目录

[1.引言 2](#_Toc10760)

[1.1 背景与现状 2](#_Toc1885)

[1.2 研究意义 2](#_Toc25843)

[2.项目综述 2](#_Toc9248)

[3.项目结果 3](#_Toc29890)

[3.1 功能完成度 3](#_Toc22157)

[3.2 性能指标 4](#_Toc31621)

[3.3 文档与测试 4](#_Toc7491)

[4.项目方法 4](#_Toc30468)

[4.1 技术架构 4](#_Toc6660)

[4.2 核心算法流程 5](#_Toc6314)

[4.3 开发与部署流程 5](#_Toc15478)

[5.讨论 5](#_Toc11719)

[5.1 项目优势 5](#_Toc20453)

[5.2 存在不足 6](#_Toc8906)

[5.3 改进方向 6](#_Toc27659)

[6.结论 6](#_Toc16742)

[7.参考文献 7](#_Toc2350)

[8. 附录 7](#_Toc2236)

[8.1 项目结构 7](#_Toc14865)

[8.2 核心API端点列表 8](#_Toc14338)

# 1.引言

## 1.1 背景与现状

随着健康监测技术的普及，心率作为基础生命体征之一，其实时监测在健康管理、运动指导等场景中具有重要价值。传统心率检测依赖专业设备（如心率胸带、医疗监护仪），存在操作复杂、成本较高等问题。近年来，基于计算机视觉的无创心率检测技术逐渐发展，通过普通摄像头捕捉面部皮肤血液流动引起的光反射变化，实现心率估算，降低了设备门槛。

现有开源项目webcam-pulse-detector已实现桌面端心率检测功能，但缺乏便捷的Web交互界面和跨设备访问能力。针对这一现状，本项目在原有算法基础上，对其进行了优化，并构建了完整的Web应用系统，实现了浏览器端实时心率检测与数据可视化服务。

## 1.2 研究意义

本项目的研究意义主要体现在：

（1）技术可行性验证：探索基于普通摄像头和Web技术实现无创心率检测的可行性，为低成本健康监测方案提供参考。

（2）用户体验优化：通过Web界面降低操作门槛，使普通用户可通过浏览器快速使用心率检测功能。

（3）功能扩展性：提供标准化API和模块化架构，为后续集成用户管理、历史数据分析等功能奠定基础。

（4）部署便捷性：支持本地开发和Docker容器化两种部署方式，满足不同场景的使用需求。

# 2.项目综述

本项目（Heart Rate Online）是基于webcam-pulse-detector的Web版本扩展，通过前端React框架和后端FastAPI构建，实现了实时心率检测（50-180 BPM）、数据可视化及多设备适配功能。

项目核心特性包括：

（1）实时心率检测：通过普通网络摄像头捕捉面部信号，基于FFT频谱分析计算心率。

（2）现代化Web界面：采用React+TypeScript开发，支持响应式布局和暗色主题。

（3）数据可视化：集成ECharts实现时域信号图和FFT频谱图实时展示。

（4）多摄像头支持：可切换不同摄像头设备，适应多样化硬件环境。

（5）灵活部署：提供Windows本地开发和Docker容器化两种部署方式，满足不同用户需求。

项目技术栈：

（1）后端：Python 3.8+、FastAPI、WebSocket（实时通信）。

（2）前端：React、TypeScript、Ant Design、ECharts。

（3）算法核心：Haar Cascade人脸检测、绿色通道信号提取、FFT频谱分析。

（4）部署工具：Docker、Docker Compose、Nginx。

# 3.项目结果

## 3.1 功能完成度

项目总体完成度达95%，核心功能实现100%覆盖，具体包括：

（1）后端服务：10+ RESTful API端点（如摄像头列表获取、检测启停控制、数据导出）和WebSocket实时流服务。

（2）前端应用：视频流显示、心率数值展示、信号质量指示、数据图表可视化、控制面板交互等组件。

（3）算法模块：人脸检测（Haar Cascade模型）、额头区域隔离、绿色通道提取、信号采集（250样本）、FFT频谱分析、心率计算（50-180 BPM范围）。

（4）部署配置：完整的Docker Compose编排、Windows启动/停止脚本、环境变量配置模板。

## 3.2 性能指标

视频延迟：<100ms，满足实时交互需求。

帧率：稳定支持30fps，保证信号采集精度。

心率更新：实时计算并展示，响应延迟<500ms。

首屏加载：前端页面加载时间<2s，提升用户体验。

兼容性：支持主流浏览器（Chrome、Firefox、Edge）和Windows 10/11操作系统。

## 3.3 文档与测试

文档完整性：100%覆盖，包括README.md（使用说明）、API.md（接口文档）、DEPLOYMENT.md（部署指南）等。

测试覆盖：核心算法单元测试完成，API集成测试和前端组件测试待补充（当前覆盖率60%）。

# 4.项目方法

## 4.1 技术架构

项目采用前后端分离架构，具体设计如下：

### 4.1.1 后端架构

核心层：复用lib目录下的原有算法（device.py设备处理、processors.py信号处理、interface.py工具函数）。

应用层：基于FastAPI构建，包括api（路由）、core（业务逻辑）、models（数据模型）三个模块。

通信层：通过HTTP协议提供RESTful API，通过WebSocket实现实时视频流和心率数据传输。

### 4.1.2 前端架构

组件层：封装VideoStream（视频流显示）、BPMDisplay（心率展示）、DataChart（数据可视化）、ControlPanel（控制面板）等组件。

