.10 故障转移和恢复测试 20](#_Toc187443834)

[3.1.11 配置测试 21](#_Toc187443835)

[3.1.12 安装测试 22](#_Toc187443836)

[3.2 工具 23](#_Toc187443837)

[4. 资源 24](#_Toc187443838)

[4.1 角色 24](#_Toc187443839)

[4.2 系统 25](#_Toc187443840)

[5. 项目里程碑 26](#_Toc187443841)

[6. 可交付工件 27](#_Toc187443842)

[6.1 测试模型 27](#_Toc187443843)

[6.2 测试日志 27](#_Toc187443844)

[6.3 缺陷报告 27](#_Toc187443845)

[7. 附录 A：项目任务 28](#_Toc187443846)

测试计划

# 简介

## 目的

基于 PyQt5 的分布式边端协同智能监控平台的这一“测试计划”文档有助于实现以下目标：

* 确定现有项目的信息和应测试的软件构件。
* 列出推荐的测试需求（高层次）。
* 推荐可采用的测试策略，并对这些策略加以说明。
* 确定所需的资源，并对测试的工作量进行估计。
* 列出测试项目的可交付元素

## 背景

随着城市化进程的加速和人们对安全问题的日益关注，传统的防盗措施已无法满足现代家庭与企业的需求。本项目起初源于对家庭安全领域的调研，发现用户对易用性强、功能全面且价格合理的智能监控系统有着广泛需求。随后，通过整合云计算、物联网和移动应用技术，本项目逐步发展为一套针对家庭和小型企业的智能监控解决方案，旨在为用户提供高效、可靠的安全保障。本项目聚焦于面向家庭和小型企业的智能监控系统，旨在满足现代社会对安全性、便捷性和智能化的需求。该系统整合了多种先进技术，提供全天候的实时监控、警报通知、远程查看和证据保存等功能。其面向场景的主要特征如下：

1. **提升安全性**：通过实时监控家庭或企业环境，及时发现异常，预防入室盗窃以及其他潜在的安全隐患。
2. **远程监控**：用户可通过智能手机或电脑随时随地查看监控画面，方便安全管理。
3. **证据保存**：系统支持视频记录与云存储，关键时刻可提供重要的影像证据。
4. **智能化管理**：与智能家居系统联动，可实现自动化管理，提升用户的生活便捷性和安全性。

该智能监控系统采用模块化设计，主要由以下部分组成：

* **硬件设备**：各类端侧设备（PC、智能手机）的摄像头和独立摄像头。
* **云存储服务**：用于存储监控视频和日志数据，确保数据安全及可用性。
* **多平台应用端**：支持Windows、Linux、MacOS 多平台，提供丰富的设备管理功能。
* **后端服务器**：负责数据处理、用户管理、警报推送及与云存储的交互。
* **网络通信模块**：通过Wi-Fi或移动网络实现设备与服务器及客户端的通信。

## 范围

本文档所描述测试的目标为：

* 测试系统稳健性，确保各个软件构件能够长时间稳定运行；
* 测试系统性能，确保该监控平台在不同系统上能够高效运行；
* 测试系统兼容性，确保该监控平台在不同操作系统和硬件条件下均能正常执行；

具体应参与测试的特性与功能如下：

1. **实时监控**：
   * 摄像头与传感器的监控能力。
   * 视频画质、延迟和实时性。
2. **模型推理功能**：
   * 深度学习模型正常加载、接入和卸载。
   * 深度学习模型流畅执行。
3. **视频画面捕获与保存**：
   * 画面捕获的实时性和准确性。
   * 基本不影响监控平台的其他功能执行。
4. **系统性能**：
   * 系统在高负载下的性能表现（并发用户、多设备连接）。
   * 在不同摄像头硬件和操作系统上能够稳定高性能执行。

本测试文档不考虑以下特性与功能的测试：

1. **硬件设备的物理性能**：
   * 摄像头和传感器的硬件缺陷。
   * 摄像头与传感器的物理连接问题。
2. **非主流平台的兼容和测试：**
   * 除了 Linux、Windows 和 MacOS 外的平台不参与测试。
   * 非主流摄像头和相应 SDK 不参与测试。
3. **模型性能测试**：
   * 模型的精确度不参与测试。
   * 模型底层推理性能优化不在测试考虑范围内。

本测试文档基于以下假设设计测试过程：

1. 系统的所有硬件设备（如摄像头、传感器）均已通过供应商的质量验证。
2. 用户设备（如手机、电脑）满足最低系统要求。
3. 项目中提供的API接口均已正确实现并通过初步验证。
4. 测试环境（网络、服务器、设备）与实际运行环境接近，测试结果具有可参考性。

