

万世杰

应用开发 全栈开发 制造业数字化

男 户籍：安徽 六安 工作地点：安徽 芜湖
电话 微信： (微信同号) :
邮箱： 备用：
期望地点：安徽省内

个人简介

安徽工程大学人工智能专业 届本科毕业生，现任 (维智捷) 芜湖精益专员。具备全栈业务系统、 应用与微信小程序开发经验，参与制造现场物料拉动系统建设，独立开展 、 企业知识库和本地大模型应用。能够将业务需求拆解成清晰的 与开发任务，结合 工具、低代码 workflow、服务器部署和文档交付，推进从需求到维护的完整流程。

教育背景

安徽工程大学 人工智能专业 本科

核心课程：数据结构与算法、数据库系统、软件工程、计算机网络、机器学习、计算机视觉、自然语言处理、神经网络与深度学习、机电设备 控制、计算机视觉算法及应用。

工作经历

(维智捷) 芜湖

精益专员 汽车研发 制造 至今

生产物料拉动系统：参与现场与仓库物料拉动系统建设，负责前后端、数据库和微信小程序开发维护，组织扫码、补货任务、标签打印、收货确认与异常追踪；在业务入口处理重复请求和盒子状态校验。

业务流程数字化：梳理现场叫料、仓库配料、标签打印、工位配送流程，主导设计标准化配送标签和状态管理机制，减少口头沟通、人工判断、错配漏配与送错工位。

系统落地成果：叫料响应由约 分钟缩短至 分钟，仓库配料效率提升约 ，人工沟通减少约 ，配送异常减少约 ，任务可追溯率达到 。

企业 应用：独立完成 文档识别与结构化处理能力建设，搭建 企业知识库，支持内部资料检索、智能问答、信息引用和业务数据辅助处理。

本地大模型部署：独立搭建本地 模型环境，完成模型部署、 接入及业务应用，用于文档分析、知识问答与结构化数据生成。

系统重构与维护：参与内部老旧系统重构与工艺文档处理平台建设，组织前端交互、 业务服务与 文件处理引擎，提供文件生成、核对、任务进度及结果下载；持续处理接口、数据和部署环境中的异常。

质量数据与现场协作：参与质量数据整理、现场异常记录、问题跟进及资料归档；接触物理检测、 工具和汽车质量体系文件，使用 、 、 处理报表及汇报材料。

核心能力

、 与自动化：把需求转换为明确的 与结构化输出约束；使用 、 、 、 等工具辅助开发，结合 、 、 、 和 落地业务 workflow。

全栈与移动端交付：业务系统；与微信小程序开发。

工业视觉与制造现场：数据采集、清洗、标注、训练和部署；工业场景识别与定位、嵌入式与机器人集成实践；物料拉动、配送标签和现场流程梳理。

部署、数据与文档：云服务器与网站运维；网络及企业共享盘实践；数据处理，和交付。

项目经历

生产物料拉动系统

工作项目 至今

- 把现场扫码、仓库补料、配送标签和收货确认接在同一条任务流程里，让每一次叫料都有状态、有记录，也能找到对应的物料和工位。
- 技术方向：微信小程序
- 参与现场需求梳理，负责前后端、数据库与微信小程序开发维护。围绕物料、工位、标签、周转盒和补料任务建立业务关联，将现场与仓库的操作集中到系统中。
 - 实现扫码叫料、仓库任务、标签打印、库存查询和日志追踪；补充收货确认与异常上报，让配送任务能够闭环，并通过推送任务变化。
 - 统一配送标签中的物料、数量、仓库、工位地址、人员、图片和二维码。扫码入口清理扫码枪控制字符，兼容普通条码、二维码及键值对内容，避免不同码制影响现场操作。
 - 按个人提供的现场成果记录，叫料响应由约 分钟缩短至 分钟，配料效率提升约 ，人工沟通减少约 ，配送异常减少约 ，任务可追溯率达到 。
 - 为后续库存联动、配送和生产计划协同保留扩展基础，具体上线范围仍以现场实施情况为准。
 - 对重复请求建立幂等记录，并在扫码业务中加锁；已有补货任务时返回原任务，减少连续扫码生成重复单据。双盒切换前校验当前盒与备用盒状态，遇到配对异常或备用盒未满时停止切换并记录告警。

背景与目标

生产现场缺料时，最急的事情是把正确的物料送到正确的工位。过去，这个动作常常要经过口头叫料、仓库确认、配料和配送几次转述。物料号、仓库料号、工位地址和数量没有放在同一处，接收信息的人需要再次核对。等物料送出以后，现场还需要知道任务到哪一步，仓库也需要确认这一单是否已经结束。就是从这条日常流程开始做的。

我的目标是把一次叫料变成有编号、有状态、有操作记录的任务。现场扫码后，系统能够识别物料和工位，仓库能够集中接收补料需求，配送标签能够带着同一组信息继续流转，现场收货后能够确认闭环。这样，后续排查一次送错、漏配或重复叫料时，不必从头问一圈人，而是先找到对应任务和它的操作过程。

这是一项工作项目。我参与现场需求梳理和系统建设，负责前后端、数据库及微信小程序的开发维护，也参与服务器部署、问题排查和版本迭代。项目仓库还包含 扫码端、打印代理和部署资料。不同客户端服务于同一条物料流程，但仓库存在多个版本，代码中的能力与现场实际启用的功能需要分别说明。

现场问题与业务约束

不同编码需要落到同一笔业务 现场叫得出口的物料号、仓库使用的料号、打印在标签上的编码，不一定是同一种标识。扫码枪读到的也不总是一串干净的数字，可能带回车、，也可能读出 或键值对内容。因此，系统不能把扫描文本直接当成唯一业务编号；它需要先整理输入，再查找候选标签，最后确认是否存在明确、唯一的匹配。

另一类问题来自连续操作。操作员可能再次扫同一个码，设备可能重试网络请求，仓库任务还没有出发时也可能收到同一份需求。如果每次请求都直接新增任务，看板上的数量就会失真，仓库也容易重复配料。扫码入口必须区分一次请求

的重复提交和同一标签的重复业务需求，并在生成任务之前处理这些情况。

双盒物料又多了一层状态约束：正在使用的盒子扫空后，备用盒需要能够接上。不是任意两个盒子都可以切换，配对关系、当前使用状态和备用盒满料状态都需要校验。若数据不完整或备用盒还未准备好，继续执行会把异常带到现场。因此，这一段逻辑需要把正常流转与异常停止写得同样清楚。

技术架构与职责分工

□□□□□□□□□□负责业务，□□□负责操作界面 后端使用□□□□□□和□□□□□□□□□□，按□□□□□□□□□□、□□□□□□□□、□□□□□□□□、□□□□□□□□□□组织代码。□□□□□□□□□□接收扫码、任务、标签和导入请求，□□□□□□□□处理状态转换与业务规则，□□□□□□□□□□负责数据访问。□□□□□用于业务数据存储，项目依赖也保留了□□。这样的分层让界面调整、业务条件变化和数据库操作不必挤在同一个接口方法里。

