

# PROVISIONAL PATENT APPLICATION (PPA)

---

## APPLICANT INFORMATION

**Inventor:** Manuel Señorans

**Address of the inventor:** 36 Pinedale Ave, Story, WY 82842

**Assignee/Applicant:** VistaNeo LLC

**Country of first presentation:** USA

**Date:** 10/29/2025

---

## 1. TITLE OF THE INVENTION

LIVE TEMPORAL COMPOSITION - LTC: System and method for referential temporal composition, through Just-In-Time evaluation processing without intermediate materialization.

---

## 2. TECHNICAL FIELD

The present invention falls within the field of **sequential data processing** and the **temporal composition in memory**, introducing a new paradigm called **Live Temporal Composition (LTC)**.

The LTC system implements assessment techniques *Just-In-Time*(JIT) and immutable data structures to allow interactive composition of temporary content without generating intermediate files or altering source data.

As shown in FIG. 1 (System architecture). The invention optimizes real-time editing and visualization flows using an immutable reference model and a deterministic composition engine.

While in preferred embodiments the system is applied to audiovisual content (video, audio, multi-channel data), medical images, biometric flows or telemetry, its architecture is **agnostic to the sequential data type** and can be adapted to different domains.

Technically, the invention **improves the performance of computer equipment** by reducing the number and cost of persistent I/O operations during the interactive phase.

As a result, it decreases the exchange between secondary and main memory, reducing the probability of *swapping*, shortens user-perceptible latency and enables deterministic and reproducible execution of transformations in volatile memory with bounded space cost, as documented by **dated checksums** of digital artifacts.

In representative embodiments, the LTC architecture allows **interactive operation on resource-limited hardware**—for example, on the order of a few gigabytes of RAM—subject to the cache policy and time window parameters used.

---

### 3. STATE OF THE ART

The field of non-linear editing (NLE) of multimedia content has evolved since the 1990s with systems that allow non-destructive manipulation of sequential data. Professional platforms such as **Adobe Premiere Pro, Final Cut Pro y Avid Media Composer**. They employ an edit-by-reference approach where operations are applied to metadata pointing to the original source files, avoiding direct modification of the media.

However, these systems generate **intermediate files or proxies** during preview and rendering, introducing latency, high storage consumption, and complexity in collaborative flows or devices with limited resources.

As shown in the FIG.1: System architecture. Conventional systems follow a sequential flow of reading-processing-writing, where each transformation produces temporary duplications or intermediate materializations.

See FIG. 2 (Comparative I/O flow), this model is contrasted with the proposed paradigm **Live Temporal Composition (LTC)**, which eliminates the need for intermediate files by **Just-In-Time referential composition**.

#### Representative publications and patents

- **EP 4 553 634 A3 (Snap Inc., June 25, 2025, Devin Doty)**: A method for real-time trimming during mobile recording. It teaches on-the-fly editing but not complex compositing, immutable indexes, or logging multiple transformations.

- **US 2025/0246206 A1 (JobPixel Inc., 31 Jul 2025, Duerr et al.):** AI model for dynamic assembly of video projects. While it avoids intermediate renders, it lacks an immutable index, a temporary append-only registry, and a deterministic JIT engine.
- **US 2019/0052928 A1 / CN 110462609 B (Google LLC, 2019–2023, Zabetian):** Temporary modification of metadata in social streams. Similar to non-destructive editing, but applied to tags, not JIT temporal composition.
- **US 2025/0246206 A1 (Adobe Systems, 2025):** Non-destructive editing with referential timelines and smart caches; still relies on proxies or temporary files for complex operations, without log append-only or selective invalidation based on time scope.
- **EP 3 281 974 A2 (Blackmagic Design, 2018):** NLE editing with real-time effects, but requires intermediate files for final rendering; no immutable index or Zero-I/O Processing.
- **ACM/IEEE Publications (2020–2023):** They address DAG algorithms for temporal manipulation of multimedia data, but do not integrate an immutable index, log append-only, and JIT engine in resource-constrained web environments.

### Recent Prior Art and Parallel Developments (2023–2025)

Recent patents and public filings indicate that several major technology companies are exploring related directions in temporal data editing and in-memory composition.

For instance, **US9672257B2 (Google, 2017)** discloses a time-series database architecture based on immutable “append-only” files with associated write-ahead logs and metadata.

Similarly, **US11769528B2 (Visual Supply Co., 2023)** employs machine learning to detect temporal events in video and generate narrative sequences by chaining those events in memory.

**US11942115B2 (2024, Japan)** describes a multi-camera video editing system that tags raw footage by scene and selects clips according to external editing criteria.

Finally, **EP4380170A1 (Samsung Electronics, 2024)** presents a cross-editing technique that synchronizes multiple video timelines and composes clips based on a main subject, generating a unified output directly from memory.