测试设计、开发和实施过程中可能出现的风险和意外事件有：

1. **硬件故障**：摄像头或传感器可能在测试期间出现故障，导致测试中断。
2. **网络不稳定**：测试环境中的网络波动可能影响测试结果的准确性。
3. **需求变更**：测试中需求的频繁变动可能导致测试计划和用例需要重新设计。
4. **测试工具不兼容**：部分自动化测试工具可能无法完全支持目标平台或功能。
5. **系统缺陷积累**：早期发现的重大缺陷可能导致后续测试阶段延迟。

本测试面临的约束如下：

1. **时间约束**：测试周期有限，可能需要优先测试关键功能。
2. **资源约束**：
   * 测试设备数量有限，可能无法覆盖所有型号的终端。
   * 测试人员数量有限，可能影响测试覆盖率。
3. **预算约束**：项目预算限制可能限制某些高成本测试（如全面的安全渗透测试）。
4. **环境约束**：测试环境可能无法完全模拟真实运行环境（如设备连接条件不同）。
5. **技术约束**：部分测试工具在特定系统或设备上可能存在局限性。

#### **1. 单元测试**

* **目标**：验证系统中最小单元（如函数、模块）的正确性，确保输入输出符合预期。
* **测试类型**：
  + 功能测试：检查单个模块的功能是否正确。
  + 边界测试：验证模块在边界输入或极端情况下的表现。
* **测试对象**：
  + 后端：视频流捕获逻辑和模型推理功能。
  + 前端：各功能页面的基本逻辑。
* **不测试的特性**：
  + 不涉及系统整体行为和模块间交互。

#### **2. 集成测试**

* **目标**：验证模块之间的交互是否正确，确保数据流和功能协作正常。
* **测试类型**：
  + 功能测试：测试模块间的交互功能。
  + 接口测试：验证模块之间的API调用是否正确。
* **测试对象**：
  + 前端与后端的数据交互。
  + 视频流捕获显示与模型推理结果的显示结合。
  + 移动应用与硬件设备（如摄像头、传感器）的集成。
* **不测试的特性**：
  + 不测试模块的内部逻辑，只测试模块之间的交互。

#### **3. 系统测试**

* **目标**：验证整个系统的功能、性能和可靠性，确保系统满足设计需求。
* **测试类型**：
  + 功能测试：验证所有功能是否按照需求文档工作。
  + 性能测试：测试系统的响应速度、稳定性和吞吐量。
  + 兼容性测试：验证系统在不同设备、操作系统上的表现。
  + 易用性测试：评估系统的用户体验。
* **测试对象**：
  + 整个系统的端到端功能。
  + 视频流捕获与显示、模型的加载和卸载以及推理结果的集成显示。
* **不测试的特性**：
  + 不测试与项目无关的硬件设备或第三方依赖的内部逻辑。

## 项目标识

下表列出了制定*测试计划*所用的文档，并标明了文档的可用性：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **文档 （版本/日期）** | **已创建或可用** | **已被接受或已经过复审** | **作者或来源** | **备注** |
| 立项申请书 | 是 | 是 | 第四组成员 |  |
| 迭代计划 | 是 | 是 | 第四组成员 |  |
| 迭代评估报告 | 是 | 是 | 第四组成员 |  |
| 软件需求规约文档 | 是 | 是 | 第四组成员 |  |
| 软件架构文档 | 是 | 是 | 第四组成员 |  |
| 源代码 | 是 | 否 | 第四组成员 | 源代码上传至github |
| 软件测试计划 | 是 | 否 | 第四组成员 |  |
| 安装包 | 是 | 否 | 第四组成员 |  |
| 项目总结报告 | 是 | 否 | 第四组成员 |  |

