河马的CC商业实战 · 会员专属资料 #1

AI 视频制作工具百科 · 实操手册

代码即剪刀

用 Vibe Coding 玩转视频创作

从零开始，用对话的方式「写」出专业视频

涵盖 HyperFrames / Remotion / video-use / FFmpeg MCP / Claude Code 完整工具生态

河马的CC商业实战 出品

2026年8月 · 会员专属版

会员专属资料 · 第 1 期

代码即剪刀

用 Vibe Coding 玩转视频创作

出品 河马的CC商业实战（知识星球）

版本 2026年8月 · 会员专属版

读者 视频创作者、内容从业者、所有想用 AI 提效的非程序员

> 本资料为「河马的CC商业实战」星球会员专属，每周/不定期更新。欢迎转发给需要的朋友——如果他们想持续收到这个系列，封底有入口。文中工具版本与数据以写作时为准，开源世界变化快，上手前以各项目官方页面为准。

河马卷首语

星友们，这是「会员专属资料」的第 1 期。

我一直觉得，AI 时代最值钱的本事，不是会剪视频，而是会「描述你要什么」。这本手册讲的就是这件事：不写一行代码，用对话把视频「说」出来——行话叫 Vibe Coding，我给这个系列起了个更直白的名字：代码即剪刀。

这本手册不薄，但别被吓到，它不是让你从头读到尾的。我的建议是：先翻第 18 章的工具选型决策树，找到你的场景，再倒回去看对应章节；附录 B 的提示词模板可以直接复制用，附录 C 的报错速查留着救急。

用好它。做出第一个「说」出来的视频，记得来星球里晒。

—— 河马，2026年8月

目 录

PART ONE

认知篇

视频制作正在被重新定义

第1章 从时间线到对话框——视频剪辑的第三次革命

第2章 三件套——GitHub项目、Skill、MCP 到底是什么关系?

第3章 你的AI视频工作台——环境准备全攻略

PART TWO

兵器篇

AI视频制作工具全景

第4章 HyperFrames——用写网页的方式做视频

第5章 Remotion——用React组件"编程"视频

第6章 Revideo——代码驱动视频的另一种选择

第7章 video-use——对话式视频编辑的"剪辑师"

第8章 FFmpeg + MCP——视频处理的瑞士军刀

第9章 Skill 技能包——给AI装上"剪辑师大脑"

第10章 AI编程助手选择指南——Claude Code vsCursor vs Kiro

PART THREE

实战篇

从素材到成品的完整工作流

第11章 口播博主实战——录一段话，AI帮你变成精品视频

第12章 知识分享实战——文稿/PPT变动画讲解视频

第13章 产品演示实战——让产品自己"说话"

第14章 批量生产实战——一套模板，量产100个视频

PART FOUR

进阶篇

从能用到好用

第15章 提示词的艺术——怎么"说"AI才能剪出你想要的视频

第16章 工作流编排——打造你的自动化视频流水线

第17章 质量把控——AI剪的视频如何避免"AI味"

第18章 工具选型决策树——我该用什么组合?

PART FIVE

附录

速查手册 · 模板库 · 报错 · 资源 · 术语

附录A 全工具安装配置一站式速查手册

附录B 提示词模板库（可直接复制使用）

附录C 常见报错与解决方案 Top 20

附录D 推荐学习资源

附录E 术语表（中英对照）

前言

如果你是一位视频创作者，这本书可能会改变你的工作方式。

2025年，前特斯拉AI总监 Andrej Karpathy 提出了一个概念——Vibe Coding。他说，这是一种新的编程方式：你不再一行行手写代码，而是用自然语言描述需求，让AI来完成实现。开发者从"写代码的人"变成了"描述需求的人"。

这个概念最初是为软件开发提出的。但很快，创作者们发现了一个更有意思的用法：Vibe Coding 不只可以用来写软件，还可以用来"写"视频。

想想看，视频的本质是什么？是一帧帧连续的画面，是精确到毫秒的时间轴，是音频波形和视频画面的同步。这些东西，用代码可以精确描述。而用AI来写代码，你只需要会说话就行。

你不再需要学习 Premiere 的复杂时间线，不需要记忆 After Effects 的关键帧快捷键，不需要在剪映里一帧一帧地调整。你只需要打开一个AI编程助手，用聊天对话框告诉它："帮我把这段20分钟的口播剪成3分钟精华，去掉所有口癖，加上字幕和背景音乐。"然后，它会自己完成。

这听起来像科幻小说，但在2026年的今天，这已经是实实在在可以使用的技术栈。

本书将带你系统了解这套AI视频制作的全新工作流。我们会从底层原理讲起，逐一介绍生态中的每个工具——从用HTML描述画面的 HyperFrames，到用React组件编程视频的 Remotion，再到能让AI直接操控剪映的 Skill 技能包，以及充当"翻译官"角色的 MCP 服务。每个工具都会给出从零上手的完整步骤，以及可以直接复制使用的提示词模板。

这不是一本编程教材。这是一本写给视频创作者的工具书。你的技术背景不重要——只要你愿意尝试新的工作方式，这本书就能帮到你。

让我们开始吧。

—— 河马，2026年5月

PART ONE

认知篇

视频制作正在被重新定义

第1章 从时间线到对话框——视频剪辑的第三次革命

第2章 三件套——GitHub项目、Skill、MCP 到底是什么关系？

第3章 你的AI视频工作台——环境准备全攻略

第1章 从时间线到对话框——视频剪辑的第三次革命

1.1 那个让剪辑师崩溃的下午

想象一下这个场景：

你是一位知识分享类博主，刚录制完一段20分钟的教学口播。内容很棒，但过程中你说了127个"嗯"、83个"啊"、还有6处说错了重来的段落。现在你需要把它剪成一段3分钟的精华视频。

你打开 Premiere Pro。导入素材，把视频拖到时间线上。然后开始一帧一帧地听，找到每一处口癖，用剃刀工具切开，删掉，再拼接回去。加字幕？需要先导出音频，扔到某个字幕软件里识别，生成SRT文件，再导回来调整位置、样式、动画。加片头片尾？打开After Effects，找到之前的模板，修改文字，渲染，再导回Premiere。整个过程，4个小时过去了。

这是2020年代视频创作者每天都在经历的事情。手动剪辑时代，视频制作的核心技能是"操作软件"——你要像钢琴家熟悉琴键一样熟悉时间线上的每一个快捷键。

1.2 三个时代的进化

视频剪辑技术的发展，大致可以分成三个时代。

第一时代：专业剪辑时代（2000-2018）

代表工具：Premiere Pro、Final Cut Pro、Avid Media Composer特征：专业软件，陡峭的学习曲线，时间线为核心操作界面。每一个转场、每一段字幕、每一个特效，都需要手动操作。这个时代的视频制作是一个手艺活，好的剪辑师需要多年经验积累。

痛点：学习成本极高，效率取决于操作者的熟练程度。一条3分钟的短视频，后期可能需要3-6小时。

第二时代：模板化剪辑时代（2018-2024）

代表工具：剪映/CapCut、Canva、快影特征：模板驱动，一键成片。剪映的"图文成片"功能可以把一段文字直接变成视频。各种预设的转场、滤镜、字幕动画，让非专业人士也能快速产出看起来不错的视频。

进步：大幅降低入门门槛，30分钟可以完成之前3小时的工作。

痛点：模板化带来同质化。所有人的视频看起来都差不多。当你想要更个性化的效果时，模板就不够用了，你还是得回到手动调整。而且批量生产依然困难——你没法让剪映自动帮你做100个不同的视频。

第三时代：对话式剪辑时代（2024-now）

代表工具：Claude Code + video-use、HyperFrames、Remotion + AI助手特征：用自然语言描述需求，AI帮你完成剪辑。你不再操作时间线，而是操作对话。AI理解你的意图，调用底层工具（FFmpeg、剪映API、代码框架），自动完成所有操作。

核心理念：视频 = 代码可以描述的结构化对象。 既然代码可以描述视频，而AI可以写代码，那么你只需要学会"怎么跟AI描述你想要的视频"。

优势：无限个性化 + 自动化能力兼备。你可以描述任何你想要的效果（不受模板限制），同时可以批量生产（代码天然支持循环和数据驱动）。

表1-1 三个视频剪辑时代的对比

维度	专业剪辑时代	模板化剪辑时代	对话式剪辑时代
核心操作	手动拖拽时间线	选择模板 + 微调	自然语言描述
学习成本	数月到数年	数小时到数天	数小时
个性化程度	无限（取决于技术）	有限（模板范围内）	无限（语言可描述的范围）
批量生产	极难	困难	容易（代码天然支持）
代表工具	Premiere, FCPX	剪映, Canva	Claude Code + 工具链
所需技能	软件操作	审美 + 基础操作	表达需求的能力

1.3 什么是 Vibe Coding?

2025年2月，Andrej Karpathy（前特斯拉AI总监、OpenAI创始成员）在一条推文里提出了Vibe Coding这个词。他说：

"有一种新的编程方式，我称之为'氛围编码'（Vibe Coding）：你完全沉浸在氛围中，接受指数级的代码生成，忘却代码的存在。这是用自然语言进行的编程，英文就是新的编程语言。"简单来说，Vibe Coding 的核心就是：用自然语言代替手写代码，让AI来完成具体的实现。

传统的编程方式是：你在脑中构思需求 -> 把它翻译成编程语言的语法 -> 一行行敲出来 -> 调试运行。

Vibe Coding 的方式是：你在脑中构思需求 -> 用中文/英文描述给AI -> AI生成代码 -> 你审核结果 -> 继续对话迭代。

这个概念最初针对的是软件开发，但很快延伸到了"用代码做一切事情"的领域——包括视频制作。

因为视频的本质是可结构化的数据（帧序列、音频轨道、时间轴），而代码恰好最擅长描述结构化数据。

小贴士： Vibe Coding 不是"不用学编程了"，而是"编程的方式变了"。你仍然需要理解底层发生了什么（比如知道什么是视频编码、帧率、分辨率），但不再需要记住具体的命令语法。

AI是你的"翻译官"，你把想法告诉它，它翻译成机器能理解的指令。

1.4 为什么 Vibe Coding 天然适合视频制作?

视频制作和软件开发有一个关键区别：视频是"结构化对象"的极佳范例。

一个视频文件，本质上可以被精确描述为：

一个时间轴（比如180秒）

一条或多条视频轨道（每个轨道上有若干片段，每个片段有起始时间、结束时间、源文件）

一条或多条音频轨道若干个字幕条目（每段字幕有文本内容、出现时间、消失时间、位置、样式）

特效和滤镜参数所有这些，用代码来描述非常自然。一段描述"从第5秒到第15秒播放素材A，同时显示字幕'Hello World'"的代码，比手动在时间线上拖拽要精确得多。

而在 Vibe Coding 的范式下，你不需要自己写这段代码——你只需要说："在视频第5秒开始放素材A，持续10秒，加上字幕'Hello World'。"AI会帮你翻译成精确的代码/命令。

这就是 Vibe Coding 和视频制作的天作之合：视频的结构化本质让代码描述成为可能，AI的自然语言理解能力让代码描述变得 effortless。

1.5 这套方法适合谁？不适合谁？

诚实地说，Vibe Coding 做视频并不是银弹。它有非常适合的场景，也有不适合的场景。

非常适合的人群

口播博主/知识分享类创作者：有大量口播素材需要剪辑（去口癖、加字幕、做精简），工作流程相对标准化短视频运营者：需要批量产出同类内容（比如每天5条带货视频），适合用模板+数据驱动的方式企业市场部/品牌部：需要稳定产出品牌视频，有明确的视觉规范和重复性需求有编程背景的内容创作者：能更快理解底层逻辑，更容易自定义工作流技术科普/教育类创作者：经常需要制作数据可视化、代码演示类视频

不太适合的场景

高度创意性的艺术短片：目前AI对"艺术感"的理解还很有限，纯创意性工作还是人类更擅长对实时性要求极高的直播剪辑：AI工作流需要一定的计算时间，不适合"边播边剪"极其复杂的影视级后期：涉及大量手工调色、精细抠像、VFX特效的场景，传统工作流仍然更高效完全不愿意接触任何技术概念的人：虽然不需要写代码，但你至少需要理解"文件路径""分辨率""帧率"等基本概念避坑指南： 不要把 Vibe Coding 当作"完全不需要学习"的捷径。虽然不需要学剪辑软件的界面操作，但你仍然需要学习"如何有效地向AI描述视频需求"——这其实是一门新的技能，我们后面会详细展开。

1.6 一个完整案例预览

为了让你对这套工作流有直观感受，我们来看一个完整的例子——从一段20分钟的口播录像到3分钟精品视频的全过程。

场景：你录了一段关于"如何学习Python"的教学视频，原始素材20分钟，充满口癖和停顿。

目标：剪成3分钟精华版，配上字幕和背景音乐，加上片头片尾。

传统工作流（预计4-6小时）：

1. 把视频导入 Premiere（5分钟）
2. 一帧帧听，找到口癖位置，切开删掉（2小时）
3. 整体听一遍，选出精华段落，重新排列（30分钟）
4. 导出音频到字幕软件，生成SRT（10分钟）
5. 导入字幕，调整样式、位置、动画（30分钟）
6. 在AE里做片头片尾（1小时）
7. 找背景音乐，调整音量平衡（20分钟）
8. 渲染导出（30分钟）

Vibe Coding 工作流（预计20-40分钟）：

1. 把视频文件放到项目文件夹（1分钟）
2. 打开 Claude Code，输入："帮我剪辑这个口播视频，去掉所有嗯啊之类的口癖，保留核心内

容，控制在3分钟左右"（AI自动处理：语音转文字 -> 语义分析 -> 识别口癖 -> 自动剪辑，约5-10分钟）

3. 继续对话："给视频加上中文字幕，用白色粗体，放在画面底部，加上逐字出现的动画"（AI

自动处理：字幕生成 -> 样式设置 -> 烧录到视频，约5分钟）

4. 继续对话："加一个5秒的片头，标题是'Python入门指南'，用深色背景+渐变文字效果"（AI用

HyperFrames 生成片头并拼接，约5-10分钟）

5. 继续对话："给视频配一段轻快的背景音乐，音量调小一点不要盖过人声"（AI自动处理，约3

分钟）

6. 导出成品（自动完成）

整个过程中，你不需要打开任何剪辑软件。所有的操作都在一个聊天对话框里完成。AI是你的"剪辑助理"，它理解你的意图，调用底层工具完成具体工作。

小贴士： 上面的案例中，AI并不是"魔法"——它背后调用的仍然是FFmpeg（视频处理）、Whisper（语音识别）、各种代码框架（片头生成）等工具。但在Vibe Coding的范式下，这些工具的复杂性被完全封装了。你只需要关心"我想要什么"，不需要关心"具体怎么实现"。

本章小结

1. 视频剪辑正在经历第三次革命：从手动剪辑到模板化剪辑，再到对话式剪辑（Vibe

Coding）。

2. Vibe Coding = 用自然语言描述需求 + AI完成实现。这个概念由Andrej Karpathy在2025年提出。

3. 视频的本质是结构化数据（时间轴、轨道、片段、字幕），这让它天然适合用代码描述，也天然适合Vibe Coding。

4. 这套方法特别适合口播博主、短视频运营者、企业市场团队等需要标准化、批量产出视频的场景。

5. 它不能替代所有类型的视频制作——高度创意性的艺术工作仍然需要人类主导。

动手试试练习1：列出你过去一个月做过的所有视频制作任务，标注每个任务大概花了多少时间。思考哪些环节是"重复性的机械劳动"，哪些环节需要"人类的创意判断"。

练习2：尝试用一句话描述你最近制作的一条视频（比如"帮我把这个20分钟的口播剪成3分钟精华，去掉口癖，加字幕"）。这句话，就是你在Vibe Coding工作流中会对AI说的第一句话。

第2章 三件套——GitHub项目、Skill、MCP 到底是什么关系?

2.1 一个厨房的故事

在你真正开始使用这些工具之前，需要理解生态中四个核心角色的关系。让我用一个生活化的类比来解释。

想象你要做一顿大餐。你的厨房里有什么?

厨房里的电器 = GitHub 开源项目

你的厨房里有烤箱、搅拌机、微波炉、电饭煲……每个电器有特定的功能。烤箱擅长烘焙，搅拌机擅长打汁，微波炉擅长快速加热。你可以把它们组合起来完成一道复杂的菜。

在视频制作的生态里，GitHub 开源项目就是这些"电器":

HyperFrames = 烤箱（把HTML描述的画面"烤"成视频）

Remotion = 料理机（用React组件组合出复杂视频）

FFmpeg = 厨房的基础设施（所有电器都离不开电和水，FFmpeg就是视频领域的"水电"）

每个GitHub项目解决一个特定的问题领域。它们各自独立，但可以互相配合。

厨房的水电气接口 = MCP 服务

现在问题来了：你请了一个超级聪明的厨师（AI助手），但它不知道怎么操作你家的电器。你家烤箱的开关在哪？微波炉怎么设定时间？搅拌机有几个档位?