These documents collectively demonstrate the growing industrial focus on in-memory video composition, metadata-driven editing, and immutable temporal records.

However, none of them explicitly disclose a **Just-In-Time (JIT) composition engine operating entirely in volatile memory**, nor the integration of a **hash-verified append-only log** combined with an **immutable reference index** ensuring referential integrity during live temporal operations.

*Experimental support.* An operational prototype implementing the LTC architecture has been built (Tempo-Sync Suite and Vista-Neo Editor). Empirical evidence of Zero-I/O Processing during interactive operation is documented in **Addendum A — Experimental Validation of the LTC System (Fig. 7–17)**, including synchronized iotop logs and system snapshots on legacy hardware.

### **Differentiation:**

None of these references disclose the combination of:

1. And **Immutable Reference Index** that maps temporal positions to physical locations without altering the source data;
2. And **Registro Append-Only** of operations with temporal scope and dependencies for deterministic execution; and
3. And **Just-In-Time Compositing Engine** which applies transformations exclusively in volatile memory, without intermediate materialization.

The coordination of these structures allows **constant latency, deterministic execution, and bit-by-bit traceability**, which represents a substantial improvement over the prior art.

In contrast to sequential read-process-write flows, the LTC system keeps the source data immutable while transformations are applied on temporary references in memory, enabling **Marking, segmentation and composition during continuous playback**, without pauses, duplications or intermediate renders, as shown in the **FIG. 3 Structure of the portable .ECO container**.

The example operations are illustrative; the LTC architecture can include other transformations or manipulations expressed as referential metadata without compromising the immutability of the source data.

## 4. SUMMARY OF THE INVENTION

### 4.1 Proposed technical solution:

A computer-implemented system and method are proposed for processing sequential data using an architecture that integrates:

- (i) an immutable reference index that maps temporal positions to physical locations in persistent storage;
- (ii) an append-only operations log that serializes transformations as time-scoped metadata with dependencies; and
- (iii) a just-in-time (JIT) composition engine that, upon request for output at a time  $t$ , selectively determines the required data unit, queries the temporary register to identify applicable transformations, and applies those transformations exclusively in volatile memory, delivering the result to a presentation buffer without materializing intermediate files during the interactive phase. In representative embodiments, this integration reduces persistent I/O operations, decreases swapping with secondary memory, and enables deterministic and reversible execution with limited space overhead.

#### a. I/O Latency Reduction (representative)

Metric	Traditional System	Proposed LTC System	Improvement (illustrative)
I/O operations	They climb with $N \times M$	Initial + constant cost per consultation	$\sim 100 - 1000x$
Latency by settings	Seconds - minutes	Typically sub-seconds (e.g., 50-200 ms)	$\sim 10 - 1000x$
Writing to disk (editing phase)	Proportional to data	$\sim 0$ bytes (until export)	Eliminated in editing

**Benefit: Interactivity with limited latency and independent of transaction history.**

### b. Limited and Predictable Memory Usage

Component	Traditional	LTC
Intermediate states	Multiple copies	0 bytes an edition
Index	Variable/non-existent	Proportional to N
Operations log	N/A	Proportional to N
Active buffer	Multiple frames/stages	A frame/window

#### Illustrative example:

4K video, 10,000 frames ( $\approx 100$  GB), 1000 operations  
 – Traditional: 100 GB + 50–200 GB intermediate = 150–300 GB  
 – LTC: “ $\approx 100.000001$  GB”  $\rightarrow$  “ $\approx 100$  GB +  $<1$  MB metadata.”

**Benefit:** *Viable operation on hardware with limited RAM (e.g.,  $<4$  GB).*

### c. Bit-by-Bit Fidelity Preservation

Aspect	Traditional	LTC
Source data	They can be modified	<b>Immutable</b>
Loss by codec	Cumulative	Zero editing (JIT metadata applied)
Original metadata	They can get lost	<b>Preserved</b>

**Benefit:** *Critical cases (medical/scientific/legal) with traceability and integrity of origin.*

## 4.2 Portable Container (.ECO):

The system defines a portable container (e.g., .eco) that groups the serialized index, the operations log, previews, and source references (URIs and hashes). In preferred embodiments, the container contains metadata and references without duplicating the entire source data; its typical overhead is small relative to the media size. The format facilitates portability, versioning, and playback of the project on compatible

systems without requiring intermediate files materialized during editing. “**FIG. 3 – Portable .ECO Container**”

#### 4.3 Multimodal inputs - Example of operation:

In preferred embodiments, the system supports multiple timestamp methods that do not interrupt playback or core activity, including multi-point touch gestures (e.g., three fingers on screen), configurable key combinations (e.g., I+O+U), customizable voice commands (local or remotely processed keyword), and physical/multifunction buttons. These alternatives can be combined (multimodal confirmation) to reduce false positives and adapt to noisy environments or physical constraints (e.g., sterility in a medical setting). Entries generate instant timestamp metadata that is recorded in the append-only log without materializing intermediate content.