# 测试需求

下面列出了那些已被确定为测试对象的项目（功能性需求和非功能性需求）。此列表说明了测试的对象。

|  |  |  |
| --- | --- | --- |
| 序号 | 测试项类型 | 测试项内容 |
| 1 | 功能性需求 | 选定多个摄像头进行视频录制或抓图 |
| 2 | 功能性需求 | 选定单个摄像头进行画面缩放与移动控制 |
| 3 | 功能性需求 | 对特定窗口，切换监视视频源 |
| 4 | 功能性需求 | 从多种模型中选择一种进行推理 |
| 5 | 功能性需求 | 设置模型配置，包括推理张量大小、置信度阈值等 |
| 6 | 功能性需求 | 实时图像面板显示目标类型的检测数量 |
| 7 | 功能性需求 | 日志面板回显系统正常运行的信息与用户操作后反馈 |
| 8 | 功能性需求 | 实时监控画面上显示模型推理的渲染结果 |
| 9 | 用户界面测试 | 切换为全局视图时，缩放无卡顿，无bug |
| 10 | 用户界面测试 | 用户可以通过拖拽锚点自行选择报警区域 |
| 11 | 用户界面测试 | 用户可以根据显示器配置来调整主监控区画面大小 |
| 12 | 性能评价 | 观察统计系统运行实时帧率 |
| 13 | 性能评价 | 观察统计数据面板显示的模型推理时间 |
| 14 | 性能评价 | 观察统计数据面板显示的实时掉帧率 |
| 15 | 性能评价 | 观察判断目标检测模型的查全与查准情况 |
| 16 | 负载测试 | 运行前指定同时处理的实时流数目，观察软件承载力 |
| 17 | 负载测试 | 选用较高精度的实时流或视频，观察软件解码情况 |
| 17 | 强度测试 | 使用较低性能的笔记本电脑，测试能够支持的视频上限数目 |
| 18 | 强度测试 | 使用性能较高的台式机，测试能够支持的视频上限数目 |
| 19 | 容量测试 | 长时间运行系统，观察内存及磁盘占用情况 |
| 20 | 容量测试 | 设置较大的推理batch size等，观察显存容量情况 |
| 21 | 配置测试 | 开启其他视频播放程序，观察软件性能是否受影响 |
| 22 | 配置测试 | 在Linux和Windows操作系统上运行，观察是否有差异 |
| 23 | 安装测试 | 对于新用户，测试正确配置环境后能否运行程序主界面 |
| 24 | 安装测试 | 对于老用户，测试更新代码后是否会出现新bug |

# 测试策略

测试策略提供了推荐用于测试对象的方法。上一节“测试需求”中说明了将要测试哪些对象，而本节则要说明如何对测试对象进行测试。

下面列出了在进行每项测试时需考虑的事项，除此之外，测试还只应在安全的环境中使用已知的、受控的数据库来执行。

## 测试类型

### 数据和数据库完整性测试

本项目为轻量化平台，生成数据量较少，不需数据库进行管理，故不做测试

### 功能测试

|  |  |
| --- | --- |
| 测试目标： | 功能测试的主要目标是验证系统功能是否按预期工作，并确保系统能够正确处理输入、数据处理和输出。特别是，要确保以下功能能正常工作：   * 用户能选择多个摄像头进行视频录制或抓图。 * 用户能够控制单个摄像头的画面缩放与移动。 * 系统能在特定窗口中切换视频监视源。 * 能从多种目标检测模型中选择一个进行推理，并确保推理功能正常。 * 用户可以设置推理模型的配置（如推理张量大小、置信度阈值等）并生效。 * 实时图像面板能够准确显示目标类型的检测数量。 * 日志面板能回显系统的正常运行信息，并能根据用户操作提供反馈。 * 实时监控画面能正确渲染模型推理结果。 |
| 方法： |  **功能用例设计**：基于系统功能需求设计详细的功能测试用例，确保覆盖所有核心业务功能。例如，选择摄像头、进行推理模型配置等。   **交互式测试**：使用图形用户界面（GUI）与应用程序交互，模拟用户的操作行为（如选择摄像头、配置推理模型、切换视频源等），观察系统的响应并验证功能是否正常。   **输入输出验证**：测试每个功能点时，验证应用程序的输入输出是否符合预期，如正确显示目标检测数量、切换视频源等。   **边界测试**：在极端条件下进行测试，如选择极多的摄像头、设置异常的推理张量大小等，检查系统是否能正确处理。   **日志验证**：对日志面板的反馈进行验证，确保其能够回显系统运行信息，且能根据用户操作准确显示反馈内容。 |
| 完成标准： |  **功能覆盖率**：所有核心功能和用例都已执行，且每项功能能按预期完成，无功能缺失或不一致的情况。   **无功能性缺陷**：所有功能测试项通过验证，系统无功能性缺陷，所有输入和输出均符合预期要求。   **稳定性和一致性**：系统在进行多次测试后，表现稳定且一致，未出现不可预测的错误或崩溃。   **日志验证通过**：日志面板能准确记录并回显系统状态信息，并且根据用户的操作提供有效反馈。 |
| 需考虑的特殊事项： |  **用户权限**：确保功能测试中考虑不同用户角色的权限限制（如管理员与普通用户可能有不同的操作权限）。   **系统响应时间**：在进行视频处理、推理和目标检测等操作时，需测试系统的响应时间，确保在用户交互过程中系统能够及时反馈。   **跨设备兼容性**：如果平台支持不同设备接入（如手机、电脑等），需要确保这些功能在不同设备上也能正常使用。   **环境依赖性**：测试时需要注意系统是否依赖于特定的硬件环境（如特定的摄像头、GPU等），并考虑如何在不同硬件配置下执行测试。   **数据一致性**：在测试过程中，必须确保输入的每一个数据（如摄像头选择、推理参数等）与系统返回的结果一致，并且没有数据丢失或错误的现象。 |

