

网球检测系统实验报告

1 项目概述

本项目旨在开发一个高精度的网球检测系统，主要用于机器人捡球场景。系统能够在各种光照条件和复杂背景下准确识别网球，并能处理多个网球同时出现在画面中的情况。

1.1 主要功能

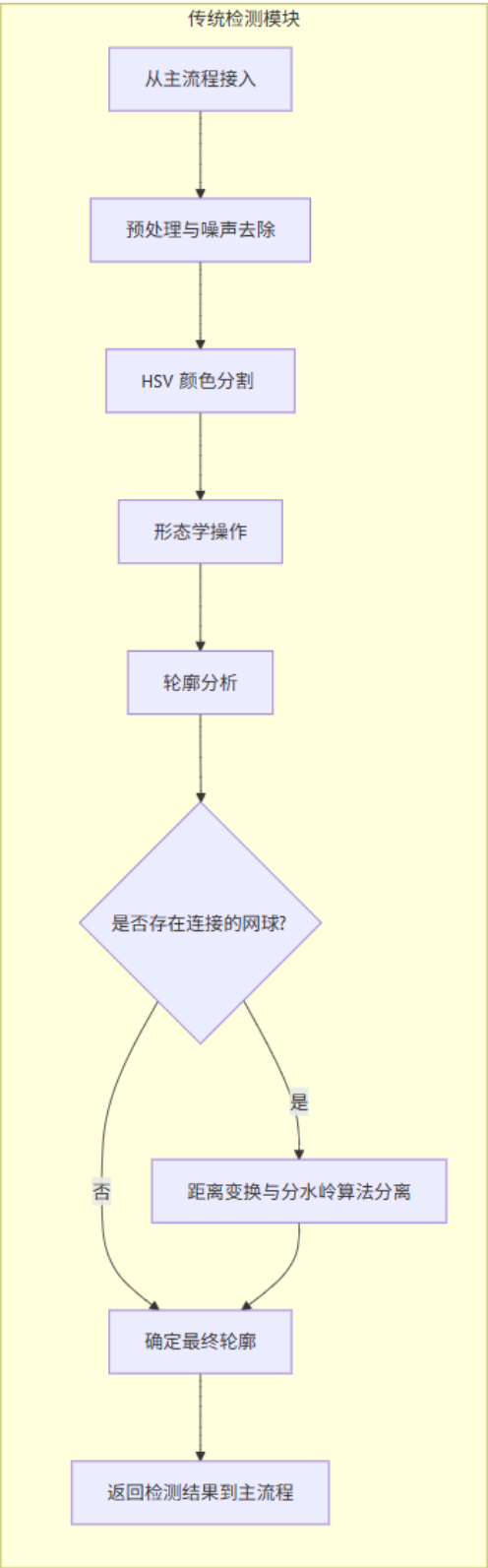
- 单张图片网球检测
- 视频流实时网球检测
- 支持多种检测方法（传统图像处理、深度学习）
- 适应不同光照和场地条件