前端使用□□□□□、□□□□□□□□□□、□□□□□和□□□□□□□□□□，结合□□□□□调用业务接口，□□□□□□□支持数据展示。任务、物料、映射和标签等页面围绕现场及仓库的操作展开，而不是只展示一组统计数字。前后端之间通过明确的数据结构传递任务编号、状态和关联信息，使列表、操作入口和后端校验能够对应到同一笔业务。

实时变化通过□□□□□□的□□□□□□□□□□消息通道推送。任务生成、打印任务提交等动作能够发布到对应主题，减少看板完全依赖手动刷新。推送服务同时记录□□□□□□信息，失败时留下原因。这里的价值是让业务操作和界面更新有连接，也保留排查线索；它并不意味着网络故障时的消息已经获得端到端必达保证。

部署方面，□□□□□□分支保留了□□□□□□□□□□、□□□□□□□□□□和□□□□□□的运维说明。□□□□□提供前端静态文件，并将□□□□和□□□□□□□□□□请求反向代理给本机后端，浏览器通过同一入口访问。服务启动、日志查看、版本更新和地址变化都需要纳入交付考虑。对现场工具来说，能开机恢复、能找到入口、出问题能定位，与页面功能同样重要。

关键功能与实现

□□□扫码解析与业务匹配 □□□□□□□□□□□□□□□□□□□□先清理扫描内容边界和控制字符，限制异常长度，再从普通内容、□□□□□字段、键值对和启用的规则中提取查询候选。候选命中结果按标签记录去重，最后要求唯一匹配。找不到标签或找到多个标签时返回明确的业务错误，而不是随便取第一条。规则表提供了前缀、精确值、正则等匹配入口，便于后续增加码制时减少核心流程改动。

□□□□□□□□□□在标签解析之后继续处理叫料。标签不存在时还会进入物料号叫料路径；标签存在时检查关键字段与状态，记录扫描时间、设备和操作员，并关联周转盒及补货任务。这个顺序把输入解析、基础数据检查和任务生成分开，避免一个看似有效的码在缺少数量、物料或送达工位的情况下继续流转。

重复请求与双盒切换

请求侧使用幂等键和请求摘要记录执行状态，区分正在处理、已完成和失败的请求；扫码业务还通过业务锁围绕同一个扫描对象执行。业务侧则查询尚未出发的补货任务，遇到重复扫码时返回已有任务编号并写入审计记录。这两层处理分别应对网络重试与现场重复操作，目的都是减少重复单据，而不是用前端按钮禁用来代替后端约束。

双盒流程要求配对数量正确，当前盒处于允许的状态，备用盒已经满料。正常情况下，当前盒转入等待补货，备用盒转为使用中，再建立对应补货任务；异常情况下记录健康状态、锁定原因和告警，并停止切换。这样的状态管理让□不能继续□的原因也成为业务数据，便于仓库或现场人员找到需要修正的位置。

任务流转、收货与异常

□□□□□□□□□□把接单、开始配料、配料完成、配送、到达和完成等动作放进状态转换逻辑。每个动作先校验任务当前状态，再更新状态及相关信息。例如，不能把刚创建的任务直接当成已经送达，也不能忽略配料前的库存关联。□□□□□□□□□□还提供现场收货确认，把扫描内容解析到配送任务，再交由任务服务处理闭环；填写空盒信息时，流程能够衔接空盒回收。

后台刷新使用 `onRefresh` 的 `onRefresh`，配合电费与课表的 `onRefresh` 展示桌面信息。组件提供的是 `onRefresh` 桌面入口，与应用内完整页面分工：桌面适合快速看当前余额或课程，页面适合查看历史、筛选信息和调整设置。代码也包含深色主题、帮助、隐私说明和更新入口，让应用具备持续使用所需的外围内容。

核心功能与实现

多周期电费分析 分别读取最近 天、 天和 天的数据，整理日期与余额，再从相邻记录中区分用电下降与充值上升。消耗记录不仅保留金额，也保留跨越的时间，计算时按有效天数得到日均值。近期数据与长期数据的选择取决于是否有足够记录，而不是把每一种周期都当成同样可靠。

代码对有短期与中期记录的情况进行加权，例如采用近期日均值并参考中期变化；若近期记录不足，则寻找更长期的历史。没有有效消耗记录但存在充值记录时，会尝试分析充值模式；仍缺少依据时再使用估计值回退。数据来源通过枚举与界面状态传递，让使用者能够区分 □ 来自历史 □ 与 □ 暂时估计 □。

剩余天数计算还考虑不同阶段的日均消耗变化，并把最终估算限制在 10 天上限。这个上限是应用的保守规则，不是性能指标。它避免少量历史或较低日均值把结果放大成不合理的长期预测。实际消耗仍受人数、季节和用电习惯影响，所以这一功能的用途是提供充值判断参考，而不是替代真实计量。

历史保存与文件导入导出

查询结果保存进本地数据库，变化检测帮助组织电费历史。应用保留时间、余额、楼栋、房间、用电量和充值等字段，后续分析能够围绕这些记录继续进行。历史与当前值分开后，界面可以查看已有数据，也能在网络异常时保留之前的观察依据，而不必把所有内容绑定到一次实时请求。

使用 和文件 读写 、 。 导出保留结构化字段， 提供便于表格软件查看的形式；导入时检查必要字段并解析时间与余额。文件入口的意义是让用户能够转移、保存和检查自己的历史，而不是把数据只能留在应用内部。它也让版本维护 and 用户排错有了更直接的数据依据。

课表导入与桌面展示

课表模块通过 `openCoursePage` 打开教务登录页面，在用户已有登录状态的基础上注入解析脚本。
`loadPage` 组织页面加载、`executeScript` 执行和 `callback` 回调，接收解析后的课程数组；代码还包含进度提示、学期选择相关适配和备用解析路径，用于应对页面结构及返回内容的差异。

课程进入本地课表数据模型后，由 `CourseTable`、`Course` 和 `Section` 管理，再提供到课表页面与桌面组件。这里需要处理的不只是课程名称，还包括教师、教室、星期、周次和节次。如果这些维度在导入阶段混淆，之后的桌面显示再精致也会给出错误时间。因此，解析结果的字段含义与本地存储结构必须一致。

校园卡信息与分页流水

[illegible]

流水保存的目的，是让□查到了多少记录□和□当前显示哪一批记录□有清晰依据。遇到账号缺失、响应异常或记录字段不完整时，□□□□□□□□□□返回明确结果，映射过程中也处理无法使用的数据。它没有把一次网络成功简单等同于数据完整，而是把账号、分页、数据转换和保存分别作为需要处理的环节。

后台刷新和组件同步

在执行时检查当前时间及登录信息，符合条件后获取电费、保存数据、检查变化并更新桌面组件。查询没有获得有效结果时会进入重试或失败返回，避免假装刷新完成。组件刷新与数据库更新放在同一条工作流程里，是为了减少主页面已有新值而桌面仍显示旧值的情况。

