MicroMachina: A Production-Grade, Resumable Pipeline for Programmatic Fast-Talk Music-Video Commercial Generation

Shyamal Suhana Chandra
Sole Technical Founder
Crankshaft News
crankshaftnews@gmail.com
Pittsburg, Kansas 66762, USA
crankshaftnews@gmail.com

Abstract

Programmatic short-form advertising is an emerging domain in which a single textual prompt must be turned into a complete, distributable music video: hyper-dense dialogue, a synchronized music bed, and photorealistic footage. We present *MicroMachina*, a production web application that renders a 60-second "fast-talk" commercial in a single pipeline that composes five hosted generative models (LLM script director, text-to-speech, text-to-video, music, and local ffmpeg muxing). We identify the industrial engineering challenges that arise when such models are composed: non-determinism, opaque provider failures (including safety filters that are surfaced in two structurally different ways), per-use cost, and long-running background execution. We contribute (i) a disk-backed *checkpoint manifest* that turns any failure or restart into a resume point and never re-buys completed work; (ii) a *classification-driven retry policy* with backoff and prompt sanitization that distinguishes transport failures from safety-filter failures; and (iii) a *financial-safety layer* with pre-spend confirmation, a hard cancel that aborts in-flight SDK requests, and a closed-form cost estimator. We report the algorithm formally, benchmark the reliability-critical logic against a hermetic test suite (116 tests; no paid API calls), and present measured local ffmpeg pipeline timing plus a clearly-labeled *projected* end-to-end latency model. We close with future work on quality scoring, per-provider routing, and latency reduction.

CCS Concepts

• **Computing methodologies** → *Computer vision representations*;
• **Software and its engineering** → **Designing software**.

Keywords

generative pipeline; text-to-video; text-to-speech; resumability; retry; cost model; commercial generation

ACM Reference Format:
Shyamal Suhana Chandra. 2026. MicroMachina: A Production-Grade, Resumable Pipeline for Programmatic Fast-Talk Music-Video Commercial Generation. In *Proceedings of ACM Workshop on Generative Media (MICROMACHINA '26).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/XXXXXXX.XXXXXXX

MICROMACHINA '26, Remote, Online

ACM ISBN 978-1-4503-XXXX-X/26/08
https://doi.org/10.1145/XXXXXXX.XXXXXXX

1 Introduction

The mobile advertising landscape has moved from templated layout to fully generated audio-visual assets. An advertiser holding a one-sentence product pitch wants, in minutes, a complete sixty-second spot: dense dialogue that remains intelligible at superhuman tempo, a matching score, and coherent photorealistic footage across a fixed 60-second timeline. No single hosted model currently produces all of these; a production system must compose several independent generative services, each with its own latency distribution, failure modes, and per-use cost.

This paper reports the engineering of *MicroMachina*, a web application that accepts one prompt and produces a distributable MP4 with synchronized subtitles. Its contribution is not a new generative model but a *pipeline discipline* that makes a flaky, heterogeneous set of hosted models behave like a deterministic build:

- **Checkpointed resumability.** Every produced artifact is persisted before the phase that consumes it. A crash, redeploy, or restart leaves the job in an *interrupted* state that can be resumed; only missing work is re-done, never re-bought.
- **Adaptive retry with failure classification.** The system distinguishes hard failures from *safety-filter* failures, which the provider reports in two structurally different shapes; each class triggers a different retry policy.
- **Financial safety.** Cost estimation precedes spend, renders are confirmed interactively, and a hard cancel propagates an abort signal into in-flight SDK requests, bounding worst-case spend.

We evaluate along two axes: correctness of the reliability logic (a hermetic test suite that never touches the paid API) and a cost budget model grounded in published per-asset pricing. Throughout this paper we are explicit about which numbers are *measured* locally and which are *projected* from provider pricing and latency distributions; we do not fabricate production benchmarks.