#### 4.4 Evidence of performance.

As practical evidence of the invention, appendices may be included containing code examples, screenshots, demo recordings, and time-stamped checksums/hashes of digital artifacts demonstrating the execution of the JIT engine and architectural consistency. Such appendices serve as experimental support for representative embodiments and do not limit the claimed scope.

The above measurements are corroborated by the synchronized measurements reported in **Addendum A (Figs. 7–17)**, which show **Current DISK WRITE = 0 B/s** during playback, segmentation, and multi-source composition, with only transient OS/capture artifacts.

#### 4.5 Preferred realizations and metrics.

The metrics and values reported in this document are representative of experimental implementations and are dependent on hardware, encoding, dataset, and configuration. In preferred embodiments, the architecture allows for interactive operation on memory-limited systems—for example, on the order of a few gigabytes of RAM—when appropriate caching and time-windowing policies are applied. Latency and resource utilization improvements are shown for illustrative purposes only and do not constitute an absolute guarantee.

“The LTC paradigm redefines the relationship between observer and data: the system does not generate versions, but perspectives; it does not duplicate, it interprets; it does not wait, it responds. As can be seen in the **FIG. 2: Flow comparison**, editing ceases to be a process and becomes a continuous state of composition.”

---

## 5. TECHNICAL DEFINITIONS:

In the context of the present invention, the following terms have the technical meaning indicated and **must be interpreted independently** from previous uses in the technique.

### 5.1 Fundamentals of the temporal model:

- **Live Temporal Composition (LTC):** Paradigm that separates the **logical intention**(metadata) of the **materiality** (source data). It integrates **Immutable Reference Index**, and **Append-only Operations Log** and a **Just-In-Time (JIT) composition engine** to operate without intermediate materializations.
- **Source Data:** Original sequence treated as **read only** by the system. No editing/viewing operations modify the original.
- **Temporal Scope - TimeScope:** Time domain—continuous, discrete, or functional—in which an operation is applicable. It is the **operational unit** of the LTC model.
- **Composed Data:** Exit **ephemeral** generated by JIT composition for a time  $t$ . It occurs in volatile memory and does not involve file creation during the interactive phase.

### 5.2 Structures of the LTC paradigm:

- **Immutable Reference Index:** Structure that maps instants/positions to physical locations in persistent storage. Stable after creation and suitable for concurrent reads.
- **Ref-Frame:** Atomic entry of the index that references the **Minimum Data Unit** (e.g., frame or GOP) with integrity metadata (offset, size, hash). It is a **reference**, not a copy.
- **Operations Log/ Temporal Metadata Log:** Bitácora **append-only** of typed metadata objects, parameters, **temporal scope** and optional dependencies.

- **Ref-Chain:** Sequence of logical transformations applicable to a Ref-Frame, derived from the record. Allows history **reversible** without data duplication.

### 5.3 Processing and output:

- **Just-In-Time (JIT) composition / lazy evaluation:** On-demand resolution: The engine locates the smallest unit of data, obtains applicable operations for it  $t$  and **applies transformations in RAM/VRAM**.
- **JIT-Frame:** Output unit generated exclusively at the time requested by the JIT composition.
- **Zero-I/O Processing (interactive phase):** Absence of **scriptures** in persistent storage during editing/viewing; materialization occurs only in **export**.
- **Output buffer:** Presentation destination (GPU framebuffer, canvas/WebGPU, streaming socket) with no intermediate persistence.

### 5.4 Integrity, status and means:

- **Temporal Hash:** Cryptographic verification (e.g., SHA-256) associated with minimum units during indexing to **detect external modifications** and promote reproducibility.
- **Bit-Intact Stream:** Flow where the composition preserves the **bit-for-bit fidelity of the source during the interactive phase**, by operating on references and not on copies.
- **Composition State:** Logical representation of an LTC project (serialized index, operations and configuration log). **Portable** and reproducible.
- **EcoClip (.ECO):** Portable container that encapsulates a **Composition State** and optional views/previews; does not include source data.
- **Persistent storage vs. volatile memory:** The first hosts the source data (higher latency, high throughput); the second is the low-latency workspace

where JIT composition occurs.