- 将业务需求拆解为高质量 输入输出约束和执行步骤，改进对话准确性、格式稳定性及文档 表格输出。
- 开展语音识别、文本分析和图像理解等多模态处理，接入大模型 并组织结构化数据流程。
- 通过 、 、 搭建知识库问答、 技能调用、文本生成和表格处理流程。
- 了解并实践增量预训练、指令微调、强化学习和 相关微调，评估模型训练、推理与部署方案。

校园业务管理系统

既有全栈项目经历

开发校园综合业务管理平台，涉及教务、后勤和设备管理，完成前端交互、业务接口、数据库及服务器部署。

技术 方向： 并部署到云服务器，持续维护运行环境。

- 开展业务需求梳理、前端页面与交互设计，优化操作流程。
- 参与数据库建模与查询优化，设计 并部署到云服务器，持续维护运行环境。

语音助手与 交互

智能硬件项目实践

探索把语音对话和设备操作接到同一套工具接口中，通过 暴露设备查询与控制能力，并组织语音识别、模型回复和语音合成流程。

技术 方向： 处理会话启动、状态返回和关闭时的任务取消。当前音频累积为简化实现，仍需完善语音活动检测和真实设备验收。

- 在服务端组织 工具注册、 与 协议接入，将设备列表、属性读取和控制操作封装成可调用工具。
- 使用统一的 与设备管理器登记设备类型和实例，让不同设备通过相同入口查询和调用，减少协议处理与设备逻辑之间的耦合。
- 为客户端建立独立的异步音频队列，串联 、 与 ，处理会话启动、状态返回和关闭时的任务取消。当前音频累积为简化实现，仍需完善语音活动检测和真实设备验收。

背景与目标

这个项目探索的是让语音对话具备实际操作入口。用户说出一句话后，系统需要把声音变成文本，让模型生成回复，再把回复转成声音；如果还涉及设备，就需要能够查询设备、读取属性或执行动作。单独接一个聊天接口只能完成其中一段，所以我尝试把模型、语音与设备能力组织在同一套服务中。

项目使用 、 和 作为服务与工具入口，通过 和 组织连接，设备侧使用 抽象管理类型、实例、属性和方法。主要源码在 的 分支。由于仓库是私有的，公开案例只描述通用架构和代码实现，不公开配置、凭据或内部资源，也不将它混同为企业 与 项目。

对话与设备交互的问题

语音对话是一条包含多个等待点的流程。客户端上传音频后，识别服务可能没有得到文字，模型调用可能失败，语音合成也可能需要较长时间。如果界面只等一个最终结果，用户无法判断系统是否还在工作。因此，会话需要能够返回识别、思考、播报、空闲和错误等状态，并在连接关闭后停止相关任务。

设备操作则需要另一种明确性。服务需要知道可用设备有哪些，设备暴露什么属性，某个动作需要哪些参数。不能把一段自然语言直接变成任意设备命令。因此，设备查询和控制被封装为单独的工具，模型或客户端通过这些已定义的入口进行操作，设备逻辑与连接协议保持相对独立。

技术组织与组件职责

模型、语音和设备分别管理 定义加载和生成接口， 负责本地模型， 负责外部接口， 根据配置选择并管理当前模型。本地分支使用 的 和 ，根据 是否可用选择运行设备及精度，并传入生成长度、 、 等参数。这说明项目已有本地与 两条接入路线，但不代表所有模型都经过部署验证。

模块通过 调用 ，将音频转换成文本，识别不了或请求失败时返回空结果。代码预留了其他引擎的扩展方向，但当前不能写成已经完成本地语音识别。使用独立的引擎与管理模块处理合成请求，和模型生成分开，使替换语音服务时不必重写整条对话流程。

服务入口组织 工具的动态注册，以及 、 协议的启动与关闭。不同协议接收的请求最后进入统一处理入口，再按原连接方式返回结果。这样，设备能力不必分别为每一种网络连接复制一份，而连接层也不需要了解每一个设备的业务细节。

关键实现与数据流程

独立会话和异步音频队列 为每个客户端持有独立的 和处理任务。会话启动时创建后台协程，接收音频块时放入队列，主循环依次累积音频、调用 、取得模型回复，再组织 输出。过程中通过 返回状态与识别内容，客户端可以区分当前正在处理哪一段。

关闭会话时设置停止状态，并取消已有处理任务，捕获 完成正常清理。这个处理的目的是避免页面离开以后，服务仍继续占用识别、生成和音频输出资源。代码目前采用简化的固定音频累积方式，注释也明确指出需要更完整的 。因此，这是一条已有实现的语音会话链路，尚不能称为成熟的实时语音产品。

统一的设备注册与工具调用

保存设备类型、创建器和已注册实例，创建设备时调用相应创建逻辑，再将实例加入管理器。 工具提供设备列表、单个设备信息、属性查询和方法调用入口。工具函数先找到设备，再进入对应能力，异常时组织可返回的结果，减少连接处理代码直接操作具体设备对象的情况。

这层抽象解决的是扩展方式：新增一种设备时，重点补充它的类型、属性和方法，而不是重新设计所有查询接口。仓库中包含智能灯、虚拟 和系统监视器等设备模块，能够说明接口组织方式。具体物理设备是否连接、控制动作是否生效，仍需要硬件与运行环境中的实际验收。

已完成的目的与当前范围

项目已经将模型接入、语音识别、语音合成、设备管理和工具接口分成可组合的模块，形成语音输入到回复输出，以及设备查询到动作调用的代码路线。它完成的目的，是为 可以交流，也可以调用工具 的助手建立基础结构，让对话与设备操作不再只是两份相互分离的脚本。

公开展示不宣称完整实时性、低延迟指标或物理设备覆盖数量。 依赖网络服务，本地模型受算力与模型配置限制， 中的状态反馈也不等于完整端到端流式推理。已有代码可以核对模块接口与处理顺序，可靠性则需要实际运行和异常场景记录。

后续完善与实践收获

后续工作应围绕语音活动检测、会话取消、网络断开、模型失败以及设备错误继续验收。语音输入到达的速度和处理速度不同，还需要检查队列与超时行为；设备操作也需要明确哪些动作允许执行、哪些结果能够确认。项目让我理解到，模型只是其中一环，服务生命周期、协议和设备抽象决定了助手能否真正接入应用。

路由管理实践

路由与网络工具实践

围绕路由配置、局域网共享与远程任务运行开展实践，并结合多设备项目中的 适配，探索在支持容器的 上部署 。

宿舍电费监控

开源工具

用 0 0 0 0 0 定时查询宿舍剩余电量，把查询时间和余额追加到 0 0 0，再生成趋势图。低余额时在运行终端给出提醒。

技术 方向：

- ☐ 使用 □□□□□□□□□□□□□□□□ 调用校园一卡通查询接口，从响应文本提取剩余电量，按查询周期记录时间与余额，便于保留连续观测数据。
- ☐ 通过 □□□□□□ 追加 □□□□，再用 □□□□□□□□□□□□ 导出趋势图；配置余额阈值和检查间隔，低于阈值时输出控制台提示。
- ☐ 当前仓库的消耗量字段仍为占位值，邮件和微信通知尚未实现；它是独立的 □□□□□□ 监测工具，与 □□□□□□□□ 校园助手中的历史分析和预测分开展示。

