The video editor playbook

How to edit and animate your own talking-head videos with Claude: cuts, captions, motion graphics, audio, export and review. Written for creators and founders with no studio, a laptop, and Claude Code.

by Taku

What the skill is

Give Claude your raw footage. Get back a cut, captioned, loudness-correct video that passed an automated check, plus every file that made it.

The skill is the operating procedure Claude follows: what to run, in what order, where to stop and ask you, and how to check the result. The scripts do the media work with free tools on your own machine. Nothing is uploaded anywhere.

What it does

1. Checks your setup and tells you what to install.
2. Makes an editing proxy and a word-timed transcript, then lists the words it may have misheard for a check.
3. Drafts a cut list: trims dead air, shortens long pauses, drops lone "um"s, and flags retakes and filler for you to approve.
4. Renders the cut in one frame-exact pass, with silent joins, re-checks every join, and has you watch the cut with sound.
5. Optionally lays your approved motion graphics onto the cut (stills first, then motion).
6. Burns in captions last, on top of everything (word by word or phrases), inside the platform safe zone. It shows you stills of each version first; that is where Claude checks the captions stay off your face, and where you choose.
7. Normalises loudness (-14 LUFS for social by default) and measures it on the file that ships.
8. Exports for Reels, TikTok and Shorts (9:16, one preset for all three), YouTube (16:9), square, and a phone copy under a size cap.
9. Runs QC (aspect, codec, frame rate, loudness, true peak, black and frozen frames, silences) and gives you a watch-through list.

How to make the motion graphics, taste and review are covered in `PLAYBOOK.md` and the reference files. Claude asks before any visual choice and shows you versions.

It never deletes or overwrites your originals. Everything it makes goes into an `edit/` folder next to your footage.

Install it

Claude Code (recommended): copy this folder to `~/.claude/skills/video-editor/` to use it in every project, or to `.claude/skills/video-editor/` inside one project (Claude docs).

```
mkdir -p ~/.claude/skills && cp -r video-editor ~/.claude/skills/
```

Claude apps: zip the folder and upload it under Customize > Skills, with code execution on (Claude help). The apps run skills in a sandbox, which suits planning and cut lists more than rendering. See `reference/01-setup.md`.

Requirements

- Python 3.9 or newer, with numpy (and Pillow if your ffmpeg has no libass)
- ffmpeg and ffprobe
- whisper.cpp (`whisper-cli` plus a model) or faster-whisper
- Node.js only for Remotion or HyperFrames motion graphics

Check everything at once:

```
python3 ~/.claude/skills/video-editor/scripts/check_setup.py
```

It prints the install command for your OS for anything missing; Python packages go into a virtual environment (`python3 -m venv .venv && . .venv/bin/activate`, then `pip install ...`). Full instructions for macOS, Windows and Linux are in `reference/01-setup.md`.

Your first prompt

Open Claude Code in the folder with your footage and say:

> Edit `my-clip.mov` into a 9:16 reel for Instagram and TikTok, word-by-word captions, as tight as it reads. Show me the cut list before you render.

Claude checks setup, asks what it cannot see from the file, transcribes, shows you the cut list, waits for your yes, shows caption stills, burns the one you pick, normalises, exports and runs QC. Then you watch it with sound.

More prompts for every stage are in `reference/10-prompts.md`.

The skill folder and this PDF: velorum-ai.com/resources/editor

How it works

Everything here comes from three places: the official docs of the tools and standards bodies, practitioners on X who posted their methods with numbers, and a working studio that edits this way every week. Every number carries its source as a link. Where a number is a studio habit rather than a published figure, it says "our working default". Fetched and checked on 2026-11-22.

This guide pairs with the `video-editor` skill in the same folder. The skill tells Claude what to do. This playbook tells you why, so you can steer it.

1 **Setup**

ffmpeg, whisper.cpp, Claude Code skills

2 **Transcribe and review**

whisper.cpp, faster-whisper

You approve: Names, numbers and misheard phrases fixed

3 **Cut list and cut**

ffmpeg filters

You approve: Yes to the cut list before any render; then you watch the cut with sound

4 **Motion**

HyperFrames, Remotion

You approve: Yes to the still, then to the motion

5 **Captions**

BBC subtitle guidelines, Netflix style guide

You approve: Stills at phone size

6 **Loudness**

ffmpeg loudnorm, EBU R128

7 **Export**

YouTube encoding

8 **QC and human watch**

Claude vision, Anthropic, harness design

You approve: Final yes

Contents

1. How AI editing works, end to end
2. Set up once
3. Transcribe with word timestamps
4. Make the cut
5. Choose a motion framework
6. Motion principles in numbers
7. Captions
8. Effects
9. Audio
10. Export
11. Taste: how to guide Claude on visual work
12. The review loop
13. Practitioner tips
14. Common mistakes and their fixes
15. Resources

1 How AI editing works, end to end

The mental model

Claude does not edit video the way a person drags clips in Premiere. It writes commands and code, and other tools do the media work. As one practitioner guide puts it, quoting Gideon Frost: "Claude Opus 5.5 does not generate pixels. It writes code that paints pixels." (@flxrnc)

A working setup splits the job across three tools:

Job	Tool	Why
Word timings	Whisper (whisper.cpp or faster-whisper)	Every cut, caption and graphic hangs off a word time
Media work: cut, join, audio, encode	ffmpeg	Fast, exact, scriptable, free
Captions and graphics	A motion framework: HyperFrames or Remotion	Animated type and UI need a renderer

@notmissing_ describes the same idea: the whole edit is code, and Claude times the edits from the transcript of your voice. Our rule follows from it: tie every cut, caption, graphic and sound to a word time, and hardcode nothing.