你需要一套标准化的接口，让厨师能统一地操作所有电器。

MCP（Model Context Protocol）就是这个接口层。它是Anthropic在2024年底推出的开放协议，定义了AI助手如何调用外部工具的标准方式。

MCP服务就像在每个电器上安装了一个"智能插座"。AI助手不需要知道每个电器的具体操作细节——它只需要按照MCP协议发指令，MCP服务会翻译成电器能理解的命令。

菜谱 = Skill 技能包

厨师很聪明，知道怎么用电器了。但它不知道做红烧肉的步骤——是先炒糖色还是先焯水？炖多久？什么时候放盐?

Skill 技能包就是"菜谱"。它是一份写好的指南，告诉AI助手完成某个任务的完整步骤。比如"剪辑口播视频"的Skill会包含:

1. 先用Whisper把语音转文字

2. 用AI分析文本，标记出口癖和重复句

3. 用FFmpeg提取有效片段

4. 拼接片段，生成剪辑版视频

5. 生成字幕文件

6. 把字幕烧录到视频上

Skill的本质是经验沉淀。一个厉害的Skill作者，会把视频剪辑的最佳实践（比如"静音阈值设为

0.3秒""字幕安全区域是画面高度的90%"）都写进Skill里。AI助手按照这个"菜谱"操作，就能产出

专业水准的结果。

厨师 = AI 编程助手

现在厨房（电脑）准备好了，电器（GitHub项目）买好了，智能插座（MCP服务）装好了，菜谱（Skill）也打印出来了。谁来实际操作?

AI 编程助手就是这位厨师。Claude Code、Cursor、Kiro，它们都是不同品牌的"厨师"——有的擅长快速上手（Cursor），有的擅长复杂大菜（Claude Code），有的擅长按规矩来（Kiro）。

厨师会阅读菜谱（Skill），按照步骤操作电器（通过MCP服务调用GitHub项目），最后把菜端上桌（输出成品视频）。

2.2 四者协作流程

让我们用一个具体的例子来看这四者是如何配合的：

你的需求："帮我把这个口播视频的前5秒片头删掉，然后加上字幕。"协作过程：

1. 你 -> AI助手：用自然语言描述需求（"删前5秒，加字幕"）

2. AI助手 -> Skill：读取"视频剪辑"Skill，了解"加字幕"的标准流程

3. AI助手 -> MCP服务：通过ffmpeg-mcp服务，调用FFmpeg执行"删除前5秒"的操作

4. MCP服务 -> GitHub项目：ffmpeg-mcp底层调用FFmpeg（一个GitHub上的开源项目）

5. AI助手：使用Whisper（另一个工具）进行语音识别，生成字幕文本

6. AI助手 -> MCP服务：再次通过ffmpeg-mcp，调用FFmpeg将字幕烧录到视频上

7. AI助手 -> 你：返回成品视频路径

【此处建议配图：四者协作关系图。中心是"AI助手（Claude Code/Cursor/Kiro）"，左侧连接"Skill技能包（菜谱）"，右侧连接"MCP服务（接口层）"，MCP服务下方连接多个"GitHub开源项目（电器）"。用户在最上方，通过自然语言与AI助手交互。】

2.3 对比表格：每种"武器"的定义和用法

表2-1 四者对比速查

角色	类比	本质	你需要做什么	上手难度
GitHub开源项目	厨房电器	具体功能的软件工具（如FFmpeg、HyperFrames）	安装到电脑上	中等（需要命令行安装）
MCP服务 Skill技能包	智能插座/接口菜谱	让AI能调用工具的"翻译层"告诉AI如何完成任务的指南文件	安装并配置到AI助手中下载到指定目录	低（修改配置文件）低（复制粘贴）
AI编程助手	厨师	执行任务的智能体（Claude Code等）	安装软件，开始对话	低（安装后聊天即可）

2.4 常见误解澄清

误解1："是不是装了Skill就不需要GitHub项目了？"

不是。 Skill和GitHub项目是完全不同的层级。Skill是"菜谱"，GitHub项目是"电器"。没有电器，菜谱再详细也做不出菜。Skill需要依赖GitHub项目来完成实际工作。比如videocut-skills（剪辑Skill）底层依赖FFmpeg来处理视频文件。

误解2："MCP和Skill是不是同一个东西？"

不是。 MCP是"接口标准"，Skill是"操作指南"。MCP解决的是"AI怎么调用工具"的问题（技术层面），Skill解决的是"AI应该按什么步骤工作"的问题（流程层面）。两者配合使用：Skill告诉AI"先做什么后做什么"，MCP让AI"能够去做"。

误解3："我必须把这四个东西都装齐才能开始吗？"

不一定。 最简单的起步方案只需要两样东西：一个AI助手（如Claude Code） + 一个GitHub项目（如video-use，它内置了所有需要的功能）。video-use甚至不需要你手动安装FFmpeg——它会自动处理。Skill和MCP是进阶玩法，让你能组合更多工具、完成更复杂的任务。初学者完全可以"先开枪再瞄准"，从一个简单的工具开始，逐步扩展。

误解4："这四种东西是不是每个平台都支持？"

支持程度不同。 目前（2026年4月）的生态系统支持情况如下：

表2-2 各AI助手对Skill和MCP的支持情况

AI助手	MCP支持	Skill支持	备注
Claude Code	原生支持	原生支持（~/.claude/skills/）	生态最成熟，推荐首选
Cursor	支持（需配置）	支持（多种加载方式）	IDE体验最好
Kiro	支持	支持	AWS生态整合最深
其他（Windsurf等）	部分支持	部分支持	生态尚在建设中

2.5 你的第一步应该怎么走?

理解了四者的关系后，你可以根据自己的情况选择起步路径：

最简路径：安装Claude Code -> 安装video-use -> 直接开始对话（不需要手动配置MCP或Skill）

标准路径：安装Claude Code -> 配置ffmpeg-mcp -> 安装videocut-skills -> 获得完整的剪辑能力进阶路径：安装Claude Code -> 配置多个MCP（ffmpeg-mcp、Video_Editor_MCP等）->安装多个Skill（videocut、jianying-editor、vectcut）-> 按需组合使用国内用户提示：国内用户特别注意：由于网络环境原因，GitHub访问可能不稳定。建议提前配置好网络加速工具，或者使用国内镜像（如Gitee）来克隆仓库。npm安装时可以使用淘宝镜像（npm config set registry https://registry.npmmirror.com）。具体方法见第3章的"国内用户特别提示"部分。

本章小结

1. 视频制作的Vibe Coding生态由四个核心角色组成：GitHub开源项目（电器）、MCP服

务（接口）、Skill（菜谱）、AI助手（厨师）。

2. GitHub项目提供具体功能（如FFmpeg处理视频、HyperFrames生成画面），MCP让AI

能调用这些功能，Skill告诉AI完成任务的步骤，AI助手负责统筹执行。

3. 四者是协作关系而非替代关系。Skill依赖GitHub项目完成实际工作，MCP是连接AI和

项目的桥梁。

4. 初学者可以从最简单的路径开始（一个AI助手+一个项目），逐步扩展。

动手试试练习1：在纸上画出你自己的"厨房"——列出你电脑上已经安装的软件，思考它们分别对应"电器""插座""菜谱""厨师"中的哪个角色。

练习：访问https://github.com/heygen-com/hyperframes和https://github.com/browser-use/video-use，浏览它们的README页面，不需要看懂所有内容，只需要感受它们各自是做什么的"电器"。

第3章 你的AI视频工作台——环境准备全攻略

3.1 硬件要求

好消息是，AI视频制作对硬件的要求比传统视频剪辑低得多。因为大部分计算（语音识别、代码生成等）要么在云端完成，要么由AI助手的服务器处理。你的电脑主要承担"运行基础工具"的角色。

表3-1 硬件配置建议

配置项	最低配置	推荐配置	说明
CPU	任意现代CPU（2018年后）	i5/R5 或以上	视频编码主要吃CPU
内存	8GB	16GB+	同时运行AI助手和多个工具时内存需求大
硬盘	20GB可用空间	100GB+ SSD	视频素材占用空间大
显卡	不需要独立显卡	有则更好	FFmpeg等工具可调用GPU加速编码
网络	能访问GitHub	稳定宽带	需要下载工具和模型

如果你的电脑能流畅运行微信和浏览器，那它大概率能胜任AI视频制作的工作。

成本提示： 大部分工具都是免费的（开源项目），AI助手有免费额度。初期投入几乎为零。

唯一可能需要付费的是AI助手的订阅（Claude Code约$20/月，Cursor Pro约$20/月），但这些费用相比于你节省的时间成本来说微不足道。

3.2 软件清单与安装顺序

整个环境搭建有四层，需要按顺序安装（后面的依赖前面的）：

1. 基础层：Node.js、Python、FFmpeg、Git
2. AI助手层：Claude Code / Cursor / Kiro（三选一）
3. 项目层：HyperFrames / Remotion / video-use 等（按需安装）
4. 能力层：MCP服务配置、Skill安装

3.3 基础层安装

3.3.1 安装 Git

Git是代码管理工具，用来从GitHub下载项目。

macOS：

打开终端，输入

brew install git

验证安装

git --version

应显示类似：git version 2.40.0

Windows：

1. 访问 https://git-scm.com/download/win 下载安装程序

2. 双击安装，全部默认选项即可

3. 打开 PowerShell 或 CMD，验证安装

git --version国内用户提示： 国内用户提示：GitHub访问可能较慢。可以考虑以下加速方案：

1. 使用国内的GitHub镜像（如 https://gitclone.com 或 https://ghfast.top）
2. 配置Git 代理（如果你有可用的代理工具）： git config --global http.proxy

http://127.0.0.1:7890

3. 使用Gitee导入GitHub仓库后从Gitee克隆

3.3.2 安装 Node.js

Node.js是运行JavaScript的环境，HyperFrames、Remotion等项目都需要它。

macOS：

使用 Homebrew 安装

brew install node

验证安装

node --version

应显示 v18.x.x 或更高版本

npm --version Windows：

1. 访问 https://nodejs.org 下载LTS版本

2. 双击安装程序，全部默认选项

3. 验证安装

node --version npm --version国内用户提示： 国内用户提示：npm安装包时可以从淘宝镜像下载，速度快很多：

npm config set registry https://registry.npmmirror.com后续所有 npm install 命令都会自动使用国内镜像。

3.3.3 安装 Python

Python是另一个重要的编程环境，video-use、剪映Skill等需要它。

macOS：

macOS 通常自带Python3，先检查

python3 --version

如果没有，用 Homebrew 安装

brew install python Windows:

1. 访问 https://python.org 下载Python 3.10+

2. 安装时勾选"Add Python to PATH"

3. 验证安装

python --version

3.3.4 安装 FFmpeg

FFmpeg是整个工具链的基石——几乎所有视频处理操作最终都会调用它。它是免费开源的。

macOS:

最简单的方式

brew install ffmpeg

验证安装

ffmpeg -version

应显示版本信息和编译参数

Windows:

方式1：使用包管理器（推荐）

先安装 chocolatey（Windows的包管理器），以管理员身份打开PowerShell：

Set-ExecutionPolicy Bypass-Scope Process-Force;

[System.Net.ServicePointManager]::SecurityProtocol= [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps 1'))

然后用 chocolatey 安装 ffmpeg

choco install ffmpeg

方式2：手动安装

1. 访问 https://ffmpeg.org/download.html#build-windows

2. 下载Windows build，解压到 C:\ffmpeg

3. 将 C:\ffmpeg\bin 添加到系统环境变量 PATH 中

验证安装

ffmpeg -version小贴士： FFmpeg安装是最容易出现问题的环节。如果你在Windows上手动安装，一定要记得把bin目录添加到PATH环境变量，否则后续所有工具都会找不到FFmpeg。验证方法是打开一个新的终端窗口，输入ffmpeg -version，如果能显示版本号就说明成功了。

3.4 AI助手层安装

三选一即可。对于视频制作，我们推荐Claude Code作为首选。

3.4.1 安装 Claude Code（推荐）

Claude Code是Anthropic出品的终端AI助手，原生支持MCP和Skill，是目前生态最成熟的选择（截至2026年4月，GitHub 30,000+ stars）。

macOS / Linux

brew install claude-code

Windows（使用npm）

npm install -g @anthropic-ai/claude-code

验证安装

claude --version首次运行需要登录：

claude

会提示你打开浏览器进行认证

按照提示完成即可

费用（截至2026年4月）：

Claude Pro：$20/月，包含标准使用量的Claude Code Claude Max：$100-200/月，更高的使用量限制 Claude Team：$30/座位/月，团队协作功能

3.4.2 安装 Cursor

Cursor是基于VS Code的AI原生IDE，有图形界面，适合习惯IDE操作的用户。

访问 https://cursor.com 下载安装程序

支持 macOS / Windows / Linux

安装后打开，注册账号即可使用

费用（截至2026年4月）：

免费版：有限的功能Pro：$20/月Pro+：$60/月Ultra：$200/月

3.4.3 安装 Kiro

Kiro是AWS出品的AI原生IDE，2025年中推出，主打"规格驱动开发"。

访问 https://kiro.dev 下载

支持 macOS / Windows / Linux

需要使用AWS账号登录

费用（截至2026年4月）：

免费版：50 credits/月 + 注册奖励500 credits（30天有效）

Pro：$20/月，1000 credits Pro+：$40/月，2000 credits超出部分：$0.04/credit

3.5 项目层安装（按需）

不需要一次性安装所有项目。根据你的需求选择：

3.5.1 安装 video-use（最简单起步）

video-use是browser-use团队出品的对话式视频编辑工具，安装最简单，适合作为第一个项目。

1. 克隆仓库

git clone https://github.com/browser-use/video-use.git

2. 进入目录

cd video-use

3. 按照README中的install.md进行安装

一般流程：

pip install -r requirements.txt

4. 配置环境变量

cp .env.example .env

编辑 .env 文件，填入必要的API密钥

3.5.2 安装 HyperFrames

HyperFrames适合需要从零"写"出视频的场景（如片头片尾、动画讲解）。

1. 克隆仓库

git clone https://github.com/heygen-com/hyperframes.git

2. 进入目录

cd hyperframes

3. 安装依赖（使用bun）

npm install -g bun # 先安装bun（如果还没有）

bun install

4. 或使用npm

npm install

3.5.3 安装 Remotion

Remotion适合复杂动画和批量生产场景。

使用npm初始化项目

npx create-video@latest my-video-project

进入项目

cd my-video-project

安装依赖

npm install

3.6 能力层配置

3.6.1 配置 MCP 服务

以ffmpeg-mcp为例，展示如何在Claude Code中配置：

1. 克隆ffmpeg-mcp仓库

git clone https://github.com/video-creator/ffmpeg-mcp.git

2. 进入目录并安装依赖

cd ffmpeg-mcp pip install -r requirements.txt # 或按照仓库的README安装

3. 编辑Claude Code的MCP配置文件

配置文件路径：

macOS: ~/.config/claude-code/mcp_settings.json

Windows: %APPDATA%\claude-code\mcp_settings.json

添加以下内容：

{"mcpServers": {"ffmpeg-mcp": {"command": "python","args": ["/path/to/ffmpeg-mcp/main.py"]

}}}

3.6.2 安装 Skill

以videocut-skills为例：

macOS / Linux

git clone https://github.com/Ceeon/videocut-skills.git ~/.claude/skills/videocut

Windows

git clone https://github.com/Ceeon/videocut-skills.git%USERPROFILE%.claude\skills\videocut然后在Claude Code中使用：输入 /videocut:安装 触发安装命令。

3.7 环境验证

安装完成后，运行以下命令验证环境是否就绪：

验证基础层

git --version # 应显示版本号node --version # 应显示 v18+python3 --version # 应显示 3.10+ffmpeg -version # 应显示版本信息

验证AI助手

claude --version # 或 cursor --version

验证项目层（如果安装了）

cd video-use && python main.py --help # 应显示帮助信息

验证MCP（在Claude Code中）

打开Claude Code，输入：

"请列出你可用的MCP工具"

应返回已配置的MCP服务列表

3.8 常见安装问题 Top 10 排查清单

表3-2 常见安装问题及解决方案

问题	原因	解决方案
ffmpeg: command not found	FFmpeg未添加到PATH	重新安装并确保bin目录在PATH中
npm install 卡住不动	网络问题	切换npm镜像：npm config set registry https://registry.npmmirror.com
git clone 失败pip install 报错	GitHub访问问题Python版本不兼容	使用镜像站或代理确保Python 3.10+，使用venv隔离环境
Claude Code登录失败 MCP服务连接不上Skill命令不识别	网络或地区限制配置文件路径或格式错误Skill目录位置不对	检查网络连接，确认所在地区支持检查JSON格式，确认路径正确确认放在正确的skills目录下
bun install 报错视频渲染为空文件	bun未正确安装FFmpeg编码器缺失	尝试用npm install替代重新安装FFmpeg，确保包含libx264
字幕中文显示为方框	字体缺失	安装中文字体（如Noto Sans CJK）