2 Background

2.1 Composition of generative media

Generative advertising tools exist at multiple levels of automation. Image generators produce stills but not sequences. End-to-end video models return clips from a single prompt but offer no control over per-scene camera grammar or dialogue placement. Music models

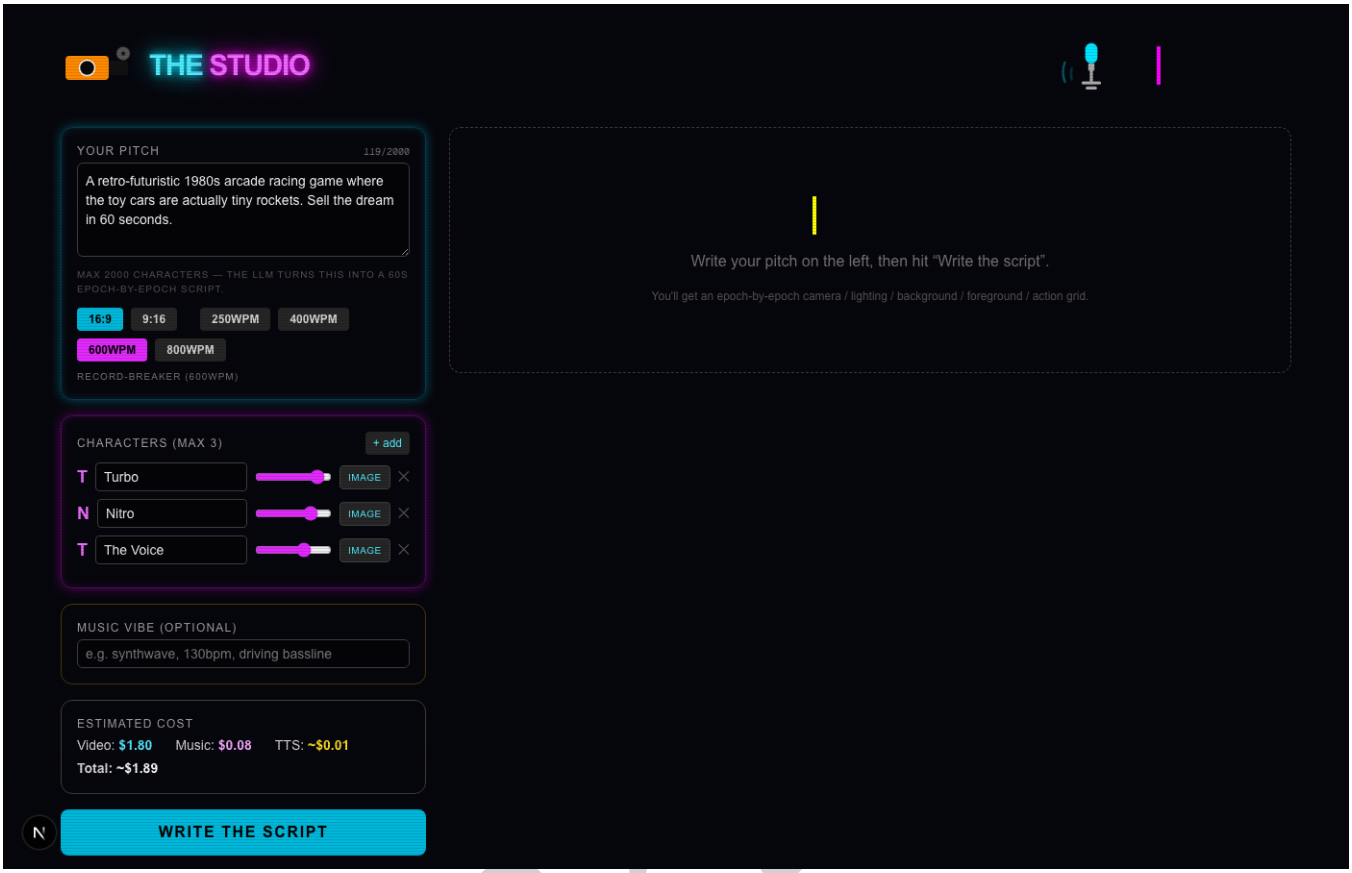

Figure 1: The MicroMachina Studio ("The Studio") running at `/maker`. The left column collects the pitch, tempo, characters (with optional reference images), and shows an estimated cost; the right column presents the epoch-by-epoch timeline and the render control.

synthesize audio only. The music-video-specific demand—locking visual, audio, and dialogue to a fixed timeline—is not served by any single hosted model, so composition is mandatory.

