AI视频流水线技术架构与工程实现详解

一段 AI 视频流水线的技术架构与工程实现详解

> 这篇文章写给真正要做"AI 口播视频批量生产"的人：不是讲概念，而是讲**可复现的工程方案**——架构怎么拆、数据结构怎么设计、两个核心难题（读对音 / 不换脸）在工程上到底怎么解决、以及实际踩过的坑。
>
> 全篇不含任何密钥、内部路径或具体配置值；示例 ID 与地址均为占位符。

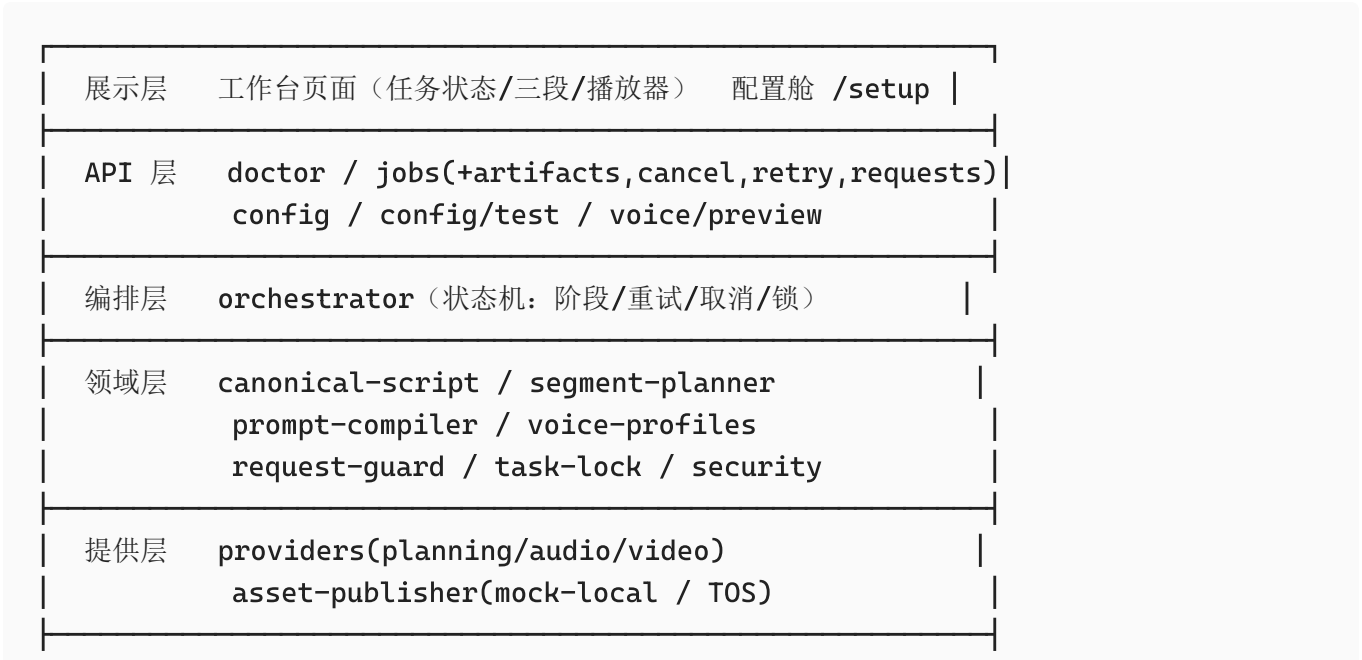
〇、参考价值声明

本文是下面这套系统真实实现的提炼，可直接作为你搭建同类系统的参考：

- **技术栈**：Next.js 14（App Router）+ React 18 + TypeScript(strict) + Zod + Vitest + FFmpeg/FFprobe
- **本地优先**：无数据库、无 Redis、无 Docker，任务与产物全部落本地文件
- **默认零成本**：不配任何真实模型也能跑通全流程（演练模式），真实能力按序点亮