本章小结

1. 环境搭建分四层：基础层（Node.js/Python/FFmpeg/Git）-> AI助手层 -> 项目层 -> 能力层（MCP/Skill）。
2. 硬件要求很低——能运行浏览器的电脑基本都能胜任。
3. AI助手推荐首选Claude Code（生态最成熟），也可以根据习惯选择Cursor或Kiro。
4. 项目不需要一次性全部安装，按需选择即可。
5. 国内用户注意配置npm镜像和GitHub加速。

动手试试练习1：按照本章的步骤，至少完成基础层的安装（Git、Node.js、Python、FFmpeg），并运行验证命令确认一切就绪。

练习2：安装Claude Code并完成登录认证，在对话框中输入"你好"，确认AI助手能正常响应。

PART TWO

兵器篇

AI视频制作工具全景

第4章 HyperFrames——用写网页的方式做视频

第5章 Remotion——用React组件"编程"视频

第6章 Revideo——代码驱动视频的另一种选择

第7章 video-use——对话式视频编辑的"剪辑师"

第8章 FFmpeg + MCP——视频处理的瑞士军刀

第9章 Skill 技能包——给AI装上"剪辑师大脑"

第10章 AI编程助手选择指南——Claude Code vsCursor vs Kiro

cd hyperframes

bun install

bun run --help

步骤2：第一个Hello World视频

// 创建一个 hello.html 文件// 内容：

然后在Claude Code中（或终端中）运行：

bun run hyperframes render hello.html --output hello.mp4 --duration 3

步骤3：理解项目结构

hyperframes/ ├── packages/ # 核心代码

│ │ │ ├── skills/ ├── registry/ └── docs/ ├── core/ ├── cli/ └── studio/

4.6 进阶用法

自定义模板

HyperFrames的registry目录包含了大量预设的过渡效果和动画模板。你可以直接调用这些模板，也可以自己创建新的。

动画效果

利用CSS动画和GSAP（一个强大的JavaScript动画库），可以实现非常复杂的动画效果！

// 使用GSAP做复杂动画import gsap from 'gsap';

// 在HTML中定义元素// 用GSAP控制动画时间线gsap.timeline()

.from('.title', { opacity: 0, y: 50, duration: 1 })

.from('.subtitle', { opacity: 0, x: -50, duration: 0.8 }, '-=0.5')

.to('.bg', { scale: 1.1, duration: 3 }, 0);

与真人素材合成

HyperFrames可以输出透明背景的视频（使用WebM格式或带Alpha通道的PNG序列），然后在FFmpeg中与实拍视频合成：

ffmpeg -i title.mp4 -i footage.mp4 -filter_complex "[0:v][1:v]concat=n=2:v=1:a=0" output.mp4

4.7 实战案例：做一个3分钟知识口播视频

需求：做一个关于"什么是MCP协议"的讲解视频，包含片头、3段内容页、片尾。

在Claude Code中的对话：

你：用HyperFrames帮我做一个知识讲解视频的片头，标题是"什么是MCP协议"，风格要科技感，深色背景，持续3秒。

AI：我来为你创建这个片头。我会使用HyperFrames生成一个HTML描述的科技感片头。

[AI自动生成HTML文件，调用HyperFrames渲染]

✅ 片头已生成：assets/title.mp4你：很好，现在做第一段内容页。标题"MCP的定义"，正文内容：

"MCP是Model Context Protocol的缩写，由Anthropic在2024年推出。

它定义了AI助手如何调用外部工具的标准方式。"持续5秒。

AI：正在生成第一段内容页...

✅ 第一段已生成：assets/section1.mp4你：继续生成第二段和第三段，然后帮我把所有片段拼接起来。

AI：所有片段已生成并拼接完成。最终视频：final_video.mp4本章小结

1. HyperFrames是HeyGen出品的开源框架，核心是用HTML/CSS描述画面，渲染成视频。

2. 截至2026年4月，GitHub 12,700+ stars，最新版本v0.4.34。

3. 特别适合片头片尾、动画讲解、数据可视化等"从零创作"的场景。

4. 不适合已有素材的剪辑类工作。

5. AI非常擅长用它，因为HTML是AI最熟悉的技术之一。

动手试试练习1：安装HyperFrames并运行Hello World示例，确认环境正常。

练习2：在Claude Code中描述一个你想要的片头效果，观察AI如何生成HTML并调用HyperFrames渲染。

第5章 Remotion——用React组件"编程"视频

5.1 项目背景与社区生态

Remotion 是目前代码驱动视频领域最成熟、生态最丰富的框架。它的核心理念是：用 React 组件来定义视频的每一帧。如果你会写React（或者你的AI会写React），你就能做视频。

Remotion由瑞士开发者Jonny Burger创建，自2021年发布以来，已经成为程序化视频生成的事实标准。它的Business Source License license允许年收入低于$1M的公司免费使用。

表5-1 Remotion 基本信息（截至2026年4月）

项目	详情
GitHub仓库	https://github.com/remotion-dev/remotion
Stars	45,100+
Forks	3,100+
最新版本	v4.0.454（2026年4月28日）
Commits	32,147
贡献者	331人
开发语言	TypeScript（74.2%）、PHP、MDX、JavaScript
License	Business Source License（收入<$1M免费）
官网	https://remotion.dev
【待核实：截至	2026年4月】Remotion社区非常活跃，官方维护了大量教程和模板，还有活跃的

Discord社区（14,000+成员）。

5.2 核心原理

Remotion的核心思想可以概括为：

React组件 = 视频的每一帧：你写的React组件就是视频的画面props = 动画参数：通过组件的属性控制动画行为useCurrentFrame() = 时间轴：一个React hook，返回当前帧号，让你基于时间做动画Composition = 视频片段：定义视频的分辨率、帧率、时长等元信息简单来说，Remotion会在后台打开一个浏览器，逐帧渲染你的React组件（就像给网页截图），然后把所有帧合成视频。

5.3 和 HyperFrames 的区别

表5-2 HyperFrames vs Remotion 对比

维度	HyperFrames	Remotion
技术基础	HTML/CSS + Puppeteer	React组件 + 浏览器渲染
学习曲线	低（会写网页即可）	中（需要了解React）
动画能力	中等（CSS动画 + GSAP）	强（React生态 + 自定义hooks）
生态丰富度	较新，生态在建设中	非常成熟，大量插件和模板
批量生产	支持	非常擅长（Lambda/Cloud Run部署）
服务器渲染	有限	成熟（remotion/lambda包支持AWS Lambda）
最适合场景	片头片尾、简单动画	复杂动画、批量生产、数据驱动视频
许可费用	免费（Apache 2.0）	免费（年收入<$1M），否则需付费

5.4 从零上手

步骤1：创建项目

npx create-video@latest my-video-project

按照提示选择模板（初学者推荐"Hello World"模板）

cd my-video-project npm install

步骤2：编写你的第一个视频组件

// src/HelloWorld.tsx import { useCurrentFrame, useVideoConfig } from 'remotion';

export const HelloWorld = () => {const frame = useCurrentFrame(); // 获取当前帧号const { fps, durationInFrames } = useVideoConfig(); // 获取视频配置// 根据帧号计算动画参数const opacity = Math.min(1, frame / 30); // 前30帧淡入const scale = 1 + frame * 0.001; // 缓慢放大return (

scale(${scale}),}}>Hello Remotion!

);

};

步骤3：定义Composition

// src/Root.tsx import { Composition } from 'remotion';

import { HelloWorld } from './HelloWorld';

export const RemotionRoot = () => {return ();

};

步骤4：预览和渲染

启动预览服务器

npm run dev

在浏览器中打开 http://localhost:3000 预览

渲染最终视频

npx remotion render HelloWorld output.mp4

5.5 核心优势

React生态全套可用：任何React库（图表库、UI组件库、动画库）都可以直接用在视频里数据驱动：可以轻松地将JSON数据映射为视频内容，实现批量个性化生成程序化动画：用JavaScript精确控制每一帧的每一个像素服务器端渲染：支持部署到Lambda、Cloud Run等，实现云端批量渲染实时预览：开发时可以实时在浏览器中预览效果

5.6 实战案例：数据可视化短视频

需求：做一个展示"2024年各季度销售额"的数据动画视频。

// src/SalesChart.tsx import { useCurrentFrame, useVideoConfig, interpolate } from 'remotion';

const data = [{ quarter: 'Q1', value: 120 },{ quarter: 'Q2', value: 180 },{ quarter: 'Q3', value: 150 },{ quarter: 'Q4', value: 240 },];

export const SalesChart = () => {const frame = useCurrentFrame();

const { fps } = useVideoConfig();

// 动画进度：0到1 const progress = interpolate(frame, [0, 3 * fps], [0, 1], {extrapolateRight: 'clamp',});

return (

2024年销售数据

{data.map((item, i) => {// 每个柱子依次出现const itemProgress = interpolate(progress,[i * 0.2, i * 0.2 + 0.4],[0, 1],{ extrapolateLeft: 'clamp', extrapolateRight: 'clamp' });

return (

<div style={{height: `${item.value * 1.5 * itemProgress}px`,background: '#0f3460',borderRadius: '8px 8px 0 0',transition: 'height 0.1s',{item.quarter}
{Math.round(item.value * itemProgress)}万
);

})}

);
};
在Claude Code中，你只需要说："帮我做一个展示2024年四个季度销售额（Q1 120万、Q2 180万、Q3 150万、Q4 240万）的动画柱状图视频，持续3秒，蓝色柱子依次出现。"AI会自动生成上面的代码。
本章小结
Remotion是代码驱动视频领域最成熟的框架（45,100+ GitHub stars），用React组件定义视频。
截至2026年4月，最新版本v4.0.454，331位贡献者，生态非常丰富。
相比HyperFrames，Remotion的动画能力更强，更适合复杂视觉效果和批量生产。
年收入低于$1M可免费使用（Business Source License）。
数据驱动视频生成是它的杀手级应用场景。
动手试试练习1：运行npx create-video@latest创建一个Remotion项目，预览Hello World模板。
练习2：修改Hello World中的文字和颜色，重新渲染，观察效果变化。

第6章 Revideo——代码驱动视频的另一种选择

6.1 项目定位与特色

Revideo是一个基于TypeScript的开源视频编辑框架，它是从Motion Canvas项目fork出来的。

Motion Canvas本身是一个用代码创建动画的工具，但维护者并不打算把它作为库来使用。

Revideo团队在使用Motion Canvas的过程中发现了这个限制，于是决定fork并改造它，目标是变成一个开发者可以用来构建完整视频编辑应用的库。

Revideo的GitHub仓库地址： https://github.com/redotvideo/revideo【待核实：截至2026年4月】Revideo的具体stars数量我们未能获取到最新数据，但它在代码驱动视频领域是一个值得关注的替代方案。

6.2 和 Remotion 的异同

表6-1 Revideo vs Remotion 对比

维度	Revideo	Remotion
技术基础	TypeScript生成器/协程（fork自Motion Canvas）	React组件
动画模型	yield-based时间线控制	基于帧的useCurrentFrame
渲染方式	HTML Canvas 2D + WebCodecs API	浏览器逐帧截图
音频支持	内置Audio标签，音频同步良好	支持
实时预览	有内置编辑器	有Remotion Studio
服务器渲染	支持（改进的渲染管线）	非常成熟（Lambda/Cloud Run）
License React库支持	MIT（完全免费）不支持	Business Source License完整支持所有React库

6.3 核心优势

完全免费：MIT许可证，没有任何商业使用限制更快的渲染速度：使用Canvas 2D直接渲染（而非浏览器截图），渲染速度通常快于Remotion更好的动画控制：yield-based的动画模型提供了非常精确的时间线控制内置编辑器：有可视化编辑界面，可以实时预览和调整无头渲染：支持纯命令行渲染，不需要打开浏览器UI

6.4 适合什么场景

不需要React生态支持的项目对渲染速度有较高要求偏好生成器/协程风格的动画编程需要完全免费无商业限制的方案构建视频编辑应用（Revideo的库优先设计非常适合这个场景）

6.5 快速上手指南

步骤1：初始化项目

npm init @revideo@latest my-revideo-project

选择项目目录

cd my-revideo-project npm install

步骤2：编写场景

import {Audio, Img, Video, makeScene2D} from '@revideo/2d';

import {all, chain, createRef, waitFor} from '@revideo/core';

export default makeScene2D(function* (view) {const logoRef = createRef();

// 添加背景视频和音频yield view.add(<>

yield* waitFor(1);

// 添加Logo并做动画view.add(,);

// 缩放+旋转动画yield* chain(all(logoRef().scale(40, 2), logoRef().rotation(360, 2)),logoRef().scale(60, 1),);

});

步骤3：渲染

使用内置编辑器预览和渲染

npm start

打开 http://localhost:9000，点击Render按钮

或用代码渲染

import {renderVideo} from '@revideo/renderer';

const file = await renderVideo({projectFile: './src/project.ts',settings: {logProgress: true},});

console.log(`Rendered to ${file}`);

本章小结

1. Revideo是从Motion Canvas fork的TypeScript视频编辑框架，MIT许可证完全免费。
2. 使用yield-based动画模型和Canvas 2D渲染，速度通常快于Remotion。
3. 不支持React生态，这是与Remotion的核心差异。
4. 特别适合需要免费方案、追求渲染速度、或打算基于它构建视频编辑应用的开发者。

动手试试练习1 ： 运行npm init @revideo@latest创建一个Revideo 项目，与Remotion 的Hello World做对比。

练习2：尝试用Revideo的makeScene2D语法编写一个简单的Logo缩放动画。

第7章 video-use——对话式视频编辑的"剪辑师"

7.1 项目背景

video-use是由browser-use团队出品的对话式视频编辑工具。browser-use是一个知名的浏览器自动化项目，其核心思想是用自然语言操控浏览器完成复杂任务。video-use把这个理念延伸到了视频编辑领域。

与HyperFrames和Remotion不同，video-use不是"从零生成视频"的框架，而是"操作已有素材"的剪辑工具。它的定位更像是一位AI剪辑师——你给它素材，用自然语言告诉它怎么剪，它帮你完成。

表7-1 video-use 基本信息（截至2026年4月）

项目	详情
GitHub仓库	https://github.com/browser-use/video-use
Stars	5,400+
Forks	
最新提交	2026年4月24日
Commits	
开发语言	Python（75.8%）、HTML（23.1%）
许可证待确认（仓库未明确标注）	

7.2 核心理念

video-use的核心设计理念是：不写代码，用自然语言对话完成视频编辑。

它的工作流程非常简洁：

1. 你把原始素材（视频文件）放到一个文件夹里
2. 打开Claude Code（或其他支持的AI助手），进入该文件夹
3. 用自然语言告诉AI你想怎么剪
4. AI调用video-use的能力，自动完成所有操作
5. 在edit/目录下输出最终视频

video-use内置了一个SKILL.md文件（见第9章关于Skill的详细解释），这个文件就是它的"菜谱"，告诉AI助手如何操作视频。

7.3 和前面几个框架的本质区别

表7-2 video-use vs HyperFrames/Remotion

维度	video-use	HyperFrames/Remotion
工作对象	已有视频素材	从零创建视频
核心操作	剪辑、拼接、调整	生成、渲染、合成
使用方式	纯自然语言对话	需要写代码（AI可以代写）
输出方式	自动导出final.mp4	需要运行渲染命令
适合场景	口播剪辑、素材整理	片头片尾、动画制作
安装复杂度	低（内置Skill）	中（需配置环境）

7.4 内置能力

根据video-use的README（截至2026年4月24日），它具备以下核心能力：

自动去除填充词：识别并删除"嗯""啊"等口癖和错误开头自动调色：为每个片段应用色彩分级（暖色电影感、中性风格或自定义FFmpeg滤镜链）

音频淡入淡出：在每个剪切点添加30ms的音频淡入淡出，避免爆音字幕生成：生成字幕并烧录到视频中，默认使用两字大写分块风格，支持自定义动画叠加层：通过Manim、Remotion或PIL生成动画覆盖层，以并行子Agent方式运行输出质量自检：在每个剪切点边界自我评估渲染输出会话记忆：在project.md中持久化会话记忆，下次继续工作

7.5 从零上手指南

步骤1：克隆仓库

git clone https://github.com/browser-use/video-use.git

读取安装说明

cd video-use cat install.md

步骤2：一键设置（推荐方式）

video-use提供了一个非常方便的"一键设置"提示词，你可以直接粘贴给Claude Code：

Set up https://github.com/browser-use/video-use for me.

Read install.md first to install this repo, wire up ffmpeg,register the skill with whichever agent you're running under,and set up the ElevenLabs API key — ask me to paste it when you need it.

