

lpr camera

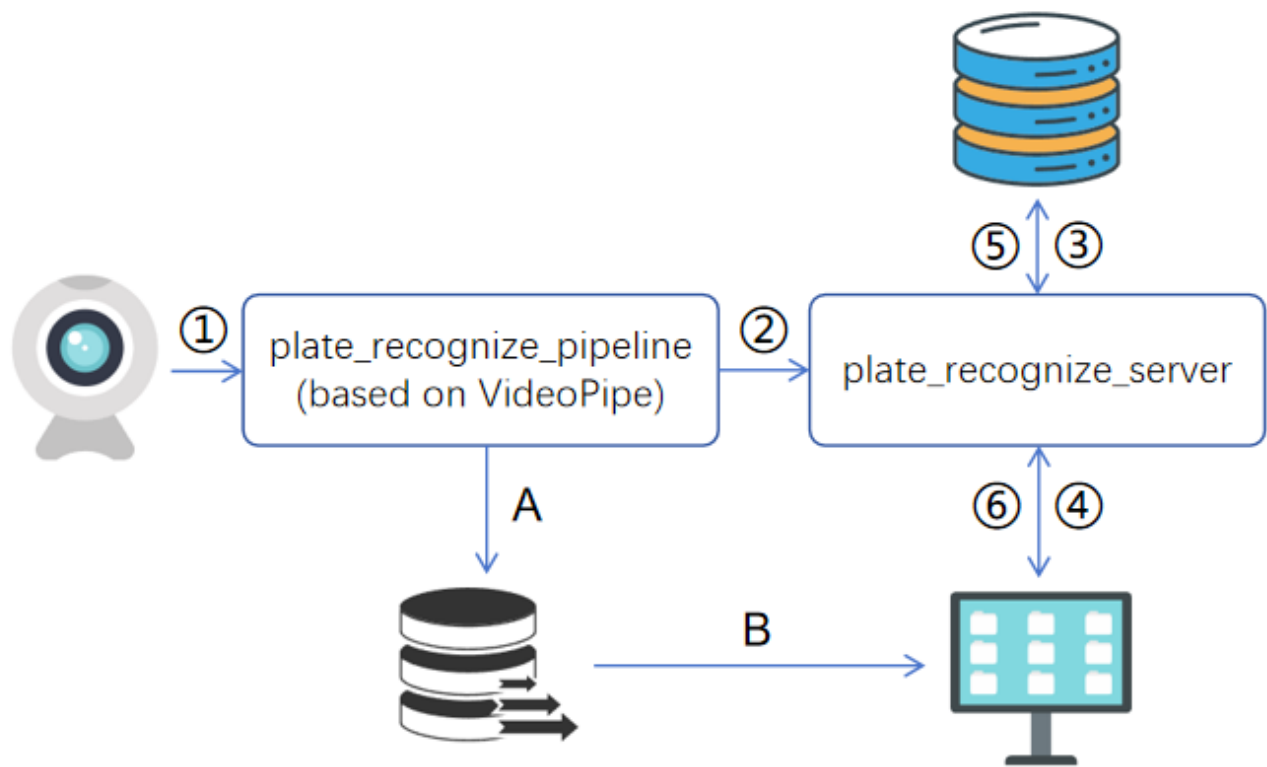
使用VideoPipe做车牌识别相机（LPR Camera）。

提前准备

- 1. 已准备好 VideoPipe 编译/运行环境（参见视频教程）
- 2. 已准备好本案例所需的测试模型和测试数据（即 vp_data，具体参见视频教程）
- 3. 已安装好本案例中 Python3 所需的依赖库（主要是numpy,pandas,opencv,flask,faiss等，其余按需安装即可）

主要文件

- 1. plate_recognize_pipeline.cpp 运行VideoPipe管道读取车牌视频，检测+识别车牌（一级推理），并将牌识结果以Socket/UDP的方式发送给车牌识别后台服务。
- 2. plate_recognize_server.py 后台服务接收车牌识别结果，存入数据库，并提供一个基于 flask 的 Web UI界面供数据查询。
- 3. clean.sh 帮助清空当前workspace下面的所有缓存数据。