一、总体架构

1.1 分层



```
└  存储层    任务JSON + events.jsonl + 媒体产物 + 配置JSON    ┘
```

1.2 核心数据流

```
idea + script
  → Canonical Script（解析人物/性别/台词）
  → Master Audio Prompt ──→ Master Audio（定节奏/时长）
  → 时长校准 ──→ 三段拆分（严格覆盖全文）
  → 每段 Audio Prompt ──→ 每段音频 ──→ 尾部处理 ──→ ffprobe 精确时长
  → 每段 Video Prompt ──→ Segment 1 成片（锚点）
                            ↳ Segment 2/3 以 Segment1 为参考，并发生成
  → 后处理混流（音视频对齐）──→ 三段成品
```

1.3 模块职责（为什么这么拆）

模块	职责	为什么独立
orchestrator	任务状态机、阶段调度、锁、取消、重试	编排是"流程"，不该和"怎么调模型"耦合
canonical-script	人物/性别/台词解析，纯函数	可单测、可复用（前后端都用）
segment-planner	三段拆分 + 覆盖率校验	拆错段会直接毁掉视频，必须可验证
voice-profiles	动态角色声音档案 + 提示词	声音设计是业务规则，独立演进
providers	各模型厂商适配层	换模型只改这一层，不动编排
asset-publisher	产物发布（本地/TOS）+ 预签名	发布方式可插拔
request-guard	幂等 + 预算 + 哈希	防重放、防超支
security	统一脱敏	密钥/URL 不落日志是硬要求

二、发音正确性：音频参考的完整链路

2.1 失败模式：视频模型的"隐式语音合成"

视频生成模型吃 `text + 画面` 输入时，内部会自己把文本"读"出来再驱动口型。这套隐式 TTS 在以下场景必然出问题：

- 金融术语（等待期/保障范围/投保条件）——逐字直读或断错句；
- 数字与金额（3.5% / 二十年 / 保额五十万）——最容易崩；

- 多音字与同音词——无领域知识，选错读音；
- 语速/重音/停顿不可控——"像机器念稿"。

结论：发音是视频模型最弱的环节，而恰是内容最不能错的环节。

2.2 解法：把"发音"和"画面"解耦

```
TTS（语音合成）──► 生成"已读对"的音频（术语/数字/重音可控）
    │
    ▼
音频发布（本地/TOS）──► 得到可访问的音频引用（预签名 URL）
    │
    ▼
视频模型：不再念台词，只"对音频张嘴"（口型+画面）
```

视频请求里的音频参考，真实结构类似（占位示例）：

```
{
  "model": "<视频模型ID>",
  "content": [
    { "type": "text", "text": "<画面提示词>" },
    {
      "type": "audio_url",
      "audio_url": { "url": "<音频的预签名URL>" },
      "role": "reference_audio"    // 关键：以音频为条件
    }
  ],
  "generate_audio": false,         // 不再让视频模型自己出声
  "ratio": "9:16",
  "duration": 3
}
```

2.3 为什么有效（本质）

- 台词不是视频模型"念"出来的，而是被一份**已经念对的音频**锁死；
- 视频模型退化为"配音的嘴"：只负责口型、表情、画面；
- 读音正确性从"视频模型的弱项"（隐式TTS）转移到"语音合成的强项"（显式TTS）。

2.4 这条链路里的真实工程约束

约束	原因	处理
音频先转码成标准格式再发布	统一格式、可被服务端校验	WAV(PCM) 转码 + SHA-256 校验
音频要"可被视频模型访问"	视频任务在云端/远端跑，需要 URL 而非本地路径	发布到对象存储，生成预签名 URL
预签名 URL 的 TTL 必须 > 视频任务超时	视频生成是异步长任务（分钟级），URL 过期则任务失败	TTL 与任务超时一起配置，测试里校验
音频内容必须精确对应当前段	参考错了，口型就错了	每段音频由"本段台词 + 本段声音档案"独立生成