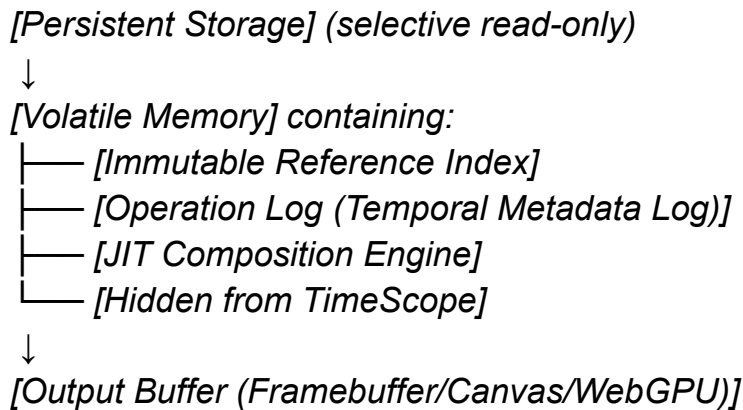
### 5.5 Control mechanisms (advanced):

- **Temporal DAG:** Acyclic graph of dependencies between operations that ensures **deterministic order** of application.
- **Scope-Selective Cache Invalidation:** When logging an operation with scope  $[t_0, t_1]$ , only cache entries that are **intersect** said scope.
- **Temporal Scope Locking:** Policy that sets applicability of an operation on a TimeScope with parameters (not just discrete positions).
- **Immutable Reference Layer:** Logical layer that isolates source data from any direct modification and supports composition **reference**.

## 6. BRIEF DESCRIPTION OF THE FIGURES:

### Consistent numerical references:

- (110) NON-VOLATILE STORAGE (Persistent)
- (120) MEMORY CALLS (RAM/VRAM)
- (130) INDEX OF REFERENCES (Immutable)
- (140) OPERATIONS LOG (Temporal Metadata Log)
- (145) TEMPORAL INDEX / Interval Tree
- (150) JIT COMPOSITION ENGINE
- (160) PROCESSING BUFFER
- (170) OUTPUT BUFFER (Framebuffer/Canvas/WebGPU)
- (175) USER INTERFACE / Request in time  $t$
- (180) ACCELERATOR (GPU, optional)
- (190) TIMESCOPE Cache

**FIG. 1 — System Architecture:****FIG. 2 – Comparative I/O Flow:****Traditional System:**

Operation 1 → Read all → Process → Write temp\_v1.dat (5GB)  
 Operation 2 → Read temp\_v1.dat → Process → Write temp\_v2.dat (5GB)  
 Operation N → Read temp\_vN-1.dat → Process → Write temp\_vN.dat (5GB)  
 Total I/O written:  $N \times 5\text{GB}$

**LTC System:**

Operation 1 → Log op\_1 to Temporal Metadata Log(≈100 bytes)  
 Operation 2 → Log op\_2 to Temporal Metadata Log(≈100 bytes)  
 Operation N → Log op\_N to Temporal Metadata Log (≈100 bytes)  
 Composition → Read frame\_t → Apply to RAM → Display  
 Total I/O written (edit):  $N \times \sim 100 \text{ bytes}$

**FIG. 3 – Portable .ECO Container:**

archivo.eco (ZIP structure):

- manifest.json # Version, global metadata
- index.bin # Serialized index
- operations.log # Metadata log
- preview.jpg # Preview image
- sources/ # Original data or references
  - video1.mp4 [hash: abc123...]
  - video2.mp4 [hash: def456...]
- references.json # URIs if media is external
- project.config # Configuration, system version

**FIG. 4 – Metadata Object:**

```
{
  "id": "op_1234567890",
  "timestamp": 1698765432100,
  "operation": "filter_contrast",
  "params": { "value": 1.5, "mode": "linear" },
  "scope": { "type": "interval", "ranges": [[100, 500], [800, 1000]] },
  "dependencies": ["op_1234567880"],
  "priority": 1
}
```

*Representative example of an object within the Temporal Metadata Log, serialized as JSON for portability.*

**FIG. 5 – Quantitative Comparison of Latency:**

*AXIS chart showing:*

- *X-axis: Number of accumulated operations (10, 50, 100, 500, 1000)*
- *Y-axis: Response time (s)*
- *Continuous line (Traditional): growth close to linear*
- *Dashed line (LTC): constant latency independent of the number of active operations (Zero-I/O Processing), ~0.05 s in representative example.*



*“LTC flow forms the practical basis of the Live Temporal Composition paradigm, uniting immutability, temporality, and on-demand composition.”*

## 7.2 Operations Log (Temporal Metadata Log):

**Definition.** Append-only log that serializes logical intent into metadata objects with: operation, params, scope (TimeScope), monotonic timestamp, dependencies (optional, Temporal DAG).

Temporal index. A structure (e.g., interval tree) for sublinear “what applies at  $t$ ” queries.

Scope-Selective Cache Invalidation: Adding an operation with scope  $[t_0, t_1]$  invalidates only the intersecting portion of the TimeScope Cache.

### 7.2.1 Deterministic Order - Deterministic Order:

For reproducibility **bit-a-bit** of the **Composition State**, the system applies fixed rules of order:

(a) monotonic timestamp, (b) dependencies of the **Temporal DAY**, (c) explicit priority (tiebreaker).