### 业务周期测试

本项目从学期开开始至结题即为执行结束，整体研发时间端，不需作多周期管理，除了持续集成测试，全方面的评估仅需要一次。

### 用户界面测试

[通过用户界面 (UI) 测试来核实用户与软件的交互。UI 测试的目标在于确保用户界面向用户提供了适当的访问和浏览测试对象功能的操作。除此之外，UI 测试还要确保 UI 功能内部的对象符合预期要求，并遵循公司或行业的标准。]

|  |  |
| --- | --- |
| 测试目标： | 用户界面（UI）测试的主要目标是确保软件的界面符合用户需求，提供直观、无障碍的交互体验，并且操作功能顺畅、无误。具体目标包括：   * **全局视图切换**：确保用户切换为全局视图时，界面没有卡顿或崩溃，操作顺畅。 * **报警区域选择**：验证用户能够通过拖拽锚点自由选择报警区域，并确保选择区域功能准确无误。 * **画面调整**：验证用户能根据显示器的配置来调整主监控区的画面大小，确保操作灵活且不出现错位或显示异常。 |
| 方法： |  **用户操作模拟**：模拟用户在界面上的操作，确保切换全局视图时不会出现卡顿或错误。使用鼠标、键盘或触摸屏等输入方式验证UI响应。   **边界测试**：通过极限条件进行测试，例如拖拽报警区域的锚点到最大或最小尺寸，调整监控区到显示器的最大分辨率等，确保在各种条件下界面功能的稳定性。   **跨设备/浏览器测试**：测试平台在不同设备或不同显示器尺寸下的UI表现，确保能够根据显示器配置调整主监控区的画面大小。   **UI一致性检查**：对UI元素的布局、颜色、字体、按钮和图标等进行检查，确保它们符合公司或行业的标准，符合设计规范。   **功能验证**：特别针对用户能够通过拖拽锚点选择报警区域和调整画面大小等操作进行验证，确保操作流畅且能正确反映用户意图。 |
| 完成标准： |  **界面流畅性**：所有UI操作（如全局视图切换、报警区域选择、画面调整等）能够快速响应且无明显卡顿。   **操作准确性**：用户界面的各项操作能够按照预期正确完成，如用户拖拽锚点能正确选择报警区域、调整主监控区画面大小时不发生错位或显示异常。   **设计一致性**：界面元素（按钮、菜单、图标、字体等）符合设计标准和公司/行业规范，具有一致的风格和易于理解的布局。   **跨设备兼容性**：UI在不同设备或显示器配置下能自适应，并且用户界面功能在各种设备上都能正常使用。 |
| 需考虑的特殊事项： |  **不同屏幕分辨率**：测试过程中要确保界面能适应不同的屏幕分辨率，尤其在调整画面大小时，确保没有内容溢出或显示不全的情况。   **操作流畅性**：特别在移动设备或较低配置的电脑上测试UI响应速度，确保即使在设备性能较低的情况下，操作依然流畅。   **兼容性测试**：在不同操作系统（Windows、Mac、Linux）和不同浏览器（Chrome、Firefox、Edge）下测试UI，以确保没有平台特定的显示问题或交互障碍。   **无障碍设计**：对于特殊需求的用户（如视力障碍者），需要确保UI具有足够的可访问性，如高对比度、可调字体大小、键盘快捷键支持等。   **交互的直观性**：测试用户界面的交互是否符合用户习惯，确保界面直观易用，且功能描述清晰明确，避免误操作或困惑。 |