The pipeline

Stage	What happens	You approve
1. Setup	Claude runs `check_setup.py` : ffmpeg, Whisper, Node and Python, with fix commands	
2. Transcribe and review	A proxy and a 16 kHz mono WAV, word-level timestamps for the whole WAV, then a read of the transcript against your voice	Names, numbers and misheard phrases fixed
3. Cut list and cut	A list of kept segments (an EDL) with the reason for every cut, then picture and voice cut in one frame-exact pass	Yes to the cut list before any render; then you watch the cut with sound
4. Motion	Graphics as stills first, several versions, then the picked one in motion, laid onto the cut	Yes to the still, then to the motion
5. Captions	Word-timed captions inside the safe zone, burned last, on top of everything	Stills at phone size
6. Loudness	Two-pass loudness to a target, measured on the encoded file	

Stage	What happens	You approve
7. Export	Platform encode plus a phone copy	
8. QC and human watch	Automated checks (aspect, codec, frame rate, loudness, true peak, black and frozen frames), then you watch the whole thing with sound	Final yes

Two things to know before you start.

- **Claude reviews stills, not playback.** Claude reads video as frames (Anthropic, prompting best practices). When @PrajwalTomar_ asked Claude to check a finished reel, it said it had only checked still frames and he should watch it with sound. That is the honest answer, and the final watch is your job.
- **This is a Claude Code job first.** Skills on the Claude API run with no network access and cannot install packages (Anthropic, skills overview), and editing needs ffmpeg, Whisper and a renderer on your machine. Claude Code on your own computer is the right home for it.

What it is good at, and where it is weak

- Strong: tightening a talking head, finding clean takes, word-timed captions, kinetic type, rebuilding interface as live animation, charts, and repeatable checks.
- Weaker: camera moves and 3D cameras. @motion_conquest found Claude designs good UI elements but struggles to animate them well, especially with a 3D camera. Plan extra review there.
- Not one prompt. @timkochjar built a full demo video in one Remotion timeline and said plainly that it is doable but not easy. @Shpigford reports about 2 to 3 hours for a product demo.

2 Set up once

The full install steps per operating system are in `reference/01-setup.md`. The short version:

- **You need:** ffmpeg, Python 3.9 or newer, a Whisper engine with a model, and Node.js if you use a motion framework (HyperFrames needs Node 22+ and FFmpeg, per its README; on Linux, install Node from NodeSource or nvm, since distro packages can be older).
- **Python packages go in a virtual environment:** `python3 -m venv .venv && . .venv/bin/activate`, then `pip install faster-whisper "av<16"` if you use faster-whisper.
- **Install the skill:** put the `video-editor` folder in `~/.claude/skills/` (for all your projects) or `.claude/skills/` inside one project (Anthropic, skills overview). Claude Code picks up changes without a restart (Claude Code docs).
- **In the Claude apps:** zip the folder, then Customize > Skills > Upload a skill (Claude help). Code execution must be on. The apps cannot run your local ffmpeg, so use them for planning and prompts, and Claude Code for the edit.
- **Model:** Anthropic suggests starting with Claude Opus 5.5 for most work (models overview). Opus 5.5 is also better at reading where things sit in a picture and what changed between two versions (prompting Opus 5.5), which is exactly what frame review needs.
- **First move in any session:** ask Claude to run `scripts/check_setup.py`. It reports what is missing, including whether your ffmpeg can burn captions. Some ffmpeg builds have no libass and no drawtext, so captions must be drawn another way. The skill handles that.

Keep your originals untouched. The skill writes everything to an `edit/` folder next to your footage.

3 Transcribe with word timestamps

Full method: `reference/02-transcribe.md`.

- **Word-level, not sentence-level.** You need a start and end time for every word. Sentence timestamps cannot drive cuts or word captions.
- **Work from a proxy.** Never decode 4K, 8K or HEVC phone footage whole for editing. Make a light proxy and a 16 kHz mono WAV, and edit on those (our working default). The final render can come from the proxy, which is fine for social video, or from the original: `cut.py --source path/to/original.mov` applies the same cut list to the full-quality file.
- **Word ends are not cut points.** In the studio's tests one engine put a word end 120 ms after the voice had decayed, and another put a word start 0.3 s into silence (our measurements). Cut on measured silence, then use the words to know what was said.
- **Transcribe the whole WAV once.** `transcribe.py` runs over the full 16 kHz WAV. Only if captions drift on a long cut, re-transcribe the kept pieces with `transcribe.py --pieces edit/cut.json`, which pads each piece and adds its start time back.
- **Fix names and numbers.** Whisper mishears proper nouns and numbers. Pass the names with `--prompt`, and keep a small fix table that maps the misheard word to the right one.
- **Always read the whole transcript before captions.** Misheard words are not only names. In our test, both engines misheard a common three-word phrase from the reel: one wrote "turn extra output", the other "10-inch up". Neither the engine nor the first pass of the skill flagged it. `review_transcript.py` lists the low-confidence words and, with `--other`, diffs against a second model or engine (any second one helps if you have no bigger model). Read the transcript against what you said, then fix each error with `transcribe.py --fix` before captions, so the captions and the SRT start from the corrected words.
- **A repeat is real only if you can see it.** Whisper sometimes merges "X, X" into one long word, and a prompt that encourages disfluencies can invent repeats. A caption word stretched over about 1 s is the tell; check the waveform for two onsets (our working default).

4 Make the cut

Full method: `reference/03-cut-list.md`. The scripts are `silences.py`, `suggest_edl.py` and `cut.py`.

The cut list first

Ask for a cut list (an EDL) before any render: kept segments in source seconds, each with the words it contains and the reason for every cut. @MushtaqBilalPhD gives the beginner version: ask Claude for a clean cut that removes silences, false starts and bad takes, then review it. @notmissing_ adds the check: make sure Claude did not cut you off mid-sentence.

Picking clean takes

- `suggest_edl.py` flags retakes from repeated words (the same 3 or more words again within 12 s; it keeps the last), plus repeated single words, hesitations and soft fillers. That catches only some: in our test the real false starts repeated no words. So Claude also reads the transcript for restarts.
- Keep the last good take of a repeated line, including a repeated call to action.
- When you drop the middle of a self-correction, drop every earlier try too.
- **List fillers, do not cut them silently.** Flags go on a list with timestamps; they are applied with `--apply` only after a yes. The script drops a lone "um" or "uh" by itself only when there is silence on both sides.

