软件层 MVP 工具链与开源项目清单

只围绕离线视频软件验证，梳理需要的软件和开源项目。

这份清单的定位

这份文档只服务于当前这个目标：

> 用现成泳池视频做离线软件验证，验证检测、跟踪、规则和结果输出这条链路是否成立。

所以这份清单不以硬件部署为中心，也不以自训练为中心。

这份文档也不重复讲“具体操作顺序”。

如果你现在要照着一步一步推进，优先看：

- 离线视频软件 MVP 操作手册 (/software-mvp/)

先说推荐的最小组合

如果今天就开始做第一版软件 MVP，最小推荐组合是：

1. 代码获取：GitHub 或 GitCode / Gitee / ZIP
2. 视频预处理：FFmpeg
3. 视频读写与可视化：OpenCV
4. 检测与跟踪：Ultralytics YOLO Track
5. 事件规则：Python 脚本 + YAML / JSON 配置
6. 结果导出：CSV / JSON
7. 人工复盘：Excel / 飞书表格 或一个简单网页

这已经足够做出第一版。

这次 MVP 里哪些是必须的，哪些是暂缓的

第一版必须

- 能拿到代码
- 能处理视频文件
- 能跑检测和跟踪
- 能输出事件结果
- 能做人审复盘

第一版可以暂缓

- 自训练
- 标注平台
- 在线接口
- 数据库
- 实时消息推送
- 企业级后台

1. 代码获取与协作

GitHub

作用

- 看开源项目文档
- 下载代码
- 查看 issue 和版本说明

现实提醒

非技术发起人不一定非要学会 Git 命令。

他只要知道：

- 这是代码常用来源
- 技术方可能会从这里拿项目
- 如果访问不稳定，可以用国内镜像或压缩包替代

参考：

- About Git - GitHub Docs (https://docs.github.com/en/get-started/using-git/about-git)

GitCode / Gitee

作用

- 国内镜像或迁移备份
- 给不能稳定访问 GitHub 的团队使用

参考：

- 迁移项目 | GitCode 帮助文档 (https://docs.gitcode.com/v1-docs/docs/repo/migrate/)
- GitHub 仓库快速导入 Gitee 及同步更新 - Gitee (https://gitee.com/help/articles/4284)

ZIP 压缩包

什么时候最实用

如果发起人不懂 Git、也没有科学上网条件，第一版最现实的办法通常是：

1. 技术方准备好仓库
2. 打包成 ZIP
3. 连同说明文档一起交付

这对非技术发起人更友好。

2. 视频预处理

FFmpeg

作用

- 把视频转成统一格式
- 抽帧
- 裁剪片段
- 生成复盘片段

在这次 MVP 里的价值

它不是可选项，几乎是必须项。

因为第一步就要把来源不同的视频处理成统一格式。

参考：

- FFmpeg Documentation (https://ffmpeg.org/documentation.html)

OpenCV

作用

- 读取视频帧
- 画检测框和轨迹
- 写出带标注的视频

在这次 MVP 里的价值

它是这次软件 MVP 最常见的底层库之一。

即使技术方最后用了别的工程框架，OpenCV 仍然很常见。

参考：

- OpenCV VideoCapture (https://docs.opencv.org/4.x/d8/dfe/classcv_1_1VideoCapture.html)

3. 检测与跟踪

Ultralytics YOLO

作用

- 做目标检测
- 做跟踪入口
- 快速得到基线结果

为什么它适合第一版

- 文档清楚
- 社区成熟
- 上手快
- 不必先自己训练

参考：

- Ultralytics YOLO Track Mode (https://docs.ultralytics.com/modes/track/)

ByteTrack

作用

- 多目标跟踪

在这次 MVP 中怎么用

更推荐把它当作“跟踪方案”，而不是一开始就单独维护一个复杂仓库。

更现实的做法是：

- 先用 Ultralytics 的跟踪模式
- 需要更细调优时，再研究 ByteTrack

参考：

- FoundationVision/ByteTrack (https://github.com/FoundationVision/ByteTrack)