背景与目标

宿舍剩余电量通常要进入校园一卡通页面查询，查完以后结果就留在那一次页面里。单次余额能告诉我现在还有多少，却不能说明最近变化得快不快。□□的□□□□□□工具从这个简单需求出发：定时查询，把时间和余额留下来，再用图表查看变化，余额较低时在运行终端提醒。

这个项目是独立脚本，与 `校园助手` 分开。`0.0.0.0` 版本后来组织了本地数据库、多周期分析和桌面组件；`0.0.0.0` 仓库目前关注的是查询、`0.0.0` 记录、趋势输出和控制台提示。展示时明确这条边界，是为了让读代码的人能够看到脚本真实完成了哪些工作，而不是把另一个应用的功能一起算进来。

数据获取与提醒的问题

查询依赖校园系统的有效会话。脚本需要用已有登录信息访问接口，响应中还要提取真正的余额文本，不能把 0000 请求结束就当成已经取得有效数据。登录信息失效、响应结构变化或无法找到余额，都会影响后续记录。因此，自动轮询首先要确认拿到的是可以使用的数据，再把它追加到历史。

低余额提示也有明确范围。当前实现通过控制台输出提醒，并没有邮件、微信或手机通知链路。如果脚本无人查看地运行，终端提醒不能自动变成能够送达用户的通知。这个限制影响使用方式：脚本适合本地持续观测或作为采集基础，进一步做远程告警还需要增加发送、去重、失败记录与重试机制。

技术选择与脚本结构

项目使用 `os` 和 `requests` 执行请求，`table` 组织表格数据，`image` 生成图像，`check` 按配置间隔触发检查。配置与主流程分开，宿舍查询需要的参数、余额阈值和检查周期不直接散落在每个处理函数中。脚本结构适合一个持续运行的小工具，没有把简单轮询扩成复杂服务。

□ □ □ 是这个阶段的数据存储形式。每次记录带查询时间、余额和消耗字段，能够用表格软件查看，也便于之后转入其他统计程序。对于个人脚本，它比一开始就引入数据库更直接；但 □ □ □ 也没有现成的数据去重、并发写入或事务管理，后续扩展需要根据运行方式选择新的存储方案。

查询、记录与可视化实现

□ □ □ 从响应文本提取余额 主流程向校园一卡通查询接口提交请求，余额提取函数使用正则从响应文本中寻找 □ 剩余电量 □ 之后的数值。这个处理让业务值与整段页面或响应内容分离，但匹配规则依赖当前文字和数字格式；如果上游页面改了措辞、格式或返回结构，就需要重新核对提取逻辑，而不是默认每次都成功。

获得余额以后，脚本整理为带时间戳的记录，通过 `cat >> file` 追加到 `file`。连续数据可以帮助比较不同日期的余额，但当前代码中的 `cat >> file` 仍赋值为 `0`，并带有待实现说明。因此，`file` 中存在消耗列不代表已准确计算用电量，也不能从输出图上的这个系列推定真实消耗变化。

逐步缩小重复检测范围

扫描器先把文件按大小分组，只有同样大小且不止一个文件的分组继续参与检测。随后计算局部指纹，把候选再次归并，再对仍存在多个成员的分组计算完整哈希。这样的分阶段处理避免所有文件一开始都读取完整内容，将较昂贵的比较集中到更可能重复的候选上。

重复分组还需要与可移除文件列表区分。检测说明内容相同，不说明路径用途相同，因此界面与清理报告保留分组信息供检查。项目没有附带能够证明所有目录规模下的性能数字，我也不把这套筛选路线写成未经测量的速度提升百分比。能够核对的是处理顺序和减少候选的机制。

回收目录与恢复路径

删除先把文件移入应用的回收目录，并记录原路径、文件名、大小、类别和删除时间。操作结束返回已移动与失败列表，索引记录用于后续展示和恢复。恢复时先创建原目录，原路径已经有文件则寻找不冲突的目标名称，避免简单覆盖；另一个入口才负责彻底删除回收记录对应的文件。

这套回收机制使清理与不可恢复删除成为两种不同操作。根据实际返回结果更新报告，将已移动和已经不存在的路径从当前数据中移除，保留失败说明。文件操作不是一个布尔值就能表达的事情，结果列表能够帮助用户知道这次到底处理了哪些文件。

规则分类与风险提示

将路径统一为便于比较的形式，对相册、聊天应用和数据目录等敏感位置生成提示。的根据路径关键字、文件名前后缀匹配日志、临时文件、缩略图和旧备份，并记录规则的风险等级。它们提供的是可解释的线索：为什么这个文件被推荐查看，以及删除前需要注意什么。

和都保留了本地处理的思路。扫描结果、分类与清理操作围绕当前设备展开，不能因为能识别缓存就声称可以安全清理任何应用数据。平台权限、文件占用和目标目录的实际状态仍会影响结果，工具需要接受部分失败，并把它展示出来。

完成目的与交付内容

项目已经从列出文件走到汇总空间、发现候选、检查风险、执行操作、保留恢复信息。与分别提供可以阅读的源码，扫描、分类、重复分析、清理规则和回收逻辑都有具体实现。它完成的目的，是让存储整理有依据，让用户先看到信息再作决定。

目前不提供虚构的清理容量、设备覆盖率或扫描速度指标。代码也不能代替对不同系统版本、权限条件和真实文件结构的验收。尤其在相册及聊天目录中，规则只是提示，不能替用户判断个人资料是否可以删除。

实践收获与后续方向

这个项目让我把文件系统、异步界面、缓存索引和风险控制放在同一条流程里考虑。扫描时要持续反馈，报告要能够逐层查看，删除后要与当前列表一致，恢复时又要处理路径变化和同名文件。后续可以继续改进大目录体验、增量索引、规则解释与预览，但每一项优化都应保留清楚的操作结果。

源码：[https://github.com/xxf001/xxf001.github.io](#)

多设备开发协作

个人项目

让一台电脑上的开发会话协调自己的其他设备执行任务。把会话状态、远程工作区和执行结果留在自托管服务里，减少跨机器开发时反复搬文件和重新交代上下文。

技术方向：[https://github.com/xxf001/xxf001.github.io](#)

采用与分离的结构：控制端维护项目、会话、租约和结果，执行端承担设备上的任务，客户端提供开发工具入口。

- 为会话附加与释放实现单写租约和持久状态，协调已授权客户端接管；执行工作区支持 □ □ □ □ □ □ □ □ □ □，将任务开发与原工作区分离。
- 将源码与产物打成带元数据的工作区 □ □ □ □ □ □，校验摘要，并对回收的远程产物检查大小与 □ □ □ □ □ □ □ □，降低传输错误和结果错配风险。
- 提供桌面、服务器及 □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ 的安装与运行适配。当前处于 □ □ □ □ □ □ 阶段，全新设备安装、真实多机并发和跨客户端接管仍以项目验收矩阵为准。