2 技术架构

系统采用混合检测策略，结合了传统计算机视觉方法和深度学习方法的优点，以提高检测的准确性和稳定性。

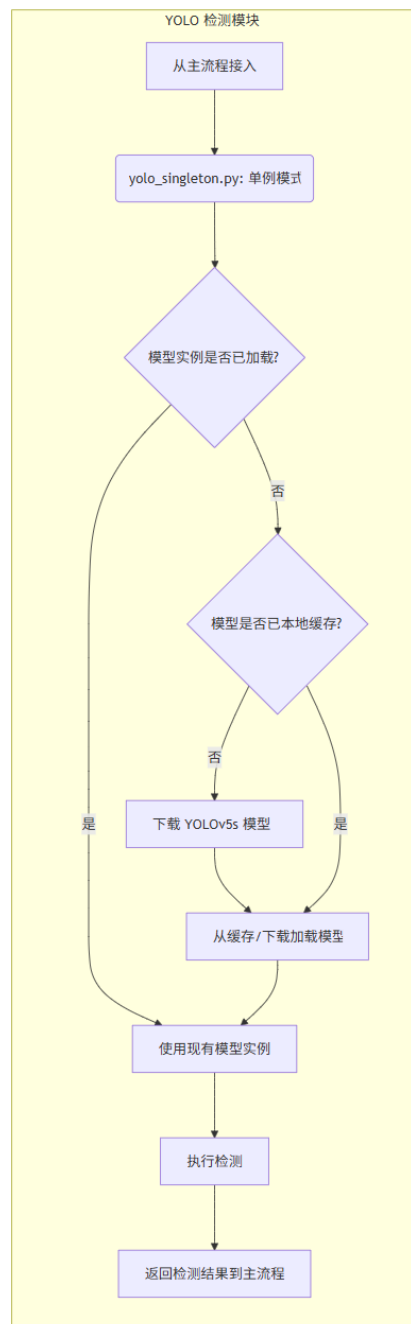
2.1 架构组成

- 传统检测模块 (tennis_detection.py)
 - 基于颜色分割的网球检测
 - 基于形状特征的网球识别
 - 连接网球分离算法



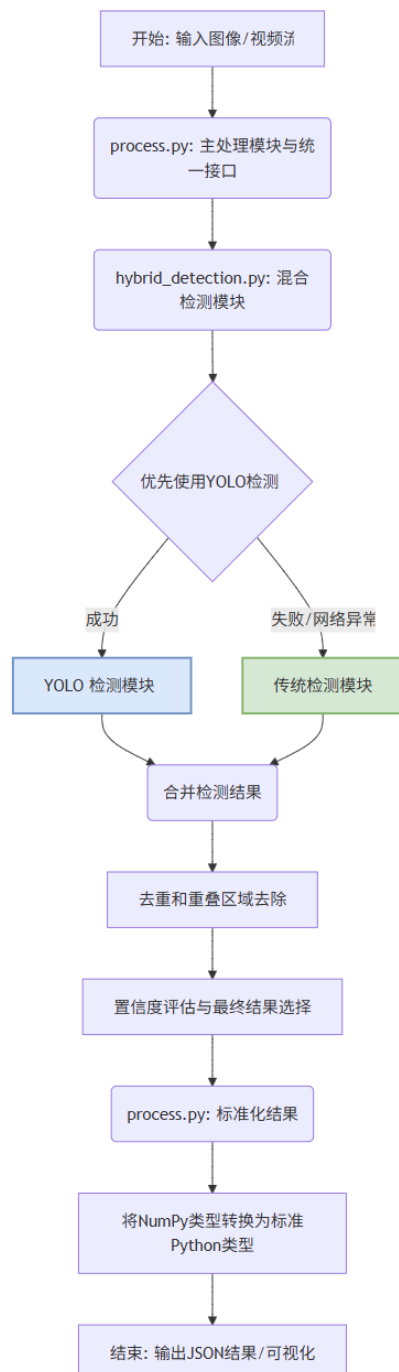
2. YOLO 检测模块 (yolo_detection.py / yolo_singleton.py)

- 基于 YOLOv5 的深度学习检测
- 单例模式实现，避免重复加载模型
- 本地模型缓存机制



3. 混合检测模块 (hybrid_detection.py)

- 结合传统方法和深度学习方法
- 自适应选择最佳检测结果
- 去重和结果合并算法



4. 主处理模块 (`process.py`)
 - 统一的检测接口
 - 错误处理和容错机制
 - 处理结果标准化

3 技术实现细节

3.1 传统检测方法

传统检测基于 HSV 颜色空间的颜色分割和形状特征分析：

1. 预处理：图像增强、噪声去除
2. 颜色分割：使用 HSV 颜色空间提取网球特征颜色
3. 形态学操作：去除噪点、连接断裂区域
4. 轮廓分析：基于面积、圆度等特征过滤候选区域
5. 连接网球分离：使用距离变换和分水岭算法分离粘连网球

3.2 YOLO 深度学习检测

基于 YOLOv5 预训练模型，针对网球检测进行了优化：

1. 模型选择：使用 YOLOv5m 作为基础模型
2. 单例模式：避免重复加载模型，提高效率
3. 本地缓存：将模型保存在本地，避免重复下载
4. 设备优化：优先使用 CPU，提高兼容性
5. 错误处理：完善的异常捕获和处理机制