Then read SKILL.md for daily usage, and always read helpers/because that's where the editing scripts live.

After install, don't transcribe anything on your own —

just tell me it's ready and wait for me to drop footage into a folder.

AI 助手会自动完成：克隆仓库 -> 安装依赖 -> 连接FFmpeg -> 注册Skill -> 请求你提供ElevenLabs API密钥。

步骤3：放置素材并开始编辑

把你的视频素材放到一个文件夹里

cd /path/to/your/videos

启动Claude Code

claude

在对话中输入编辑指令

> edit these into a launch video

AI会：盘点素材 -> 提出剪辑策略 -> 等待你确认 -> 执行剪辑 -> 输出到 edit/final.mp4。

7.6 实战案例：把20分钟口播剪成3分钟精华

完整对话过程示例：

你：edit these into a launch video AI：我发现了以下素材：

- raw_take_1.mp4（8分钟）
- raw_take_2.mp4（12分钟）

- intro_backup.mp4（30秒）

我的剪辑策略：

1. 分析两段素材的语音内容，识别口癖和停顿
2. 提取两段的核心观点，去除重复内容
3. 拼接精华部分，控制在3分钟以内
4. 添加字幕和音频优化

是否确认执行?

你：确认AI：正在执行...

[步骤1/5] 语音转文字...

[步骤2/5] 识别口癖位置（找到127个"嗯"、83个"啊"）...

[步骤3/5] 语义分析，提取关键段落...

[步骤4/5] 自动剪辑和拼接...

[步骤5/5] 添加字幕和音频淡入淡出...

✅ 剪辑完成!

输出文件：edit/final.mp4时长：2分58秒共删除：67%的原始内容保留：18个核心段落小贴士： video-use的最大优势是"零配置上手"。你不需要手动安装MCP服务，不需要配置Skill目录——它自带了完整的SKILL.md，AI助手安装后就能直接使用。这使它成为初学者的理想入门选择。

避坑指南： video-use项目目前只有10个commits，是一个相对早期的项目。它的稳定性和功能完整性可能不如Remotion等成熟项目。建议把它作为"快速体验对话式剪辑"的入口，如果遇到问题，可以考虑切换到更底层的工具组合（如ffmpeg-mcp + videocut-skills）。

本章小结

1. video-use是browser-use团队出品的对话式视频编辑工具，5,400+ GitHub stars。
2. 它和HyperFrames/Remotion的本质区别在于：它操作已有素材（剪辑），而不是从零

生成视频。

3. 内置了完整的SKILL.md，支持一键设置，是初学者最友好的入门选择。
4. 内置能力包括：去口癖、自动调色、音频淡入淡出、字幕生成、动画叠加等。
5. 项目较早期（10 commits），稳定性和功能完整性有待观察。

动手试试练习1：克隆video-use仓库，按照README中的一键设置提示词完成安装。

练习2：准备一个你自己的口播视频（任何长度），把它放到video-use项目文件夹中，然后对AI说"edit this video"，观察AI如何处理。

第8章 FFmpeg + MCP——视频处理的瑞士军刀

8.1 FFmpeg 是什么?

如果说前面的工具是"厨房里的电器"，那FFmpeg就是整个厨房的电力系统。它是几乎所有视频处理操作的基础设施。

FFmpeg是一套开源的音视频处理工具集，诞生于2000年，至今已经发展了二十多年。它可以完成：

视频格式转换（MP4转MOV、AVI转WebM……任何格式之间互转）

视频剪辑（截取片段、拼接多个视频）

音频处理（提取音频、替换音轨、调整音量）

字幕处理（嵌入字幕、提取字幕、格式转换）

画面处理（缩放、裁剪、旋转、调速、调滤镜）

流媒体处理（推流、拉流、转码）

可以说，你看到的几乎所有视频软件（Premiere、剪映、YouTube、TikTok），底层都在调用FFmpeg。

8.2 为什么普通人不直接用FFmpeg?

因为FFmpeg的命令行接口非常复杂和晦涩。比如，一个简单的"把视频前5秒删掉"的操作，FFmpeg命令长这样：

ffmpeg -i input.mp4 -ss 00:00:05 -c copy output.mp4而"把视频缩放成1080p并加上中文字幕"的操作：

ffmpeg-i input.mp4-vf"scale=1920:1080,subtitles=subtitle.srt:force_style='FontName=Noto Sans CJK SC,FontSize=24,PrimaryColour=&HFFFFFF'" -c:a copy output.mp4这些命令对人类来说很难记忆和理解。你需要知道每个参数的含义、正确的顺序、各种编解码器的区别……

但AI不存在这个问题。AI可以：

准确理解你的自然语言需求（"把视频前5秒删掉"）

自动生成正确的FFmpeg命令处理各种边界情况（比如视频时长不足5秒时的错误处理）

在命令执行失败时自动调试和重试这就是MCP服务的价值所在。MCP服务是AI和FFmpeg之间的"翻译官"，让AI可以用标准化的方式调用FFmpeg。

8.3 主要 MCP 服务介绍

8.3.1 ffmpeg-mcp（video-creator版）

GitHub: https://github.com/video-creator/ffmpeg-mcp这是目前最成熟的FFmpeg MCP服务之一。它封装了FFmpeg的常用操作，让AI可以通过MCP协议调用。

主要功能：

视频搜索、裁剪、拼接、播放视频搜索（按名称在目录中查找视频）

视频裁剪和拼接视频信息查询（时长、分辨率、码率等）

格式转换安装方式：

使用UV包管理器（推荐）

git clone https://github.com/video-creator/ffmpeg-mcp.git cd ffmpeg-mcp uv sync

或使用pip

pip install ffmpeg-mcp

8.3.2 Video_Editor_MCP（by Kush Agrawal）

GitHub: https://github.com/Kush36Agrawal/Video_Editor_MCP这是另一个流行的视频编辑MCP服务，基于FFmpeg构建。

主要功能：

execute_ffmpeg：执行FFmpeg命令的核心工具支持裁剪、拼接、格式转换、音频提取等操作内置安全和错误处理机制支持批量操作

8.3.3 其他视频相关MCP服务

截至2026年4月，生态中还有一些其他值得关注的视频MCP服务：

sora-2-mcp（GitHub: https://github.com/writingmate/sora-2-mcp）：用于调用OpenAI Sora 2视频生成API的MCP服务，支持从文本生成视频、视频合并、淡入淡出动画video-audio-mcp（by Misbah Syed）：另一个基于FFmpeg的MCP服务，支持基础视频和音频编辑video-editing-mcp（by Burningion）：用于Video Jungle平台的MCP接口vibestudio（by Wizenheimer）：无运行时视频处理工具小贴士： MCP生态发展非常快，几乎每周都有新的MCP服务出现。建议关注https://github.com/punkpeye/awesome-mcp-servers 这个awesome-list，它汇总了社区中所有MCP服务。你也可以直接用自然语言问AI"有哪些视频相关的MCP服务"，它会帮你搜索最新信息。

8.4 MCP 安装与配置方法

以在Claude Code中配置ffmpeg-mcp为例：

步骤1：安装MCP服务

克隆仓库

git clone https://github.com/video-creator/ffmpeg-mcp.git

进入目录

cd ffmpeg-mcp

安装依赖（使用UV）

uv sync

步骤2：编辑Claude Code的MCP配置

macOS 配置文件路径

~/.config/claude-code/mcp_settings.json

Windows 配置文件路径

%APPDATA%\claude-code\mcp_settings.json添加以下内容（注意替换实际路径）：

{"mcpServers": {"ffmpeg-mcp": {"command": "uv","args": ["--directory","/path/to/ffmpeg-mcp","run","ffmpeg-mcp"],"transportType": "stdio"}}}

步骤3：验证配置

重启Claude Code后，输入：

请列出你可用的MCP工具

如果配置正确，应返回类似：

- ffmpeg-mcp: video_search, video_trim, video_concat, ...

8.5 20个高频操作速查表

以下是你最常用到的视频操作，以及对应的自然语言描述和AI执行的命令：

表8-1 自然语言描述与FFmpeg命令对照

自然语言描述AI生成的FFmpeg命令示例把视频前5秒删掉ffmpeg -i input.mp4 -ss 5 -c copy output.mp4截取视频第10秒到ffmpeg -i input.mp4 -ss 00:00:10 -to 00:00:20 -c copy output.mp4 20秒的片段把两个视频拼接起ffmpeg -f concat -safe 0 -i files.txt -c copy output.mp4来把视频转成16:9 ffmpeg -i input.mp4 -vf "scale=1920:1080,setsar=1:1" -c:a copy output.mp4把视频音量提高ffmpeg -i input.mp4 -af "volume=1.5" -c:v copy output.mp4 50%提取视频的音频ffmpeg -i input.mp4 -vn -acodec copy output.aac给视频加上中文字ffmpeg -i input.mp4 -vf "subtitles=sub.srt" -c:a copy output.mp4幕把视频加速2倍ffmpeg -i input.mp4 -vf "setpts=0.5*PTS" -af "atempo=2.0" output.mp4 *把视频放慢到0.5 ffmpeg -i input.mp4 -vf "setpts=2*PTS" -af "atempo=0.5" output.mp4*倍ffmpeg -i input.mp4 -vf "drawtext=text='Hello':x=10:y=10:fontsize=24"给视频加文字水印output.mp4把视频静音ffmpeg -i input.mp4 -an -c:v copy output.mp4替换视频音轨ffmpeg -i video.mp4 -i audio.mp3 -c:v copy -map 0:v:0 -map 1:a:0 output.mp4把视频转成GIF ffmpeg -i input.mp4 -vf "fps=10,scale=480:-1" output.gif旋转视频90度ffmpeg -i input.mp4 -vf "transpose=1" output.mp4裁剪视频画面（去ffmpeg -i input.mp4 -vf "crop=iw:ih*0.8:0:ih*0.1" output.mp4掉上下黑边）

查看视频信息ffprobe -v error -show_format -show_streams input.mp4压缩视频文件大小ffmpeg -i input.mp4 -vcodec libx264 -crf 28 -preset fast output.mp4 ffmpeg -i video.mp4 -i bgm.mp3 -filter_complex "[0:a]

给视频加背景音乐[1:a]amix=inputs=2:duration=shortest" output.mp4 ffmpeg -i input.mp4 -c copy -map 0 -segment_time 60 -f segment把视频切成多output_%03d.mp4个片段ffmpeg -i input.mp4 -vf "split[original][copy];

给视频加模糊[copy]scale=1920:1080,boxblur=20[blurred];[blurred][original]overlay=(W-w)/2:

背景（竖屏转(H-h)/2" output.mp4横屏）

小贴士： 不需要背诵这些命令！这就是Vibe Coding的意义所在。你只需要用自然语言描述需求，AI会自动生成正确的命令。但了解一些基本的FFmpeg概念（如-c copy表示直接复制而不重新编码，-vf表示视频滤镜，-af表示音频滤镜）会有助于你和AI的沟通。

8.6 实战案例：FFmpeg MCP工作流

场景：你有一个vlog视频，需要做以下处理：删掉开头3秒、把音量调大30%、加上字幕文件、最后压缩一下文件大小。

传统方式：需要记住4条不同的FFmpeg命令，逐一执行，注意输入输出文件的衔接。

Vibe Coding方式：只需要一句话：

你：帮我把input.mp4处理一下：删掉前3秒，音量调大30%，加上subtitle.srt字幕，最后压缩文件大小。

AI：我来按步骤处理：

步骤1： 删除前3秒ffmpeg -i input.mp4 -ss 3 -c copy step1.mp4步骤2： 音量调大30%ffmpeg -i step1.mp4 -af "volume=1.3" -c:v copy step2.mp4 步骤3： 加上字幕ffmpeg -i step2.mp4 -vf "subtitles=subtitle.srt" -c:a copy step3.mp4步骤4：压缩ffmpeg -i step3.mp4 -vcodec libx264 -crf 28 -preset fast final.mp4✅ 处理完成！输出文件：final.mp4原始大小：156MB处理后大小：42MB压缩率：73%本章小结

1. FFmpeg是视频处理的底层基础设施，几乎所有视频软件都在调用它。
2. FFmpeg命令复杂难记，但AI通过MCP服务可以自动生成和执行这些命令。
3. 主要的视频MCP服务包括：ffmpeg-mcp（video-creator版）、Video_Editor_MCP、

sora-2-mcp等。

4. 你只需要用自然语言描述需求（如"删前5秒""加字幕""调音量"），AI会自动翻译成

FFmpeg命令。

5. 建议在Claude Code中配置至少一个FFmpeg MCP服务，它是所有视频操作的基础。

动手试试练习1：在Claude Code中配置ffmpeg-mcp（或其他视频MCP服务），确认能列出可用工具。

练习2：准备一个视频文件，用自然语言让AI完成以下操作：查看视频信息 -> 截取前10秒 ->调大音量 -> 压缩文件。

第9章 Skill 技能包——给AI装上"剪辑师大脑"

9.1 Skill 机制详解

在前面第2章的"厨房"类比中，我们把Skill比作"菜谱"。现在来深入理解这个机制。

Skill本质上是一份结构化的指南文件，告诉AI助手如何完成某个特定的任务。它通常包含：

任务描述：这个Skill是做什么的前置条件：需要什么环境、什么工具操作步骤：详细的执行流程参数说明：每个配置项的含义和推荐值示例：具体的输入输出示例最佳实践：经验总结和注意事项

Skill 文件格式

Skill通常以 SKILL.md 文件的形式存在。在Claude Code中，Skill放在 ~/.claude/skills/ 目录下，每个Skill一个子目录。

Claude Code 的Skill目录结构

~/.claude/skills/├── videocut/ # videocut-skills │ ├── SKILL.md │ └── ...

├── jianying-editor/ # jianying-editor-skill │ ├── SKILL.md │ ├── rules/ │ └── ...

└── my-custom-skill/ # 你自己的Skill └── SKILL.md当你在Claude Code中输入 /skillname:command格式的消息时，Claude Code会自动加载对应的Skill文件，并按照其中的指南执行任务。

9.2 视频剪辑相关 Skill 全景

9.2.1 videocut-skills

GitHub: https://github.com/Ceeon/videocut-skills Stars: 1,500+ 这是一个专为口播视频剪辑设计的Skill。它的核心能力是：用Claude的语义理解能力来识别口播中的问题（口癖、重复句、停顿），然后自动剪辑。

主要功能：

表9-1 videocut-skills 功能一览

功能	说明	剪映对比
语义理解剪辑	AI逐句分析，识别重说/纠正/卡顿	剪映只能模式匹配
静音检测	大于0.3秒自动标记，阈值可调	剪映固定阈值
重复句检测	相邻句开头≥5字相同则删除前句	无此功能
句内重复检测	如"好我们接下来好我们接下来做"自动删重复	无此功能
词典纠错	自定义专业术语词典，避免语音识别错误	无此功能
字幕生成	基于Whisper生成字幕，支持词典纠错	基础字幕功能
自更新能力安装方式：	记住你的偏好，越用越准	无此功能
# macOS / Linux		

git clone https://github.com/Ceeon/videocut-skills.git ~/.claude/skills/videocut

Windows

git clone https://github.com/Ceeon/videocut-skills.git%USERPROFILE%.claude\skills\videocut

然后在Claude Code中输入：

/videocut:安装

AI会自动安装依赖并下载模型（约5GB）

9.2.2 jianying-editor-skill

GitHub: https://github.com/luoluoluo22/jianying-editor-skill这是国内创作者最需要关注的一个Skill。它可以让AI直接操控剪映专业版，完成从素材导入到视频导出的全套操作。

核心特点：

不是替代剪映：而是把剪映当成底层执行器，AI负责"自动点击"覆盖剪映的几乎所有功能：素材导入、时间轴排列、字幕、配音、特效、转场、导出文件注入驱动：AI直接操作剪映底层的草稿数据文件，而非模拟鼠标点击支持网页动效转视频：可以用HTML/JS/Canvas写动画，自动导入剪映支持录屏+智能变焦：录制屏幕操作，自动给鼠标位置加缩放和标记功能清单：

表9-2 jianying-editor-skill 功能清单

项目	详情
功能类别具体能力	
素材操作导入视频、音频、图片，自动排列到时间轴	
AI配音输入文案自动生成语音（支持剪映原生音色和微软语音）	
字幕生成自动拆句、逐句对齐，支持打字机等动画效果	
配乐本地音乐或剪映素材库云端音乐	
特效/转场/滤镜按名字搜索剪映特效库，一句话应用	
关键帧动画缩放、位移、透明度关键帧，做出运镜效果	
网页动效转视频	HTML/JS/Canvas动画自动录屏导入剪映
录屏+智能变焦录制屏幕操作，自动加缩放和红圈标记	
影视解说	AI分析视频内容，自动生成分镜脚本
自动导出剪完直接导出	MP4，支持1080P到4K