### 性能评价

|  |  |
| --- | --- |
| 测试目标： |  **实时帧率**：确保系统能够稳定处理实时视频流，保证视频画面的流畅显示。   **模型推理时间**：验证目标检测模型的推理时间是否符合预期，确保模型能在合理的时间内完成推理。   **掉帧率**：测试系统在高负载条件下的表现，确保实时视频处理不会出现频繁掉帧现象。   **查全率与查准率**：评估目标检测模型的准确性，包括目标检测的查全率（能否识别所有目标）和查准率（识别的目标是否准确），确保系统的检测效果符合业务需求。 |
| 方法： |  **实时帧率测试**：在不同的负载和摄像头接入数量下，监控系统的实时帧率，确保其满足系统设计要求（例如30帧/秒）。可以使用性能分析工具（如FFmpeg）监控视频流的帧率。   **推理时间测试**：记录并分析每次模型推理所需的时间。测试不同类型的视频流（如高清、低分辨率视频）时，确保推理时间不会过长，并且能够在规定的时间内完成。   **掉帧率监控**：通过系统的统计面板，实时观察和记录视频流的掉帧率。在高负载和多任务条件下，测试系统是否出现频繁的掉帧，并评估其对系统性能的影响。 |
| 完成标准： |  **实时帧率稳定**：在大多数场景下，系统的实时帧率能够稳定保持在预期范围内（如30帧/秒或更高），即便在高负载或多个摄像头接入的情况下，也不应出现明显的卡顿现象。   **推理时间符合预期**：目标检测模型的推理时间应在设定的阈值内（如小于200毫秒），并在多种场景下都能达到预期的实时处理要求。   **掉帧率控制在可接受范围**：掉帧率应保持在合理范围内（如少于5%），确保实时视频监控不会受到严重影响。   **查全率与查准率达到预期**：根据业务需求，查全率和查准率应符合设定的最低标准（如查全率大于90%，查准率大于85%），确保目标检测的准确性。 |
| 需考虑的特殊事项： |  **并发负载**：需要模拟不同摄像头接入、不同视频流数量、不同模型推理任务的并发执行情况，评估系统在高负载下的表现，确保性能不下降。   **网络带宽**：测试时需要考虑网络带宽的影响，尤其是在涉及远程摄像头或分布式系统时，网络延迟和带宽不足可能会影响帧率和掉帧率。   **实时反馈**：目标检测模型的推理时间和准确度应及时反馈给用户，特别是在实时监控系统中，确保用户能够根据检测结果做出快速响应。   **环境变化**：考虑不同环境因素（如光照、物体遮挡等）对目标检测效果的影响，确保模型在不同场景下依然能够保持较好的查全率和查准率。 |

### 负载测试

|  |  |
| --- | --- |
| 测试目标： | 负载测试的主要目标是验证系统在不同工作量条件下的性能表现，确保系统能够在超出最大预期工作量的情况下仍然保持稳定运行，并且能够有效处理更高的负载。具体目标包括：   * **承载能力评估**：确保系统在处理多个实时视频流时，不会出现性能瓶颈或崩溃。 * **解码性能测试**：评估系统在处理高精度视频流时的解码能力，确保即使面对较高分辨率或更复杂的编码格式，系统仍能顺畅处理。 |
| 方法： |  **负载模拟**：模拟不同数量和类型的实时视频流并观察系统表现。逐步增加同时处理的实时流数量，直到系统达到负载极限，以评估系统在高负载条件下的性能表现。可使用工具（如JMeter、LoadRunner）进行负载模拟。   **视频流解码测试**：选择不同分辨率和编码方式（如高精度的高清视频流、H.264、H.265等格式），观察系统的解码情况，包括解码速度和解码错误等，测试系统在高精度视频流下的处理能力。   **性能监控**：通过实时监控软件的CPU、内存、磁盘、网络等资源的使用情况，记录在不同工作量下系统资源消耗的变化，确保系统不会因资源不足而导致性能下降。 |
| 完成标准： |  **负载能力达标**：系统在超出最大预期工作量（如多摄像头接入、大规模视频流处理）时仍能正常运行，不卡顿、不崩溃，且能够承载更多实时流的处理。   **解码性能满足要求**：在高精度视频流或高分辨率视频流的处理过程中，系统能够顺利完成视频解码，保持稳定的处理速度，避免出现画面卡顿、延迟或解码错误。   **资源利用效率**：系统在负载测试过程中，资源利用率（CPU、内存、带宽等）应保持在合理范围内，避免因资源瓶颈导致性能下降。   **响应时间稳定**：即使在大规模负载下，系统响应时间仍然符合预定要求（如保持在毫秒级别），不会因处理大量流而导致响应延迟过长。 |
| 需考虑的特殊事项： |  **系统规模**：负载测试时需要考虑不同规模的系统配置（如单机与分布式架构）。在分布式环境下，需测试各个节点的负载均衡能力和数据传输效率。   **网络带宽**：视频流的带宽要求较高，负载测试时需要考虑网络带宽的影响。高并发视频流的传输可能会受到网络带宽和延迟的限制，因此需要确保网络设施能够支撑大流量数据传输。 |