Silence and pauses in numbers

Setting	Value	Source
Cut in measured silence, not on Whisper word edges	Words clipped and misheard when cut on word edges	@socialwithaayan
Tail after the last word	0.15 s, measured on the actual audio	@ladprofit
Silence threshold	`silences.py`: 22 dB under the speech level (its 90th percentile), at least 6 dB over the noise floor, never above -20 dBFS; runs of 0.12 s or more count. 30 dB under found almost no silence for a continuous speaker	our working default (`--below`, `--min`)
Pauses kept, `reel` preset (also TikTok and Shorts)	Between phrases 0.18 s, at a beat 0.30 s, after a sentence 0.22 s; 0.08 s before the first word, 0.15 s after the last	our working default (`suggest_edl.py`)
Pauses kept, `talking-head` preset	Between phrases 0.30 s, at a beat 0.40 s, after a sentence 0.50 s; 0.12 s lead, 0.25 s tail	our working default (`suggest_edl.py`)

Setting	Value	Source
Where the kept pause sits	55% of it after the last word, the rest before the next	our working default
Room tone at a section change	Keep a real pause; 50 ms read as a stutter	our working default
Shortest shot	No shot under 1.2 s	@ladprofit
Speed	Never speed a voice up to hit a length	@ladprofit, our working default

Base the threshold on the speaker's own level, not a fixed dB. Voices and rooms differ.

Placing the boundaries

- **Start** a piece just before the first voiced sound of the kept word, and after any inhale. The glide at the start of a word like "which" begins about 100 ms before the hard sound (our measurement).
- **End** a piece in the dip before the next word, not at Whisper's end time.
- **A snap must not cross voice.** `suggest_edl.py` moves a boundary into a measured pause only when the pause is within 0.3 s of the word edge; anything it cannot place cleanly is flagged "check by ear".
- **Land on a frame inside the silence.** Otherwise rounding to a frame can move the boundary back onto sound.

Rendering a cut that stays in sync

- **One pass, frame-exact.** Snap each boundary to a frame once, then cut picture with `trim` and voice with `atrim` at the same times and join both through the `concat` filter. Cutting each piece with `-ss` / `-t` and joining with the concat demuxer drifted lips 1 to 3 frames behind the voice in the studio's films (our measurement).
- **Short fades at joins.** `cut.py` fades every join in over 12 ms and out over 15 ms (our working default); anything from 12 to 30 ms stops clicks. Fade only at real cuts. The script merges touching pieces from the same take, or the fades would put notches in the middle of a word.
- **Build the voice once from the source audio.** Joining clips that each carry their own AAC audio slid the voice 8 to 16 ms per join (our measurement).
- **Count frames after every mux.** `-shortest` with AAC can drop the last video frame.
- **External mic?** Its offset from the camera audio differs per clip (60 to 76 ms in the studio's footage). Measure it on every clip; never reuse another clip's number.

The join check

Run it on every join of `edit/cut.mp4`, after `cut.py` and before the user watches the cut. Claude cannot listen, so it re-reads each join instead.

1. Find the joins: each piece's `out_end` in `edit/cut.json`, except the last, is the time J of a join in the cut.
2. Cut a 3-second slice centred on the join (1.5 s each side) and transcribe it on its own:

```
J=13.8 ffmpeg -v error -ss $(python3 -c "print(max(0, $J - 1.5))") -t 3 -i edit/cut.mp4 -vn -ac
```

3. Compare `edit/join_check_words.txt` with the words of `edit/cut_words.json` that start between J - 1.5 and J + 1.5 s. Read the words either side of J most closely.
4. Pass: the same words in the same order. Ignore the first and last word of the slice: a word cut in half by the slice edge is often dropped, guessed ("And" for a breath) or marked `(?)`. Ignore spelling and contractions ("I'll" for "I will").
5. A word missing, doubled or merged next to J is a suspect. Tell a whisper artefact from a real fault by transcribing a wider slice, 4 s each side (`-ss` J - 4, `-t 8`). If the wider slice reads right, it was an artefact of the short slice. If the same word is still missing or doubled at J, the join is wrong: move the cut point into the pause (`silences.py`), render again, and check again.
6. Report every join to the user by its time in the cut, with the result, and keep them listed as "not listened yet" until the user has watched the cut with sound (gate 1b). The check finds dropped and doubled words; only a person hears a click, a breath cut in half or a change in tone.

In our test, the join at 13.8 s read "...learn how to do this, comment editor and I'll send you a." against the cut's "...learn how to do this, Comment editor and I will send you a": a pass (an extra "And" at the slice start, a contraction, and the last word cut by the slice edge).

Then watch the cut with sound. This is your gate before any graphics or captions go on. A join that sounds wrong is cheapest to fix now.

5 Choose a motion framework

You do not always need one. Cuts, a fixed caption style and a logo can be done with ffmpeg alone. Once captions need word-by-word animation, highlights or moving interface, use a framework. The decision tree with install steps is in `reference/06-motion.md`.

Honest comparison

Licence terms and prices as published on 2026-11-22; each row links to its source.

