The Claude Video Editing Stack

The five tools that let Claude Code edit your videos, and the skill file that ties them together.

Why I stopped looking for the one AI editing app

There is no single app that lets Claude edit your videos. What works is five small tools, each one handing Claude a sense it does not have on its own, plus one skill file telling it how to use them.

Claude Code is a coding agent. It has no timeline to click on and no preview window to scrub, so it edits video the way it edits code, by writing and running files. That is why people get stuck. They open Claude Code, ask it to edit a clip, and it has nothing to work with. It cannot hear the audio, it cannot see the frames, and it has no project to write into.

Give it those things and the whole job changes. HyperFrames or Remotion holds the project. Whisper gives it the words and the timings. Clack tells it what each sound is. Demucs pulls music away from effects. Blender builds 3D objects for free so you stop paying a video model for every shot. By the end of this guide you will have all five installed and one skill file that makes Claude use them in the right order.

01

How it actually works

1. You describe the edit in plain language. Claude reads the skill file, sees which tool covers which part of the job, and starts with the transcript rather than the cut.
2. It runs the tools from the terminal and writes the results into the project as files. The transcript becomes a caption track, the stems become separate audio layers, a Blender script becomes a rendered 3D shot sitting in a folder.
3. At the end it renders once, to MP4. Because the whole video is code, you change one line and re-render instead of redoing the edit, and you can produce the same video a hundred times with different text.

02

The one thing almost everyone gets wrong

Installing the tools does nothing on its own. Claude does not scan your machine and discover that Whisper is available, so until you write the skill file it will keep guessing at caption timings and skipping Demucs entirely.

The skill file is where you say what each tool is for and what order to use it in. It is the difference between a one-off result and a system that behaves the same way every time. Mine is about forty lines and most of it is rules, not code.

> **WATCH OUT**
>
> The folder name has to match the name field inside SKILL.md exactly. If the folder is video-editing and the file says name: video-editor, the skill never loads and nothing tells you why.

03

The files

Two files. The first is the skill that teaches Claude the stack. Save it as SKILL.md inside a folder called video-editor. The second is the set of commands it runs underneath, which is worth keeping nearby so you can read what Claude is doing.

SKILL.MD

```
---
name: video-editor
description: Edit short-form video with HyperFrames or Remotion, Whisper, Clack, Demucs
and Blender. Use whenever I ask to cut, caption, score or render a video.
---

# Video editor

## The stack
- Project and render: HyperFrames (HTML and CSS) or Remotion (React). One project
folder per video.
- Transcript and caption timing: Whisper. Always word level.
- Sound identification: Clack. Use it to name what is in the audio before touching the
mix.
- Stem separation: Demucs. Split music from effects before any ducking.
- 3D assets: Blender, scripted in Python. Never pay a video model for a shot Blender
can build.

## Rules
1. Transcribe before you cut. Run Whisper first and read the transcript back to me.
2. Captions come from the Whisper word timings, never from a guess.
3. If there is music under the voice, run Demucs before mixing anything.
4. Build 3D shots in Blender headless, render the frames, then import them.
5. Do not render the project until I ask. Rendering is the slow step.
6. Tell me the output path and the duration when you are done.

## Paths
- Source footage: ./footage
- Audio and stems: ./audio
- Blender scripts: ./blender
- Renders: ./out
```

COMMANDS.SH

```
# 1. transcribe with word-level timings
whisper footage/raw.mp4 --model small --word_timestamps True \
  --output_format srt --output_dir audio

# 2. split music from everything else before mixing
python3 -m demucs --two-stems=vocals audio/raw.wav -o audio/stems

# 3. build a 3D shot without ever opening Blender
blender --background --python blender/shot01.py

# 4. render, once, at the end (Remotion)
npx remotion render src/index.ts Main out/final.mp4
```

If you went with HyperFrames instead of Remotion, the only line that changes is the last one. HyperFrames has its own render command, so ask Claude to run that and leave the rest of the file alone.