避坑指南： jianying-editor-skill的自动导出功能目前只支持剪映专业版5.9及以下版本。

6.0+版本有大量弹窗会干扰自动化脚本。使用时请务必注意版本兼容性。

安装方式：

Claude Code

git.claude/skills/jianying-editor# Trae IDE git.trae/skills/jianying-editor	clone clone	https://github.com/luoluoluo22/jianying-editor-skill.git https://github.com/luoluoluo22/jianying-editor-skill.git
# Cursor / VSCode git skills/jianying-editor	clone	https://github.com/luoluoluo22/jianying-editor-skill.git

9.2.3 vectcut-skill（VectCutAPI Skill封装）

GitHub: https://github.com/sun-guannan/VectCutAPI（含vectcut-skill子目录）

Stars: 800+VectCutAPI是一个面向AI Agent的视频编辑API/Skill项目，支持剪映/CapCut（国际版剪映）。

它提供了一整套云端编辑API，同时也有本地Skill封装。

核心能力：

草稿管理：创建和保存剪映/CapCut草稿文件视频处理：导入、剪辑、转场、特效音频编辑：多音轨、音量控制、音效处理字幕系统：SRT字幕导入、样式设置、时间同步特效引擎：视觉特效、滤镜、转场动画关键帧动画：属性动画、时间线控制、缓动函数VectCutAPI的独特之处在于它同时提供了HTTP API和MCP协议两种接口，可以集成到各种Agent环境中（Coze、Dify、N8N、OpenClaw、Claude Code等）。

9.2.4 其他视频相关Skill

根据腾讯云开发者社区的整理（截至2026年4月），生态中还有以下视频相关Skill值得关注：

remotion-dev/skills：Remotion官方维护的Skill集合，包含最佳实践和常用模板Youtube-clipper-skill（by op7418）：YouTube视频剪辑相关Skill bibigpt-skill（by JimmyLv）：与BibiGPT（视频摘要工具）集成的Skill narrator-ai-cli-skill（by jieshuo-ai）：解说视频生成相关Skill remotion-best-practices（by openclaw）：Remotion最佳实践Skill

9.3 如何安装 Skill

通用的Skill安装步骤（以Claude Code为例）：

1. 找到Skill的GitHub仓库
2. 克隆到对应的skills目录：git clone [仓库地址] ~/.claude/skills/[skill-name]
3. 在Claude Code中使用 /skillname:command 格式触发
4. 按照AI的提示完成后续安装步骤

9.4 如何自己写/修改 Skill

写Skill不需要编程能力——它是用自然语言写的指南文件。以下是一个简单的Skill模板：

---name: my-video-skill description: 一个自定义的视频处理Skill---

My Video Skill

目标

帮助用户快速处理口播视频，包括格式转换和基础优化。

前置条件

- FFmpeg已安装
- 输入视频格式为MP4或MOV

操作步骤

1. 检查输入文件

确认文件存在且格式正确。如果不正确，先进行格式转换：

ffmpeg -i input.avi -c:v libx264 -c:a aac output.mp4

2. 基础优化

执行以下标准优化流程：

- 压缩文件大小（CRF 28）
- 标准化音频音量
- 生成低分辨率预览版

3. 输出文件

- 主文件：output_optimized.mp4
- 预览文件：output_preview.mp4

参数说明

- CRF值：默认28（范围18-30，越小质量越高文件越大）
- 预览分辨率：480p

示例

输入：raw_video.mp4（500MB）

输出：raw_video_optimized.mp4（45MB）+ raw_video_preview.mp4（5MB）

9.5 Skill 的局限性和注意事项

版本兼容性：Skill依赖特定版本的底层工具，升级后可能失效平台差异：有些Skill只在特定平台测试过（如jianying-editor-skill主要在Windows上开发）

安全性：安装第三方Skill前建议检查其内容，避免执行恶意命令维护状态：开源项目可能停止维护，使用前要检查最近更新日期

9.6 特别话题：剪映Skill vs 纯代码方案，怎么选?

表9-3 剪映Skill vs 纯代码方案对比

维度	剪映Skill（jianying-editor）	纯代码方案（FFmpeg+代码框架）
上手难度	低（保留剪映的图形化预览和导出）	中（需要理解命令行操作）
特效丰富度	极高（剪映海量特效库）	有限（需要自己实现或调用外部库）
批量生产能力	中（依赖剪映的导出速度）	极高（纯代码渲染，可并行化）
可定制性	中（受限于剪映提供的功能）	极高（任何效果都可以代码实现）
稳定性	中（依赖剪映版本兼容性）	高（不依赖第三方GUI软件）
国内适用性	很好（剪映是国内最普及的工具）	好（FFmpeg全球通用）
推荐场景	日常口播剪辑、需要丰富特效	批量生产、复杂定制、自动化流水线

小贴士： 两者不是互斥的！最佳实践是：日常快速剪辑用剪映Skill（利用剪映的特效库和导出体验），批量生产或需要深度定制的场景用纯代码方案。你可以在同个项目中同时使用两种方式。

本章小结

1. Skill是告诉AI如何完成任务的"菜谱"，以SKILL.md文件形式存在。
2. 核心视频Skill包括：videocut-skills（口播剪辑）、jianying-editor-skill（剪映自动化）、vectcut-skill（CapCut API）。
3. videocut-skills 1,500+ stars，强项是语义理解剪辑和词典纠错。
4. jianying-editor-skill让AI操控剪映，覆盖从素材导入到导出的全流程，是国内的刚需工具。
5. Skill可以自己写，用自然语言描述步骤即可，不需要编程能力。
6. 剪映Skill和纯代码方案各有优劣，可以组合使用。

动手试试练习1：安装videocut-skills，用它处理一个你自己的口播视频，观察AI如何识别口癖和重复句。

练习2：浏览jianying-editor-skill的GitHub仓库，尝试在Claude Code中安装并使用它完成一次简单的视频导入和导出操作。

第10章 AI编程助手选择指南——Claude Code vsCursor vs Kiro

10.1 三大主流AI编程助手对比

前面章节多次提到了Claude Code、Cursor和Kiro。它们是你视频制作工作流中的"总指挥官"——你通过它们与所有工具交互。选择一个适合自己的AI助手非常重要。

表10-1 三大AI编程助手对比（截至2026年4月）

维度	Claude Code	Cursor	Kiro
出品公司	Anthropic	Anysphere	Amazon AWS
界面类型	终端（CLI）+ VS Code扩展	AI原生IDE（VS Code fork）	AI原生IDE（VS Code fork）
底层模型	Claude Sonnet/Opus	Claude/GPT/Gemini（多模型）	Claude Sonnet/Nova（via Bedrock）
MCP支持 Skill支持	原生支持原生支持（~/.claude/skills/）	支持（需配置）支持（多种加载方式）	支持支持
上下文窗口	1M tokens	~272k tokens	128K tokens
免费额度 GitHub Stars 核心优势	有限（需Pro订阅$20/月） 30,000+Agent能力最强，终端原生	有限（Pro $20/月）未公开（产品型公司）IDE体验最好，迭代快	50 credits/月 + 500注册奖励未公开规格驱动，AWS生态整合
主要劣势	纯终端，无图形界面	价格较高，Ultra $200/月	生态较新，社区较小

10.2 各自的优势场景

Claude Code——Agent能力最强

Claude Code是Anthropic在2025年初推出的终端AI助手，经过一年多的迭代，已经成为AI编程工具领域的标杆。2026年Q1，Anthropic发布了120+个新功能，包括Claude Code的40+次更新。

优势场景：

自主执行复杂任务：给它一个目标，它可以自主规划多步操作，全程不需要人工介入大型代码库操作：1M token的上下文窗口，可以一次性理解整个项目CI/CD集成：原生支持在持续集成环境中运行多Agent协作：/simplify命令可以启动并行Agent审查代码MCP生态最成熟：原生MCP支持，配置最简单

Cursor——IDE体验最好

Cursor是目前最受欢迎的AI原生IDE，基于VS Code构建，所以继承了VS Code的所有插件和快捷键。它的Supermaven自动补全模型有72%的接受率，业界最高。

优势场景：

日常编码工作：图形界面，有Tab补全、inline diff等便捷功能快速迭代：Composer模式可以做多文件编辑，反馈速度快多模型选择：可以在Claude、GPT、Gemini之间切换背景Agent：可以在后台运行多个Agent任务

Kiro——规格驱动开发

Kiro是AWS在2025年中推出的AI IDE，核心理念与其他两家完全不同：先写规格文档，再生成代码。它会在写代码之前先生成requirements.md、design.md、tasks.md三份文档。

优势场景：

复杂功能开发：规格驱动方法能有效减少后期返工团队协作：规格文档就是沟通媒介，降低协作成本AWS生态用户：原生整合AWS服务企业级项目：Agent Hooks可以自动化测试、lint、文档更新

10.3 推荐组合方案

表10-2 按用户类型推荐方案

用户类型	推荐工具	理由
纯小白/首次尝试	Claude Code + video-use	video-use内置Skill，零配置上手
口播博主	Claude Code + videocut-skills	语义理解剪辑能力最强
剪映重度用户	Claude Code/Cursor + jianying-editor-skill	保留剪映习惯，AI接管重复操作
批量生产需求	Claude Code + Remotion/HyperFrames	代码驱动，适合自动化流水线
习惯图形界面	Cursor + 各种MCP/Skill	IDE体验最好，学习成本低
AWS生态用户	Kiro + 视频工具链	深度整合AWS服务
专业开发者	Claude Code（CLI）+ Cursor（IDE）	两者互补，各取所长

10.4 其他可选工具简介

Windsurf（by Codeium）：$15/月，Cascade功能提供自主编码流， generous free tier GitHub Copilot：$10/月，支持40+ IDE，生态最广Trae（by ByteDance）：免费，国内访问友好，支持Skill Antigravity：国内开发者社区活跃的AI IDE OpenClaw：开源的Claude Code替代品国内用户提示： 国内

用户特别提示：Claude Code和Cursor在国内访问可能需要网络加速工具。如果你遇到访问问题，可以考虑以下替代方案：

1. Trae（字节跳动出品，国内可直接访问，免费）
2. 通过API代理服务使用Claude Code（如OfoxAI等API网关）
3. GitHub Copilot（国内访问相对稳定）

本章小结

1. Claude Code（Anthropic）：Agent能力最强，终端原生，MCP生态最成熟，推荐作为

视频制作首选。

2. Cursor（Anysphere）：IDE体验最好，图形界面，适合习惯VS Code的用户。
3. Kiro（AWS）：规格驱动开发，适合复杂项目和AWS生态用户。
4. 初学者推荐从Claude Code + video-use开始，剪映用户推荐jianying-editor-skill。
5. 国内用户可以考虑Trae作为访问友好的替代方案。

动手试试练习1：如果你还没有安装AI助手，根据你的使用习惯选择Claude Code、Cursor或Trae中的一个，完成安装和认证。

练习2：在你选择的AI助手中，尝试用自然语言让它帮你列出"当前可用的所有工具和能力"，观察它如何展示自己的能力范围。

PART THREE

实战篇

从素材到成品的完整工作流

第11章 口播博主实战——录一段话，AI帮你变成精品视频

第12章 知识分享实战——文稿/PPT变动画讲解视频

第13章 产品演示实战——让产品自己"说话"

第14章 批量生产实战——一套模板，量产100个视频

第11章 口播博主实战——录一段话，AI帮你变成精品视频

11.1 场景描述

你是一位知识分享类博主，刚录制完一段20分钟的教学口播。原始素材中存在大量口癖（"嗯""啊""那个""就是说"）、停顿、重复表述。你的目标是在30分钟内把它变成一条3分钟的精品短视频，配字幕、加片头片尾、调音量，可以直接发布到抖音/小红书/B站。

11.2 完整工作流

步骤1：素材准备与导入

创建项目文件夹

mkdir ~/video-projects/tutorial-001 cd ~/video-projects/tutorial-001

把原始视频素材复制进来

cp /path/to/your/raw-recording.mp4 ./input.mp4

验证文件

ls -lh input.mp4

确认文件大小和格式正常

在Claude Code中：

你：我有一份口播素材input.mp4，大约20分钟，需要剪辑成精华版。

请帮我分析一下这个视频的基本信息。

AI：我来分析视频信息...

视频时长：19分42秒分辨率：1920x1080帧率：30fps音频采样率：48kHz文件大小：2.3GB建议处理方案：

1. 先进行语音转文字，识别口癖和停顿位置
2. 基于语义分析提取精华段落
3. 自动剪辑并生成字幕

是否开始?

你：开始吧。

步骤2：语音转文字 + 智能断句

AI：正在使用Whisper模型进行语音转文字...

转写完成!

- 总时长：19分42秒
- 识别到的文字数：3,847字
- 检测到句末停顿：312处
- 平均语速：195字/分钟

正在生成带时间轴的字幕文件...

✅ 字幕文件已生成：output/transcript.srt✅ 带时间戳的文本：output/transcript.txt

步骤3：去口癖

AI：正在分析口癖...

检测结果：

- "嗯"：127次
- "啊"：83次
- "那个"：45次
- "就是说"：29次
- 其他填充词：38次
- 总填充词：322次（占说话时间的12%）

正在使用语义理解进行智能剪辑...

- 删除纯填充词（不影响语义的）：289次

- 保留必要的停顿（段落过渡等）：33次
- 预计删除后时长：约7分钟

是否继续?

你：继续，目标控制在3分钟左右。

步骤4：内容精选

AI：正在基于语义进行内容精选...

原始内容结构分析：

1. 开场白（0:00-1:30）-> 标记为"可精简"
2. 核心观点A（1:30-5:20）-> 保留
3. 过渡闲聊（5:20-7:10）-> 删除
4. 核心观点B（7:10-11:40）-> 保留精华部分
5. 举例说明（11:40-14:30）-> 保留最有代表性的1个例子
6. 补充内容（14:30-17:20）-> 删除（与核心观点重复）
7. 总结（17:20-19:42）-> 保留核心总结句

预计剪辑后时长：2分50秒保留内容比例：约14%剪辑策略已生成，是否确认执行?

你：确认。

步骤5：加字幕

AI：正在为剪辑后的视频生成字幕...

字幕样式设置：

- 字体：Noto Sans CJK SC（黑体）
- 字号：24pt
- 颜色：白色（#FFFFFF）
- 描边：黑色2px（确保在任何背景下清晰）
- 位置：画面底部（安全区域内）
- 动画：逐字出现（ karaoke 效果）

字幕分段：

- 总句数：42句

- 平均每句字数：18字
- 平均每句显示时长：3.8秒

✅ 字幕已烧录到视频：output/with_subtitles.mp4

步骤6：加片头片尾

你：帮加一个5秒的片头，标题"Python入门指南"，科技感风格。

AI：正在使用HyperFrames生成片头...

[自动生成HTML描述]

[渲染5秒片头动画]

✅ 片头已生成：output/title.mp4正在拼接片头+正文...

✅ 拼接完成：output/with_title.mp4你：再加一个3秒的片尾，显示"关注我看更多教程"和二维码占位。

AI：正在生成片尾...

✅ 片尾已添加：output/with_outro.mp4

步骤7：配背景音乐

AI：正在为视频添加背景音乐...

音频处理参数：

- BGM音量：原音量的20%（确保不盖过人声）
- 淡入：前2秒从0到20%
- 淡出：最后3秒从20%到0%
- 人声保持原始音量

✅ 背景音乐已添加：output/with_bgm.mp4

步骤8：调色/滤镜

AI：正在为视频应用色彩调整...

调色参数：

- 亮度：+5%
- 对比度：+10%
- 饱和度：-5%（让画面更柔和）
- 色温：略微偏暖（+200K）

- 锐化：+15%（口播视频需要清晰的人脸）

✅ 调色完成：output/color_graded.mp4

步骤9：导出成品

AI：正在进行最终导出...