Framework	Good for	Licence and cost	How it renders	Agent support
ffmpeg only	Cuts, reframes, a fixed caption style, logo overlay, audio	LGPL 2.1+, GPL optional, free	Shell commands	No official skill; Claude knows it well
HyperFrames	Captions, overlays, kinetic type, titles, charts; for people who do not want React	Apache 2.0, "no per-render fees or commercial-use thresholds"	Seeks each frame in headless Chrome, encodes with FFmpeg	Official skills, Claude Code plugin, MCP connector, about 450 ready-made blocks
Remotion	Captions, kinetic type, UI mockups, charts, 3D; the older, larger ecosystem (since 2021)	Free for individuals, non-profits, companies of up to 3 employees, and evaluation. Otherwise a Company License: Creators $25/mo per seat, Automators $0.01 per render ($100/mo minimum), Enterprise from $500/mo	React, frame by frame in headless Chromium; CLI, Studio, Node, Lambda	Official Agent Skills (13 skills), llms.txt, LLM system prompt. The hosted MCP server shut down on 2026-10-26
three.js / React Three Fiber	Product hero shots, logo reveals, depth camera moves	MIT / MIT, free	No video export of its own: render inside Remotion with @remotion/three or as a HyperFrames 3D block	llms.txt; no official skill found
Manim	Maths, equations, graphs	MIT, free	`manim` CLI to MP4 (Python)	Community skills, such as manim_skill
Motion Canvas	Diagram and code explainers synced to voice	MIT, free	Browser editor saves an image sequence	No skill; last release December 2024
Revideo	Template rendering on a server	MIT, free	Headless, serverless	No agent tooling found

How to pick

- **Solo creator, no React, no licence questions:** HyperFrames. Its pitch, from its lead @liu8in, is that anything that works in a browser works in it: CSS animations, GSAP, Lottie, three.js, D3, Google Fonts. Language models have seen far more HTML than React video code, a point @mvanhorn relays from HeyGen's team; he now uses HyperFrames for one-off videos and Remotion for templated or team work.
- **You know React, or want the older, larger ecosystem:** Remotion. Read its licence if you work in a company of 4 or more.
- **You want 3D:** three.js inside Remotion or a HyperFrames 3D block, and budget extra rounds. Lighting, scale and camera are where agents get it wrong.
- **You explain maths:** Manim.
- **Pros mix tools.** One editor, @thacreativeguru, uses HyperFrames with Claude Code to animate real screenshots and still finishes in Premiere. That is a fine workflow.

6 Motion principles in numbers

Adjectives get you the model's defaults. Numbers get you what you meant. @0xclayo puts it as: adjectives are where the defaults live.

Easing

- **Entering: ease-out.** It starts fast and settles, so the eye knows where it stops (NN/g, Emil Kowalski).
- **Leaving: ease-in. Moving across the screen and staying: ease-in-out** (Material 3).
- **Never linear for type or UI.** Remotion's `interpolate()` is linear unless you add an easing (Remotion docs), and linear reads as mechanical.

Curve	Values	Source
Standard (on screen to on screen)	`cubic-bezier(0.4, 0, 0.2, 1)`	Material 2
Decelerate (entering)	`cubic-bezier(0, 0, 0.2, 1)`	Material 2
Accelerate (exiting)	`cubic-bezier(0.4, 0, 1, 1)`	Material 2
Stronger exit	`cubic-bezier(0.7, 0, 0.84, 0)`	@flxrnc (practitioner)
Calm, premium ease-out	`cubic-bezier(0.16, 1, 0.3, 1)`	our working default
Calm ease-in-out	`cubic-bezier(0.645, 0.045, 0.355, 1)`	our working default

Durations

Interface guides agree on a band. Video can sit toward the long end, because nobody is waiting to click.

Source	Numbers
NN/g	Most animations 100 to 500 ms; enter about 300 ms, exit 200 to 250 ms
Material 3	Emphasized 500 ms, emphasized decelerate 400 ms, emphasized accelerate 200 ms, standard 300 ms
Material 2	Drawer opens 250 ms, closes 200 ms; card 300 ms open, 250 ms close
Emil Kowalski	Interface animations usually under 300 ms
@flxrnc **(practitioner, for video)**	UI moves 200 to 500 ms, hero reveals 600 to 900 ms

- **Exits are shorter than entrances.** Material says exits need less attention (Material 3). @flxrnc uses exits 40 to 50% shorter than the entrance.
- **In frames at 30 fps:** one frame is 33.3 ms, so 100 ms is 3 frames, 200 ms is 6, 300 ms is 9, 500 ms is 15, 900 ms is 27. At 60 fps double the counts.
- **Never scale from 0.** Start at 0.9 or higher (Emil Kowalski).

Springs without bounce

Remotion's `spring()` defaults (mass 1, damping 10, stiffness 100) bounce. Raise damping to remove the bounce (Remotion docs); Remotion's own transition example uses `damping: 200` (TransitionSeries).

@flxrnc suggests describing a spring by one number, the damping ratio, instead of a word like "bouncy". The arithmetic: damping ratio = damping / (2 x square root of (stiffness x mass)). The defaults give 10 / (2 x 10) = 0.5, which overshoots. A ratio of 1 or more does not overshoot. With stiffness 100 and mass 1, that means damping 20 or more.

Stagger

No standards body publishes a stagger number. The principle behind it is overlapping action, parts moving at different rates (twelve principles of animation). As a starting point, @flxrnc uses letters 1 frame apart and words 60 to 120 ms apart. Keep the whole group landing inside the durations above.

Motion blur, frame rate, flashes

- A 180 degree shutter is the film norm (rotary disc shutter). In Remotion, `shutterAngle={180}`; at 30 fps that equals 1/60 s exposure (Remotion motion blur). More samples look smoother and render slower.
- Render graphics at the footage frame rate. YouTube asks for uploads at the recorded rate (YouTube encoding).
- Nothing should flash more than three times in one second (WCAG 2.3.1). That rules out rapid negative-flash hits.

Rules that keep motion calm

- **Purpose first.** Apple's guidance: do not add motion for its own sake (Apple HIG).
- **One moving thing at a time.** Staging means pointing the eye at what matters most (twelve principles).
- **Land on the word.** A graphic should arrive 0 to 3 frames before the word it illustrates, never late (our working default).
- **Never text through text.** In a dissolve, the old text is fully gone before the new text starts to appear (our working default).