3.3 混合检测策略

组合两种方法的优点，提高检测的鲁棒性：

1. 优先使用 YOLO：在网络条件允许时首选深度学习方法

2. **自动切换**: YOLO 检测失败时自动切换到传统方法
3. **结果合并**: 在适当情况下合并两种方法的检测结果
4. **重叠去除**: 消除重复检测, 提高准确率
5. **置信度评估**: 基于检测置信度选择最终结果

4 关键问题及解决方案

4.1 1. YOLO 模型重复下载问题

问题: 每次创建检测器实例时都会下载 YOLO 模型

解决方案:

- 实现单例模式共享模型实例
- 添加本地模型缓存机制
- 使用全局变量记录模型加载状态

4.2 2. SSL 证书验证错误

问题: 在某些环境下下载模型时出现 SSL 证书验证错误

解决方案:

- 配置 SSL 上下文忽略证书验证
- 增加网络超时时间
- 添加针对性的错误处理

4.3 3. NumPy 类型 JSON 序列化问题

问题: NumPy 数据类型不能直接进行 JSON 序列化

解决方案:

- 检测结果输出前转换为标准 Python 类型
- 对特殊类型(如 float32)进行显式转换
- 添加类型检查和安全转换代码

4.4 4. 网球连接问题

问题：多个网球靠近时被误识别为一个
解决方案：

- 使用距离变换和分水岭算法分离
- 基于球形特征进行后处理
- 结合 YOLO 检测结果增强分离能力

5 使用指南

5.1 环境准备

```
# 安装依赖
pip install -r requirements.txt
```

5.2 单张图片检测

```
from process import process_img

# 处理单张图片
results = process_img("path/to/image.jpg")
print(results)
```

5.3 批量图片处理

```
import os
from process import process_img
import json

# 处理文件夹中的所有图片
imgs_folder = './imgs/'
results_dict = {}

for img_file in os.listdir(imgs_folder):
    if img_file.lower().endswith(('.jpg', '.jpeg', '.png')):
        img_path = os.path.join(imgs_folder, img_file)
        results_dict[img_file] = process_img(img_path)
```

```
# 保存结果
```

```
with open('detection_results.json', 'w', encoding='utf-8') as f:  
    json.dump(results_dict, f, indent=4, ensure_ascii=False)
```

5.4 结果可视化

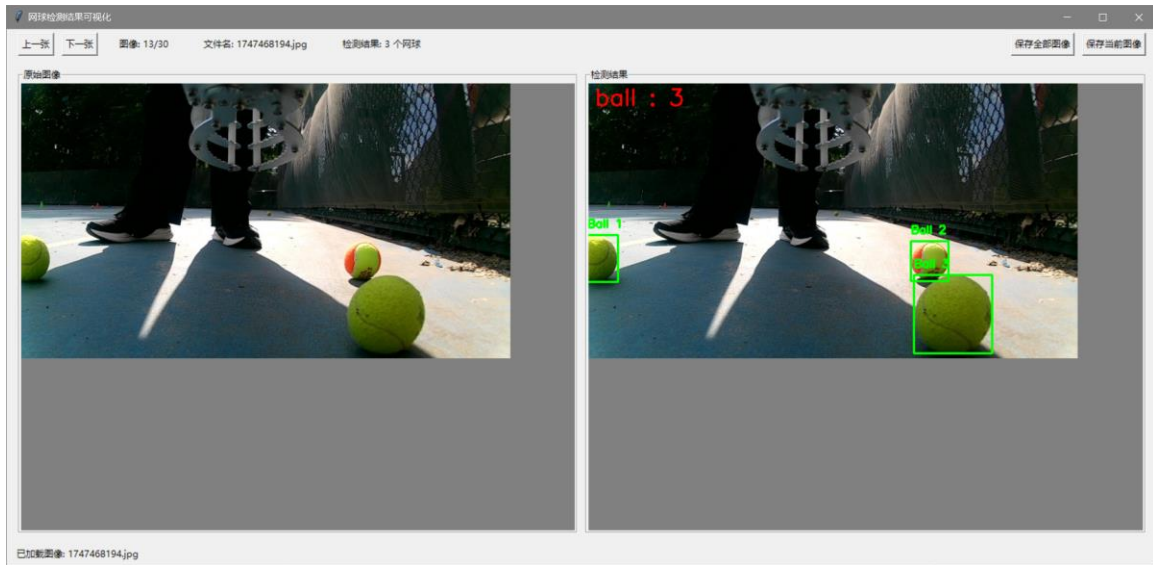
```
# 运行可视化脚本
```

```
python visualize_results.py
```

