

COVER SHEET: PROVISIONAL APPLICATION FOR PATENT

This is a provisional application for patent under 35 U.S.C. § 111(b) and 37 CFR § 1.53(c).

Field	Detail
Title of Invention	Systems and Methods for Phase-Coupled Deterministic Audiovisual Timeline Composition and Authority-Gated Transcript Mutation
Inventor	Brennan William DeCrow, Northridge, California, United States
Correspondence Address	18617 Chase Street, Northridge, CA 91324, United States
Attorney Docket Number	MM-SLASH-PROV-003
Related Disclosures	U.S. Provisional Application No. 64/159,586 (filed September 22, 2026) and U.S. Provisional Application No. 64/161,921 (filed September 24, 2026), by the same inventor.
Entity Status	Micro-Entity (Certification Form PTO/SB/15A)
Government Interest	None.

This document serves as the provisional application cover sheet under 37 CFR § 1.51(c)(1). Brennan William DeCrow is the sole inventor identified for this application. No attorney or agent is appointed. No U.S. Government agency has a property interest in the invention. The applicant requests treatment as a provisional application under 35 U.S.C. § 111(b) and 37 CFR § 1.53(c). If the filing fee is not paid on submission, the applicant will timely submit the required filing fee and any applicable late surcharge in response to the Office's notice.

Signature: _____ Date: September 25, 2026

Brennan William DeCrow

Sole Inventor & Pro Se Applicant

SPECIFICATION: SYSTEMS AND METHODS FOR PHASE-COUPLED DETERMINISTIC AUDIOVISUAL TIMELINE COMPOSITION AND AUTHORITY- GATED TRANSCRIPT MUTATION

1. Technical Field

[0001] This disclosure relates generally to digital media processing and computer-implemented video editing, and more particularly to systems, methods, and non-transitory computer-readable media for deterministic, authority-gated transcript-driven video editing and non-linear phase-coupled audiovisual timeline synchronization in browser and distributed runtime environments.

2. Background of the Invention

[0002] Modern digital non-linear video editors (NLEs) and transcript-based video editing applications allow users or automated machine agents to edit video files by editing corresponding text transcripts. However, conventional architectures suffer from several fundamental technical deficiencies:

[0003] First, conventional transcript-based editing tools decouple the transcript editing interface from the underlying video/audio media stream revision state. If an audio or video track is updated, transcoded, trimmed, or substituted, existing systems frequently allow downstream edits against stale or unaligned word-boundary timestamps. This causes severe audio-visual drift, clipped syllables, and corrupted video edit decision lists (EDLs).

[0004] Second, existing automated timeline “tightening” algorithms (e.g., removing pauses, filler words, or silence gaps) operate with blunt threshold filters. They cannot differentiate between dispensable hesitation syllables (“um”, “uh”) and critical brand names, technical terms, or legal disclaimers that may happen to fall adjacent to silence, leading to inadvertent and unrecoverable content destruction.

[0005] Third, synchronizing visual elements (such as caption animations, B-roll cut transitions, and visual HUD overlays) with audio rhythm and speech cadence has traditionally relied on crude, static keyframing or computationally prohibitive audio spectral analysis that fails to produce natural, organic pacing. Existing

timeline engines do not possess an intrinsic mathematical framework for coupling visual keyframe snap intervals to audio beat transients across multi-modal tracks.

[0006] Fourth, conventional video rendering pipelines require heavyweight, platform-dependent native binary wrappers (such as FFmpeg executables or proprietary C++ rendering libraries), preventing lightweight, zero-dependency, deterministic video synthesis directly within sandboxed browser environments or WebWorker threads.

[0007] Accordingly, there is an urgent technical need in the art for an authority-gated, deterministic, phase-coupled media timeline engine that prevents unaligned mutations, preserves protected transcript entities, synchronizes visual keyframes to audio dynamics via coupled non-linear oscillators, and executes deterministically across heterogeneous compute platforms.