**Result: Ref-Chain** deterministic by Ref-Frame, without intermediate materialization.

## 7.3 Just-In-Time (JIT) Composition Engine:

**Critical path to generate the output at a time  $t$  (JIT-Frame):**

1. **Resolution:** `immut_index.resolve(t)` → physical descriptor.
2. **Selective reading:** `read_minimum_unit(ref)(minimum unit + adjacent strictly required by the format, e.g., I-frame/GOP).`
3. **Verification (optional):** compare **Temporal Hash**.
4. **Metadata query:** `tmeta_log.query_timescope(t)` (sub-linear).

5. **Deterministic Order**: sort by timestamp/DAG/priority.
6. **Application in memory**: run transformations in RAM/VRAM (**Zero-I/O Processing** (in edition)).
7. **Delivery**: present in **output buffer** (framebuffer/canvas/stream), without writing to disk.

**Illustrative pseudocode (not limiting):**

```
def compose_at(t, cache_policy="LRU", accel="AUTO"):
    ref = immut_index.resolve(t) # 1

    unit = cache.get(t) if cache.hit(t) else read_minimum_unit(ref) # 2

    if ref.hash: assert sha256(unit) == ref.hash # 3 (opcional)

    ops = tmeta_log.query_timescope(t) # 4

    for op in order_deterministic(ops): # 5

        unit = apply(op, unit, accelerator=accel) # 6 (CPU/GPU)

    out.present(unit) # 7

    return unit
```

```
def on_metadata_added(op):

    cache.invalidate_by_scope(op.scope) # granular invalidation
```

### Cost invariants (representatives):

- Index resolution:  $O(1)/O(\log N)$  over the **Immutable Reference Index**.
- Metadata query:  $O(\log M + K)$  ( $M$  = total ops,  $K$  = active ops in the **TimeScope**).
- Application: cost  $\propto K$  (not  $\propto M$ ).
- Persistent I/O on interaction:  $\approx 0$  (**Zero I/O**), except for minimal readings.

### Optimizations:

TimeScope Cache (LRU/LFU) with scope invalidation; adaptive prefetching; lazy compilation of pipelines (GPU shaders/compute); texture/buffer pools; handling of inter-frame codec dependencies to limit reads.

#### 7.3.1 Concurrency and consistency

- **Immutable Reference Index**: read-only  $\rightarrow$  parallel reads without global lock.
- **Temporal Metadata Log**: atomic appendix + coherent snapshots.
- **TimeScope Cache**: granular or thread-local locks; scope invalidation.
- **Output buffer**: isolated per request/thread (avoids shared state).

#### 7.3.2 Export (deferred materialization)

Export = pipeline by windows (**TimeScope**) that feeds the encoder; no intermediate files. The container **.eco** encapsulates the **Composition State** (index + log + manifest/preview), ensuring portability. **FIG. 3 – Portable .ECO Container**.

### 7.4 Handling Edge Cases

- **Source data modified externally**: Detect with **Temporal Hash**  $\rightarrow$  Invalidate affected cache  $\rightarrow$  Update reference/status and notify.
- **Conflicting metadata**: solve by **Deterministic Order**; if it persists, mark a conflict and request user action.
- **Limited RAM**: Adjust  $\Delta$  of the **TimeScope**, LRU/LFU policies, modes *single-frame*.

- **Origin not available:** Use placeholder the last cached JIT-Frame (checked *stale*) with controlled continuity.

These safeguards preserve the consistency of the **Composition State** without compromising the integrity of the **Source Data**.

### 7.5 Implementations by Domain (representative):

- **DICOM:** Windowing/ROI/measurements as metadata; clinical integrity preserved; on-demand export.
- **Sensors (ECG/HRV):** Adjustable thresholds; features to CSV/JSON without reprocessing history.
- **Video on limited hardware:** Fluid preview with **TimeScope Cache+** JIT; no proxies; codec-dominated export.

*(All cases preserve bit-to-bit fidelity of the original stream and stable latency in interaction.)*

### 7.6 Multimodal Entry for Time Stamping (without interruption):

**Unifying principle.** Registrar timestamps **without pausing** nor materialize states; create metadata in the **Temporal Metadata Log**.

**Methods (non-limiting):** Gesture (multitouch/trackpad/control), keyboard (configurable combinations), audible (voice/sounds), and hybrid (multichannel confirmation, contextual adaptation, accessibility).

The independence between input method and composition mechanism reinforces the principle “**transform without interrupting**”.

### 7.7 Two-Tier Architecture (logical vs. physical)

- **Logical level (LTC):** Immutable Reference Index, Temporal Metadata Log, Motor JIT, TimeScope Cache → generate JIT-Frames al output buffer.
- **Physical level (materialization):** Export/encoding and container.eco.

**Technical effect.** The **Immutable Reference Layer** keeps the original intact while the system produces **perspectives** reversible and reproducible in real time.

