READ THIS FIRST

If you're not technical, don't build this. Book a call and I'll build it for you, on your machine, with your look on it. **Book a call with Keira**

calendly.com/realmissai/new-meeting

MISS AI · THE METHOD

How that video edited itself

You commented MISS AI, so here it is. The pipeline that cut the video you just watched, the numbers that control how it feels, the parts that don't work yet, and the parts I'm keeping.

~~2:29~~ → **1:24**

65 SECONDS OF NOTHING, GONE, WITHOUT WATCHING A TIMELINE ONCE

> Fair warning first: this isn't a product you can download. It's a method, built out of four free tools plus a few hundred lines of glue.
>
> I'm giving you the method and the numbers, because that's the part that took me the time. Rebuilding the glue once you know the shape is a weekend, not a month.
>
> If it turns out you'd rather not spend the weekend, there's a way to skip it at the bottom.

The idea the whole thing rests on

The unit is the line, not the video.

That sounds like nothing. It's the entire trick.

If you record yourself rambling for three minutes and then ask software to find the good bits, you're asking it to guess. Guessing is why every auto-editor you've tried gave you clips that start halfway through a sentence.

If you write the lines first, read them to camera, fluff them, say them again better, then the software isn't guessing. It's matching what you said to the line you meant to say. That's alignment, and alignment is reliable.

Same footage. Completely different problem to solve.

How to record so the software can do its job

This part is free and it's most of the win. Get the recording right and the rest is arithmetic.

- **One continuous recording. Don't stop and start.**
  The whole point is that it finds every attempt at every line and keeps the best one, so messing up costs you nothing.

- **Fluff it on purpose.**
  Say a line, hate it, say it again. Say it four times. More attempts means a better winner, and you're no longer performing, you're just talking until one comes out right.

- **Leave about a second of silence between takes.**
  That gap is what lets the cutter find clean edges. It's the single thing most likely to make your first attempt go badly if you skip it.

- **Don't be word perfect.**
  The matcher expects paraphrase, dropped filler and small reorderings. Say it how it comes out.

- **Talking between lines is fine.**
  "Sorry, again" is fine. Junk at the top and the tail is fine. It all gets cut.

- **Skipping a line is fine.**
  It comes back flagged as missing rather than quietly vanishing, which is a rule worth holding everywhere in a pipeline like this.

The video you watched was one pass in my kitchen with the phone propped up. Full frame, face fills it, no split screen.

The seven steps

01

Write the script as numbered lines.

Not paragraphs. Lines, each tagged with what it's doing: hook, what-it-is, how, result, value, cta.

The tag isn't decoration and it does two jobs. It stops you recording something shapeless, because you can see when you've written four explanations and no payoff. And later it tells the graphics layer which asset is even eligible at that moment, so a callout can't turn up on your hook.

02

Read it to camera. Fluff it as much as you like.

Everything in the section above. Don't stop recording, don't do anything clever.

03

Transcribe with word-level timings.

WhisperX, free, runs on your machine. You need per-word start and end times, not sentence-level subtitles. Everything downstream keys off word edges, and a cut placed on a sentence boundary is exactly how you get clipped words.

First run pulls down a few gigabytes of models. Budget for that once and never think about it again.

04

Find every attempt at every line.

Score each stretch of transcript against each script line. The scoring has to tolerate you rewording things, because you will. One line in that video was scripted as "a skill doesn't, it's still there tomorrow" and I said "a skill will be inside your AI every single time you use it". Three words in common. Obviously the right take.

Watch for the stupid stuff here. A transcriber writes "fifteen" one time and "15" the next for the exact same word, said the exact same way, and suddenly one take scores as more complete than another for a reason that has nothing to do with your delivery. Numbers have to be flattened to one form on both sides before anything is compared.

05

Pick the winner in script order.

This is the bit people get wrong. Don't pick each line's best match on its own. Assign them in order, maximising confidence across the whole script.

Order is a stronger signal than word overlap, and it's what rescues a line you rewrote completely, because there's only one place it could sit. It also means overlaps sort themselves out: two lines can't claim the same seconds if the sequence has to hold.

The cost of that assumption, stated plainly: if a recording wanders badly out of script order, this is the thing that breaks.

06

Cut on word edges, then kill the silence.

Three numbers, and these are measured off a working pipeline, not invented:

VALUE	NAME	WHY
0.04s	lead	Padding before the first word, so it's never clipped.
0.06s	tail	More than the lead, because opening in dead air reads as a slow cut and ending a fraction late reads as breathing room.
0.22s	gap	The longest silence allowed to survive inside a take.

That last one is the tightness dial and it's the one you'll want to fiddle with. It's why the finished video runs at about 250 words a minute instead of 180.

One consequence worth knowing before it confuses you: a single line often comes out as several separate pieces, because any silence longer than the gap gets removed by splitting the clip in two. Your line count and your clip count won't match, and that's correct.