3. Summary of the Invention

[0008] To solve the aforementioned problems, the present disclosure provides systems, methods, and software architectures for:

[0009] (1) Authority-Gated Transcript Edit State:

A media project maintains an immutable media revision counter (ρ_{media}) and a timing authority register. Every mutation operation (e.g., transcript word deletion, gap tightening, B-roll take insertion) is intercepted by a trusted gate that enforces verified timing authority. When underlying media changes or is replaced, the media revision increments, immediately revoking timing edit authority and setting the transcript timing state to unaligned (`unalignedTiming("media_changed", \rho_{\text{media}})`). Any subsequent cut attempt is immediately blocked with a typed refusal (`UNALIGNED_TRANSCRIPT_CUT`), preventing out-of-sync audio/video destruction before it can occur.

[0010] (2) Non-Linear Kuramoto Phase Coupling for Audiovisual Synchronization:

Timeline keyframe snapping, caption transition clocks, and cut intervals are dynamically synchronized to audio beat transients and cadence using a community-aware Kuramoto coupled oscillator network:

$$\frac{d\theta_i}{dt} = \omega_i + KR \sin(\psi - \theta_i)$$

where $Re^{i\psi} = \frac{1}{N} \sum_{j=1}^N e^{i\theta_j}$ represents the complex order parameter, K

represents the coupling strength, and ω_i is the natural frequency of the i -th timeline track element. Intra-community coupling ($K_{\text{in}} = K \cdot 1.8$) and inter-community coupling ($K_{\text{out}} = K \cdot 0.4$) ensure that localized cluster tracks (e.g., speech syllables and subtitle words) lock into phase coherence first ($R_c \rightarrow 1$), followed by global timeline synchronization ($R \rightarrow 1$) as coupling parameter K crosses the critical synchronization threshold K_c .

[0011] **(3) Non-Destructive Mutation and Protected Entity Filtering:**

Transcript deletions do not alter absolute timeline geometry or discard timing intervals; instead, words are toggled to `cut: true` while preserving their exact microsecond start/end timestamps ($t_{\text{start}}, t_{\text{end}}$), allowing bit-identical undo/redo and non-destructive re-assembly. Automated pause tightening accepts an immutable protection set ($\mathcal{P}$), ensuring protected proper names, numbers, and user-specified tokens are mathematically immune to removal.

[0012] **(4) Deterministic Receipted State Tracking:**

Every mutation produces a timestamped, tamper-evident cryptographic receipt (`stamp(project, move, payload)`) appended to a verifiable project history log, enabling third-party auditability and bit-identical timeline replay.

4. Detailed Description of Preferred Embodiments

4.1 Authority-Gated Edit Architecture

[0013] Figure 1 illustrates the state machine governing transcript mutation. A project object P comprises a duration D , a media revision $\rho_m \in \mathbb{N}$, a word list $W = [w_1, w_2, \dots, w_n]$, a gap list G , and a cryptographic receipt log $\mathcal{R}$.

[0014] Each word w_i comprises a stable identifier id_i , text token t_i , start timestamp s_i , end timestamp e_i , a boolean cut flag $c_i \in \{0, 1\}$, a filler classification flag f_i , and an explicit timing provenance record τ_i :

$$w_i = (\text{id}_i, t_i, s_i, e_i, c_i, f_i, \tau_i)$$

[0015] In an embodiment, verified timing authority is maintained in a private, unforgeable memory reference set $\mathcal{A}_{\text{timing}}$ (such as an ECMAScript `WeakSet` or hardware-isolated capability table). When a mutation command (such as `applyTranscriptDelete(P, \text{wordId})` or `confirmTighten(P, \text{proposal})`) is dispatched, the gate evaluates:

$$\text{Gate}(P) = \begin{cases} \text{ALLOW}, & \text{if } P \in \mathcal{A}_{\text{timing}} \\ \text{REFUSE}(\text{UNALIGNED_TRANSCRIPT_CUT}), & \text{otherwise} \end{cases}$$

[0016] When the media source is altered or replaced, the system invokes `invalidateTimingForMedia(P)`, which atomically executes: 1. $\mathcal{A}_{\text{timing}} \leftarrow \mathcal{A}_{\text{timing}} \setminus \{P\}$ (authority revoked); 2. $\rho_m \leftarrow \rho_m + 1$ (media revision incremented); 3. $P.\text{transcriptTiming} \leftarrow (\text{status} = \text{"unaligned"}, \text{provenance} = \text{"media_changed"}, \text{revision} = \rho_m)$.