- **No dead holds.** A graphic that does not change for more than about 18 frames looks frozen, except for 0.8 to 0.9 s of reading time after a payoff lands (our working default).
- **Every frame is a pure function of time.** @0xCarnagee and @0xclayo both arrive at this. An animation that adds velocity frame by frame looks different when the renderer jumps to frame 300, a point @flxrnc also makes. Compute every value from the frame number.

7 Captions

The full catalogue of caption types is in `reference/05-captions.md` (section "Caption styles"). How to time and burn them is in the same file.

Reading speed and length

Standard	Figure
BBC	160 to 180 words per minute, about 0.33 to 0.375 s per word; 2 lines max on 16:9, 3 on 9:16; width 68% of a 16:9 frame, 90% of a 9:16 frame; about 25 characters per line in vertical
Netflix	Adults up to 20 characters per second, 42 characters per line, two lines max
DCMP	Upper level not above 160 wpm; mixed case reads better than all caps

Use the BBC range for talking heads and check characters per second against Netflix. Break lines at natural points, ideally punctuation (BBC).

Legibility

- Contrast at least 4.5:1, or 3:1 for large text (WCAG 1.4.3). White on a white shirt measured 1.3:1 in the studio; put the word on a soft dark pill when the background is light (our measurement).
- Text people must read should be about 20 px or more at 1080 wide. Smaller card text does not read on a phone (our working default).
- Check every caption at phone size, and check your face stays in frame (@notmissing_).
- Caption rules from @ladprofit: never over the face, one idea per caption, never split a brand name across two captions. Place them low in the safe zone: on 9:16, `captions.py` puts the block about 60% down, above the app's bottom UI, with its bottom edge at y 1248. Check the face on the caption stills; with chest-up framing, the safe zone and the face can overlap.

Timing that holds (our working defaults)

- Each word visible at least 3 frames at full opacity; 4 frames when words replace each other. Merge tiny words like "a", "to", "of" into the next.
- A word with nothing after it stays at most 0.55 s.
- Do not fade a replacing word in from zero opacity. In the studio's test, most word changes had a near-empty first frame and read as flicker. Hard-cut word to word, or ease scale only.
- Land each word about one frame before its voiced onset.
- Target: every caption within 2 frames of its word, no caption over silence.
- Re-check timing on the rendered file, not the source (@PrajwalTomar_). Bad captions often mean a bad cut upstream, as @socialwithaayan found: the fix was a better cut, not a better caption.

Accessibility and reach

- Proper captions name who speaks and include meaningful sounds (WCAG 1.2.2). Burned-in social captions do not replace that. Upload an SRT as well, in sentence case with normal punctuation.

Safe zones

These figures come from secondary guides that measured each platform's template files. Meta's own safe-zone page gives no pixel figures (it says Reels ads with a disclaimer should keep the bottom 40% clear); TikTok's says the zone varies with caption length. Treat them as guides and check against the current app.

Platform (1080x1920 unless noted)	Top	Bottom	Sides	Source
Instagram and Facebook Reels	14% (269 px)	35% (672 px)	6% (65 px) each	AdsUploader (secondary)

Platform (1080x1920 unless noted)	Top	Bottom	Sides	Source
TikTok in-feed	240 px	660 px	120 px, plus a 180 px button column on the right	PixFit (secondary); TikTok says the zone varies with caption length (TikTok)
YouTube Shorts	288 px	672 px	left 48 px, right 192 px	Clear Cortex (secondary)
16:9 (1920x1080), broadcast graphics-safe area	5% (54 px)	5%	5% (96 px)	EBU R 95 (official, broadcast; YouTube publishes no figure)

One layout for all three vertical apps: keep text inside x 120 to 780 px and y 288 to 1,248 px on a 1080x1920 frame (Clear Cortex, secondary). Above 840 px down you can widen to x 888. These are ad-placement numbers; organic posts are similar.

8 Effects

The full effects catalogue with one-line recipes is in `reference/06-motion.md`. The short version:

Effect	Use it for	Overdone when
Jump cut	Removing pauses, repeats and false starts	Mid-breath or mid-word; every cut paired with a zoom
Punch-in zoom	Hiding a jump cut by changing the framing	Zooming on every cut; zooming soft 1080p footage
J and L cuts	Letting the voice lead into or out of b-roll	Rarely; watch for mismatched room tone
Match cut	Linking two ideas by matching composition	The match needs explaining
Whip or blur transition	Energetic topic changes	In calm founder films; every transition a whip
Crossfade	Time passing, soft section change	Used to hide every bad cut
Screen-recording zoom	Pointing at the part of a capture you talk about	Constant drift; zooming into blur
Highlight or underline	One phrase in a real document	Highlighting everything
Counter	A number with a source	Counting a number you cannot source
Sound hits	Marking a transition or click	On every caption; louder than the voice

Rules:

- TikTok's own ad guide recommends jump cuts, zooms or text animation to avoid a static talking head (TikTok for Business). No credible source gives an "effects per minute" number, so do not trust one.
- Reset a zoom on the first frame of a new piece. An ease that starts at the in-point leaves the old zoom for one frame (our measurement).
- Never upscale a screen capture past its own pixels. For a big readable detail, rebuild it as type from the real value and keep the capture for context (our working default).
- Show real things. A named product, company or person on screen should be the real logo, photo or capture, with its source written down (our working default).

9 Audio

Full chain: `reference/04-audio.md`. Script: `loudness.py --target social` (-14 LUFS, -1 dBTP; `podcast`, `speech-stream` and `broadcast` set -16, -18 and -23). It runs after captions are burned, on the file you will export.

Targets

Destination	Integrated loudness	True peak	Source
Spotify	Normalises to -14 LUFS	Below -1 dBTP; below -2 dBTP if louder than -14	Spotify
Apple Podcasts	About -16 LKFS, plus or minus 1 dB	Not above -1 dBFS	Apple
AES streaming, speech	-18 LUFS	Not above -1 dBTP	AES TD1008
EBU R128 broadcast	-23 LUFS	Not above -1 dBTP	EBU R128
YouTube, Instagram, TikTok	No target published in any official page found		

A practitioner who measured YouTube in 2018 reports it only turns loud videos down, never quiet ones up (Admiral Bumblebee); this may have changed since. If it still holds, a quiet mix stays quiet. Our working target for voice-only social video is -14 LUFS integrated with true peak at or below -1 dBTP, borrowed from Spotify's figure.