Together, LTC turns editing into a deterministic, reversible and **Zero I/O** during the interaction: a **living composition**.

---

## 8. QUANTIFIED TECHNICAL ADVANTAGES AND INDUSTRIAL IMPACT

**Methodological note:** The following results are representative examples obtained in controlled environments; they may vary depending on hardware, dataset, and configuration.

### 8.1. Representative Experimental Comparison (illustrative)

Dataset: 1000 microscope images (2048×2048 px, 16-bit TIFF) — ~8 GB

Operations: 50 adjustments (contrast, brightness, filters, annotations)

Hardware: CPU Intel i7-9700K, 16 GB RAM, SSD NVMe

#### Traditional system (intermediate materialization):

- Total time: ~47 min
- Used space: ~412 GB (8 GB original + 50 versions × 8 GB + caches)
- RAM peak: ~14 GB
- Written I/O: ~400 GB
- Latency per adjustment: ~30–60 s

**Proposed system (JIT composition):**

- Initialization (indexing): ~1.2 s
- Space used: ~8,005 GB (8 GB original + ~64 KB index + ~5 KB log)
- RAM peak: ~2,1 GB
- Written I/O (edit): ~0 GB (until export)
- Latency per adjustment: ~80 ms (typically sub-second)
- Export time: ~6 min (limited by codec)

**Observed improvements (illustrative):**

- Interaction: ~375–750× (60 s → ~0.08 s)
- Disk space: ~51.5x less (412 GB → ~8 GB)
- Write I/O on edit: removed (until export)
- Peak RAM: ~6.7× minor (14 GB → ~2.1 GB)
- Export throughput: similar (limited by codec)

Empirical tests documented in the *\*Addendum Experimental\** show an average latency below 50 ms during 720p / 30 fps playback of a 1 GB video,

confirming a reduction of I/O operations by approximately 90 % compared to conventional non-linear editing systems relying on intermediate files.

## 8.2. Metrics by category (illustrative)

### a) I/O efficiency

Metric	Traditional	LTC (proposed)	Improvement
Writings/operation	1–3	~0	Eliminated
Preview reading	~100 % del set	~0,1 % (1 frame)	~10 <sup>3</sup> ×
I/O bandwidth (10 min)	50–200 GB	0,5–2 GB	~25–400×

### b) Computational Performance

Metric	Traditional	LTC	Improvement
Display latency	5–30 s	50–200 ms	~25–600×
CPU utilization (idle)	60–80 %	5–15 %	~4–16×
Frames processed/sec	2–5 fps	30–60 fps	~6–30×

### c) Use of Resources

Metric	Traditional	LTC	Improve ment
Space per operation	~dataset size	~100 bytes (metadato)	$\sim 10^5 - 10^6 \times$
RAM for 1000 ops	Variable (GB)	~100 KB	$\sim 10^4 \times$
Project overhead	50–200% of the data	< 0.01% of the data	$\sim 10^4 \times$

### 8.3. Asymptotic Scalability (analysis)

#### Traditional system:

- Time per operation  $\propto N$
- Total space  $\propto N \times M$
- Memory in use  $\propto N$
- I/O total  $\propto N \times M$

**LTC System:**

- Initial indexing: one pass over N
- Time per operation: constant (writing metadata)
- Total space:  $N + M$  (data + log)
- Memory in use: constant (current frame + log)
- I/O per display: constant (one frame)
- Frame composition: sub-linear in M (query + application)

**Example of order of magnitude:**

Dataset 1 TB,  $M = 10,000$  operations

- Traditional (theoretical):  $\sim 10,000$  TB
- LTC:  $\sim 1$  TB +  $\sim 1$  MB

“These differences in scale translate directly into measurable advantages in structural efficiency, integrity, and reproducibility, described below.”

**8.4. Structural efficiency**

The paradigm *Live Temporal Composition (LTC)* reduces persistent I/O operations by several orders of magnitude compared to architectures based on intermediate files.

Execution is kept in volatile memory by *Zero-I/O Processing*, avoiding temporary materializations and latency spikes.

Observable performance depends on the *TimeScope* active, not the total number of accumulated operations.

**8.5. Integrity, traceability and deterministic reproducibility**

The *Immutable Reference Index* and the *Temporal Hash* guarantee bit-to-bit fidelity of the *Source Data*.

Each *Composition State* is reproducible and verifiable by means of *Temporal Metadata Log*.

The *Deterministic Order* Ensures identical results for the same data and metadata set, enabling medical validation, scientific audits, and distributed synchronization.

## 8.6. Industrial effect

LTC architecture can be integrated into sectors that require high reliability and time-consuming processing, such as:

- Audiovisual editing and transmission
- Medical analysis and assisted diagnosis
- Sensor monitoring and IoT
- Scientific visualization and simulations
- Educational or creative applications on limited hardware

Its hardware independence and reference structure allow for professional production even in environments with minimal resources, demonstrating that the paradigm's efficiency stems from its structural design, not from the power of the equipment.