[0017] Under this invariant, it is impossible for any automated agent, planner, or client interface to execute timeline cuts on unverified timing data.

4.2 Kuramoto Phase-Coupled Audiovisual Synchronization

[0018] In an embodiment, visual timeline snapping and caption display clocks are governed by a network of N coupled non-linear phase oscillators. Each timeline track or UI layer element possesses a phase $\theta_i \in [0, 2\pi)$ and an intrinsic natural frequency ω_i .

[0019] The global order parameter $R(t) \in [0, 1]$ and collective phase $\psi(t) \in [0, 2\pi)$ are evaluated in $O(N)$ linear time via mean-field vector summation:

$$Re^{i\psi} = \frac{1}{N} \sum_{j=1}^N e^{i\theta_j} = \left(\frac{1}{N} \sum_{j=1}^N \cos \theta_j \right) + i \left(\frac{1}{N} \sum_{j=1}^N \sin \theta_j \right)$$

[0020] To provide hierarchical synchronization between localized word tracks (syllables, words) and global media tracks (scene cuts, background audio tempo), the network partitions oscillators into communities $C_1, C_2, \dots, C_M$. The equation of motion for oscillator $i \in C_k$ is:

$$\frac{d\theta_i}{dt} = \omega_i + K_{\text{in}} R_{C_k} \sin(\psi_{C_k} - \theta_i) + K_{\text{out}} R_{\text{global}} \sin(\psi_{\text{global}} - \theta_i)$$

where $R_{C_k} e^{i\psi_{C_k}} = \frac{1}{|C_k|} \sum_{j \in C_k} e^{i\theta_j}$, $K_{\text{in}} = 1.8K$, and $K_{\text{out}} = 0.4K$.

[0021] When audio energy crosses a transient threshold (e.g., a spoken plosive or musical downbeat), the coupling coefficient K is impulsively driven above critical threshold K_c . This causes the localized community order parameter $R_{C_k} \rightarrow 1$, snapping subtitle display and visual HUD cards into phase alignment with zero perceptual latency and zero jitter.

4.3 Protected Entity Gap Tightening

[0022] Automated timeline pause tightening evaluates all inter-word silence intervals $g_j = (s_{j+1} - e_j)$. For any silence duration exceeding threshold Δ_{gap} (default 0.35 seconds), the engine constructs a candidate tightening proposal.

[0023] Candidate word deletion is restricted to tokens identified as filler ($f_i = 1$, such as “um”, “uh”, “like”) that are strictly disjoint from an immutable caller-specified protection set $\mathcal{P}$:

$$W_{\text{cut}} = \{w_i \in W \mid f_i = 1 \wedge \text{id}_i \notin \mathcal{P} \wedge c_i = 0\}$$

[0024] This ensures that critical proper nouns, technical acronyms, or numbers that match filler phonetic substrings are preserved inviolate during automated processing.

CLAIMS: SYSTEMS AND METHODS FOR PHASE- COUPLED DETERMINISTIC AUDIOVISUAL TIMELINE COMPOSITION AND AUTHORITY- GATED TRANSCRIPT MUTATION

- 1.** A computer-implemented media editing system comprising: storage holding a media project data structure, the media project data structure comprising a reference to a media stream, a stored media revision counter, a transcript comprising a sequence of timed word records, each timed word record having a start timestamp and an end timestamp relative to the media stream, and a timing authority state; and processing circuitry configured to: receive an edit command requesting a mutation of the sequence of timed word records; evaluate the timing authority state of the media project data structure against a declared verified timing authority criteria; in response to the timing authority state satisfying the criteria, apply the mutation to the sequence of timed word records and append a tamper-evident receipt record to a receipt log associated with the media project data structure; and in response to the timing authority state failing the criteria, reject the edit command with a typed error code distinguishing an unaligned transcript cut, without modifying the sequence of timed word records.
- 2.** The system of claim 1, wherein the processing circuitry is further configured to: receive an update altering the media stream; and in response to the update, atomically increment the stored media revision counter, revoke the verified timing authority state, and record an unaligned timing status associated with the incremented media revision counter.
- 3.** The system of claim 1, wherein the timing authority state is maintained in an unforgeable in-memory reference set isolated from untrusted client execution environments.
- 4.** The system of claim 1, wherein the mutation marks a target timed word record as deleted by asserting a cut flag while preserving the start timestamp and the end timestamp of the target timed word record in the sequence without altering timing coordinates of adjacent word records.
- 5.** The system of claim 1, wherein the edit command comprises a pause tightening command specifying a gap duration threshold and a protected token set, and wherein applying the mutation comprises: identifying inter-word silence intervals in the sequence of timed word records exceeding the gap duration threshold; identifying filler words within the sequence of timed word records that