- ① 读取车牌视频（用文件模拟）
- ② 车牌识别结果通过Socket转发到Server
- ③ 车牌识别记录存入数据库
- ④ 客户端请求车牌识别历史记录
- ⑤ 向数据库查询车牌识别历史记录
- ⑥ 车牌识别历史记录返回到客户端
- A 将实时车牌识别结果编码推流到rtmp服务器
- B 浏览器播放rtmp视频流（直播）

如何使用

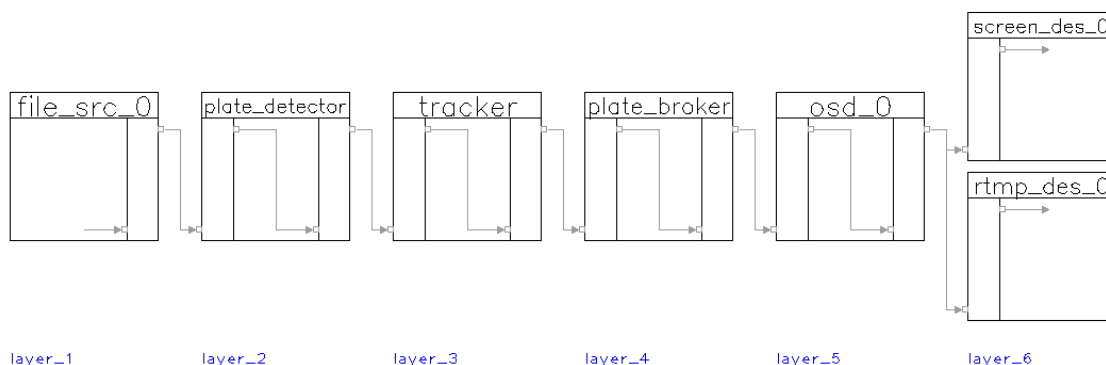
1. CMake编译 `plate_recognize_pipeline.cpp` 生成二进制可执行文件并运行，它会检测并识别车牌。(编译之前请先修改代码中 `des_ip` 和 `rtmp推流地址`)
2. 运行 `python3 plate_recognize_server.py` 脚本, 然后打开浏览器访问 `http://server-ip:9696` 查看牌识记录。(运行前, 请先修改脚本中 `bind_ip` 和 `live_stream_0` 的值)

数据流向

1. [pipeline -> license plate results -> send via socket]
2. [receive license plate results -> display results from web]

please run ``sh clean.sh`` if you want to restart sample from scratch.

VideoPipe管道介绍



1. `layer_1` 模拟获取车牌视频数据，实际可以换成rtsp/udp或者直接从摄像机Sensor抓图。
2. `layer_2` 完成车牌检测&识别，创建 `vp_frame_target` 对象，利用 `vp_frame_target.primary_label` 存储车牌颜色和车牌号码。注意这里用的是一级推理模式，直接从大图中检测车牌，不是先检测车辆、再抠图去检测车牌的二级推理方式。
3. `layer_3` 完成车牌跟踪，更新 `vp_frame_target` 的 `track_id` 和 `tracks`，避免重复提交的问题。
4. `layer_4` 完成数据转发，将管道中的车牌识别结果（`vp_frame_target.primary_label` 存储车牌颜色和车牌号码）按自定义格式，以Socket/UDP的方式发送给Python后台服务。
5. `layer_5` 完成结果绘制，主要是将当前画面中的车牌结果（`vp_frame_target.primary_label`）绘制出来。
6. `layer_6`
 - `screen_des_0` 负责GUI输出显示，显示基本无延时。
 - `rtmp_des_0` 负责rtmp编码推流，Web UI通过rtmp地址可以实时播放，有若干秒延时。

重要知识点

1. 如何避免车牌数据重复提交？
2. 认识VideoPipe 中一级推理和二级推理的区别？VideoPipe仓库中包含2种车牌检测的节点，可用于不同场景。
3. `vp_frame_target` 一般用来表示目标检测算法的输出结果，如何适用于车牌检测&识别这种先检测、后识别于一体的场景？

- 4. 浏览器如何播放 VideoPipe 管道推送的rtmp实时流？
- 5. 直接从摄像机Sensor抓图有哪些优势？

Sample运行截图