背景与目标

[illegible]

项目使用 `libvirt` 和 `qemu`，采用自托管 `libvirt`、`qemu` 与 `libvirt` 客户端分离的结构。当前处于 `libvirt` 阶段，仓库已经包含会话、设备、工作区和执行机制，稳定发布仍需要证明新设备安装、真实多机并发和跨客户端接管。这里的目标与已实现的代码都会分别说明，不把产品方向当成验收结果。

多设备开发的关键问题

分布式执行首先要解决状态属于谁。聊天窗口关闭以后，任务是否还存在，设备是否仍在执行，另一台客户端能否继续项目，都不能只依靠上一轮对话文本。项目因此让常驻 □□□□□□□□□□ 持有项目、会话、事件、检查点和结果，□□□□ 客户端提供工具入口，□□□□□□□□ 负责在设备上执行，而不是让所有状态留在客户端内存里。

第二个问题是不能让多台设备任意覆盖彼此的工作。任务需要清楚的工作区，执行需要知道使用哪一份源码，回收的产物也需要能证明来自哪一轮操作。连接中断时，如果客户端简单重发命令，还可能重复启动带副作用的任务。因此，工作区隔离、稳定操作标识、恢复与结果校验都进入了项目设计。

技术架构与组件分工

[illegible]

设备注册信息不只有一个 □ □。它还包含平台、架构、能力、角色、最后活动时间和心跳状态。设备选择可以根据所需平台、能力、角色以及在线、空闲条件寻找合适目标。这样，调度才有机会区分 □ 这台机器可以跑 □ 和 □ 这台机器适合这项任务 □，也为后续多种设备分工保留入口。

传输支持本地、□□□□□□通道和□□□路由等形式，持久状态与连接方式相对分离。□□□相关路线保留主机密钥校验；远端□□□□□□通过配对与设备凭据建立连接。项目不要求由作者运营中心中继服务，设备和□□□□□□□□□□都由使用者自行部署，可选网络方案也不被写成必要条件。

会话、工作区与执行实现

□ □ □ 单写租约与会话接管 □ □ □ □ □ □ □ □ □ □ 暴露会话附加和释放入口，读取持久快照中的租约，判断当前所有者及过期时间。已有其他客户端仍持有有效租约时，不默认静默夺取；释放、等待过期或明确接管是不同处理路线。这个机制针对的是多个开发客户端同时写入同一会话的问题，而不是单纯让所有客户端看到相同聊天记录。

项目、会话、任务事件、检查点和结果共同组成恢复依据。客户端连接恢复后需要先理解当前状态，再决定继续或接管，不能把□重新连上服务器□直接等同于□所有执行都已恢复□。因此，当前□□□□□的验收中特别保留了跨电脑接管和执行中任务协调，而不只检查进程能够启动。

- 将获取的记录导出为 □ □ □ 或 □ □ □ □，整理不同记录的字段，避免页面查询结果只能临时查看。
- 可视化页面使用 □ □ □ □ □ □ □ □ □ □ 读取本地 □ □ □ □，按月份处理交易，展示收支汇总、商户排行、消费类别、星期与时段分布及余额趋势。

背景与目标

校园卡交易页面适合查看最近几笔记录，却不方便回顾较长时间的消费。连续翻页以后，仍很难知道哪个月花得多、主要在哪些商户消费，或者余额在什么时候发生变化。这个项目把查询与分析分成两段：先通过桌面工具取得个人流水并保存，再用网页查看数据的分布。

实现使用 □ □ □ □ □ □、□ □ □ □ □ □ □ □ 和浏览器页面。桌面端完成登录、查询进度与文件导出；分析端使用 □ □ □ □ □ □ □ □ □ □ 和 □ □ □ □ □ □ □ □ □ □，把本地 □ □ □ □ 组织成汇总与图表。这样，已经导出的数据可以反复查看，不必每次打开分析页都重新登录校园服务，也不需要把个人交易上传到一个新的分析服务器。

查询与分析需要解决的问题

流水通常是分页返回的。查询工具需要知道开始与结束页、当前获得多少记录、是否还在继续，以及中途失败或用户停止时如何反馈。若把所有网络请求都放在界面主线程，用户很难知道进程是在等待还是已经卡住。因此，查询过程需要与界面更新分开，并让每一批数据与当前进度同步。

交易记录也不能只按金额排序后直接解释。收入与支出、交易类型、商户、日期和余额分别回答不同问题。分析页面需要先整理有效记录，再按月份、星期和时段等维度聚合；分类规则只是查看数据的一种方式，并不能替用户判断每笔消费的真实用途。

桌面工具与网页分析架构

□ □ □ □ □ □ □ □ 界面内嵌校园卡网页，让用户自己完成登录，再读取当前会话用于查询。获取任务放在 □ □ □ □ □ □ □ □ 中，通过 □ □ □ □ □ □ □ □ 将页码进度、记录批次和完成状态传回界面。开始查询、停止查询、保存数据使用不同入口，避免网络函数直接承担按钮状态与提示文字等界面职责。

文件导出提供 □ □ □ 与 □ □ □ □ □。□ □ □ □ 遍历记录字段组织表头，方便在表格软件查看；□ □ □ □ □ 保留结构，适合后续网页读取。分析页通过 □ □ □ □ □ □ □ □ □ □ □ □ 读取用户选中的本地 □ □ □ □，并在浏览器中执行解析和聚合。这里的 □ □ □ □ 本地读取 □ □ 说明交易数据的处理位置，不代表页面引用的第三方图表资源全部离线可用。

分页获取与图表组织

□ □ □ □ 后台分页与进度反馈 获取线程按设置的分页范围调用查询函数，检查响应中的 □ □ □ □，累积记录并逐批发送给界面。根据 □ □ □ □ □ □ 和当前返回记录数估算总页数，在页与页之间保留短暂停顿，减少连续请求。□ □ □ □ □ 修改运行标志，循环在继续下一页前检查，给用户提供中途结束的控制入口。

程序同时区分一次请求失败与全部完成。失败时返回具体页码相关说明，完成时汇总取得的记录数。这个机制让用户知道哪些部分已经拿到，哪些部分可能需要重新查询。不过，分页循环中的边界条件仍需要运行验收，不能只凭 □ □ 支持所有页 □ □ 的界面选项就保证任意记录量都能完整下载。

数据处理与不同图表维度

网页先解析 □ □ □ □，再由 □ □ □ □ □ □ □ □ □ □ □ □ □ □ 组织交易，生成整体统计和月份数据。汇总卡展示金额和交易数等概要，月份选择用于切换对应图表。商户消费排行、交易类型、消费类别、星期和小时分布分别提供不同观察角度，余额趋势则保留时间变化，避免所有问题都挤进同一张图。