导出参数：

- 格式：MP4（H.264编码）
- 分辨率：1920x1080
- 帧率：30fps
- 视频码率：8Mbps
- 音频码率：192kbps AAC
- 目标平台：全平台兼容

✅ 成品视频已导出：output/final.mp4导出统计：

- 原始素材：19分42秒，2.3GB
- 成品视频：2分58秒，187MB
- 压缩率：92%
- 处理时间：18分钟

11.3 推荐工具组合

表11-1 口播剪辑工具组合

层级	最佳方案	备选方案
AI助手	Claude Code	Cursor
主要Skill	videocut-skills	video-use
视频处理	ffmpeg-mcp	Video_Editor_MCP
片头片尾	HyperFrames	Remotion
字幕生成	Whisper（videocut内置）	剪映AI字幕

11.4 时间与成本对比

表11-2 传统方式 vs AI方式对比		
维度	传统方式（剪映/Premiere手动剪辑）	AI方式（Vibe Coding）
去口癖	手动找、手动删，约2小时	AI自动识别删除，约3分钟
内容精选	反复听、手动裁切，约30分钟	AI语义分析自动精选，约2分钟
加字幕	导出->识别->导入->调样式，约30分钟	一键生成，约2分钟
片头片尾	打开AE做动画，约1小时	AI生成HTML并渲染，约5分钟
总耗时	4-6小时	20-40分钟
人力成本	需要剪辑师专注操作	AI运行期间可做其他事
AI助手费用	0	$20/月（无限使用）

11.5 常见问题与踩坑记录

表11-3 口播剪辑常见问题

问题	原因	解决方案
AI删除了不该删的内容字幕显示不完整音频不同步	语义理解误判字幕超出安全区域剪辑后音画未对齐	在对话中明确告诉AI"保留xxx段落"，或使用更保守的剪辑策略调整字幕字号或分行策略，确保在安全区域内使用FFmpeg的-itsoffset参数微调，或让AI重新处理
背景音乐盖过人声	BGM音量过高	明确指定"BGM音量不超过人声的20%"
导出文件太大	码率设置过高	指定CRF值（如28）或目标文件大小
AI剪辑后逻辑不通	跳剪过于激进	要求AI在剪辑时保持段落间的过渡语句

本章小结

1. 口播剪辑的完整工作流：素材导入 -> 语音转文字 -> 去口癖 -> 内容精选 -> 加字幕 -> 片头片尾 -> 背景音乐 -> 调色 -> 导出。
2. 核心工具组合：Claude Code + videocut-skills + ffmpeg-mcp + HyperFrames。
3. AI方式相比传统方式可将口播剪辑时间从4-6小时压缩到20-40分钟。
4. 关键技巧：在对话中明确表达需求，必要时要求AI保留特定内容。

动手试试练习1：录制一段5分钟的教学口播（故意加一些口癖和停顿），用上述工作流完成完整剪辑，记录每个步骤的耗时。

练习2：对比AI剪辑结果和你手动剪辑（用剪映）的结果，分析AI哪里做得好、哪里需要改进。在下次对话中把改进意见告诉AI，观察它是否能做得更好。

第12章 知识分享实战——文稿/PPT变动画讲解视频

12.1 场景描述

你写了一篇关于"区块链技术原理"的公众号文章，约3000字。现在你想把它变成一个5分钟的动画讲解视频，可以在抖音/B站/视频号上发布。视频需要有画面动画、配音解说、字幕。

12.2 完整工作流

步骤1：文稿 -> 视频脚本

你：我有一篇关于区块链的文章，帮我改写成视频脚本。

要求是：5分钟左右的解说词，口语化表达，每段配一个画面描述。

[粘贴文章内容]

AI：正在将文章改写成视频脚本...

视频脚本：

---【画面1：片头动画，科技感背景，标题出现】

（0:00-0:05，5秒）

配音："今天5分钟，搞懂区块链到底是什么。"【画面2：一个传统账本的动画，纸页翻动】

（0:05-0:45，40秒）

配音："我们先从一个简单的比喻开始。想象一下，...

[继续生成完整脚本]

---✅ 脚本已生成，共5分12秒，分9个画面段落。

步骤2：脚本 -> 分镜

你：帮我把每个画面写成具体的分镜描述，要足够详细让AI能生成。

AI：分镜描述：

画面1（0:00-0:05）：片头

- 背景：深蓝色渐变（#0a1628到#1a3a5c）
- 中心：大标题"区块链原理"，白色无衬线字体，60px
- 动画：标题从下方滑入，带淡入效果

- 底部：小字"5分钟搞懂系列"

画面2（0:05-0:45）：传统账本比喻

- 背景：浅色纹理（模拟纸张）
- 主体：一个翻开的账本，左侧显示交易记录
- 动画：手写风格的文字逐行出现
- 标注："中心化的记账方式"

[继续生成所有分镜]

✅ 分镜描述已生成，共9个画面。

步骤3：分镜 -> 动画帧（HyperFrames/Remotion生成）

AI：正在使用HyperFrames生成分镜动画...

[画面1] 生成片头... ✅ 0:05[画面2] 生成账本动画... ✅ 0:40[画面3] 生成区块链示意图... ✅ 0:45[画面4] 生成分布式网络图... ✅ 0:38[画面5] 生成哈希计算动画... ✅ 0:42[画面6] 生成区块链接动画... ✅ 0:50[画面7] 生成挖矿过程示意... ✅ 0:35[画面8] 生成应用场景展示... ✅ 0:45[画面9] 生成片尾... ✅ 0:05所有分镜已生成，正在拼接...

✅ 拼接完成：output/visual_track.mp4（5分05秒）

步骤4：配音（TTS）

AI：正在为脚本生成配音...

使用ElevenLabs TTS API：

- 声音风格：专业男声（中文）
- 语速：适中（约200字/分钟）
- 语调：自然、有起伏

逐段生成：

段落1/9... ✅段落2/9... ✅...

段落9/9... ✅音频总时长：5分08秒✅ 配音文件：output/narration.mp3

步骤5：合成与调整

AI：正在合成视频...

合成步骤：

1. 对齐画面和配音（使用音频波形自动对齐）

2. 添加字幕（根据配音文本自动生成，同步时间轴）

3. 添加过渡效果（画面间淡入淡出）

4. 音量平衡（配音100%，背景音乐15%）

5. 最终渲染

✅ 成品视频：output/final_video.mp4时长：5分08秒文件大小：245MB

12.3 实战案例：把一篇公众号文章变成讲解视频

假设原文标题是《Web3.0是什么？一文读懂下一代互联网》，以下是关键步骤的对话实录：

你：帮我把这篇文章改成5分钟的视频脚本。

AI：分析完成。原文3200字，预计需要压缩到约1000字的口语化解说词。

改写要点：

- 删除过于技术化的术语解释
- 增加类比和比喻
- 每60-90秒设置一个"钩子"保持注意力

[生成完整脚本]

你：现在为每个段落生成画面描述。

AI：

段落1："想象一下互联网是一场盛大的派对..."画面：一个卡通风格的派对场景，有各种互联网图标（浏览器、App图标）

以拟人化形象出现段落2："Web1.0时代，派对是只读的..."画面：时间轴动画，1990年代风格的网页截图，标"只读"[继续生成]

你：用HyperFrames生成这些画面。

AI：[调用HyperFrames渲染每个画面]

你：加上中文配音。

AI：[调用TTS生成配音]

你：合成最终视频。

AI：[FFmpeg合成所有轨道]

✅ 最终视频已生成！

本章小结

1. 文稿变视频的完整工作流：文稿 -> 脚本 -> 分镜 -> 动画生成 -> 配音 -> 合成。
2. HyperFrames适合生成解说类动画画面（HTML描述，渲染成视频）。
3. Remotion也可以用于此场景，特别是需要数据可视化或复杂动画时。
4. TTS配音目前ElevenLabs的中文效果较好，注意选择合适的声音风格。
5. 最终合成需要精确对齐画面和音频，FFmpeg是核心工具。

动手试试练习1：找一篇你自己的文章（或任意一篇公众号文章），用AI助手将其改写成3分钟的视频脚本。

练习2：选择脚本中的前3个画面，用HyperFrames生成对应的动画片段。

第13章 产品演示实战——让产品自己"说话"

13.1 场景描述

你是一位产品经理，需要为公司的SaaS产品制作一段2分钟的演示视频，用于官网展示和销售拜访。视频需要展示产品的核心功能界面，配上操作解说和字幕。

13.2 完整工作流

步骤1：录屏/截图素材准备

使用系统自带录屏工具或专业录屏软件录制产品操作

录制要点：

- 分辨率：1920x1080

- 帧率：30fps

- 操作节奏：每个功能点展示5-10秒

- 鼠标移动：缓慢、清晰

或者使用 jianying-editor-skill 的录屏功能：

"帮我启动录屏工具，录制一段产品演示"

步骤2：编写解说脚本

你：我需要为这段产品录屏配解说词。产品是一个项目管理工具，核心功能包括：任务看板、甘特图、团队协作、自动化工作流。

帮我写一段2分钟的解说词。

AI：产品演示解说脚本：

（0:00-0:05）片头"欢迎来到TaskFlow，让项目管理变得简单高效。"（0:05-0:30）任务看板"这是我们的智能看板。所有任务一目了然，拖拽即可变更状态，比传统表格直观10倍。"（0:30-0:55）甘特图"一键切换到甘特图视图，项目进度尽在掌握。

关键路径自动高亮，延期风险提前预警。"（0:55-1:20）团队协作"@团队成员，分配任务只需一秒。

实时评论、文件共享，沟通零延迟。"（1:20-1:45）自动化工作流"更强大的是我们的自动化引擎。

当任务状态变更时，自动触发通知、更新报表、甚至集成第三方工具。"（1:45-2:00）片尾CTA"立即免费试用TaskFlow，让团队效率翻倍。

访问 taskflow.com 开始体验。"✅ 脚本已生成，共2分钟。

步骤3：动画包装（画面标注、高亮、缩放）

AI：正在为录屏素材添加动画包装...

处理内容：

1. 添加圆角边框（让画面更精致）
2. 鼠标点击位置添加高亮圆圈
3. 重要区域添加缩放聚焦效果
4. 功能切换处添加过渡动画
5. 底部添加产品Logo水印

✅ 动画包装完成：output/edited_screencast.mp4

步骤4：字幕解说合成

AI：正在生成配音和字幕...

配音参数：

- 声音：专业女声（中文）
- 语速：180字/分钟（产品演示稍慢）
- 风格：自信、亲和

字幕样式：

- 底部居中
- 白色字体+黑色描边
- 逐句出现，句间停顿0.3秒

✅ 配音和字幕已合成：output/with_voiceover.mp4

步骤5：成品导出

AI：正在进行最终导出...

导出设置（针对官网展示优化）：

- 分辨率：1920x1080
- 视频码率：5Mbps（平衡质量和大小）
- 音频：AAC 192kbps
- 格式：MP4

✅ 成品：output/product_demo_final.mp4文件大小：78MB时长：2分00秒小贴士： 产品演示视频的核心是"清晰"而非"花哨"。观众的注意力应该集中在产品功能上，而不是被过多动画分散。建议在包装时保持简洁，重点使用"鼠标高亮""区域放大"等功能性动画。

本章小结

1. 产品演示视频的完整工作流：录屏素材 -> 解说脚本 -> 动画包装 -> 配音字幕 -> 导出。
2. 录屏时保持操作节奏缓慢清晰，分辨率不低于1080p。
3. 动画包装要服务于功能展示（高亮、缩放、标注），避免过度设计。
4. 产品演示的配音风格应该自信、专业、语速适中。

动手试试练习1：选择一个你常用的软件/网站，录制一段30秒的操作演示，然后用AI助手为其添加字幕和简单的画面包装。

第14章 批量生产实战——一套模板，量产100个视频

14.1 场景描述

你运营一个电商账号，需要为100个不同商品各制作一条15秒的展示视频。每条视频的结构相同：

3秒片头（商品名称）+ 8秒产品展示（图片轮播+卖点文字）+ 4秒片尾（价格和CTA）。只有内容不同（商品名称、图片、卖点、价格）。

传统方式：每条视频手动制作，100条视频需要约50小时。

Vibe Coding方式：制作一个模板 + 准备数据文件，批量渲染，约2小时。

14.2 模板化思路

批量生产的核心是固定结构 + 变量替换。具体步骤：

1. 设计一个视频模板（用代码描述视频结构和样式）
2. 准备一个数据文件（JSON/CSV格式，包含每个视频的变量内容）
3. 写一个渲染脚本，读取数据文件，为每条数据生成一个视频
4. 批量执行渲染

14.3 用 Remotion 批量渲染的方法

步骤1：设计模板组件

// src/ProductVideo.tsx import { useVideoConfig, useCurrentFrame, interpolate } from 'remotion';

import { AbsoluteFill, staticFile } from 'remotion';

// 定义props接口 - 这些就是变量interface ProductVideoProps {productName: string;

images: string[]; // 商品图片路径sellingPoints: string[];

price: string;

cta: string; // 行动号召文字}export const ProductVideo: React.FC = ({productName, images, sellingPoints, price, cta}) => {const { fps, durationInFrames, width, height } = useVideoConfig();

const frame = useCurrentFrame();

// 时间线划分（30fps）

// 0-3秒：片头 | 3-11秒：产品展示 | 11-15秒：片尾const showTitle = frame < 3 * fps;

const showProduct = frame >= 3 * fps && frame < 11 * fps;

const showCta = frame >= 11 * fps;

// 产品展示阶段的图片索引const productFrame = frame - 3 * fps;

const imageIndex = Math.floor((productFrame / (8 * fps)) * images.length);

return ({/* *片头：商品名称* /}{showTitle && (

{productName}

)}{/* *产品展示：图片轮播 + 卖点* /}{showProduct && (style={{ width: '100%', height: '70%', objectFit: 'cover' }}/>
{sellingPoints.map((point, i) => (
{point}
))}
)}{/* *片尾：价格和CTA* /}{showCta && (
{price}
{cta}
)});

};

步骤2：准备数据文件

// data/products.json[{"id": "001","productName": "智能保温杯","images": ["products/001_1.jpg", "products/001_2.jpg"],"sellingPoints": ["24小时恒温", "OLED温度显示", "IPX7防水"],"price": "¥199","cta": "立即抢购 >>"},{"id": "002","productName": "无线降噪耳机","images": ["products/002_1.jpg", "products/002_2.jpg", "products/002_3.jpg"],"sellingPoints": ["40dB深度降噪", "36小时续航", "Hi-Res音质"],"price": "¥599","cta": "限时特惠 >>"}// ... 更多商品数据]

步骤3：编写批量渲染脚本

// src/render-batch.ts import { renderMedia, selectComposition } from '@remotion/renderer';

import fs from 'fs';

import path from 'path';

const products = JSON.parse(fs.readFileSync('./data/products.json', 'utf-8'));

async function renderBatch() {for (let i = 0; i < products.length; i++) {const product = products[i];

console.log(`[${i + 1}/${products.length}] 渲染：${product.productName}`);

const composition = await selectComposition({serveUrl: 'http://localhost:3000',id: 'ProductVideo',inputProps: product,});

await renderMedia({composition,serveUrl: 'http://localhost:3000',outputLocation: `output/product_${product.id}.mp4`,inputProps: product,});

console.log(`✅ 完成：output/product_${product.id}.mp4`);

}console.log('🎉 全部渲染完成！');

}renderBatch();

步骤4：执行批量渲染

启动Remotion开发服务器

npm run dev

在另一个终端窗口中运行批量渲染

npx ts-node src/render-batch.ts

输出示例：

[1/100] 渲染：智能保温杯

✅ 完成：output/product_001.mp4

[2/100] 渲染：无线降噪耳机

✅ 完成：output/product_002.mp4

...