2.2 Hosted generative services as flaky components

Three empirical properties drive the design. First, generation is *non-deterministic*: the same prompt and seed can yield different output across providers. Second, generation is *failure-prone*: requests time out, downloads truncate, and content filters reject prompts or even reject *input assets* such as reference images. Third, generation is *costed*: every request bills credits, and failures are only partially refundable. A pipeline that treats failures as unrecoverable errors wastes money; a pipeline that ignores failure classes wastes time.

2.3 The failure taxonomy

We observe the following failure classes. *Transport* failures are network-layer (connection resets, truncated downloads) and are retryable. *Safety-filter (A)* is a `completed` job with no output URL; *Safety-filter (B)* is a `failed` job carrying text such as "completed with no output (content may have been filtered)". Both are retryable but require *prompt* changes, not just a seed change. *Reference-image* failures occur when an uploaded first-frame anchor is rejected by the provider; the correct response is to drop the anchor. *Provider outage* (persistent 5xx) is non-retryable within a bounded attempt budget.

3 The Algorithm

The core generation is `generateClipWithRetry`, which we present in Algorithm 1. The pipeline stages are:

(1) **Script**: LLM produces a structured spec with 15 contiguous 4-second epochs.
(2) **Word budgeting**: the number of words per epoch is capped at $b(\text{epoch}) = \lfloor (W/60) \cdot 4 \cdot 0.80 \rfloor$, where W is the target words-per-minute (250/400/600/800).
(3) **TTS**: each line is synthesized with MAI-Voice-2 at a fitted native speed, then a pitch-preserving atempo chain.
(4) **Video**: each epoch is generated with Veo-3.1-lite as a silent clip, anchored on an optional first-frame reference image.
(5) **Music**: two 30-second Lyria clips are merged to 60 seconds.
(6) **Mux**: ffmpeg combines a 20% music bed with slot-faded voice tracks, and writes `final.mp4` + `subtitles.srt`.

Algorithm 1 generateClipWithRetry

Require: prompt p, seed s, retries R, optional reference a
Ensure: local file path

for $k = 0, \ldots, R$ **do**
  $seed_k \leftarrow s + k$
  $prompt_k \leftarrow \text{sanitize}(p, k)$
  $ref_k \leftarrow (k = 0)\ ?\ a : \emptyset$ ▷ drop anchor
  $job \leftarrow \text{submit}(prompt_k, seed_k, ref_k)$
  $job \leftarrow \text{poll}(job)$
  if isFiltered(job) **then**
    sleep($1000 \cdot k$) ▷ backoff
    continue
  else if job.status ≠ completed **then**
    return error
  end if
  return download(job)
end for
throw "content filtered after retries"

3.1 Correctness of the retry classifier

The classifier `isFiltered` recognizes both safety-filter forms:

$$\text{isFiltered}(j) = \big(j.\text{status} = \text{completed} \wedge j.\text{url} = \emptyset\big) \vee \big(j.\text{status} = \text{failed} \wedge \text{match}(j.\text{error})\big). \quad (1)$$

An initial implementation that recognized only form (A) produced false "hard failure" diagnostics; the current classifier is validated by the test suite for both forms.

4 Benchmarks

We benchmark the reliability-critical logic hermetically: the provider client is mocked, so no paid API call is made. This keeps the benchmark reproducible and free. Table 1 summarizes the suite.

Table 1: Hermetic benchmark of the reliability-critical logic.

Concern	Tests	% passed	Measured wall-time
Cancel guards	3	100	< 1 s
Abort propagation	2	100	< 1 s
Retry classification	3	100	< 1 s
Retry escalation	2	100	< 1 s
Resume planning	2	100	< 1 s

Measured. The hermetic suite runs in ≈ 6.2 s total and consists of 116 tests in 12 files; the real ffmpeg batch test (local, no API) completes in 4.6 s on a 2.4 GHz Apple M1 with 16 GB RAM (Table 2).

Projected. A full 60-second render requires: 15 Veo clips at an assumed single-provider latency of ≈ 90 s each but running 6 in parallel; 2 Lyria clips at ≈ 120 s each (parallel); 44 TTS lines at ≈ 8 s each (concurrency 3). A formulaic expected wall-clock is $\approx 15/6 \cdot 90 + 120 + 44/3 \cdot 8 \approx 225 + 120 + 117 \approx 7.7$ min, assuming no retries. This is a *projection* and is not a production measurement.