### 强度测试

|  |  |
| --- | --- |
| 测试目标： | 强度测试的主要目标是通过模拟资源不足或资源争用的情形，找出系统在资源紧张条件下可能出现的错误或性能瓶颈。具体目标包括：   * **资源不足导致的错误**：评估系统在内存、磁盘空间或其他关键资源不足时的表现，找出系统是否能够在资源不足的情况下稳定运行，及时处理异常。 * **资源争用问题**：模拟多个资源（如数据库锁、网络带宽等）争用的场景，评估系统在共享资源受到限制时的表现，确保系统能够有效处理资源争用问题。 * **最大工作量评估**：测试系统能够处理的最大负载量，特别是在不同硬件配置下（如低性能笔记本与高性能台式机），确定系统的工作负载上限。 |
| 方法： |  **异常行为模拟**：通过人为限制系统资源（如手动占用大量内存或磁盘空间、限制网络带宽等），模拟极端的资源争用或资源不足条件，观察系统是否会崩溃、卡顿或发生其他异常行为。   **性能下降评估**：在高负载条件下，记录系统性能下降的情况，测试系统是否能及时报告资源瓶颈、自动释放资源或采取其他补救措施。   **最大工作量测试**：通过在不同硬件配置下增加视频流数目，测试系统的最大承载能力，帮助确定系统能够处理的最大工作量。 |
| 完成标准： |  **无崩溃和严重异常**：系统在资源不足或资源争用的条件下不应出现崩溃或重大异常行为。即使在资源紧张时，系统也应能稳定运行，尽可能提供容错机制。   **合理资源分配**：系统应能够合理分配有限的资源，避免出现资源争用导致的性能瓶颈，如数据库锁死、带宽抢占等问题。   **性能下降可控**：在高负载条件下，系统的性能应呈现平稳的下降趋势，且能及时报告资源瓶颈，避免不可预测的性能崩溃。   **最大工作量确定**：根据测试结果，能够明确系统在不同硬件配置下的最大工作量，如支持的最大视频流数量，并根据此进行进一步优化。 |
| 需考虑的特殊事项： |  **资源限制**：强度测试时，需要人为限制系统资源（如手动占用内存或带宽），模拟实际的资源不足情况，测试系统的反应和稳定性。   **网络带宽和延迟**：网络带宽可能是视频流和目标检测模型处理中的瓶颈，测试时需要考虑网络带宽的限制，并模拟低带宽环境下的表现。 |

### 容量测试

|  |  |
| --- | --- |
| 测试目标： | 容量测试的主要目标是评估系统在处理大量数据时的表现，确保系统能够稳定运行并有效处理超大工作量或数据集。具体目标包括：   * **内存和磁盘占用情况**：测试系统在长时间运行时，特别是在处理大量数据或持续工作负载的情况下，系统的内存和磁盘占用情况。确保系统能够在长时间运行过程中有效管理资源，避免内存泄漏或磁盘空间不足的情况。 * **显存容量**：测试系统在处理较大推理批次（batch size）时，观察显存的使用情况，确保显卡能够在较大负载下正常工作，不出现内存溢出或性能下降的问题。 |
| 方法： |  **长时间运行测试**：持续运行系统，尤其是在处理大量数据或高负载任务（如多摄像头视频流处理、目标检测推理）时，监控内存和磁盘的占用情况。逐步增加负载和任务持续时间，观察系统的资源占用变化，确保资源不会因长时间运行而耗尽。   **大规模数据处理**：模拟系统处理大量数据的场景（如长时间视频流记录、存储大量检测结果），检查系统是否能够在不断增加的数据量下仍然保持稳定运行，确保数据处理不会导致性能瓶颈或崩溃。 |
| 完成标准： |  **内存和磁盘占用稳定**：系统在长时间运行过程中，内存和磁盘占用应保持在合理范围内，避免内存泄漏、磁盘空间溢出或资源使用异常。   **显存使用有效**：显存容量应根据推理批次（batch size）适当调整，不会发生显存溢出或性能显著下降的现象。系统能够合理管理显存资源，处理大规模数据时不发生崩溃。 |
| 需考虑的特殊事项： |  **长时间运行的资源管理**：容量测试时需要特别关注系统在长时间运行后的资源管理，避免出现内存泄漏、缓存积累、磁盘空间不足等问题，确保系统能够高效管理资源。   **高负载情况下的性能衰减**：在容量测试中，随着工作负载的逐步增加，系统可能会出现性能衰减的现象。需要通过合理优化系统架构（如采用分布式计算、增加缓存等）来降低性能下降的幅度。 |