[100/100] 渲染：xxx

✅ 完成：output/product_100.mp4

🎉 全部渲染完成！

14.4 成本核算与效率对比

表14-1 批量生产成本对比（100条15秒视频）

维度	手动制作（剪映）	模板化批量（Remotion）
制作时间	约50小时（30分钟/条）	约3小时（含渲染）
人工成本	按100元/小时计算：5000元	约100元（电脑电费）
工具成本	剪映免费	Remotion免费（<$1M收入）
AI助手成本	0	$20/月（Claude Code）
质量一致性	因人而异	100%一致
修改成本	高（需逐条修改）	低（改模板即可）

14.5 进阶：并行渲染

如果100条视频串行渲染太慢，可以使用Remotion的Lambda支持实现并行渲染：

// 使用Remotion Lambda在云端并行渲染 import { renderMediaOnLambda } from '@remotion/lambda';

// 同时启动多个渲染任务 const renderPromises = products.map(product =>renderMediaOnLambda({functionName: 'remotion-render',serveUrl: 'https://your-site.com',composition: 'ProductVideo',inputProps: product,outputBucket: 'your-output-bucket',})

);

// 100个任务并行执行await Promise.all(renderPromises);

小贴士： 并行渲染需要AWS账户和一定的云计算知识。初学者建议先用本地串行渲染，熟悉流程后再考虑云端并行方案。

本章小结

1. 批量生产的核心是"固定结构 + 变量替换"：设计一个代码模板，用数据文件驱动内容变化。
2. Remotion是批量渲染的最佳选择（支持props传递、Lambda并行渲染）。
3. 100条视频的手动制作成本约5000元/50小时，模板化批量成本约100元/3小时。
4. 批量生产适合：矩阵账号、多语言版本、系列内容、电商商品视频等场景。

动手试试练习1：设计一个简单的视频模板（如"每日一句名言"），准备5条测试数据，用Remotion批量渲染5条视频。

练习2：测量批量渲染的时间，估算渲染100条、1000条视频分别需要多长时间，计算成本。

PART FOUR

进阶篇

从能用到好用

第15章 提示词的艺术——怎么"说"AI才能剪出你想要的视频

第16章 工作流编排——打造你的自动化视频流水线

第17章 质量把控——AI剪的视频如何避免"AI味"

第18章 工具选型决策树——我该用什么组合？

第15章 提示词的艺术——怎么"说"AI才能剪出你想要的视频

15.1 视频制作场景的提示词设计原则

在Vibe Coding中，提示词就是你的"剪辑指令"。提示词的好坏直接决定了产出视频的质量。以下是经过大量实践总结的设计原则：

原则1：具体而非模糊

模糊："帮我把视频剪短一点"具体："帮我把视频从15分钟剪到3分钟，去掉所有口癖和停顿，保留核心观点"

原则2：提供上下文

无上下文："加个字幕"有上下文："给这个口播视频加字幕，白色粗体字，放在画面底部安全区域内，字号24pt，逐句出现"

原则3：分步骤而非一次性

复杂任务不要试图用一句话说完。分步骤沟通，每一步确认结果后再继续。

原则4：使用视频专业术语

适当使用术语能让AI更精确地理解你的需求。比如"16:9""1080p""淡入淡出""CRF"等。

原则5：指定输出格式

明确告诉AI你期望的产出物是什么格式、什么参数。

15.2 提示词模板库

模板1：口播剪辑

请帮我剪辑这段口播视频：

- 原始时长：[XX]分钟
- 目标时长：[3-5]分钟
- 必须保留的内容：[列出核心观点/段落]
- 删除规则：去掉所有"嗯""啊""那个""就是说"等填充词，

去掉明显说错重来的部分，去掉与主题无关的闲聊

- 输出要求：保持逻辑连贯，段落间有过渡句

请先生成剪辑方案让我确认，确认后再执行。

模板2：字幕生成

请为这段视频生成字幕并烧录到画面上：

- 语言：中文
- 字体：Noto Sans CJK SC（或[你的偏好字体]）
- 字号：[24]pt
- 颜色：白色（#FFFFFF），带黑色描边[2]px
- 位置：画面底部，安全区域内
- 样式：[逐句出现/逐字出现/直接显示]
- 时间轴精度：与语音严格同步

模板3：动画片头生成

请用HyperFrames帮我生成一个片头动画：

- 时长：[3-5]秒
- 背景：[深色渐变/纯色/图片背景]，具体颜色[描述]
- 标题文字：[你的标题]
- 副标题：[你的副标题，如果有]
- 风格：[科技感/简约/活泼/商务]
- 动画效果：[淡入/滑入/缩放/打字机]
- 额外元素：[Logo/装饰图形/粒子效果]

模板4：调色/风格化

请为这段视频进行调色处理：

- 整体风格：[电影感/明亮清新/暗黑风格/复古]
- 亮度：[+5%/+10%/不变]
- 对比度：[+10%/+15%]
- 饱和度：[降低5%使画面柔和/提高10%更鲜艳]
- 色温：[偏暖+200K/偏冷-100K/中性]
- 锐度：[+15%让人脸更清晰]

- 特殊效果：[暗角/胶片颗粒/光晕]

模板5：批量生产任务

我需要批量生成[XX]条视频，结构如下：

- 模板：固定结构 + 变量替换
- 每个视频的差异：[商品名称/图片/价格/卖点]
- 视频结构：[X]秒片头 + [X]秒主体 + [X]秒片尾
- 分辨率：[1920x1080]
- 目标平台：[抖音/B站/小红书]
- 数据文件：[提供JSON/CSV文件或格式]

请帮我：1）设计模板代码 2）编写批量渲染脚本 3）执行渲染

模板6：视频拼接

请帮我把以下视频按顺序拼接：

1. [video1.mp4] - 片头，[3]秒
2. [video2.mp4] - 主体内容，[需要完整保留]
3. [video3.mp4] - 片尾，[4]秒

- 片段间过渡：[直接拼接/淡入淡出/滑动]
- 音频处理：[保持原音/统一音量/添加BGM]
- 输出分辨率：[与主体视频一致]

模板7：音频处理

请为这段视频处理音频：

- 人声音量：[100%/提高至120%]
- 背景音乐：[添加来自bgm.mp3，音量20%，淡入2秒淡出3秒]
- 噪音处理：[去除底噪/保持原样]
- 音频格式：AAC 192kbps
- 特殊要求：[确保人声清晰可辨]

模板8：格式转换与压缩

请帮我处理这段视频的文件格式和大小：

- 目标格式：[MP4(H.264)/WebM/MOV]
- 目标分辨率：[1920x1080/1280x720]
- 目标文件大小：[不超过50MB]
- 质量要求：[社交媒体发布/邮件附件/存档]
- 如需要压缩：优先降低码率，其次降低分辨率

模板9：数据可视化视频

请帮我做一个数据可视化动画视频：

- 数据：[提供JSON/CSV数据]
- 图表类型：[柱状图/折线图/饼图/混合图表]
- 动画效果：[柱子依次升起/线条绘制/扇形展开]
- 时长：[3-5]秒
- 配色：[指定配色方案]
- 文字标签：[显示数值/百分比/类别名称]

模板10：录屏视频包装

请帮我包装这段产品录屏视频：

- 添加圆角边框：[16px圆角]
- 鼠标高亮：[点击时显示黄色圆圈]
- 区域缩放：[操作重点区域放大1.5倍]
- 添加水印：[右下角显示Logo，透明度50%]
- 添加简介卡片：[开始处显示产品名称和版本]

模板11-15：更多场景

模板11：多语言字幕

请为这段视频添加[英文/日文/韩文]字幕，基于中文原文翻译，保持字幕风格与中文版一致。

模板12：视频截图提取

请从这段视频中提取[5]张关键帧截图，时间点：[片头/核心内容/结尾]，分辨率：[1920x1080]，用于[封面图/缩略图/文章配图]。

模板13：速度调整

请将这段视频中[0:30-1:00]的片段加速到[1.5]倍，其余部分保持原速，确保音频不变调。

模板14：画面裁剪

请将这段[竖屏9:16]视频转换为[横屏16:9]，使用[中心裁剪/智能跟踪/模糊背景填充]方式。

模板15：直播回放处理

请帮我处理这段直播回放：[2]小时时长，提取[5]个高光片段（每个[1-2]分钟），添加章节标记，生成带时间戳的目录。

15.3 好提示词 vs 坏提示词对比

表15-1 提示词对比示例

场景	坏提示词	好提示词
剪辑	"剪短点"	"从15分钟剪到3分钟，去掉口癖，保留第3段和第5段的核心内容"
字幕	"加个字幕"	"加白色黑边字幕，24pt，放底部，用思源黑体，逐句出现"
BGM	"配个音乐"	"添加轻快的背景音乐，音量不超过人声的20%，前2秒淡入"
导出	"导出视频"	"导出MP4(H.264)，1080p，码率5Mbps，文件不超过100MB"
调色	"调好看点"	"电影感调色，对比度+15%，色温偏暖，加轻微暗角"

15.4 迭代技巧

第一次对话很少能产出完美的视频。掌握迭代技巧至关重要：

技巧1：先框架后细节

第一轮对话先让AI完成整体结构和流程，后续对话再调整细节。

技巧2：用具体例子说明

如果你不确定怎么描述，可以给AI一个参考："类似这个视频的风格 [链接]" 或 "像B站UP主XXX的片头那样"。

技巧3：保存成功的提示词

把产出效果好的提示词保存下来，形成你自己的"提示词库"。

技巧4：利用AI的自我评估

让AI自己检查输出："你觉得这个视频还有什么可以改进的地方？"本章小结

1. 提示词质量直接决定产出质量。具体、有上下文、分步骤是三大原则。
2. 本章提供了15+个可直接复制的提示词模板，覆盖剪辑、字幕、动画、调色、批量生产等场景。
3. 好提示词的共同特征：包含具体参数（数字、颜色、时长）、明确约束条件、指定输出格式。
4. 迭代优化比一次性完美更重要。先出框架，再调细节。

动手试试练习1：选择一个你最近做过的视频任务，用本章的模板改写成一个"好提示词"版本。

练习2：用不同的提示词描述同一个需求，对比AI产出的差异，找出最有效的话术模式。

第16章 工作流编排——打造你的自动化视频流水线

16.1 什么是工作流编排?

工作流编排就是把多个独立的操作串联成一个端到端的自动化流程。比如"口播剪辑工作流"可能包含10个步骤：转文字 -> 去口癖 -> 精选内容 -> 生成字幕 -> 做片头 -> 配音 -> 加BGM -> 调色 ->压缩 -> 导出。工作流编排让这些步骤自动依次执行。

16.2 实用工作流模板

工作流1：口播视频全自动流水线

触发条件：文件夹中有新的raw_*.mp4文件步骤1：分析视频信息（时长、分辨率、音频质量）

步骤2：语音转文字（Whisper）

步骤3：语义分析（识别口癖、重复句、核心段落）

步骤4：生成剪辑方案（供人工确认，或设置自动确认阈值）

步骤5：自动剪辑（FFmpeg）

步骤6：生成字幕（语音识别 + 样式设置）

步骤7：生成片头（HyperFrames，基于视频主题）

步骤8：TTS配音（如果需要替换原声）

步骤9：添加背景音乐步骤10：调色和压缩步骤11：导出成品到output/目录步骤12：发送通知（邮件/Slack/飞书）

工作流2：每日新闻视频自动生成

触发条件：每天早上8点（定时触发）

步骤1：从RSS/API抓取今日新闻标题和摘要步骤2：AI筛选最重要的5条新闻步骤3：为每条新闻生成30秒解说脚本步骤4：用HyperFrames生成新闻画面（标题+摘要+配图）

步骤5：TTS生成配音步骤6：拼接5条新闻片段步骤7：生成片头（"今日科技新闻"）和片尾步骤8：导出成品步骤9：上传到视频平台（可选，需平台API）

工作流3：电商商品视频批量生产

触发条件：商品数据表有更新（新增商品）

步骤1：读取新增商品数据（名称、图片、价格、卖点）

步骤2：检查商品图片是否存在步骤3：用Remotion模板渲染视频（每个商品一条）

步骤4：质量检查（文件大小、时长、分辨率）

步骤5：上传到商品管理系统步骤6：记录生产日志

16.3 任务编排技巧

技巧1：用项目文件管理上下文

video-use的project.md机制是一个很好的实践：在每个项目文件夹中维护一个project.md文件，记录项目状态、决策历史、待办事项。AI在每次处理前读取这个文件，就能保持上下文连贯。

技巧2：设置检查点

复杂工作流中，在关键步骤设置"人工确认"检查点。比如剪辑方案生成后先让人类确认，再继续执行。

技巧3：错误处理和重试

伪代码示例

try:

result = step_n.execute()

except Exception as e:

记录错误

log.error(f"Step {n} failed: {e}")

尝试自动修复

if can_auto_fix(e):

auto_fix_and_retry()

else:

通知人类介入

notify_human(e)

pause_workflow()

技巧4：并行化独立任务

如果多个步骤之间没有依赖关系，可以并行执行。比如"生成片头"和"处理音频"可以同时进行。

16.4 错误处理与断点续做

工作流中断是常见问题。以下是应对策略：

记录已完成的步骤：在project.md中标记每个步骤的状态幂等设计：每个步骤都可以安全地重复执行（执行两次和执行一次结果相同）

中间产物保存：每个步骤的输出都保存为文件，便于断点续做人类介入机制：复杂决策点预留人工确认接口本章小结

1. 工作流编排 = 把多个操作串联成端到端自动化流程。
2. 本章提供了3个完整的工作流模板：口播全自动流水线、每日新闻视频、电商批量生产。
3. 编排技巧：用project.md管理上下文、设置人工检查点、错误处理重试、并行化独立任

务。

4. 断点续做是关键能力：记录状态、幂等设计、保存中间产物。

动手试试练习1：设计一个你自己的工作流（基于你日常的视频制作需求），用伪代码或步骤清单描述出来。

练习2：在一个项目中实践project.md机制，记录AI的每一步操作和决策。

第17章 质量把控——AI剪的视频如何避免"AI味"

17.1 常见质量问题清单

表17-1 AI视频常见问题与解决方案

问题	表现	解决方案
跳剪过于生硬	句子之间突然切断，没有过渡	要求AI保留段落间的过渡句，或添加0.5秒淡入淡出
字幕不同步	字幕显示时间和语音不匹配	手动调整关键时间戳，或使用更精确的语音识别模型
音频爆音	剪辑点有"啪"的声音	在每个剪辑点添加30ms的音频淡入淡出
画面拉伸变形	16:9素材被强制转成9:16	明确指定处理方式（裁剪/居中/模糊背景填充）
BGM和人声冲突	背景音乐盖过人声	BGM音量不超过人声20%，选择无人声频段的音乐
字幕被裁切	字幕超出画面边界	确保字幕在安全区域内（画面高度的85%以下）
色彩不统一	多个片段色调差异大	统一调色参数，或使用LUT预设
片头过于花哨	动画太多分散注意力	简化片头动画，3秒内完成，突出标题即可

17.2 人工复核检查表（导出前必查10项）

1. 时长检查：确认视频时长符合目标平台要求
2. 字幕完整性：从头到尾看一遍字幕，确认没有遗漏或错别字
3. 音频质量：确认人声清晰，BGM不干扰，没有爆音
4. 画面流畅度：检查剪辑点是否自然，有没有画面卡顿
5. 色彩一致性：确认整个视频色调统一
6. 文字可读性：字幕和重要文字在各种背景下都清晰可读
7. Logo和品牌元素：确认品牌标识位置和大小合适
8. 首帧和末帧：确认视频开头和结尾没有黑帧或异常画面
9. 文件大小：确认文件大小适合目标发布平台
10. 格式兼容性：确认MP4编码在各平台都能正常播放

17.3 提升质感的实用技巧

统一的色彩风格：为系列视频定义一套调色预设，保持一致性专业的字体选择：使用思源黑体、阿里巴巴普惠体等高质量中文字体精心选择BGM：使用Epidemic Sound、Artlist等平台的免版权音乐控制信息密度：每屏不超过3个重点信息，避免拥挤留出呼吸空间：段落之间留出1-2秒的过渡时间使用安全区域：重要元素放在画面中央80%的区域内

17.4 什么时候该放弃AI、回到手动剪辑?

诚实地说，AI不是万能的。以下情况建议回到手动剪辑：

情感类内容：需要精细的情绪节奏把控（如婚礼视频、纪录片）

高度创意性作品：艺术短片、实验性视频精确的音画同步：音乐MV、舞蹈视频等对节拍要求极高的内容客户定制项目：客户有非常具体且频繁的修改意见AI多次尝试仍不满意：如果迭代3次后效果仍然不理想，手动可能更快小贴士： 最好的工作流不是"全AI"也不是"全手动"，而是"AI做批量和标准化工作，人类做创意和质量把控"。让AI处理80%的机械劳动，你把精力投入到20%的关键决策上。

本章小结

1. AI视频的常见问题包括：跳剪生硬、字幕不同步、音频爆音、画面变形等，都有对应的

解决方案。

2. 导出前必查10项：时长、字幕、音频、流畅度、色彩、可读性、品牌、首尾帧、文件大

小、格式兼容。

3. 提升质感的关键：统一色彩、专业字体、优质BGM、控制信息密度。

4. 知道什么时候用AI、什么时候手动，是专业创作者的成熟标志。

动手试试练习1：用本章的检查表审查一个你之前用AI制作的视频，记录发现的问题和改进方案。

第18章 工具选型决策树——我该用什么组合?