页面也对商户设置类别映射，用于观察预设消费类别。这样的分类便于读图，但规则可能无法覆盖所有商户，需要保留原交易信息供核对。切换或重建图表时先销毁旧实例，避免同一画布反复叠加；错误区域负责显示文件解析和图表数据问题，帮助区分没有数据与处理失败。

已完成的目的是与使用范围

项目已经连接登录查询、后台分页、文件导出和浏览器可视化，为个人校园卡记录提供了一条可以保存和反复分析的路线。它完成的是 □把流水拿出来，再看清消费结构□，而不是财务管理平台，也不尝试替用户自动判断合理消费或输出信用评价。

代码可以核对查询线程、导出函数和图表聚合入口，但完整记录、不同数据格式和页面适配仍需要真实运行验证。公开作品只展示通用功能与技术，不展示个人流水、会话 □ □ □ □ □ □ 或账号信息。数据文件属于使用者自己的记录，不作为网站的演示内容公开。

实践收获与后续方向

这个项目让我把数据采集和数据解释分开考虑。获取阶段关心是否完整、是否失败、能否停止；展示阶段关心分类是否清楚、月份是否对应、金额方向是否正确。后续可以补充分页边界、字段变化和重复记录处理的验证，并改进类别规则说明，让每一个图表都能回到它所依据的记录。

源码：

照片 000 提取与地图展示

个人项目

从带定位信息的照片中提取拍摄时间和坐标，生成可以在浏览器打开的地图，让照片的位置从文件属性中走出来。

技术 方向：□□□□□□ □□□□□□□□ □□□□□□□□ □□□□□□□□

□ 递归扫描照片目录，通过 □ □ □ □ □ □ 读取 □ □ □ □ ，整理拍摄时间、设备和 □ □ □ 信息；对打不开或缺少定位的图片跳过处理。

□ 将 □□□ 度分秒转换为十进制度，经过 □□□ 中间数据组织后生成 □□□□□□ 地图，在标记弹窗中展示坐标与时间。

□ 将相邻有效记录连接为可开关的路径图层，导出 □ □ □ □ 供浏览器查看。当前连线沿扫描顺序生成，尚未按拍摄时间排序，也未完整处理南纬、西经方向。

背景与目标

照片文件里常常保存了拍摄时间、设备和 `EXIF`，但这些信息藏在属性里，一次只能看一张。照片 `EXIF` 提取与地图展示项目想把这组信息拿出来：扫描一个目录，整理带定位的照片，再生成一份能在浏览器打开的地图，让照片与位置建立直观联系。

项目使用 `org.openstreetmap.osmosis`、`org.openstreetmap.p2pengine`、`org.openstreetmap.routematch` 和 `org.openstreetmap.util`。`osmosis.jar` 读取照片 `osmosis`，`osmosis.jar` 保存和转换中间表格，`osmosis.jar` 组织地图标记与路径图层。输出形式是 `osmosis`，适合离线保存文件后在浏览器查看，但地图底图是否能够加载仍取决于使用的资源和网络条件，不能把一个 `osmosis` 文件直接等同于完整离线地图。

文件信息与地图数据之间的差异

不是所有图片都包含定位信息。文件可能没有 `□□□□`，格式可能无法读取，`□□□` 字段也可能缺失。因此，脚本需要逐个处理并跳过没有可用信息的文件，不能把目录里的每张图片都假定为一个有效坐标点。拍摄时间、设备型号和定位信息也要与路径一起保留，方便核对数据来自哪张照片。

□□□在□□□□中以度、分、秒等结构保存，而地图需要十进制度。转换过程中还涉及南北纬与东西经方向，连接多个点时又涉及排序。当前代码完成了基础换算和连线，但没有完整处理半球方向，也没有先按拍摄时间排序。因此，它是一项照片定位展示实验，不是已经完成通用轨迹重建的地理系统。

技术组织与处理中间文件

入口沿着三个函数阶段执行：`getPhotoInfo` 整理照片信息，`convertCoordinate` 转换坐标，`generateMap` 生成地图。每一步通过 `yield` 衔接，中间结果能够分别查看，有助于判断问题来自图片读取、坐标转换还是地图生成。

目录扫描使用 `os.listdir()`，覆盖子目录中的常见图片后缀。`os.listdir()` 打开文件并读取 `os.path.splitext()`，通过 `os.path.splitext()` 和 `os.path.splitext()` 把字段标识转换成可读名称。带 `os.path.splitext()` 的照片被组织为文件路径、拍摄时间、设备品牌、型号和原始 `os.path.splitext()` 信息。读取失败时记录说明并继续处理其他图片，而不是让单个坏文件直接终止整个目录。

坐标提取与可视化实现

从 到十进制度 坐标转换阶段读取 ,对保存的 结构使用 解析,而不是直接执行字符串。随后提取纬度与经度的度分秒,按 度 分 秒 计算十进制度,保留其他照片字段并输出新的表格。当前代码仍需要补足方向参照和部分 类型的兼容,不能默认所有设备数据都符合一种结构。

□ □ □ 中间形式降低了函数之间的耦合，也暴露了格式转换的成本。原始数值结构经过字符串与表格再解析，遇到特殊数据类型时可能失败。因此，后续可以改为更直接的结构化传递，或者明确规范写入形式；当前作品展示的是已有管线，不把这种改进方向写成已经实现。

标记、弹窗和路径图层

□□□□□□ 创建地图，将有效记录转换为 □□□□□□，弹窗展示纬度、经度和拍摄时间。相邻有效点之间加入 □□□□□□□□，并放入默认隐藏的 □□□□□□□□□□□□，用户通过 □□□□□□□□□□□□ 决定是否显示路径。标记回答 □照片在哪里□，路径提供连续查看的辅助，两种信息不强制同时显示。

路径沿输入记录的处理顺序连接，因此同一条线可能反映目录遍历顺序，而不是旅行顺序。拍摄时间格式异常也可能影响展示。这些限制需要在使用时明确，不能将画出来的折线当作经过时间校准的真实行程。当前代码还会在主流程结束后删除中间 `□□□`，若要保留排查依据，需要调整这一行为。

完成目的与当前结果

项目完成了从照片目录提取信息、转换坐标到导出交互地图的基础路线，让原本藏在图片属性中的位置能够集中查看。源码里可以核对异常跳过、字段整理、标记弹窗和可开关路径图层，技术选择围绕一个明确的小需求组织，没有扩成无关的平台功能。

当前没有实现完整的方向转换、时间排序、轨迹纠偏或照片缩略图浏览，也不提供定位准确率数字。照片中保存的位置取决于原始 `location`，脚本只是整理与展示这些信息。公开作品不展示个人照片和坐标样本，避免把真实活动位置当作演示数据。

后续方向与实践收获

后续应优先补足 □ □ □ 方向和数值类型处理，按有效拍摄时间排序，并明确缺失或错误时间如何参与路径。再考虑调整地图初始视野、保留中间数据和添加可核对的来源入口。这个项目让我注意到，能生成一张地图和能正确解释一组地理数据是两件事，数据含义需要和可视化一起处理。

