ESP32-CAM 人脸识别门禁系统技术方案

项目概述

本项目基于 ESP32-CAM 摄像头、Flask 服务器和微信小程序，构建一个本地部署、可实时识别与权限控制的人脸识别门禁系统。适用于智能安防、家庭监控、访客管理等场景，系统支持识别身份后联动执行“开门”或“拒绝访问”等动作。

系统结构与工作流程

系统分为三部分：

1. **ESP32-CAM**：负责图像采集与 HTTP 服务；
2. **Flask 服务端**：接收图像并执行人脸识别；
3. **微信小程序**：用户界面，发起识别、显示结果。

基本流程如下：

> 用户点击识别按钮 → ESP32 拍照上传 → Flask 识别 → 返回识别结果 → 执行开门/拒绝操作

系统各端模块职责

ESP32-CAM 摄像头端

- 使用 `esp_camera` 驱动 OV2640 摄像头；
- 使用 `esp_http_server` 提供以下接口：
  - `/capture`：低分辨率图像，伪视频流帧；
  - `/capture_hd`：高分辨率图像，用于人脸识别；
  - `/stream`：MJPEG 实时流（可选）。

Flask 服务端

- 使用 Flask 构建 HTTP 接口 `/upload_photo`；
- 接收图像后使用 `face_recognition` 识别；
- 对比本地已知人脸库（128D 向量），返回 JSON 格式如下：

```
{
  "faces_detected": 1,
  "results": [{"name": "张三", "location": [top, right, bottom, left]}]
}
```

微信小程序端

- 定时拉取 ESP32 的 `/capture` 图像实现伪视频流展示；
- 用户点击按钮，调用 `/capture_hd` 拍照并上传到 Flask；
- 显示识别结果并提示“开门”或“禁止访问”。

功能拓展与规划

1. App 端适配鸿蒙系统，支持访问 `/stream` 实现实时视频；
2. 识别结果返回带人脸框图像（使用 Pillow 画框）；
3. 照相图像增强 + 保存至云端/本地；
4. 实现摔倒检测功能（定时上传图像 + 姿态识别模型）；
5. 陌生人识别预警机制（陌生脸三次触发警报）；
6. 录入新成员接口 + 权限验证机制（注册 + token）。

人脸识别模型技术细节

人脸识别使用 Python 的 `face_recognition` 库，封装 Dlib 的深度学习模型。流程如下：

1. **图像预处理**：JPEG 图解码为 RGB 格式；
2. **人脸检测**：`face_locations()` 使用 CNN 或 HOG 检测人脸位置；
3. **特征提取**：`face_encodings()` 使用 ResNet 提取 128 维特征向量；
4. **身份比对**：计算欧氏距离判断是否为同一人。

匹配判断公式为：

$$\text{distance} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \ldots + (x_{128} - y_{128})^2}$$

当 `distance < 0.6` 时，判断为同一人。

可使用 `face_distance()` 查看最小距离，`compare_faces()` 获取匹配布尔值。

安全与权限机制建议

- 所有服务应部署于局域网或添加 HTTPS；
- 增加上传接口的 token 校验，避免非法识别调用；
- 注册人脸操作必须绑定用户身份，后台管理员验证；
- 陌生人识别触发报警，应避免误报，需多次确认。

总结

本项目以轻量级设备 ESP32-CAM 为核心，结合本地 AI 服务和微信小程序界面，构建了一个完整的边缘智能人脸识别系统。具有成本低、可部署性强、拓展性高等优势，可作为智慧家居、校园门禁、独居老人看护等场景的应用基础。