**项目总结报告**

日期：2025年1月13日

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 组号 | 4 | 项目名称 | 基于PyQt5的分布式移动端目标检测监控平台 | |
| 编程语言 | Python | 开发平台和框架 | PyQt5 | |
| **项目工作小结** | | | | |
| 1.是否实现了项目立项时的所有需求？列出实现的新增需求和未实现的需求。  项目立项时的所有需求均得到了实现并通过了单元测试和系统测试。  在项目立项时计划的需求之外，开发和测试过程中我们根据实际应用场景对计划的需求进行了补充和丰富，具体如下：   1. 支持根据部署设备性能的不同，自定义目标跟踪模型和推理batch size。 2. 支持自定义跟踪目标类别范围和跟踪相关参数（如置信度阈值）等，以适应不同的监控场景。 3. 为了便于操作，系统支持直接在画面上对跟踪结果（包括目标边框、置信度、轨迹）进行渲染。 4. 支持采用可交互的图形化方式自定义报警区域，方便操作。   2.采用哪种架构风格？哪些设计模式？  架构风格方面，我们使用了分布式架构，将视频采集和处理分布在不同类型的设备上，并支持设备间的实时通信。  设计模式方面，使用观察者模式实现了实时目标检测的事件触发机制。  3.技术方案有哪些亮点？  人机交互层面，使用基于 PyQt5 的高效图形界面实现了极简的操作流程，增强了用户体验。  视频分析层面，支持自定义的开放词汇检测类别，应用场景更加灵活；使用累积形式对每一帧图像检测内容进行内存管理，实现了灵活时间范围的检测结果提取和分析。  4.是否做了单元测试？是否做了系统功能测试？是否做了性能测试？是否做了兼容性等其他非功能测试？  单元测试方面，完成了核心功能模块（如目标检测、数据通信）的单元测试。在系统界面开发过程中，对每个可交互元素对应的槽函数均进行了单独测试。  系统测试方面，我们针对整个系统进行了全面的功能测试，所有关键功能均通过测试。 | | | | |
| 性能测试方面，一方面，我们在单机上使用网络视频对目标检测和追踪系统进行了在线测试和验证。由于选用了高效的最新yolo系列模型，我们的系统可以实时进行目标跟踪和结果显示，无明显延迟、卡顿现象。此外，得益于我们的检测结果管理方式，平台的内存占用较低，扩大了其部署设备的范围。另一方面，我们模拟实际应用场景测试了系统在高并发场景下的性能。我们将平台部署在个人笔记本电脑上，并连接多台网络相机。考虑到实际应用场景下对视频帧率无严格要求，在根据部署设备计算性能调节检测帧率后，系统的检测、跟踪、预警功能均正常运行，无明显的延迟、卡顿现象。兼容性测试方面，完成了在 Windows 和主流Unix操作系统上的兼容性测试。由于本平台开发使用的Python包具有多操作系统兼容性，本平台可以在不同的主流操作系统上顺利运行。综上所述，本平台可以在大部分个人笔记本电脑上一键部署，并可以通过多个网络摄像头实现多种场景的实时监控。  5.是否采用大模型来辅助开发？对开发生产率约有百分之几的提升？  是。在PyQt软件包，Yolo模型推理，等接口的使用上，官方文档在检索上存在一定难度且未能全面覆盖各种使用场景，而通过互联网检索与本项目类似真实场景的代码需要消耗大量时间。因此在此部分我们使用GPT-4o模型进行对应接口的介绍和调用样例的生成。  在具体函数的编写中，我们使用了VsCode的Copilot插件。Copilot可以阅读类和函数的定义并提供建议。在实践中，我们发现Copilot能够较好地生成一些逻辑简单的代码片段或常见函数的调用代码，如使用opencv在视频上对边框和文字进行标注。此外，Copilot还在函数命名、注释编写、代码风格规范等功能之外的方面提供了大量有用的建议，减少了这些部分的重复劳动。  综上所述，我们认为对大模型的使用使得本项目的开发时间缩短了30%左右。 | | | | |
| **项目组成员对项目的贡献度（%）** | | | | |
| 徐郅浩：50%，米博宇：20%，郑心锐：15%，李盼奇：15% | | | | |
| **软件度量** | | | | |
| 软件代码行数（不包括注解行、空行和复用代码）： | | | | 5500 |
| 复用他人代码行数： | | | | 0 |

|  |
| --- |
| **经验、教训和建议** |
| 在这次项目中，我们深刻体会到团队协作的力量。通过定期站会和高效的沟通工具，团队成员能够快速共享信息、解决问题并保持项目进度一致。此外，采用敏捷开发方法让我们能够逐步迭代项目，及时响应需求变更并持续优化功能，显著提高了项目的整体质量。同时，借助专业开发工具如Github和copilot，我们在代码管理和实现上实现了显著的提升，这些工具的有效使用为项目顺利完成提供了重要保障。  尽管项目顺利完成，但开发过程也暴露了我们的一些不足之处。首先，在需求分析阶段，由于未能充分挖掘用户需求，后期频繁的调整对开发进度造成了较大影响。其次，时间管理上存在一些问题，部分任务的时间预估过于乐观，导致后期时间压力较大。此外，测试覆盖率不够全面，对某些边界条件和极端场景的考虑不足，结果在后期阶段发现了部分功能漏洞，不得不紧急修复，增加了额外工作量。  针对以上经验和教训，我们提出几点改进建议：首先，在项目初期要投入更多时间进行深入的需求调研，与用户充分沟通，确保需求明确、稳定。其次，制定更加科学的时间规划方法，例如任务分解与历史数据参考，提升时间估算的准确性。最后，要进一步加强测试环节，不仅要覆盖核心功能，还要充分考虑边界条件和潜在风险场景，并设计全面的测试用例。这些改进措施将有助于提升团队在未来项目中的开发效率和质量。 |

项目组各成员签字：