18.1 决策流程图

面对众多工具，如何选择适合自己的组合？以下是按需求类型的推荐：

场景A：口播博主

核心需求：去口癖、加字幕、精简内容推荐组合：

1. AI助手：Claude Code
2. 核心Skill：videocut-skills
3. MCP：ffmpeg-mcp
4. 片头片尾：HyperFrames
5. 备选：jianying-editor-skill（如果需要剪映特效）

场景B：知识博主

核心需求：文稿变动画视频、数据可视化、片头片尾推荐组合：

1. AI助手：Claude Code
2. 动画框架：HyperFrames（简单动画）或 Remotion（复杂动画）
3. MCP：ffmpeg-mcp
4. TTS：ElevenLabs
5. Skill：remotion-dev/skills（如果用Remotion）

场景C：产品经理

核心需求：产品演示视频、录屏包装、快速迭代推荐组合：

1. AI助手：Cursor（IDE体验好）
2. 录屏工具：jianying-editor-skill的录屏功能 或 系统自带录屏
3. 动画包装：HyperFrames
4. MCP：ffmpeg-mcp

场景D：批量生产

核心需求：矩阵账号、电商视频、系列内容推荐组合：

1. AI助手：Claude Code
2. 渲染引擎：Remotion（模板+数据驱动）
3. 批量脚本：自定义渲染脚本
4. 云端渲染：Remotion Lambda（大规模并行）
5. MCP：ffmpeg-mcp

场景E：纯小白

核心需求：最低学习成本，最快上手推荐组合：

1. AI助手：Claude Code
2. 一体化工具：video-use（内置所有功能）
3. 不需要额外配置MCP或Skill

18.2 成本对比表

表18-1 三种方案的成本对比（月度）

总成

方案本全免费方案 基础付费方案	AI助手Trae（免费） Claude Code（$20）	工具费用全部开源工具（0） ¥0 开源工具（0）~¥150	适合预算敏感，愿意花时间学习大多数创作者
效率优先方~¥320案	Claude Code（$20）+ Cursor（$20）	ElevenLabs（$5）	专业创作者，时间就是金钱

18.3 国内用户特别推荐

考虑到国内用户的网络环境和工具可用性：

AI助手：Trae（字节跳动出品，国内直接访问）或 通过API代理使用Claude Code剪辑工具：jianying-editor-skill（操控剪映，国内最普及）

视频处理：ffmpeg-mcp（FFmpeg全球通用，不受网络影响）

云存储：阿里云OSS/腾讯云COS（替代AWS S3）

TTS配音：百度语音合成、阿里云语音合成（替代ElevenLabs）

国内用户提示： 国内用户路线建议：Trae + jianying-editor-skill + ffmpeg-mcp。这个组合全部可以在国内网络环境下使用，剪映也是国内最成熟的视频工具。如果需要生成动画画面，可以用HyperFrames（需要GitHub访问，可以用Gitee镜像）。

本章小结

1. 不同用户类型有不同的最佳工具组合。口播博主用videocut-skills，知识博主用HyperFrames/Remotion，批量生产用Remotion模板。
2. 成本从0元/月到约320元/月不等，大多数创作者150元/月的基础方案足够。
3. 国内用户推荐Trae + jianying-editor-skill的组合，网络无障碍。
4. 选择工具的核心原则：从自己的实际需求出发，不要贪多求全。

动手试试练习1：根据本章的决策树，确定你自己的工具组合方案。列出你需要安装的工具清单。

练习2：计算你的月度成本预算，选择最适合你的付费方案。

PART FIVE

附录

速查手册 · 模板库 · 报错 · 资源 · 术语

附录A 全工具安装配置一站式速查手册

附录B 提示词模板库（可直接复制使用）

附录C 常见报错与解决方案 Top 20

附录D 推荐学习资源

附录E 术语表（中英对照）

附录A 全工具安装配置一站式速查手册

附录

A.1 Git

项目详情官网https://git-scm.com macOS安装brew install git Windows安装https://git-scm.com/download/win验证git --version

A.2 Node.js

项目详情官网https://nodejs.org macOS安装brew install node Windows安装官网下载安装程序验证node --version && npm --version国内镜像npm config set registry https://registry.npmmirror.com

A.3 Python

项目详情官网https://python.org macOS安装brew install python（或系统自带python3）

Windows安装官网下载，勾选"Add to PATH"验证python3 --version（或 python --version）

A.4 FFmpeg

项目详情官网https://ffmpeg.org GitHub https://github.com/FFmpeg/FFmpeg macOS安装brew install ffmpeg Windows安装choco install ffmpeg（或官网下载解压后加PATH）

验证ffmpeg -version

A.5 Claude Code

项目详情官网https://docs.anthropic.com/en/docs/claude-code GitHub https://github.com/anthropics/claude-code（30,000+ stars）

macOS安装brew install claude-code Windows安装npm install -g @anthropic-ai/claude-code验证claude --version登录claude（按提示浏览器认证）

费用Pro $20/月，Max $100-200/月，Team $30/座位/月Skill目录~/.claude/skills/（macOS/Linux）

MCP配置~/.config/claude-code/mcp_settings.json

A.6 Cursor

项目详情官网https://cursor.com下载官网下载（macOS/Windows/Linux）

费用Pro $20/月，Pro+ $60/月，Ultra $200/月

A.7 Kiro

项目详情官网https://kiro.dev下载官网下载（macOS/Windows/Linux）

费用Free 50 credits/月，Pro $20/月，超出$0.04/credit

A.8 HyperFrames

项目详情GitHub https://github.com/heygen-com/hyperframes（12,700+ stars）

最新版本v0.4.34（2026年4月28日）

安装git clone + bun install（或 npm install）

依赖bun 或 npm，Puppeteer License Apache 2.0

A.9 Remotion

项目详情GitHub https://github.com/remotion-dev/remotion（45,100+ stars）

官网https://remotion.dev最新版本v4.0.454（2026年4月28日）

安装npx create-video@latest my-project依赖Node.js 18+License Business Source License（<$1M/年免费）

A.10 Revideo

项目详情GitHub https://github.com/redotvideo/revideo安装npm init @revideo@latest my-project License MIT

A.11 video-use

项目详情GitHub https://github.com/browser-use/video-use（5,400+ stars）

最新提交2026年4月24日安装git clone + pip install（详细见仓库README）

特点内置SKILL.md，一键设置

A.12 ffmpeg-mcp

项目详情GitHub https://github.com/video-creator/ffmpeg-mcp安装git clone + uv sync（或 pip install ffmpeg-mcp）

功能视频搜索、裁剪、拼接、格式转换

A.13 Video_Editor_MCP

项目详情GitHub https://github.com/Kush36Agrawal/Video_Editor_MCP功能基于FFmpeg的视频编辑MCP服务

A.14 videocut-skills

项目详情GitHub https://github.com/Ceeon/videocut-skills（1,500+ stars）

安装git clone到~/.claude/skills/videocut触发/videocut:安装模型下载约5GB

A.15 jianying-editor-skill

项目详情GitHub https://github.com/luoluoluo22/jianying-editor-skill安装git clone到对应IDE的skills目录兼容IDE Antigravity, Trae, Claude Code, Cursor, VSCode等注意事项仅支持剪映专业版5.9及以下版本

A.16 VectCutAPI

项目详情GitHub https://github.com/sun-guannan/VectCutAPI（800+ stars）

支持平台剪映/CapCut（国际版）

接口类型HTTP API + MCP协议

附录B 提示词模板库（可直接复制使用）

B.1 口播剪辑

请帮我剪辑这段口播视频：

- 原始时长：[XX]分钟
- 目标时长：[3-5]分钟
- 必须保留的内容：[列出核心观点/段落]
- 删除规则：去掉所有"嗯""啊""那个""就是说"等填充词，

去掉明显说错重来的部分，去掉与主题无关的闲聊

- 输出要求：保持逻辑连贯，段落间有过渡句

请先生成剪辑方案让我确认，确认后再执行。

B.2 字幕生成

请为这段视频生成字幕并烧录到画面上：

- 语言：中文
- 字体：Noto Sans CJK SC
- 字号：24pt
- 颜色：白色（#FFFFFF），带黑色描边2px
- 位置：画面底部，安全区域内
- 样式：逐句出现
- 时间轴精度：与语音严格同步

B.3 动画片头

请用HyperFrames帮我生成一个片头动画：

- 时长：5秒
- 背景：深色渐变（#0a1628到#1a3a5c）
- 标题文字：[你的标题]

- 副标题：[你的副标题]
- 风格：科技感
- 动画效果：标题从下方滑入，带淡入效果
- 额外元素：底部粒子效果

B.4 调色

请为这段视频进行调色处理：

- 整体风格：电影感
- 亮度：+5%
- 对比度：+15%
- 饱和度：-5%
- 色温：偏暖+200K
- 锐度：+15%
- 特殊效果：轻微暗角

B.5 批量生产

我需要批量生成[XX]条视频，结构如下：

- 模板：固定结构 + 变量替换
- 每个视频的差异：[描述差异内容]
- 视频结构：[X]秒片头 + [X]秒主体 + [X]秒片尾
- 分辨率：1920x1080
- 目标平台：[抖音/B站/小红书]

请帮我：1）设计模板代码 2）编写批量渲染脚本 3）执行渲染

B.6 视频拼接

请帮我把以下视频按顺序拼接：

1. [video1.mp4] - 片头，3秒
2. [video2.mp4] - 主体内容，需要完整保留
3. [video3.mp4] - 片尾，4秒

- 片段间过渡：淡入淡出
- 音频处理：保持原音
- 输出分辨率：与主体视频一致

B.7 音频处理

请为这段视频处理音频：

- 人声音量：100%
- 背景音乐：添加来自bgm.mp3，音量20%，淡入2秒淡出3秒
- 噪音处理：去除底噪
- 音频格式：AAC 192kbps
- 特殊要求：确保人声清晰可辨

B.8 格式转换与压缩

请帮我处理这段视频的文件格式和大小：

- 目标格式：MP4(H.264)
- 目标分辨率：1920x1080
- 目标文件大小：不超过50MB
- 质量要求：社交媒体发布
- 如需要压缩：优先降低码率，其次降低分辨率

B.9 录屏包装

请帮我包装这段产品录屏视频：

- 添加圆角边框：16px圆角
- 鼠标高亮：点击时显示黄色圆圈
- 区域缩放：操作重点区域放大1.5倍
- 添加水印：右下角显示Logo，透明度50%
- 添加简介卡片：开始处显示产品名称和版本

B.10 多语言字幕

请为这段视频添加[英文/日文/韩文]字幕，基于中文原文翻译，保持字幕风格与中文版一致：

- 白色字体，黑色描边
- 24pt字号
- 逐句出现
- 与语音严格同步

B.11 速度调整

请将这段视频中[0:30-1:00]的片段加速到1.5倍，其余部分保持原速，确保音频不变调。

加速片段的音频需要保持清晰可懂。

B.12 画面裁剪（竖屏转横屏）

请将这段竖屏(9:16)视频转换为横屏(16:9)，使用模糊背景填充方式：

- 原视频居中显示
- 背景用原视频放大+高斯模糊填充
- 确保主体内容在安全区域内

B.13 直播回放处理

请帮我处理这段直播回放：

- 原始时长：2小时
- 任务1：提取5个高光片段（每个1-2分钟）
- 任务2：添加章节标记
- 任务3：生成带时间戳的目录
- 输出：完整回放 + 5个高光片段（单独文件）

B.14 产品演示包装

请帮我制作一个产品演示视频：

- 素材：产品录屏 + 产品截图
- 结构：3秒片头（产品名称） + 1分30秒功能演示 + 3秒片尾（CTA）
- 风格：简洁专业，蓝白配色

- 字幕：白色黑边，24pt
- BGM：轻快科技感，音量15%

B.15 数据可视化

请帮我做一个数据可视化动画：

- 数据：[提供JSON/CSV]
- 图表类型：柱状图
- 动画：柱子依次升起
- 时长：5秒
- 配色：蓝色系
- 文字标签：显示数值和类别名称

预计阅读时间：15分钟

附录C 常见报错与解决方案 Top 20

表C-1 常见问题排查

编

号	问题/报错	原因	解决方案
1	ffmpeg: command not found	FFmpeg未安装或未添加PATH	重新安装FFmpeg，确保bin目录在PATH中
2	npm install 卡住/超时	网络问题	npm config set registry https://registry.npmmirror.com
3	git clone 失败/超时	GitHub访问问题	使用ghfast.top镜像或配置代理
4	pip install 报错	Python版本不兼容	确保Python 3.10+，使用venv隔离
5 6	Claude Code登录失败 MCP服务连接不上	网络或地区限制配置格式错误或路径不对	检查网络连接，考虑使用API代理检查JSON格式，使用绝对路径
7	Skill命令不识别	Skill目录位置不对	确认在~/.claude/skills/下
8	bun install 报错	bun未正确安装	改用npm install
9	视频渲染为空/0字节	FFmpeg编码器缺失	重新安装FFmpeg，确保含libx264
10 11	字幕中文显示为方框 Whisper模型下载失败	系统缺少中文字体网络问题或磁盘空间不足	安装Noto Sans CJK SC字体检查网络和磁盘空间（需5GB+）
12 13	剪映自动化失效 HyperFrames渲染黑屏	剪映版本不兼容 Puppeteer启动失败	降级到剪映专业版5.9或以下检查Chrome/Chromium是否安装
14	Remotion预览白屏	端口冲突或组件报错	检查浏览器控制台错误信息
15 16	音频不同步BGM音量无法调低	剪辑后音画未对齐 FFmpeg滤镜语法错误	使用-itsoffset参数微调检查volume滤镜的语法格式
17	批量渲染中途崩溃	内存不足	减少并行数量，或逐条渲染
18	导出视频在某些播放器花屏	编码参数不兼容	使用更保守的编码参数：-profile:v high -level 4.1

19	TTS生成的音频质量差	选用的声音模型不合适	更换声音模型，或调整语速参数
20	project.md文件越来越大	历史记录累积	定期归档旧项目，新建project.md

附录D 推荐学习资源

D.1 GitHub 仓库合集

仓库	Stars	简介
heygen-com/hyperframes	12.7k	HTML描述画面，渲染成视频
remotion-dev/remotion	45.1k	React组件编程式视频制作
redotvideo/revideo	-	TypeScript代码驱动视频
browser-use/video-use	5.4k	对话式视频编辑
Ceeon/videocut-skills	1.5k	口播视频剪辑Skill
luoluoluo22/jianying-editor-skill	-	剪映自动化Skill
sun-guannan/VectCutAPI	800+	剪映/CapCut自动化API
video-creator/ffmpeg-mcp	-	FFmpeg MCP服务
Kush36Agrawal/Video_Editor_MCP	-	视频编辑MCP
anthropics/claude-code	30k+	Claude Code

D.2 教程资源

Remotion 官方文档：https://remotion.dev/docs HyperFrames GitHub README：https://github.com/heygen-com/hyperframes video-use 使用指南：https://github.com/browser-use/video-use/blob/main/SKILL.md FFmpeg 官方文档：https://ffmpeg.org/documentation.html MCP协议文档：https://modelcontextprotocol.io

D.3 社区与交流

Remotion Discord：https://remotion.dev/discord（14,000+成员）

Claude Code讨论区：Reddit r/ClaudeAI国内Vibe Coding交流：关注相关技术公众号和B站UP主

附录E 术语表（中英对照）

表E-1 术语表

中文	英文	释义
氛围编码/氛围编程	Vibe Coding	Andrej Karpathy提出的概念，用自然语言描述需求让AI完成代码实现
模型上下文协议	MCP (Model Context Protocol)	Anthropic推出的开放协议，定义AI助手如何调用外部工具
技能包	Skill	告诉AI如何完成特定任务的指南文件（SKILL.md）
视频编码	Video Encoding	将视频数据压缩为特定格式的过程
帧率	FPS (Frames Per Second)	每秒显示的画面帧数，常见24/30/60fps
分辨率	Resolution	画面的像素尺寸，如1920x1080
码率	Bitrate	视频每秒的数据量，单位Mbps或kbps
时间轴	Timeline	视频编辑中的时间线性编辑界面
淡入淡出	Fade In/Out	画面或音频从黑/静音渐变到正常，或反之
关键帧	Keyframe	动画中定义状态变化的关键时间点
字幕烧录	Burn Subtitles	将字幕文字直接嵌入到视频画面中
语音转文字	STT (Speech to Text)	将语音自动识别为文字的技术
文本转语音	TTS (Text to Speech)	将文字合成为语音的技术
填充词/口癖	Filler Words	说话时不自觉的填充词如"嗯""啊""那个"
安全区域	Safe Area	画面中确保在各种设备上都能完整显示的区域
色彩分级/调色	Color Grading	对视频画面进行色彩风格化处理
定常媒体框架	Remotion	用React组件编程式制作视频的框架

超帧	HyperFrames	用HTML描述画面渲染成视频的框架
浏览器使用	Browser-use	用自然语言操控浏览器的自动化工具
自动化代理/智能体	AI Agent	能自主规划和执行任务的AI系统
编程式视频	Programmatic Video	用代码而非传统软件界面制作视频
并行渲染	Parallel Rendering	同时渲染多个视频以加速处理
无头浏览器	Headless Browser	没有图形界面的浏览器，用于后台自动化
运动画布	Motion Canvas	Revideo的前身项目，用于代码创建动画
无损剪辑	Lossless Editing	不重新编码的剪辑方式，速度快但灵活性低
常量速率因子	CRF (Constant Rate Factor)	FFmpeg的质量控制参数，0-51，越小质量越高
查找表	LUT (Look-Up Table)	预设的色彩映射表，用于统一调色风格
行动号召	CTA (Call to Action)	视频结尾引导观众行动的提示语
视频滤镜	Video Filter	FFmpeg中对视频画面进行处理的操作
音频滤镜	Audio Filter	FFmpeg中对音频进行处理的操作

- 全书完 -

感谢你阅读《代码即剪刀》

祝你用Vibe Coding创作出精彩的视频作品

会员专属资料 · 持续更新

把这本手册用起来的地方

「河马的CC商业实战」星球里，每周都有看得懂、用得上的 AI 实战：

新栏目「河马带你逛GitHub」每周一期，实战长文、会员资料持续更新。

微信扫码 · 加入「河马的CC商业实战」

河马的CC商业实战 出品