2.5 边界（诚实说明）

这套方案成立的前提是：视频模型**真正支持"以音频为条件的口型驱动"**。选型必须验证——它是对给定音频对齐口型，还是自己重新合成。选错模型，方案失效。

三、人物一致性：首段锚点 + 后段并发

3.1 失败模式：多段独立生成 = 每段一个新角色

把一条口播拆成三段分别生成，模型对"这个人长什么样"没有记忆 → 圆脸变方脸、服装跳戏、光线不一致。批量越努力，一致性越差。

3.2 解法：用"参考"传递一致性，而非"让模型记住"

```
Segment 1 先成片 ──► 确立角色外观/服装/场景/画风（锚点）
        │
        ├──► Segment 2：参考 Segment 1 画面 + 本段音频 ──► 并发
        └──► Segment 3：参考 Segment 1 画面 + 本段音频 ──► 并发
```

视频请求里的画面参考，真实结构类似：

```
{
  "content": [
    { "type": "text", "text": "<画面提示词>" },
    { "type": "audio_url", "audio_url": { "url": "<本段音频URL>" }, "role":
"reference_audio" },
    {
      "type": "video_url",
      "video_url": { "url": "<Segment 1 成片的预签名URL>" },
```

```
      "role": "reference_video"   // 关键：画面参考
    }
  ],
  ...
}
```

3.3 为什么后段可以并发

依赖分析：

- Segment 2 **只依赖** Segment 1（画面参考 + 自己的音频），不依赖 Segment 3；
- Segment 3 同理，不依赖 Segment 2；
- 二者彼此无关 → 可并行（真实实现用并发调度同时跑两段），省掉串行等待时间。

3.4 提示词层的双保险

画面参考是"工程保底"，提示词是"语义约束"：

- 谁说话谁张嘴，未说话人物闭口、自然倾听；
- 人物/服装/场景/画风连续；
- 只表现当前段，不重复上一段、不提前进入下一段。

3.5 数据模型支撑

```
type FinanceSegmentPlan = {
  index: 1 | 2 | 3;
  exactText: string;            // 本段原文，严格来自全文切片
  dialogueLineIds: string[];    // 命中的台词行
  speakerIds: string[];
  audioDurationMs?: number;
  videoDurationMs?: number;
  referenceVideoPath?: string;  // Segment 2/3 指向 Segment 1 成品
  audioPrompt?: string;         // 继承角色声音档案
  videoPrompt?: string;
  status: 'pending' | 'running' | 'completed' | 'failed';
};
```

四、关键数据模型

4.1 台词与人物（Canonical Script）

```ts
type FinanceCharacter = {
  id: string;            // speaker_<label>
  displayName: string;
  gender: 'male' | 'female' | 'unknown';
  ageRange: string;
  role: string;
  voiceDescription: string;
};

type FinanceDialogueLine = {
  id: string;            // line_<n>
  speakerId: string;
  text: string;
  order: number;
  // 原文偏移 + 规范化偏移，用于三段切分与覆盖率校验
  originalStartOffset: number;
  originalEndOffset: number;
  normalizedStartOffset: number;
  normalizedEndOffset: number;
  boundaryType: 'sentence' | 'clause' | 'line';
};
```

台词是"带偏移"的结构化数据——**这是后面所有切片、校验、对音准的前提**。

4.2 阶段状态与请求记录（可靠性的地基）

```ts
type FinanceStageState = {
  status: 'pending' | 'running' | 'completed' | 'failed' | 'cancelled';
  attempt: number;                // 阶段级重试次数
  providerRequestId?: string;
  providerTaskId?: string;        // 视频任务的远端ID，用于续跑
  inputHash?: string;             // 幂等用
  outputHash?: string;
  error?: { code: string; message: string; retryable: boolean };
};

type FinanceProviderRequest = {
  id: string;
  provider: string;
  model: string;
  stage: string;
  segmentIndex?: number;
  idempotencyKey: string;         // 防重放
  inputHash: string;
  result: 'submitted' | 'completed' | 'failed' | 'cancelled';
```

```
  artifactPath?: string;
  requestSnapshot?: unknown;        // 脱敏后
  responseSnapshot?: unknown;       // 脱敏后
};
```