On the video you just watched: 2 minutes 29 down to 1 minute 24. Sixty-five seconds of nothing, gone, without me watching a timeline once.

07

Stitch, then render the graphics on top.

ffmpeg for the stitch. Use `trim` in a filter graph, not `-ss` with stream copy: copy snaps to the nearest keyframe, which on a phone recording can be seconds away from where you meant to cut.

Then Remotion for everything on top. It's React, so a caption is a div and an animation is a CSS transform. If you can build a web page you can build these graphics, which is the whole reason I stopped looking for a video editor.

The one place a human still decides

Every run spits out a page listing each line with the take it chose and every alternate it rejected, all of them playable. You skim it, you play the two you're unsure about, you override anything you'd rather have.

It takes about a minute and it's the only gate. Build one before you build anything fancy, because it's what turns "the software cut my video" from a thing you have to trust into a thing you can check.

It's also where the failures surface. A line that couldn't be matched shows up flagged and waiting, never as a silent wrong cut. That distinction is worth more than any amount of accuracy.

The part that makes graphics survive a recut

Anchor every graphic to a **line number**, never to a timestamp.

"Callout on line 6", not "callout at 24.3 seconds". Change your mind about which take wins, re-run, and every graphic still lands where it should. Anchor to seconds and one re-cut throws the entire beat sheet out by two frames and you're placing them all again by hand.

That one decision is the difference between a pipeline you use and a pipeline you abandon after three videos.

Two things that will

cost you an afternoon

Both of these took me longer than they should have, and neither is written down anywhere obvious.

Give the render far more time than feels sane. Font loading runs on the render's overall timeout, so a long video with a couple of typefaces quietly runs out of road. The error it prints never mentions fonts, and the stack it shows you points at the font loader rather than at whatever was actually slow. A short test render passing proves nothing about a long one, because more frames means everything stays alive longer.

A cancelled render poisons the next one. Stop a render partway and it leaves a half-finished cache behind, so the very next run fails with the same error for a completely unrelated reason. If a render breaks right after you killed one, just run it again before you believe a word of the message.

What it actually cost

TOOL	WHAT FOR	PRICE
`WhisperX`	word timings	`free, runs locally`
`ffmpeg`	the stitch	`free`
`Remotion`	captions and graphics	`free for individuals`
`Claude Code`	wrote the glue, and does the directing	`paid plan`

The glue is a few hundred lines of Python doing the align, choose and cut, and some React for the graphics. Claude wrote nearly all of it, which is the joke inside the joke.

What I'm not publishing, and why

You've got the shape of the whole thing. Three pieces of it I'm keeping, and I'd rather name them than have you assume I forgot.

- **The scoring.** How a stretch of transcript gets matched to the line it belongs to, and the thresholds that decide when a match is good enough to win. That's the piece that took the longest and broke twice on real footage before it held.
- **The ordering method.** Step 5 tells you to assign in order across the whole script. Doing that well, rather than greedily, is its own problem.
- **The graphics layer.** The vocabulary of beats and the style library the renderer draws from. This is the part that makes it look like mine.

The reason is straightforward. I'm turning this into a product, and a format stops working the moment fifty people run it identically. You can build all three yourself from what's here. I'd just rather you build your own version than paste mine.

What doesn't work yet

I'd rather tell you this than have you find out.

- × **No finger tracking.** "Follow my finger" in that video is a punch-in and a slow pan I placed by hand. It looks like tracking. It isn't.
- × **Short lines are fragile.** A three-word line like "Zoom in here" can get swallowed by a longer overlapping match. It shows up as a line needing a manual pick, never as a silent wrong cut, but it does need a human eye.
- × **Nothing writes the script.** Step 1 is still entirely me. That's the next thing I'm building.
- × **B-roll is manual.** The software puts an asset on screen. Choosing a good one is still a person with taste.

If you'd rather not build it

Most people who read this far won't build it, and that's a fair call rather than a failure. It's a Python environment, a transcription model and a React renderer. If those three things don't already live on your machine, the weekend I mentioned is not a weekend.

So the other option: I'll set it up for you. On your machine, with your look on the graphics, running on your own footage, and I'll show you how to drive it so you're not calling me every time you make a video.

Book a call and tell me what you're making and how you're editing it now. If you don't need me, I'll say so on the call and you'll still leave with a better answer than you came with.

BOOK A CALL WITH KEIRA

calendly.com/realmissai/new-meeting

This is the same system that cut the video you just watched. I run my own show on it every week, which is the only reason I'd put my name on it.

If you only take one thing

Script it in lines. Everything else here is downstream of that decision, and it's the one that costs you nothing to try tomorrow.

Built in public. If you rebuild this, I'd genuinely like to see it.
Keira, @realmissai