### 安全性和访问控制测试

本项目面向个体用户，仅在局域网内连接设备，不需考虑多用户和访问模式问题

### 故障转移和恢复测试

本项目核心运行设备仅包括一台笔记本电脑或台式机，仅面向局域网内小范围视频监控，未设置多台冗余备份

### 配置测试

|  |  |
| --- | --- |
| 测试目标： | 配置测试的主要目标是确保系统在不同的硬件和软件配置下均能正常运行，并且验证软件在多种运行环境中是否能够稳定工作。具体目标包括：   * **软件兼容性验证**：确保软件在不同的操作系统（如Windows与Linux）下能够稳定运行，并且无重大差异。 * **资源占用影响测试**：通过在系统中运行其他程序，检查它们是否会影响测试对象的性能，确保在多任务环境中系统能够保持预期的性能。 * **环境适应性验证**：验证系统在不同的硬件配置、操作系统、驱动程序和其他软件环境中是否能够实现一致的性能和稳定性。 |
| 方法： |  **多任务测试**：在系统运行时，启动其他视频播放程序或资源占用较高的应用程序，检查是否对软件的性能造成显著影响。例如，观察在播放高清视频时，目标检测系统是否出现卡顿或延迟。   **操作系统兼容性测试**：分别在Linux和Windows操作系统上运行测试对象，比较其在不同操作系统下的性能差异。需要验证操作系统环境对软件的运行是否有不良影响，并确保功能一致性。 |
| 完成标准： |  **无性能退化**：在多任务环境下运行时，系统应能够保持稳定的性能表现，不会因为运行其他程序或占用过多资源而出现明显的性能下降。   **操作系统兼容性**：系统应在不同操作系统（如Windows和Linux）下稳定运行，无功能缺失或不兼容的问题。操作系统间的性能差异应保持在可接受范围内。 |
| 需考虑的特殊事项： |  **不同操作系统的差异**：不同操作系统（如Windows和Linux）可能存在不同的资源管理方式、文件系统、驱动程序和系统调用等，因此在配置测试时要特别关注这些差异可能带来的影响。   **硬件和驱动程序兼容性**：硬件（如显卡、CPU、内存等）和驱动程序的不同版本可能对系统的性能和稳定性产生重要影响，测试时需要特别关注这些因素。   **多任务环境中的性能影响**：在多任务环境下，操作系统可能会根据当前负载情况调整任务的优先级和资源分配。需要检查在其他程序占用资源时，目标检测监控平台是否仍能正常运行，确保在资源争用情况下系统表现稳定。 |

### 安装测试

|  |  |
| --- | --- |
| 测试目标： | 安装测试的主要目标是确保软件能够在各种配置环境下顺利安装，并且在安装完成后可以立即正常运行。具体目标包括：   * **安装过程的正确性**：确保软件能够在所有支持的操作系统和硬件配置上顺利完成安装过程，涵盖首次安装、升级、完整安装和自定义安装等不同场景。 * **异常处理验证**：验证在异常情况下（如磁盘空间不足、缺少目录权限等），安装程序是否能够正确提示并处理这些异常，避免用户遇到无法解决的安装问题。 * **安装后的功能验证**：确保软件在安装完成后能够顺利启动，并在启动后能够正常运行，功能上没有缺失或错误。 |
| 方法： |  **首次安装测试**：测试新用户在首次安装时，是否能够顺利完成安装，并且在安装完成后能够成功启动程序主界面。需要验证所有必要的依赖是否已被正确安装，系统环境是否配置正确。   **升级安装测试**：对于老用户，测试更新版本的安装是否会出现新bug或兼容性问题。包括检查更新过程是否平滑、旧版本数据是否成功迁移，更新后的系统是否正常运行，并测试旧版本中的功能是否依然有效。 |
| 完成标准： |  **成功安装**：软件能够在所有支持的操作系统和硬件环境下成功安装，并且无安装失败、卡顿或错误提示。   **升级无问题**：对于老用户，更新安装后不应出现新的功能故障或崩溃，更新后的软件应保留原有功能，并解决已知的 bug。   **异常安装处理**：当遇到常见安装异常（如磁盘空间不足、权限不足等）时，安装程序能够正确识别并提示用户，避免发生无法恢复的错误。   **安装后软件正常运行**：安装后，软件应能够成功启动并加载主界面，且所有基础功能均能正常运行，无任何初期启动问题。 |
| 需考虑的特殊事项： |  **依赖项检查**：软件可能依赖其他程序或库（如某些特定版本的.NET框架、CUDA库等）。测试时需要确保这些依赖项能够在安装过程中正确处理并安装。   **更新过程兼容性**：在测试更新过程时，要特别关注更新前后版本的兼容性，确保旧版本的数据和设置能被成功迁移，并且更新后的版本没有引入新的问题。   **网络连接需求**：如果软件的安装过程需要从网络下载某些依赖包或数据，测试应确保在不同网络条件下（如高速网络或有限带宽下）安装流程能够顺利进行。 |

## 工具

此项目将使用以下工具：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 工具 | 厂商/自行研制 | 版本 |
| 项目管理 | github | github | 网页版 |
| 自动测试 | SonarQube | SonarQube | 8.9 LTS |