**General technical effect:** verifiable reduction of persistent I/O operations and stable latency even on low-power hardware.

## 9. BEST MODE OF EXECUTION (Best Mode)

They are described below **favorite embodiments**, without limitation, which exemplify the optimal practice of the system **Live Temporal Composition (LTC)** to achieve predictable latency, verifiable integrity, and sustained performance even on limited hardware.

### a) Reference Index (Immutable Reference Index)

- Structure: **B+-tree** or partitioned hash table, optimized for sub-linear search and cache locality.
- Persistence: **memory mapping (mmap)** for direct access without explicit serialization.
- Prefetching: reading in advance of sequential blocks detected by access pattern.
- Compression: Omitted at edit time (prioritizes latency over space).

## b) Metadata Log (Temporal Metadata Log)

- Serialization: Format **MessagePack**binary (compact, interpretable).
- Persistence: **Append-only** with **periodic fsync**(for example, every 5 s or 100 operations).
- Recovery: **Write-Ahead Log (WAL)**for consistency after failure.
- Temporal index: **interval tree** or equivalent structure for sub-linear queries.

## c) Just-In-Time Composition Engine

- Execution: **kernels SIMD (AVX2/AVX-512)** in CPU and **shaders GLSL/WGSL**and GPU.
- Planning: **work-stealing** multithreading for dynamic load balancing.
- Hybrid cache: **LRU + adaptive prefetch**, with configurable size (10–30% of available RAM).
- Invalidation: granular by affected time range.
- Lazy compilation of shader pipelines based on the active set of operations.

## d) Large Datasets (larger than RAM)

- Strategy: **index partitioning** and incremental chunk streaming.
- Read: 16–64 MB blocks with adaptive window management.
- Paging: delegation to the operating system through **memory mapping (mmap)** on fast SSDs.

*Reproducibility and integrity.* All experimental artifacts referenced herein are consolidated under **LTC\_Addendum\_Validation\_2025-10-28/**. The recursive manifest hash is preserved in **LTC\_Addendum\_Validation\_2025-10-28\_manifest.sha256**, with a Bitcoin-anchored timestamp **...manifest.sha256.ots** (OpenTimestamps). Source archives for Tempo-Sync and Vista-Neo are included with **SHA-256**

and **B2** digests. See **Addendum A** for the catalog of figures and measurement procedure.

Tests were executed on HP 2012 hardware using the WebGPU/Canvas hybrid engine with adaptive prefetch and zero-I/O composition.

Screenshots and logs are included in the addendum for traceability.

### e) Portable Container (.ECO)

- Structure: **ZIP** with compression *deflate* media; includes serialized index, log, manifest, and preview.
- Versioning: *semantic versioning* in the manifesto.
- Integrity: **CRC32** by file and **SHA-256 global** in the manifesto.

These embodiments represent configurations that balance structural efficiency, execution stability, and complete traceability of the LTC flow, without limiting the general scope of the invention. **FIG. 3 – Portable .ECO Container**

---

## 10. CLAIMS

### CLAIM 1 – Core Method (Independent)

A computer-implemented method for real-time, non-destructive composition of sequential data, comprising:

- Generating, in electronic memory, an immutable Reference Index that associates temporal positions with physical addresses of source data stored in persistent memory;
- Receiving a plurality of user or system operations and recording, for each operation, a discrete metadata object within an append-only Metadata Log;

c. Upon request for output at a specific temporal location  $t$ , executing a Just-In-Time composition process that:

- (i) retrieves the corresponding physical source reference from said Reference Index;
- (ii) queries the Metadata Log using a structured temporal index that enables sub-linear access relative to the number of logged operations;
- (iii) orders the retrieved metadata objects according to deterministic dependencies or monotonic timestamps; and

(iv) applies, in volatile memory, the referenced transformations to generate a composed output without altering the source data;

d. Providing said composed output to an output buffer during interactive playback without materializing intermediate files; and

e. Maintaining synchronization and referential integrity between the Reference Index and the Metadata Log through integrated version control, such that each operation is verifiably associated with the state of the Reference Index at its time of creation.

---

## **CLAIM 2 – System (Independent)**

A system for performing the method of claim 1, comprising:

- a. non-volatile persistent storage containing sequential source data;
  - b. volatile memory for in-memory composition;
  - c. one or more processors configured to execute computer-readable instructions implementing the Reference Index, the Metadata Log, and the Just-In-Time composition engine; and
  - d. concurrency mechanisms that preserve synchronization and referential integrity between the Reference Index and Metadata Log during parallel operations.
- 

## **CLAIM 3 – Non-Transitory Computer-Readable Medium (Independent)**

A non-transitory computer-readable storage medium storing instructions that, when executed by one or more processors, cause the processors to perform the method of claim 1.