Two-pass loudnorm

ffmpeg's `loudnorm` defaults to -24 LUFS and -2 dBTP, so always set the target (ffmpeg loudnorm). Measure first, then apply:

```
ffmpeg -i in.wav -af loudnorm=I=-14:TP=-1:LRA=11:print_format=json -f null - ffmpeg -i in.wav -af
```

- LRA 11 is our choice; set it at or above the measured LRA so linear mode holds.
- Dynamic mode upsamples to 192 kHz, so always set `-ar 48000` (ffmpeg loudnorm). 48 kHz is YouTube's recommended rate (YouTube).
- Linear mode can silently fall back to dynamic mode. Compare output LRA with input; if it collapsed, use plain gain plus a limiter instead (our measurement).
- **Measure on the encoded file**, not the WAV. A mono WAV read about 3 dB lower than the encoded stereo file, and AAC added up to about 0.4 dB of true peak (our measurements). `loudness.py` re-measures the encoded file and falls back to gain plus a limiter when linear mode will not hold.

Rules

- **Voice only unless you ask.** Music is optional, and many creators add it themselves in their editor or the platform app. The skill never adds it on its own. Remotion and most frameworks give you silent motion; audio is a separate job, as @obaid found.
- **Level your sources before cutting** so the volume does not jump at the cuts (@PrajwalTomar_).
- **If you add music:** one practitioner, @ladprofit, keeps it 8 dB under every word and the final mix at -14 LUFS. Go lower if the voice still fights it on a phone speaker.
- **Do not filter the voice after cleanup**, and do not add more compression; it lifted echo tails in the studio's tests. Turn off any silence cutter in a cleanup tool, because it breaks lip sync (our working defaults).
- **Sound effects need body.** Phone speakers roll off below about 150 Hz, so a pure sub hit is silent on a phone. Layer a 150 to 600 Hz knock (our working default).

10 Export

Exact presets: `reference/07-export.md` and `export.py`.

- **Codec:** h264, yuv420p, constant frame rate at the recorded rate, one audio track, AAC, 48 kHz (YouTube).
- **Reel:** 1080x1920. **YouTube:** 1920x1080.
- **Tag colour explicitly** as bt709 tv range. A full-range versus tv-range mismatch shows as a steady grey offset (our measurement).
- **Encode as few times as you can.** The scripts write h264 CRF 18 with AAC 192k at every stage (later stages copy AAC audio that is already 48 kHz); the final `export.py` encode always uses `-preset slow`, with High profile, BT.709 tags and faststart. For a higher-quality master, pass `--crf 14 --preset slow` to `cut.py` and `captions.py`, and `--crf 14` to `export.py` (our working default).
- **Phone copy under a size cap:** `export.py` makes a two-pass x264 encode with AAC 128k. Video bitrate in kbps = cap in MB x 8192 / duration in seconds, minus 128 kbps for audio and a 40 kbps margin, at most 9,000 kbps and never above the input's bitrate. The cap is `--max-mb`, 25 by default. Check the copy has the same duration and frame count as the master.

11 Taste: how to guide Claude on visual work

The detail is in `reference/09-taste-and-review.md`. This is the core.

Anthropic's own prompting guide warns that without guidance, models default to generic patterns users call the "AI slop" aesthetic (Anthropic). @0xCarnagee: run a vague prompt and you get the same video everyone else got. The fix is not more adjectives. It is a written design language.

Treat Claude as a brilliant new colleague

Anthropic's advice is to treat Claude like a brilliant new employee who does not know your norms (Anthropic). So:

- **Give the reason.** Claude generalises from the reason behind a rule.
- **Say what to do, not only what to avoid.** "Ease out over 18 frames" beats "no bouncy motion".
- **Use examples.** A reference still, a clip cut into frames, a past render you approved.

Numbers, not adjectives

Replace every adjective with a number or a named reference. The values below are examples; set your own.

Instead of	Write
"smooth"	`cubic-bezier(0, 0, 0.2, 1)` over 12 frames
"snappy"	6 frames in, 4 frames out
"not bouncy"	spring damping ratio 1, or `damping: 200`
"big text"	108 px, centre y 760 on 1080x1920 (inside the safe zone)
"stagger the words"	words 3 frames apart
"make it pop"	the word scales 1.0 to 1.1 over 6 frames with ease-out

Put these values in one **token file** (durations, easings, stagger, colours, sizes) and tell Claude to use only those tokens and no inline numbers. @flxrnc hands the file to Opus 5.5 with one line: use only these tokens, no inline numbers.

References as specs

- **Collect what you like** and name what to copy from each: the type, the transition, the pacing. A reference is a spec, not a mood.

- **Make references readable.** Claude only sees the first frame of an animated GIF (Anthropic vision), so send stills. @0xMovez cuts every reference MP4 into a 3x3 grid of frames. @PrajwalTomar_ ran scene detection on his editor's reels so Claude could write down what the editor does.
- **Ask for a breakdown, then a rebuild.** @motion_conquest sends videos he likes and has Claude break down the principles before recreating them. His point: move the model from what it was trained on to what you like.
- **Copy to learn.** Recreate a reference shot exactly before you make your own version, the idea behind Austin Kleon's Steal Like an Artist.

Versions, then pick

One draft is one guess. For every new visual moment, ask for several genuinely different directions: different shapes or ideas, not one drawing with five finishes. The studio's habit is five versions; a quick test is to overlay them, and if two share most of their shape, replace one (our working default). Anthropic's own design harness ran 5 to 15 rounds, each pushing toward something more distinctive (Anthropic, harness design).