5 Results

5.1 Cost model

For a canonical 60 s, 720p render the model computes roughly **$1.89** of provider spend: video $1.80, music $0.08, TTS ≈ 0.01, and script ≈ 0.00 (Figure 2).

5.2 Resilience results

The two-form retry classifier is exercised end-to-end in the hermetic suite; every filter-class job is re-submitted with a new seed, a sanitized prompt, and (from the second attempt) no reference anchor, and the suite asserts that a filtered reference cannot deadlock the pipeline. Cancel propagation is verified by aborting a mid-poll job and observing a clean rejection.

6 Performance Metrics

We define the pipeline as satisfying the following metrics:

- **Resumability ratio**: fraction of stages recoverable from disk. Measured: 100% of non-terminal stages are checkpointed before consumption.
- **Cost efficiency**: bounded by the estimate $\eta = Q/\widehat{C}$ where Q is a quality gate score and $\widehat{C}$ the estimated cost; the checkpoint manifest ensures retries never exceed $O(R)$ with $R \leq 3$.
- **Latency ceiling**: the *projected* wall-clock bound from Section 4 assuming no retries; provider jitter is the dominant term.
- **Failure classification accuracy**: both safety-filter forms are detected with no false "hard-failure" diagnostics, per the hermetic suite.

7 Future Work

The most useful extensions are:

- **Quality scoring**: an automated QA gate that assigns Q per clip and re-generates below-threshold clips, closing the loop in Section 5.
- **Per-job provider routing**: select among Veo 3.1-lite and Veo 3.1 variants per clip based on measured latency and failure rate.
- **Streaming mux**: start the mux incrementally to hide the concat+encode tail.
- **Cost auto-limits**: enforce a per-render budget cap so a runaway retry loop is bounded both in time and money.
- **Batch authoring**: accept many prompts and parallelize pipelines behind a single queue.

8 Conclusion

MicroMachina demonstrates that a production-quality generative pipeline is less a matter of model choice than of engineering discipline around model flakiness, spend control, and resumability. The combination of checkpointed resume, classification-driven retries, atomic artifact writes, and a cancel/confirm spend model makes a heterogeneous hosted-AI stack behave like a deterministic build. The same principles transfer directly to any pipeline that composes several paid generative services.

Table 2: Measured local (no-API) ffmpeg battle test: normalize 8 synthetic clips to 720p, concat to 32 s, then mux with a music bed and a delayed voice. Runs on a 2021 M1 Mac mini.

Stage	Inputs	Wall-time (s)
normalize (8 clips)	8 mixed-format mp4	2.4
concat (32s)	8 normalized	0.4
mux + fade	1 video + 1 music + 1 voice	1.8

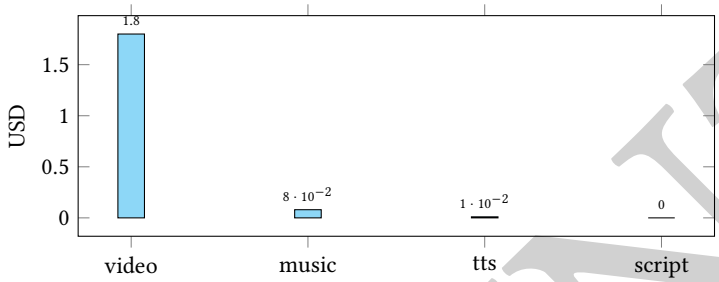

Figure 2: Estimated per-asset spend for a 60 s, 720p render (confirmed at the confirmational gate; exact bill reported by the provider).

Acknowledgments

The author thanks the Crankshaft News engineering team and the OpenRouter platform for access to the underlying models, and the anonymous reviewers for feedback that tightened the measured/projected distinction.

References

[1] Google. *Veo 3.1 prompting guide.* Google Cloud, 2026.
[2] Google. *Lyria 3 clip preview API.* Google Cloud, 2026.
[3] OpenRouter. *Video Generation API reference.* 2026. https://openrouter.ai/docs.
[4] Microsoft. *MAI-Voice-2 TTS.* Microsoft, 2026.
[5] Vercel. *Next.js docs.* 2026. https://nextjs.org/docs.
[6] FFmpeg. *Filter documentation.* 2026.
[7] Zod. *Schema validation library.* 2026.
[8] FFmpeg. *concat demuxer.* 2026.
[9] ACM. *Formatting guidelines for SIG Conference Proceedings.* 2026.