源码：

网页全景展示实验

个人实验

用一张全景图片在浏览器里建立可以转动查看的场景，探索普通网页中更直观的空间展示方式。

技术 方向：

集成 与 ，在页面容器中加载 ，配置自动旋转速度与启动时间。

- 提供全屏与设置等查看控件，完成图片资源和页面展示的基础集成；这是现有图形库的应用实验，展示重点是场景接入与交互配置。

背景与目标

一张平面照片只展示一个方向，全景图片则可以让观看者自己转动视角。我做这个网页实验，是想把全景资源放进普通浏览器页面，用一个直接的交互入口展示空间，而不要求访问者安装专门的查看软件。它适合校园介绍、场景展示或个人页面中的小型视觉体验。

项目通过 `Three.js` 集成 `Mapbox GL JS` 和 `React`，页面以一个铺满视口的容器承载全景。这里使用的是现有图形库提供的场景与交互能力，我完成资源加载、容器和参数配置，不把第三方引擎写成自己研发的 `JS` 框架。仓库名包含 `react`，但本案例的主要实现入口是全景页面，不声称原创补间动画引擎。

图片资源与页面交互的问题

全景展示首先需要合适的图片资源。普通照片并不会因为放进全景查看器就具有完整空间信息，输入图片的投影和尺寸会影响最终效果。页面容器也需要明确宽高，否则库已经初始化，用户却可能看不到正确画面。因此，实验先把资源路径、脚本加载和容器尺寸这几件基础事情连接起来。

自动旋转提供初次进入时的观察方向，但过快会让人难以看清，立即启动也可能干扰主动操作。页面因而配置旋转速度与激活等待时间，让图像在展示与查看之间保持基本节奏。实际观看舒适度仍需要浏览器与设备上检查，不能从一个配置值直接推定所有用户的体验。

技术选择与场景组织

`index.html` 引入 `Three.js` 和 `Mapbox GL JS`，`index.js` 将 `Three.js` 和容器设置为完整视口。`index.js` 通过 `Mapbox GL JS` 指定 `Mapbox`，再创建 `Mapbox` 并传入图片路径，最后将全景加入 `Three.js`。图形库负责底层渲染与查看交互，页面负责把素材和展示环境交给它们。

这种实现适合验证一张全景能否进入网站，结构短、入口直接。它尚未引入组件框架或复杂资源管理，没有把基础展示包装成多个场景的内容平台。独立页面也便于检查脚本路径与图片是否能够加载，之后再决定如何嵌入其他网页。

已实现的界面与动画配置

`index.js` 配置了自动旋转、启动等待和旋转速度，同时指定全屏、设置等控件入口。用户可以进入全屏查看，自动旋转则帮助展示画面不同方向。项目没有自行实现这些控件背后的渲染算法，展示重点是对已有能力的选择与接入。

图片通过相对资源路径加载，页面添加全景对象后由查看器呈现。当前只有一个 `index.html`，没有实现多场景跳转、热点导航、资源预加载进度或错误重试界面。把这些未实现内容留在后续方向中，能够让读者区分一个可运行的基础示例和完整全景导览产品。

完成目的与当前范围

项目已经完成浏览器全景展示的基本闭环：加载库、建立容器、接入图片、配置观看控件和自动旋转。它达到的目的是探索网页中的空间展示形式，为后来在网站中使用图形与交互提供实践基础，不属于生产级数字孪生或精细三维建模项目。

已有代码说明展示路线，但图片质量、加载耗时、移动端手势、`JS` 兼容及动画舒适度还需要实际检查。因此不使用帧率、模型精度或设备覆盖率数字，也不把一张全景图片描述成带几何和物理结构的 `JS` 模型。

后续方向

如果继续完善，我会先处理资源加载状态、失败提示和不同屏幕尺寸，再考虑多个场景之间的切换与热点说明。自动旋转还应尊重用户操作和减少动画偏好。这个实验的收获是明确素材、图形库和网页交互各自的职责，让视觉效果为内容服务，而不是只让页面一直转动。

源码：[https://github.com/maoyuanliang/react-mapbox-gl-3d](#)

页面也需要正确引用图片与脚本，避免本地目录调整后资源失效。关联仓库中的全景页面尝试了另一种图片展示方式，但仍属于图形库的集成实验。它们让前端练习从静态排版延伸到浏览行为，同时没有把已有插件的算法算作原创代码。

完成目的与作品范围

这些练习完成了校园主题内容、页面导航、图文布局与组件交互的组织，积累了前端文件结构、资源管理和浏览器调试经验。作品可以通过仓库检查具体页面和初始化代码，说明我如何把已有素材与组件组合成可浏览的页面。

目前不展示访问量或加载性能数字，也不声称建立完整校园管理系统。部分页面使用模板与第三方资源，因此对外描述明确采用「设计与集成练习」，避免把整个页面生态中的通用能力都归为个人原创。它的价值在于实现和理解完整页面，而不是以功能数量代替技术深度。

后续改进方向

继续完善时，可以先统一语义结构、资源引用与小屏布局，再检查图片体积、键盘操作和减少动画偏好。对于轮播和灯箱，应保证正文仍能独立阅读，主要入口不依赖动画才能发现。这个项目让我认识到，网页效果要和信息层次一起设计，组件多并不自动意味着页面更好用。

源码：[https://github.com/wwsjw/wwsjw.github.io](#)

「视觉学习项目」游戏画面识别与数据集处理

视觉学习项目

以游戏画面为对象整理检测数据，完成视频抽帧、标注格式转换和训练集划分。通过一个具体场景练习视觉数据准备与训练迭代。

技术方向：[https://github.com/wwsjw/wwsjw.github.io/tree/main/visual-learning](#)

- 使用「ffmpeg」按视频帧率计算抽帧间隔，批量提取画面并连续编号，作为后续标注的数据来源。
- 将「yolo」标注转换为「coco」的归一化中心坐标与框尺寸，对未登记类别给出提示，并按固定随机种子划分训练与验证图片、标签。
- 仓库保存了多轮训练目录、配置和标注数据；该项目的对象是游戏画面，与工业零件识别实践分开展示，不从训练文件直接推定现场准确率或实时性能。

背景与目标

这个视觉项目以游戏画面为对象，练习从原始视频到检测数据集的准备过程。图像检测需要的不只是一个训练命令，还需要能够持续获得图片、把标注转换成统一格式，并明确哪些数据用来训练、哪些用来验证。我围绕这些环节编写脚本，整理游戏中的角色、场景对象和界面元素。

主要源码在「visual-learning」仓库的「game」分支，「game」只保留了简短说明。项目使用「ffmpeg」、「yolo」、「coco」、「yolo」标签格式和「coco」数据集格式，仓库中还有多轮训练目录与配置。本案例明确属于游戏画面实践，不与工业零件、缺陷检测和机器人项目混为同一份证据。