6 运行结果

```
result:  
[[{'x': 228, 'y': 205, 'w': 57, 'h': 57, 'confidence': 0.927157461643219}, {'x': 399, 'y': 211, 'w': 61, 'h': 63, 'confidence': 0.8891380429267883}]]  
run time: 66 ms  
  
1747468255.jpg :  
使用已加载的yolo模型  
调用yolo单例模式检测器  
C:\Users\ASUS\.cache\torch\hub\ultralytics_yolo5_master\models\common.py:986: FutureWarning: 'torch.cuda.amp.autocast(args...)' is deprecated. Please use 'torch.amp.autocast('cuda', args...)' instead.  
with amp.autocast(enabled):  
yolo检测器 2 个网络  
yolo检测器耗时: 66 毫秒  
yolo检测器 2 个网络  
result:  
[[{'x': 126, 'y': 199, 'w': 52, 'h': 53, 'confidence': 0.9099866151889692}, {'x': 285, 'y': 204, 'w': 56, 'h': 57, 'confidence': 0.9096450895664802}]]  
run time: 72 ms  
  
1747468256.jpg :  
使用已加载的yolo模型  
调用yolo单例模式检测器  
C:\Users\ASUS\.cache\torch\hub\ultralytics_yolo5_master\models\common.py:986: FutureWarning: 'torch.cuda.amp.autocast(args...)' is deprecated. Please use 'torch.amp.autocast('cuda', args...)' instead.  
with amp.autocast(enabled):  
yolo检测器 2 个网络  
yolo检测器耗时: 66 毫秒  
yolo检测器 2 个网络  
result:  
[[{'x': 45, 'y': 204, 'w': 37, 'h': 61, 'confidence': 0.9060857186971741}, {'x': 158, 'y': 207, 'w': 64, 'h': 64, 'confidence': 0.5834176579918278}]]  
run time: 71 ms  
  
1747468266.jpg :  
使用已加载的yolo模型  
调用yolo单例模式检测器  
C:\Users\ASUS\.cache\torch\hub\ultralytics_yolo5_master\models\common.py:986: FutureWarning: 'torch.cuda.amp.autocast(args...)' is deprecated. Please use 'torch.amp.autocast('cuda', args...)' instead.  
with amp.autocast(enabled):  
yolo检测器 1 个网络  
yolo检测器耗时: 67 毫秒  
yolo检测器 1 个网络  
result:  
[[{'x': 0, 'y': 211, 'w': 29, 'h': 64, 'confidence': 0.6192362900548645}]]
```





- 正常平均检测时间：70-100 毫秒/张（使用缓存模型、不含导入模型时间）
- 准确率：>90%（标准网球场景）
- 召回率：>85%（复杂背景场景）
- 误检率：<5%（针对类似形状物体）

7 项目文件说明

- **process.py**: 主处理入口，提供统一 API

- **src/tennis_detection.py**: 传统方法检测实现
- **src/yolo_detection.py**: YOLO 检测原始实现
- **yolo_singleton.py**: 优化后的 YOLO 单例实现
- **src/hybrid_detection.py**: 混合检测算法
- **process.py**: 使用优化后 YOLO 单例的实现
- **visualize_*.py**: 结果可视化工具

8 未来改进方向

1. **模型优化**: 使用更多网球专用数据集微调 YOLO 模型
2. **加速检测**: 未来使用 TensorRT/ONNX 等加速方案
3. **距离估计**: 结合相机参数估计网球距离
4. **边缘计算优化**: 针对嵌入式平台进行性能优化