服务层：通过axios处理HTTP请求，通过WebSocket客户端实现实时通信。

状态层：使用Zustand管理应用状态（连接状态、心率数据、视频帧信息等）。

## 4.2 核心算法流程

人脸检测：采用Haar Cascade分类器识别面部区域。

区域隔离：提取额头区域作为信号采集目标（减少运动干扰）。

信号提取：分离视频帧的绿色通道数据（对血液流动最敏感）。

数据处理：采集250个样本（约8秒数据），通过FFT转换至频域。

心率计算：在50-180 BPM范围内识别频谱峰值，确定心率值。

## 4.3 开发与部署流程

### 4.3.1 本地开发流程

后端启动：创建Python虚拟环境→安装依赖（requirements.txt）→通过uvicorn启动服务（端口8000）。

前端启动：安装Node.js依赖（npm install）→启动开发服务器（端口3000）。

交互测试：前端通过WebSocket连接后端，实时获取视频流和心率数据。

### 4.3.2 Docker部署流程

环境准备：安装Docker Desktop和WSL 2。

服务启动：执行start.bat脚本，通过Docker Compose启动后端、前端服务。

访问应用：通过浏览器访问localhost:3000使用前端界面。

# 5.讨论

## 5.1 项目优势

架构设计：采用分层架构和模块化设计，代码质量高（类型安全、注释完整），易于维护和扩展。

功能完整性：覆盖心率检测全流程，包括实时监测、数据可视化、设备管理、数据导出等。

部署便捷性：提供两种部署方式，本地开发适合功能验证，Docker部署适合快速演示。

文档完善：用户文档和开发文档齐全，降低使用和二次开发门槛。

## 5.2 存在不足

测试覆盖：API集成测试、前端单元测试和E2E测试尚未完善，可能影响系统稳定性验证。

安全性：未实现用户认证和权限管理，不适合多用户共享场景。

环境限制：主要支持Windows系统，对Linux/macOS的适配需进一步验证。

算法鲁棒性：受光线条件和用户运动影响较大，复杂环境下检测精度可能下降。

## 5.3 改进方向

短期（1-2周）：补充API集成测试、前端单元测试和E2E测试，提升测试覆盖率。

中期（1个月）：添加用户认证系统、集成数据库、支持多用户管理和云存储。

长期（3个月）：优化算法鲁棒性（如引入AI增强）、实现移动端适配、开发HRV（心率变异性）分析功能。

# 6.结论

本项目成功实现了基于Web的实时心率检测系统，通过整合计算机视觉算法与现代Web技术，提供了便捷、低成本的心率监测方案。系统核心功能完整，性能指标达标（延迟<100ms，帧率30fps），文档完善，部署流程简单，已达到生产就绪状态。

项目验证了Web技术在无创心率检测领域的应用可行性，为后续健康监测类应用开发提供了参考架构。尽管存在测试覆盖不足、安全性待加强等问题，但整体功能已满足基本使用需求，可通过后续迭代持续优化。

# 7.参考文献

1.webcam-pulse-detector开源项目： https://github.com/thearn/webcam-pulse-detector

2.FastAPI官方文档：https://fastapi.tiangolo.com/

3.React官方文档：https://react.dev/

4.ECharts官方文档：https://echarts.apache.org/

5.Docker容器化技术指南：https://docs.docker.com/

# 8. 附录

## 8.1 项目结构

Heart-Rate-Online

|── backend # 后端服务

| |── app # FastAPI应用

| | |── api # API路由

| | |── core # 业务逻辑

| | |── models # 数据模型

| | |── main.py # 入口文件

| |── lib # 核心算法

| |── requirements.txt # 依赖清单

|── frontend # 前端应用

| |── src

| | |── components # React组件

| | |── services # API服务

| | |── stores # 状态管理

| |── package.json # 前端依赖

|── docs/ # 文档

| |── API.md # API文档

| |── DEPLOYMENT.md # 部署指南

|── docker-compose.yml # Docker配置

|── start.bat/stop.bat # 启动/停止脚本

|── README.md # 主文档

## 8.2 核心API端点列表

|  |  |  |
| --- | --- | --- |
| 端点 | 方法 | 功能描述 |
| /api/v1/status | GET | 获取系统状态 |
| /api/v1/cameras | GET | 获取摄像头列表 |
| /api/v1/pulse/start | POST | 启动心率检测 |
| /api/v1/pulse/stop | POST | 停止心率检测 |
| /api/v1/data/export | GET | 导出CSV格式数据 |
| /ws/pulse | WebSocket | 实时视频流与心率数据传输 |