4.3 动态角色声音档案

```
type CharacterVoiceProfile = {
  characterId: string;
  speakerName: string;
  gender: 'male' | 'female' | 'unknown';
  ageImpression?: string;
  personality?: string;
  accent?: string;
  baseVoicePrompt: string;          // 自动生成的文字声音描述
  userVoicePrompt?: string;         // 用户覆盖，优先
  referenceSpeakerId?: string;      // 可选参考底模（不固定男女）
  speechRate?: number;
  pitchRate?: number;
  loudnessRate?: number;
  source: 'auto' | 'user';
};
```

五、关键算法

5.1 人物与性别解析

两级策略，纯规则、确定性、可单测：

1. **显式标签**：台词行 `标签：内容` 提取 `speaker_<标签>`；标签直接含"男/女/顾问/客户"则直接判性别；
2. **最近性别词**：在构思（idea）同一分句里，找离该标签最近的性别词（男性/女性/男士/女士…），取最近者。

```
// 伪代码
for clause in idea.split(/[。！？；;\n]/):
  if clause.contains(label):
    return nearestGenderWord(clause, labelIndex)
return 'unknown'
```

这套规则让"男顾问 + 女客户"这类常见写法稳定判对，且不依赖模型。

5.2 三段拆分（穷举最优切点）

```
1. 候选边界：每句台词结束 + 句末标点 + 从句逗号，每个边界带质量分
2. 枚举两个边界 (a, b) 组成三段的切法
3. 过滤：任一段预估时长 > 上限(15s) 则淘汰
4. 打分：a.quality + b.quality − 三段时长不均衡惩罚
5. 取最高分 → 得到三段 exactText
6. 覆盖率校验：三段拼接 === 规范化台词全文（硬校验，失败即报错）
```

关键点：**切点选在"语义边界"（句子/从句），而不是任意字符**；三段时长由"母版音频时长"做全局校准，避免第一段超长。

5.3 动态语音档案的确定性生成

- 从构思里解析：年龄感（30岁/中年/青年）、性格（沉稳/耐心/紧张）、角色（顾问/企业主）、场景（办公室）、口音（普通话/粤语…）；
- 由性格推导语速/音高/响度（沉稳→稍慢稍低，紧张→稍快稍高）；
- 组装成中文文字声音描述：

> 30岁左右的普通话男性声音，专业、克制、有耐心，像一名顾问在办公室说话，语速适中，避免播音腔。

- **同性别区分**：当两个角色都没给出足够信息时，用 `stableHash(characterId)` 从多套人格池确定性选取——**两个男性也能声音不同**，且同输入结果稳定。

三种音色策略：

策略	行为
text_prompt_only（默认）	只发文字声音描述，**不发送** references
manual_reference	用户指定角色 → 发送该 reference 的 speakerId
auto_reference	按 性别+年龄+风格标签+描述 综合评分，**性别不是唯一条件**；无合适匹配则回退纯文字

TTS 请求的真实结构（占位）：

```
{
  "model": "<TTS模型ID>",
  "text": "<本段台词>",
  "text_prompt": "<该角色的文字声音描述>",
  "audio_config": {
```

```
    "format": "wav",
    "sample_rate": 48000,
    "speech_rate": 0,    // 来自 CharacterVoiceProfile
    "pitch_rate": 0,
    "loudness_rate": 0
  }
  // references 仅当有 referenceSpeakerId 且非 text_prompt_only 时才加
}
```

六、工程可靠性

6.1 任务锁（防并发重复执行）

- 每个任务一个 `runner.lock` 文件：写入 `{ pid, token, heartbeatAt, stage }`；
- 心跳按 `TTL/3` 间隔续写，超时判定 `now - heartbeatAt > TTL`；
- 常规冲突直接拒绝（`JOB_LOCKED`）；**重试/接管**允许在锁过期后接管；
- 释放时校验 token 归属，避免误删他人锁。

6.2 取消（可中断的优雅取消）

```
标记文件 cancelled ──► 编排层每次阶段切换都检查
                   ──► AbortController 中止 FFmpeg/HTTP
                   ──► 对视频任务调用远端 DELETE
```