Stills before motion

Render the landed frame of each version first, at final size on your real footage, with captions where they will be. Pick one. Only then build motion. A rejected still costs seconds; a rejected render costs minutes. And stills are what Claude can actually review.

Three gates, each approved by you: stills, then the picked motion clip on its own, then the stitched film. Nothing goes into the film before you pick it (our working default).

A taste file

Keep one file of what you love and hate, in your own words, dated, with red lines at the top. The newest note wins where two conflict. Every rule says what to do instead. Claude reads it at the start of every session. @ladprofit describes the same loop: each note teaches the agent your taste, one at a time. A template is in `templates/taste.md`.

Write only rules you mean

Newer models follow written rules closely. Keep the taste file short and true. Delete a rule you no longer believe.

12 The review loop

Make Claude look before it changes anything

@flxrnc tells Claude to render, open specific frames (for example 0, 24 and 48), and describe what it sees before changing code. Anthropic's design agents worked the same way: screenshot, study, then fix (Anthropic, harness design).

- **Crop the problem area at full size.** Claude makes more mistakes on small images under 200 px, and zooming into regions improves its judgement (Anthropic vision, prompting guide).
- **Keep each image at or under 3,000 px** on its longest side when you send more than 20 (Anthropic vision).

Give notes by frame number

"The punch at frame 84 peaks late, move it 2 frames earlier" is a note Claude can act on. "It feels off" is not (@flxrnc). Ask for a frame counter in a corner of the preview and a pause key, so you can name frames. Remove the counter before export.

Automated checks

Run `qc.py` before calling anything done. It checks aspect, codec, pixel format, frame rate, duration, loudness, true peak, audio tracks, and black or frozen stretches. ffmpeg's `freezedetect` prints the start and end of every stretch where nothing changes (@flxrnc):

```
ffmpeg -i final.mp4 -vf freezedetect=d=0.5 -map 0:v:0 -f null -
```

Test your detectors on a known-bad version first. If a detector finds nothing on a file you know is broken, the detector is wrong (our working default).

A separate reviewer

Agents reliably rate their own work too kindly (Anthropic, harness design). Separating the maker from the judge is a strong lever. So:

- Start a fresh Claude session, or a subagent, that did not make the work.
- Give it written criteria, not "is it good". Anthropic graded design on named criteria such as quality, originality and craft.
- Tell it to measure first: frames around every cut, caption timing, safe zones, loudness. Then look.
- Cap it at two passes per version: pass 1 the maker fixes everything, pass 2 report only, then it comes to you with the list of what is still wrong (our working default).

The human watch

No agent signs off anything checked by ear: joins, voice clarity, sound effects on headphones and on a phone speaker (our working default). Watch the whole video with sound, once on a laptop and once on a phone. That is the last gate.

13 Practitioner tips

Tips from long-form posts on X that the sections above do not already cover, fetched 2026-11-22. Each is paraphrased and linked.

1. **Plan before you build.** Ask for a plan of timing, scenes, text and graphics first. (@Shpigford)
2. **Write each scene as text plus timing.** State the exact on-screen text, then enter, hold and exit in seconds, and name one animation system. (@rryssf)
3. **In Remotion, drive everything from the frame.** Use only `useCurrentFrame`, `interpolate` and `spring`; no CSS transitions. (@rryssf)
4. **One-shot generators give a wow moment, not a film.** Add your own story and constraints. (@rryssf)
5. **Without a framework, build animations as one HTML file with a `seek(t)` function** that paints the exact frame for time t, then capture frame by frame. HyperFrames (a paused GSAP timeline on `window.__timelines`) and Remotion (`useCurrentFrame`) do the seeking for you. (@0xclayo)
6. **Chain skills.** For example, a browser skill pulls the real data and a video skill animates it. (@kaxil)
7. **Treat three.js as an advanced option.** In our search of posts on X, it appears in nearly every list of tools but in only one post that showed a finished video. Use it inside Remotion or a HyperFrames 3D block when you need depth.

14 Common mistakes and their fixes

Mistake	What you see	Fix
Cutting on Whisper word edges	Clipped first or last sounds, misheard captions	Cut in measured silence; pad about 0.15 s after the last word
Per-piece `-ss` / `-t` encodes joined with the concat demuxer	Lips drift 1 to 3 frames behind the voice	One pass: `trim` / `atrim` and the `concat` filter, frame-snapped
Joining clips that each carry their own AAC	Voice slides 8 to 16 ms per join	Build the voice once from the source audio
No fades at joins	Clicks and pops	12 to 30 ms fades at real cuts only
Fades between touching pieces of one take	A notch in the middle of a word	Merge touching pieces into one
Tight join after a sentence end	Ear hears a stutter	Keep the preset's sentence pause (0.22 s on reels)
Captions drift on a long cut	Words land 0.3 to 1 s off inside phrases	Re-transcribe the kept pieces with `transcribe.py --pieces edit/cut.json`
Trusting Whisper on names and numbers	Wrong words on screen	Prompt with names; keep a fix table; read every word
Skipping the transcript read	A misheard phrase burned into captions	Read the whole transcript, fix with `transcribe.py --fix`, then caption
Fading each caption word in from zero	Flicker	Hard-cut word to word, or ease scale
Captions on a light shirt	Unreadable white text	A soft dark pill behind the word
Captions at a fixed height during a zoom	Text touches the chin	Move captions with the zoom
Spring with default settings	Bounce you did not ask for	Damping ratio 1 or more
Animation that steps state per frame	Different result when the renderer seeks	Compute every value from the frame number
Scaling from 0	Unnatural pop	Start at 0.9 or higher