are strictly excluded from the protected token set; and marking the identified filler words as cut while preserving all word records identified in the protected token set.

6. A computer-implemented method for phase-coupled audiovisual timeline synchronization, the method comprising: maintaining, in processing circuitry, a plurality of N timeline track elements, each timeline track element being associated with a phase $\theta_i \in [0, 2\pi)$ and an intrinsic frequency ω_i ; receiving an audio stream and detecting an audio transient event in the audio stream; evaluating a complex order parameter $Re^{i\psi} = \frac{1}{N} \sum_{j=1}^N e^{i\theta_j}$ representing collective phase coherence of the plurality of timeline track elements; updating the phase of each timeline track element according to a non-linear phase-coupled equation of motion:

$$\frac{d\theta_i}{dt} = \omega_i + KR \sin(\psi - \theta_i)$$

wherein a coupling parameter K is increased in response to detecting the audio transient event; and snapping a visual timeline event or caption display to the audio stream when the complex order parameter R exceeds a declared synchronization threshold.

7. The method of claim 6, wherein evaluating the complex order parameter is performed in $O(N)$ linear computational time via separate summation of sinusoidal and cosinusoidal components.

8. The method of claim 6, wherein the plurality of timeline track elements are partitioned into a plurality of community clusters, and wherein updating the phase comprises applying an intra-community coupling constant K_{in} and an inter-community coupling constant K_{out} , wherein $K_{\text{in}} > K_{\text{out}}$, such that localized timeline elements synchronize into phase coherence prior to global timeline synchronization.

9. The method of claim 8, wherein $K_{\text{in}} \approx 1.8K$ and $K_{\text{out}} \approx 0.4K$.

10. A non-transitory computer-readable storage medium storing instructions that, when executed by processing circuitry, cause the processing circuitry to perform operations comprising: maintaining a media project comprising a duration, a media revision, an array of word objects each having microsecond start and end timestamps, and a timing authority register; intercepting a transcript deletion request; verifying that the media project is registered in an unforgeable timing authority registry; upon successful verification, marking a target word object as cut without shifting timestamps of subsequent word objects,

and generating a cryptographically hash-chained receipt record; and upon failed verification, refusing the deletion request and returning a typed unaligned cut exception without altering the array of word objects.

APPENDIX: REFERENCE IMPLEMENTATION EXCERPTS OF THE SLASH TIMELINE ENGINE AND KURAMOTO OSCILLATOR COUPLING

The following code excerpts are extracted verbatim from
`engines/slash/engine.mjs` and `deploy/slash/braingraph/kuramoto.mjs` and
demonstrate verified working reduction to practice. In a test run on an Apple M5
Max with Node.js v22, all 21 unit tests in `engines/slash/engine.test.mjs` and
`engines/slash/mcp.test.mjs` passed cleanly.