6.3 幂等与续跑（防双收费）

这是真实生产最容易被坑的点：

- 每次真实请求生成 `idempotencyKey = hash(jobId|stage|segment|provider|model|payloadHash|attempt)`；
- 请求历史里按 `idempotencyKey` 查重：已完成则直接复用产物；
- **视频任务中途失败重试时，不重新创建远端任务**，而是用 `providerTaskId` 续跑轮询——避免"重试一次 = 再付一次钱"。

6.4 请求预算

- 每任务真实请求上限（默认 8：规划1 + 母版音频1 + 三段音频3 + 三段视频3）；
- 内存预留计数 + 历史计数双校验，超限即拒绝，**不会因为并发调度偷偷超支**。

6.5 阶段级重试

- 每阶段独立状态、独立 attempt；重试只重跑失败的阶段；
- 失败的请求记录保留，重试可作为 `retryOf` 关联，审计可查。

6.6 脱敏与本地化

- 统一脱敏：Key / Bearer / X-Api-Key / 签名URL查询参数 / 配置文件密钥；
- 任务 JSON 与请求记录存的是**脱敏后**的 snapshot；
- 配置、测试状态、任务产物全部留在本地私有目录，不进 Git、不进浏览器包、不进公开接口。

七、媒体后处理

```
每段音频：生成 ──► 尾部裁剪(时长-1s 保护) ──► ffprobe 精确时长
每段视频：生成 ──► 混流（视频copy + AAC音频 + -shortest）
              ──► ffprobe 校验（h264+aac，时长对齐）
```

- 尾部留 1 秒安静保护区：防止生成模型"收尾声音被截断"；
- 精确时长来自 ffprobe，而不是估计值——三段视频时长因此严格对齐；
- 视频固定 9:16 / 480×854 / 25fps，参考链路稳定。

八、踩坑与权衡（真实经验）

问题	现象	我们的处理	教训
读错音	术语/数字念错	音频参考，发音交给 TTS	发音要"锁死"而不是"期望模型读对"
人物跳戏	每段换脸	首段锚点 + 参考视频 + 连续性提示词	一致性靠"参考"传递，不靠模型记忆
URL 过期	视频任务中途失败	TTL 必须 > 视频任务超时，测试校验	异步任务的"引用有效期"要算进架构
重试双收费	视频重试=再付一次	idempotencyKey + providerTaskId 续跑	异步任务必须支持"续跑"而非"重发"
并发超支	2/3 并发导致请求数翻倍	预算双校验（历史+预留）	并发和预算要一起设计
一致性 vs 自由度	参考太强导致动作僵硬	锚点用中性画面 + 动作幅度控制	一致性是权衡，不是绝对值

九、这套方案里可复用的 5 个模式

1. **分层解耦**：发音(TTS) / 画面(视频模型) / 编排(状态机) 各司其职——换任何一个"引擎"不动其他层。
2. **锚点 + 参考**：用"显式参考"传递一致性（画面参考、音频参考），而不是指望模型记忆。
3. **幂等续跑**：`idempotencyKey` + `providerTaskId` 续跑，是异步付费任务不出钱事故的底线。
4. **阶段可控**：每阶段独立状态/重试/取消，出错了只回滚一小步。
5. **演练先行**：默认 mock 也能跑通全流程；真实能力按序点亮、单独验证——让"采购决策"和"工程验证"解耦。

十、展望

- **人感**：从"对的嘴型"到"对的微表情/眼神/肢体"——口型正确只是及格线；
- **一致性**：锚点 + 条件生成覆盖大动作、多机位、多情绪；
- **长内容**：从"三段 15 秒"到结构化长视频自动拼接；
- **实时化**：从批处理到"边改边看"的实时预览；
- **合规自动化**：流水线内嵌内容审核与留痕，让"可追溯"成为系统能力而非人工整理。

本文是一段真实实现的 AI 视频流水线技术参考。面向企业/商业视角见《当短视频生产成为企业的水电煤》，博客向技术解说见《读对音不换脸-企业级AI视频流水线技术拆解》。