# 资源

本节列出推荐 <基于 PyQt5 的分布式边端协同智能监控平台> 项目使用的资源，及其主要职责、知识或技能。

## 角色

下表列出了在此项目的人员配备方面所作的各种假定。

|  |  |  |
| --- | --- | --- |
| 人力资源 | | |
| 角色 | 推荐的最少资源  （所分配的专职角色数量） | 具体职责或注释 |
| 测试经理：徐郅浩 | 无 | 进行管理监督。  职责：   * 提供技术指导 * 获取适当的资源 * 提供管理报告 |
| 测试设计员：米博宇 | 无 | 确定测试用例、确定测试用例的优先级并实施测试用例。  职责：   * 生成测试计划 * 生成测试模型 * 评估测试工作的有效性 |
| 测试员：郑心锐 | 无 | 执行测试。  职责：   * 执行测试 * 记录结果 * 从错误中恢复 * 记录变更请求 |
| 测试系统管理员：李盼奇 | 无 | 确保测试环境和资产得到管理和维护。  职责：   * 管理测试系统 * 授予和管理角色对测试系统的访问权 |

## 系统

下表列出了测试项目所需的系统资源。

|  |  |
| --- | --- |
| 系统资源 | |
| 资源 | 名称/类型 |
| Linux环境服务器 | 台式机，如联想ThinkStation p3 |
| —网络或子网 | 接入路由器子网 |
| 移动设备（手机） | 市面常见型号即可，如Xiaomi 14 Pro |
| —包括特殊的配置需求 | 安装能开启rtsp服务的软件，如IP摄像头 |
| Windows环境服务器 | 笔记本电脑，如联想Y7000P |
| —网络或子网 | 接入路由器子网 |
| 路由器 | 市面常见型号即可，如Xiaomi A3000 |

# 项目里程碑

对 <基于 PyQt5 的分布式边端协同智能监控平台> 的测试应包括上面各节所述的各项测试的测试活动

|  |  |  |  |
| --- | --- | --- | --- |
| **里程碑任务** | **工作量** | **开始日期** | **结束日期** |
| 制定测试计划 | 10% | 2025.01.01 | 2025.01.05 |
| 设计测试 | 20% | 2025.01.06 | 2025.01.10 |
| 实施测试 | 20% | 2025.01.11 | 2025.01.15 |
| 执行测试 | 20% | 2025.01.16 | 2025.01.20 |
| 评估测试 | 10% | 2025.01.21 | 2025.01.23 |

# 可交付工件

本节列出了将要创建的各种文档、工具和报告，及其创建人员、交付对象和交付时间。

## 测试模型

为了确保项目的质量和可靠性，我们将采用多层次的测试模型，涵盖功能测试、性能测试、集成测试和用户验收测试等方面。每个测试阶段都会生成详细的测试报告，并通过ASQ（Automated Software Quality）工具来创建和引用这些报告。ASQ工具将作为主要的自动化测试和报告生成工具，确保测试过程的高效和准确。

## 测试日志

测试报告将包括以下几部分内容：

* 测试概述：描述测试的目标、范围和实施的背景信息。包括测试对象（目标检测、跟踪、计数、异常检测等功能模块）以及测试执行的时间和资源。
* 测试用例和测试场景：列出所有测试用例，包括功能测试、边界测试、异常处理测试等。每个测试用例将定义输入数据、预期输出和实际输出，并通过ASQ工具记录和管理。
* 测试结果：展示每个测试用例的执行结果，包括成功和失败的案例、错误日志、异常记录等。失败的测试将附带详细的错误信息，并标明问题的严重性。
* 性能评估：包括系统在多摄像头接入、多任务处理下的性能评估报告，测试平台对大规模数据流的处理能力、帧率、响应时间等关键指标。

## 缺陷报告

缺陷报告主要用于记录在测试过程中发现的系统问题和性能瓶颈，便于开发团队跟踪和修复。例如以下内容：

* **性能瓶颈**：在高负载或多摄像头接入的情况下，系统可能出现响应迟缓、帧率降低、处理延时过长等问题，影响用户体验。例如，平台在处理多个实时视频流时，可能会遇到推理进程无法及时完成或帧率不足，导致视频画面卡顿或目标检测滞后。
* **实时流或推理进程的限制**：系统在实时目标检测和推理过程中，可能会因计算资源的限制而无法同时支持大量的视频流或推理进程，导致部分摄像头画面无法实时显示，或者目标检测精度降低。

# 附录 A：项目任务

以下是一些与测试有关的任务：

• 制定测试计划

* + 确定测试需求
  + 评估风险
  + 制定测试策略
  + 确定测试资源
  + 创建时间表
  + 生成测试计划

• 设计测试

- 准备工作量分析文档

- 确定并说明测试用例

- 确定并结构化测试过程

- 复审和评估测试覆盖

• 实施测试

* + 记录或通过编程创建测试脚本
  + 确定设计与实施模型中的测试专用功能
  + 建立外部数据集

• 执行测试

- 执行测试过程

- 评估测试的执行情况

- 恢复暂停的测试

- 核实结果

- 调查意外结果

- 记录缺陷

• 评估测试

- 评估测试用例覆盖

- 评估代码覆盖

- 分析缺陷

- 确定是否达到了测试完成标准与成功标准