Mistake	What you see	Fix
Two-pass loudnorm left unchecked	Flattened dynamics	Compare output LRA with input; use gain plus limiter if it collapsed
Measuring loudness on the WAV	Final file louder or peakier than you think	Measure on the encoded file, as `loudness.py` does
`-shortest` with AAC	Last frame missing	Count frames after every mux
Agent grading its own render	Everything "looks great"	A separate reviewer with written criteria
Sending a GIF for review	Claude sees one frame	Send stills or a contact sheet
Shipping without listening	Doubled words, hiss, a jingle at the start	A human watches with sound
Deleting or overwriting originals	Lost footage	Work in `edit/` ; never write into the source folder

15 Resources

Tools

- ffmpeg filters, including loudnorm and the xfade gallery
- whisper.cpp and faster-whisper
- HyperFrames (Apache 2.0) and its docs index for LLMs
- Remotion, its licence and pricing
- Remotion and three.js, three.js, React Three Fiber
- GSAP licensing (free for commercial use under Webflow's standard licence)
- Manim Community

Official and community skills

- Remotion Agent Skills (link only; no licence file)
- HyperFrames skills (Apache 2.0)
- GSAP AI skills (MIT)
- anthropics/skills, including skill-creator for packaging your own
- claude-code-video-toolkit (MIT)
- ECC video-editing skill (MIT)
- manim_skill (MIT)

Anthropic guidance

- Agent Skills overview and best practices
- Prompting best practices and prompting Opus 5.5
- Vision
- Harness design for long-running apps
- Improving frontend design through skills

Craft and standards

- Motion: NN/g animation duration, Material 3 easing and duration, Emil Kowalski, 7 practical animation tips, Apple HIG motion
- Captions: BBC subtitle guidelines, Netflix timed text style guide, WCAG contrast
- Loudness: EBU R128, AES TD1008, Spotify
- Taste: Emil Kowalski, developing taste, Paul Graham, taste for makers, Ira Glass on the gap

Posts on X worth reading

- @flxrnc on prompting motion and on springs and checks
- @ladprofit on cut and caption rules
- @PrajwalTomar_ on learning an editor's style
- @0xclayo on the seek(t) pattern
- @motion_conquest on references and taste
- @rryssf on scene prompts
- @notmissing_ on review gates

A last word

A last word on taste. Early on, your taste is ahead of your skill and your work will disappoint you. Ira Glass's advice is to do a huge volume of work (James Clear). With Claude, volume is cheap: five versions, a written rule for every note, and one honest watch with sound at the end.

You don't need us for any of this. Where we help at Velorum is the next step for a business: turning the jobs your team repeats into agents and automations that run on their own, connected to the tools you already use.

Book a call with Velorum

Sources

Every link in this guide, under the section it supports. Fetched and checked on 2026-11-22. Figures marked "our working default" or "our measurement" are the studio's own and carry no link.

1. How AI editing works, end to end
@flxrnc, whisper.cpp, faster-whisper, ffmpeg, HyperFrames, Remotion, @notmissing_, Anthropic, prompting best practices, @PrajwalTomar_, Anthropic, skills overview, @motion_conquest, @timkochjar, @Shpigford

2. Set up once
README, Anthropic, skills overview, Claude Code docs, Claude help, models overview, prompting Opus 5.5

4. Make the cut
@MushtaqBilalPhD, @notmissing_, @socialwithaayan, @ladprofit

5. Choose a motion framework
LGPL 2.1+, GPL optional, HyperFrames, Remotion, Company License, Agent Skills, shut down on 2026-10-26, @remotion/three, Manim, manim_skill, Motion Canvas, Revideo, @liu8in, @mvanhorn, @thacreativeguru

6. Motion principles in numbers
@0xclayo, NN/g, Emil Kowalski, Material 3, Remotion docs, Material 2, @flxrnc, Emil Kowalski, @flxrnc, Remotion docs, TransitionSeries, twelve principles of animation, rotary disc shutter, Remotion motion blur, YouTube encoding, WCAG 2.3.1, Apple HIG, @0xCarnagee

7. Captions
BBC, Netflix, DCMP, WCAG 1.4.3, @notmissing_, @ladprofit, @PrajwalTomar_, @socialwithaayan, WCAG 1.2.2, AdsUploader, PixFit, TikTok, Clear Cortex, EBU R 95

8. Effects
Jump cut, J and L cuts, Match cut, Whip or blur transition, TikTok for Business

9. Audio
Spotify, Apple, AES TD1008, EBU R128, Admiral Bumblebee, ffmpeg loudnorm, YouTube, @obaid, @PrajwalTomar_, @ladprofit

10. Export
YouTube

11. Taste: how to guide Claude on visual work
Anthropic, @0xCarnagee, @flxrnc, Anthropic vision, @0xMovez, @PrajwalTomar_, @motion_conquest, Steal Like an Artist, Anthropic, harness design, @ladprofit

12. The review loop
@flxrnc, Anthropic, harness design, Anthropic vision, prompting guide, @flxrnc

13. Practitioner tips
@Shpigford, @rryssf, @0xclayo, @kaxil

15. Resources

loudnorm, xfade gallery, whisper.cpp, faster-whisper, HyperFrames, docs index for LLMs, Remotion, licence, pricing, Remotion and three.js, three.js, React Three Fiber, GSAP licensing, Manim Community, Remotion Agent Skills, HyperFrames skills, GSAP AI skills, anthropics/skills, claude-code-video-toolkit, ECC video-editing skill, manim_skill, Agent Skills overview, best practices, Prompting best practices, prompting Opus 5.5, Vision, Harness design for long-running apps, Improving frontend design through skills, NN/g animation duration, Material 3 easing and duration, Emil Kowalski, 7 practical animation tips, Apple HIG motion, BBC subtitle guidelines, Netflix timed text style guide, WCAG contrast, EBU R128, AES TD1008, Spotify, Emil Kowalski, developing taste, Paul Graham, taste for makers, Ira Glass on the gap, @flxrnc on prompting motion, on springs and checks, @ladprofit on cut and caption rules, @PrajwalTomar_ on learning an editor's style, @0xclayo on the seek(t) pattern, @motion_conquest on references and taste, @rryssf on scene prompts, @notmissing_ on review gates