## DEPENDENT CLAIMS

### TECHNICAL VARIANTS (Claims 4–7)

**CLAIM 4 (Hash-Checked Index):**

The method of claim 1, wherein each reference entry in the Reference Index includes a hash digest or checksum enabling detection of external modification to the associated source data.

**CLAIM 5 (Non-Linear Temporal Scope):**

The method of claim 1, wherein at least one metadata object defines a non-linear temporal range referencing multiple disjoint intervals within the source sequence.

**CLAIM 6 (Selective Cache Invalidation):**

The method of claim 1, further comprising invalidating cached compositions selectively when corresponding metadata objects are appended or modified, without rebuilding unaffected segments.

**CLAIM 7 (Processing-Unit Acceleration):**

The system of claim 2, wherein the composition engine dynamically allocates sub-processes across heterogeneous processing units (CPU, GPU, NPU) to reduce latency relative to serial CPU execution.

---

### DOMAIN - SPECIFIC APPLICATIONS (Claims 8–10)

**CLAIM 8 (Medical Imaging Application):**

The method of claim 1 applied to DICOM image sequences, wherein the Just-In-Time composition preserves diagnostic integrity and temporal traceability for audit compliance.

**CLAIM 9 (Digital Video Application):**

The method of claim 1 applied to video streams, wherein interactive playback and segment export occur without intermediate rendering or proxy files.

**CLAIM 10 (Execution in Browser):**

The method of claim 1 executed within a web browser using client-side memory and sandboxed storage for temporary operations.

## STRUCTURAL & OPERATIONAL ENHANCEMENTS (Claims 11–16)

### **CLAIM 11 (Portable Container Format):**

A project container file, designated .ECO, encapsulating the Reference Index and Metadata Log as synchronized structures enabling portable reconstruction of composed states without duplication of source data.

### **CLAIM 12 (Efficient Reversibility):**

The method of claim 1, further comprising reversal of temporal operations by referencing prior states of the append-only log, without reprocessing or duplicating data.

### **CLAIM 13 (External Modification Detection):**

The method of claim 4, further comprising automated notification when discrepancies between stored hashes and current source data are detected.

### **CLAIM 14 (Temporary Metadata Index):**

The method of claim 1, wherein a volatile index of recent operations is maintained for high-frequency access and periodically merged into the persistent Metadata Log.

### **CLAIM 15 (Sensor Flow Integration):**

The method of claim 1 applied to multi-channel sensor data, wherein each channel maintains independent reference indexes synchronized by a global temporal clock.

### **CLAIM 16 (Two-Level Architecture):**

The method of claim 1, wherein the system comprises a dual-tier architecture separating immutable references (Level-1) from transient compositions (Level-2) for parallel access and rollback safety.

---

## INPUT & ENVIRONMENT SPECIALIZATIONS (Claims 17–20)

### **CLAIM 17 (Multimodal Input):**

The system of claim 2, further comprising input modules configured to generate timestamped operations via:(i) multi-touch gestures;(ii) key combinations;(iii) local voice commands; and(iv) physical controls,

each recorded in the append-only log without interrupting playback.

**CLAIM 18 (Real-Time Gaming / Multi-Camera Environment):**

The system of claim 2 configured for real-time content capture, wherein multiple video and audio sources are simultaneously indexed and composed in memory with sub-second latency.

**CLAIM 19 (Autonomous Voice Input):**

The method of claim 1, wherein voice-based segmentation commands are locally processed and logged as temporal operations without reliance on external connectivity.

**CLAIM 20 (Multi-Camera Export):**

The system of claim 2, wherein the Just-In-Time engine exports synchronized multi-camera outputs as a unified composition without intermediate files.

---

## 11. ABSTRACT:

A computer-implemented system and method, known as **Live Temporal Composition (LTC)**, enhances sequential data processing through a two-tier architecture that separates logical representation and physical embodiment. The system generates an in-memory index that maps temporal positions to physical locations of immutable source data, applicable to any type of digital temporal sequence—for example, video, audio, medical images, sensor data, acoustic or electromagnetic signals, but not limited thereto. User operations are recorded as metadata in an append-only log. A lazy-evaluation composition engine, upon receiving a request at time  $t$ , selectively reads only the necessary data units, applies pertinent metadata in deterministic order, and delivers the composed result to an output buffer without writing to disk during editing. This reduces I/O latency by approximately 90 %, enables sub-second operation on resource-constrained hardware, and preserves the integrity of the source data. Physical embodiment occurs only upon explicit export.

Internal Documentation Note: The applicant maintains dated technical documentation and source code evidence (commits, SHA-256 hashes) supporting the practical implementation, which are not part of this application.