BoT-SORT

作用

- 另一种常见跟踪方案

在这次 MVP 里的定位

它是 ByteTrack 的平行备选。

如果技术方想先少折腾，直接用 Ultralytics 默认配置也是合理的。

4. 事件规则层

这一层通常不需要额外的重型开源项目。

第一版更推荐这样做：

- 用 Python 写规则逻辑
- 用 YAML 或 JSON 保存阈值
- 用 CSV / JSON 输出结果

为什么这样更合适

因为这次 MVP 的关键不是炫技，而是先把判断链路做透明。

如果一开始就上复杂规则平台，反而会增加理解和调试成本。

建议第一版的规则输入

- 目标轨迹
- 速度变化
- 停留时长
- 区域或泳道信息
- 轨迹是否中断

建议第一版的规则输出

- 候选异常事件
- 风险分数
- 触发原因

5. 泳道和区域映射

这一层不是 AI 模型，而是配置。

建议做法：

- 由技术方在首帧或参考帧上画区域
- 保存成一个简单配置文件
- 用于判断目标属于哪条泳道、哪个风险区

第一版不需要引入额外 GIS 或复杂几何平台。

6. 结果导出和人工复盘

CSV / JSON

为什么推荐

- 最透明
- 最容易查错
- 最容易给非技术发起人看

典型输出

- `events.csv`
- `events.json`
- `tracks.csv`
- `video_manifest.csv`

Excel / 飞书表格

作用

- 做人工复盘
- 记录真阳性、误报、漏报

为什么第一版推荐

因为发起人和馆方通常已经熟悉表格，而不是专业标注平台。

Streamlit

是否第一版必须

不是必须。

什么时候值得加

如果技术方希望：

- 把结果集中展示

- 点击事件就跳转片段

- 让发起人不用看文件夹

那可以快速做一个内部网页。

参考：

- Streamlit Documentation (https://docs.streamlit.io/)

7. Docker

是否第一版必须

不是绝对必须，但很推荐技术方使用。

原因

它能减少“换一台机器就跑不起来”的环境问题。

对非技术发起人意味着什么

他不一定要会写 Docker 命令。

他只需要知道：

- 技术方可能会交付一个镜像

- 这是一种把运行环境打包好的方式

参考：

- Docker Engine on Ubuntu (https://docs.docker.com/engine/install/ubuntu/)

8. 哪些工具属于第二阶段再引入

Label Studio / CVAT

什么时候再引入

只有在你决定进入“数据标注和自训练”阶段时，它们才变重要。

为什么这次不优先

因为当前 MVP 首先验证的是：

- 开源现成模型能不能跑

- 规则链路有没有价值

还不是训练问题。

参考：

- Install Label Studio (https://labelstud.io/guide/install)
- CVAT Installation Guide (https://docs.cvat.ai/docs/administration/basics/installation/)

FastAPI / 数据库

什么时候再引入

如果第二阶段要做：

- 多人查看结果
- 事件检索
- 在线接口
- 多次运行统一管理

那时再上 FastAPI 和 SQLite / PostgreSQL 更合理。

第一版离线 MVP 甚至可以没有后台。

参考：

- FastAPI (https://fastapi.tiangolo.com/)

自训练与微调

什么时候才值得做

只有在下面情况同时成立时才值得进入：

1. 现成模型已经跑通链路
2. 误报和漏报模式比较稳定
3. 团队知道到底要补哪类数据
4. 业务上确认这个方向值得继续

9. 推荐的默认软件栈

如果只做“现成视频 -> 事件输出”的软件 MVP，推荐默认栈如下：

层	默认选择	原因
代码获取	GitHub + GitCode / Gitee 或 ZIP	兼顾资料完整和国内可访问性
视频标准化	FFmpeg	基本必需
视频读写	OpenCV	常见、透明

层	默认选择	原因
检测与跟踪	Ultralytics YOLO Track	跑得快、资料多
规则引擎	Python + YAML / JSON	易调试、易解释
结果输出	CSV / JSON + 标注视频	便于复盘
复盘方式	Excel / 飞书表格	非技术发起人最容易参与
可选展示	Streamlit	需要内部网页时再加
可选封装	Docker	降低环境坑

10. 对非技术发起人的一句话建议

这次不要先纠结“要不要自己训练模型”，也不要先纠结“摄像头怎么装”。

先盯住这一个问题：

> 用现成视频，系统能不能稳定输出值得人工回看的候选异常事件？

如果这一步不能成立，后面所有更重的投入都没有意义。