数据准备需要解决的问题

视频是连续内容，逐帧保存会产生大量相近图片，完全手动截图又很难控制间隔。因此，抽帧需要依据视频帧率，把时间间隔换算为帧数，再给图片统一编号。不同视频还需要检查是否能够打开、帧率是否有效，否则后续取模和采样可能出现错误。

标注工具产生的「yolo」与「coco」训练所需文本不同。类别名需要映射为类别序号，框坐标需要按图片宽高归一化；训练和验证还必须同时保留图片与对应标签。若图片已经划分，标签却没有同步复制，数据集表面完整，训练时仍可能缺少监督信息。

工具选择与处理链路

使用 ffmpeg 的 ffmpeg 批量读取视频，按帧率和设定秒数计算采样间隔，再调用 ffmpeg 保存图像。使用 ffmpeg 解析 txt 文件，将类别、类别和边界框转换成 json 文本。使用 ffmpeg 使用 ffmpeg 按固定随机种子划分图片，再通过 ffmpeg 将图片与标签复制到对应目录。

这三个脚本把采集、格式转换和数据划分分开，便于分别检查输入与输出。代码中的一些目录仍是特定机器上的路径，因此这里展示的是个人数据准备流程，不把它描述成无需配置即可处理所有数据的通用训练平台。迁移到其他设备时，需要先整理路径和数据说明。

具体实现与数据约束

按时间间隔抽帧 视频抽帧先取得 帧率，再将配置的秒数转换为 帧数。读取每一帧时检查帧序号，达到间隔就保存，并使用跨视频连续计数生成名称。代码检查视频是否打开和 帧率 是否为零，完成后释放 帧率，保留处理数量与时间的控制台信息。

这种采样便于得到间隔一致的候选图片，但不保证图片内容足够多样。连续游戏画面仍可能重复，类别也可能分布不均。后续还需要查看样本内容、去重和补足稀少目标，不能只用抽出的图片总数判断训练数据质量。

帧率到帧数的标注转换

转换脚本读取图片尺寸和每个 帧率的类别、帧率、帧率、帧率、帧率，计算框中心与宽高，再分别除以图片尺寸，输出类别序号和归一化坐标。未知类别输出提示并跳过，避免把不在类别表中的名称强行映射到错误编号。类别表的顺序也需要与训练配置保持一致。

这个过程体现了数据格式转换中最重要的约束：同一个目标在不同文件里应该有同样的含义。图片尺寸缺失、框越界或类别不一致都需要进一步检查，当前脚本没有替代完整数据集质量验收。因此不把标签能够写成文本等同于标注已经完全正确。

训练与验证目录划分

按 训练与验证 的比例划分训练和验证图片，设置 训练与验证 使相同输入下的结果更容易复现。复制前检查标签是否存在，没有对应文本就提示并跳过，图片与标签分别进入 训练与验证、训练与验证 的 训练与验证 和 训练与验证 目录。这保证目录布局与训练使用方式一致。

不过，当前随机划分以图片为单位。来自同一段视频的相似帧可能进入训练和验证两侧，影响对泛化能力的判断。更严格的后续验收需要考虑按视频或场景划分。仓库的多轮训练记录可以说明进行了迭代，但没有重新执行训练和检查完整结果时，不据此写出准确率或实时推理性能。

完成的目的与阶段结果

项目完成了视频抽帧、标注格式转换和训练验证目录整理，形成能够继续训练的基础数据路线。它帮助我把视觉项目从 使用一个现成模型 延伸到数据准备和迭代组织，理解模型效果与样本质量、类别定义和验证方式之间的关系。

当前没有将游戏识别结果直接作为工业视觉成果，也不声称已经完成自动化游戏控制、强化学习策略或现场设备部署。已有脚本与训练目录是可以核对的交付，其他应用能力需要独立实现和证据，不能从仓库名称或目标类别推定。

后续方向与实践收获

后续应补充路径参数化、标注校验、重复样本处理和按场景划分的验证，再讨论模型与部署优化。对视觉实践来说，数据是否具有代表性、标签是否一致、验证是否存在泄漏，比简单增加训练轮次更值得先检查。这也是本项目留下的主要工程经验。

源码：[https://github.com/1024x/1024x](#)

专业技能

编程语言：Python、Java、JavaScript、C++、Go、PHP。

前后端开发：Spring Boot、Django、Vue.js、React、Node.js、MySQL、Redis、Kubernetes、Docker、Linux、Git。

移动端与界面：Flutter、Swift、Kotlin、Android Studio、Xcode、Figma、Sketch、布局、UI/UX、交互设计、原型设计、敏捷开发、微信小程序、鸿蒙、小米快应用和小爱同学生态。

数据库与服务：MySQL 业务建模、表结构与索引设计、SQL 查询优化、数据迁移、备份及安全策略；接触 Redis 和微服务。

视觉与模型：OpenCV、TensorFlow、PyTorch、YOLO、Keras，数据集采集与标注、模型训练验证、推理和本地部署；工业零件识别、定位、缺陷检测及质量检查实践。

大模型与自动化：LangChain 设计与优化、RAG、Agent、Prompt Engineering、模型集成与本地部署；了解并实践增量预训练、指令微调、强化学习、LoRA 微调及视觉模型推理文件应用；LangChain、LangGraph、LangServe。

开发与运维工具：Jenkins、GitLab CI/CD、Ansible、Terraform、Kubernetes、Docker、Prometheus、Grafana、Istio、Vault、宝塔面板、Nginx、Apache、Tomcat。

网络与硬件：TCP/IP、路由配置、局域网及企业共享盘、Wi-Fi、5G 通信；服务器选型与 RAID 配置、显卡选型、显存和驱动环境；嵌入式 Linux、RTOS 相关实践。

现场与办公：精益生产、装配工艺、ISO 9001、质量数据整理、8D 工具及质量体系资料；PPT、Excel、Word、Visio、AutoCAD、SolidWorks。

语言：英语流利，能够阅读技术文档。

补充经历与个人特点

交付方式：从需求分析、架构与界面设计、开发调试到部署、用户反馈、版本维护和文档交付，重视完整流程和可维护性。

辅助开发：使用 Copilot、Cursor、GitHub Copilot、CodeGeeX 辅助代码生成、排错、文档整理与自动化，把业务需求转化为明确的开发需求和开发任务。

补充经历：独立开发并上线微信小程序；开展云服务器与 CDN 网站运维、企业共享盘、网盘云端平台文件上传下载等实践。

校园经历：校级创新创业项目获奖经历，项目涉及基于 AI 的校园智能服务系统。

持续学习与兴趣：关注 AI 前沿工具、大模型、Agent、RAG、模型微调、本地部署与硬件环境；兴趣包括智能硬件和智能家居。

职业方向

优先关注 AI 应用开发、全栈开发、大模型应用开发与自动化工具相关岗位，同时关注前端开发、制造业与质量数字化、技术支持和运维；期望工作地点为安徽省内。