1. Authority-Gated Edit State Machine (engines/slash/engine.mjs)

```
const VERIFIED_TIMING_AUTHORITY = new WeakSet();

export const UNALIGNED_TRANSCRIPT_CUT = "UNALIGNED_TRANSCRIPT_CUT";
export const UNALIGNED_TRANSCRIPT_MESSAGE =
  "Timing isn't matched to this video – cutting is unavailable in this
  build.";

function unalignedTiming(reason = "no_verified_alignment",
  mediaRevision = 0) {
  return {
    status: "unaligned",
    provenance: reason,
    mediaRevision,
    association: null,
  };
}

export function timingEditState(project) {
  const allowed = Boolean(project &&
    VERIFIED_TIMING_AUTHORITY.has(project));
  return allowed
    ? { allowed: true, code: null, message: null }
    : { allowed: false, code: UNALIGNED_TRANSCRIPT_CUT, message:
      UNALIGNED_TRANSCRIPT_MESSAGE };
}

export function requireTimingEditAuthority(project) {
  const state = timingEditState(project);
  if (state.allowed) return state;
  const error = new Error(state.message);
  error.code = state.code;
  throw error;
}

export function invalidateTimingForMedia(project) {
  VERIFIED_TIMING_AUTHORITY.delete(project);
  project.mediaRevision = Number.isSafeInteger(project.mediaRevision)
    ? project.mediaRevision + 1 : 1;
  project.transcriptTiming = unalignedTiming("media_changed",
    project.mediaRevision);
  return project.transcriptTiming;
}

export function applyTranscriptDelete(project, wordId) {
  const word = project.words.find((w) => w.id === wordId);
  if (!word || word.cut) return null;
  requireTimingEditAuthority(project);
  word.cut = true;
  return stamp(project, "transcript", { wordId, t: word.t, start:
    word.start, end: word.end });
}

export function proposeTighten(project, gapOrOpts = 0.35) {
```

```
const gap = typeof gapOrOpts === "number" ? gapOrOpts :
  (gapOrOpts.gap ?? 0.35);
const protect = new Set((typeof gapOrOpts === "object" &&
  gapOrOpts.protect) || []);
const kept = project.words.filter((w) => !w.cut);
const wordIds = kept.filter((w) => w.filler &&
  !protect.has(w.id)).map((w) => w.id);
const gaps = [];
for (let i = 1; i < kept.length; i++) {
  const start = kept[i - 1].end;
  const end = kept[i].start;
  if (end - start >= gap) gaps.push({ start, end });
}
return { wordIds, gaps };
}
```

2. Kuramoto Coupled Oscillator Synchronization

(deploy/slash/braingraph/kuramoto.mjs)

```
export const TWO_PI = Math.PI * 2;
export const COMMUNITY_INTRA = 1.8;
export const COMMUNITY_INTER = 0.4;

export function wrapPhase(theta) {
  let t = theta % TWO_PI;
  if (t < 0) t += TWO_PI;
  return t;
}

export function orderParameter(phases) {
  const N = phases.length;
  if (!N) return { R: 0, psi: 0, meanSin: 0, meanCos: 0 };
  let sumSin = 0;
  let sumCos = 0;
  for (let i = 0; i < N; i++) {
    sumSin += Math.sin(phases[i]);
    sumCos += Math.cos(phases[i]);
  }
  const meanSin = sumSin / N;
  const meanCos = sumCos / N;
  const R = Math.hypot(meanSin, meanCos);
  return { R, psi: Math.atan2(meanSin, meanCos), meanSin, meanCos };
}

export function stepPhasesMeanField(phases, omega, K, dt, communities
  = null) {
  const N = phases.length;
  if (!N) return [];
  const next = new Float64Array(N);

  if (!communities || communities.length !== N) {
    const { R, psi } = orderParameter(phases);
    const KR = K * R;
    for (let i = 0; i < N; i++) {
      const dtheta = omega[i] + KR * Math.sin(psi - phases[i]);
      next[i] = wrapPhase(phases[i] + dtheta * dt);
    }
    return next;
  }

  const { byC } = communityOrders(phases, communities);
  const global = orderParameter(phases);
  const Kin = K * COMMUNITY_INTRA;
  const Kout = K * COMMUNITY_INTER;

  for (let i = 0; i < N; i++) {
    const c = communities[i];
    const local = byC.get(c) || global;
    const pullLocal = Kin * local.R * Math.sin(local.psi - phases[i]);
    const pullGlobal = Kout * global.R * Math.sin(global.psi -
      phases[i]);
    const dtheta = omega[i] + pullLocal + pullGlobal;
  }
}
```

```
        next[i] = wrapPhase(phases[i] + dtheta * dt);  
    }  
    return next;  
}
```