Set it up in Claude Code or Codex

1. Pick your framework and create one project folder for the video. Remotion: run npx create-video@latest and follow the prompts. HyperFrames: run hyperframes init in an empty folder. Install one of them, not both.
2. Install Whisper with python3 -m pip install -U openai-whisper. Whisper reads audio out of video files using ffmpeg, so install that too: brew install ffmpeg on macOS.
3. Test Whisper on its own before you involve Claude. Run it on one short clip and open the SRT file it writes. If the timings look right, the hardest part is already done.
4. Install Demucs with python3 -m pip install -U demucs. Skip this if your edits never have music under the voice, and come back to it when they do.
5. Add Clack for sound identification. It is the optional one, so install it from its own repo and let Claude read the README before you ask it to use it.
6. Install Blender from blender.org, then check that blender --version answers in the terminal. On macOS the command often is not on your PATH even after installing. Ask Claude to add the Blender binary to your shell profile, then open a new terminal window so the change takes effect.
7. Make the skill folder. In Claude Code: mkdir -p ~/.claude/skills/video-editor and save SKILL.md inside it. The folder name and the name field in the file have to match.
8. In Codex, put the same folder in your project, or in .codex/skills/video-editor/, and add one line to AGENTS.md pointing at it so Codex knows to read it.
9. Open a new Claude Code session in the project folder and say: read the video-editor skill and tell me what the stack is. If it lists five tools back at you, it loaded.
10. Run your first real job on a thirty second clip, not a five minute one. A successful first run looks like this: Claude runs Whisper, tells you where the SRT landed, builds the caption track, and stops to ask before rendering.

> 💡 **QUICK TIPS**
>
> - Keep one project folder per video. Reusing a folder is how scenes from last week end up in this week's render.

- Put the project in git before you let Claude touch it. Then a bad edit is one command away from being undone.
- Start with the small Whisper model. Move to medium only if the transcript is actually getting words wrong.

04

The commands I actually run

These are prompts, not shell commands. Claude turns them into the terminal work. Replace anything in brackets.

TRANSCRIBE AND CAPTION

```
Transcribe [FOOTAGE FILE] with Whisper at word level, then build the caption track in the project from those timings. Do not render yet, just show me the first ten lines.
```

CUT THE DEAD SPACE

```
Using the transcript, cut every silence longer than [0.4] seconds and every false start where I repeat myself. List the cuts with timestamps before you make them.
```

FIX THE MIX

```
Run Demucs on [AUDIO FILE] to split the music out, then duck the music under my voice. Before you change anything else, use Clack to tell me what the other sounds in the clip are.
```

BUILD A 3D SHOT FOR FREE

```
Write a Blender Python script that builds [OBJECT DESCRIPTION] on a transparent background, render it headless at [1080x1920], and place it in the project at [TIMESTAMP].
```

RENDER

```
Render the project to out/ as MP4 at [30] fps and tell me the file path and the final duration.
```

When it breaks

Almost every failure on this stack is one of seven things, and five of them are a tool not being on your PATH.

ERROR	FIX
whisper: command not found	Whisper installed into a different Python than the one on your PATH. Install it again with python3 -m pip install -U openai-whisper and call it as python3 -m whisper.
Whisper fails on an .mp4 file	Whisper pulls audio out of video with ffmpeg. Install ffmpeg (brew install ffmpeg on macOS) and open a new terminal.
Captions are close but always slightly late	The --word_timestamps True flag was left off, so the SRT only has line level timings. Re-run with it on.
Captions drift further apart as the video goes on	The project frame rate does not match the source frame rate. Set the composition fps to the source fps, then rebuild the caption track.
blender: command not found	Blender is installed as an app but the binary is not on your PATH. On macOS it sits inside Blender.app/Contents/MacOS/. Ask Claude to add it to your shell profile and open a new terminal.
Demucs ran but there is no separate music file	--two-stems=vocals only ever gives you two files, vocals and everything else. Drop the flag to get the full set of stems.
Claude ignores the skill completely	The folder name does not match the name field in SKILL.md, or the session started before the file existed. Fix the name, then start a new session.

Ready to go deeper?

This playbook is the short version. Inside the hub you get the full build walkthroughs, the prompt library, and the systems I use to ship this in a weekend.

Join the Build →

https://hub.digicuratoragency.com/join