Efficient Storage Model for Spatio-Temporal Polygons Using Deltas

Lautaro Conde

January 19, 2026

This document describes a conceptual architecture for storing geometries (polygons) that change over time, prioritizing **maximum storage efficiency**, **redundancy reduction**, and **referential integrity**. The main contribution is a vertex-level temporal storage model combining delta encoding, validity intervals, and pattern-based compression. This work presents a conceptual storage model and does not aim at providing a full implementation or experimental evaluation.

1 Problem Definition

Let P be a polygon in the Cartesian plane $\mathbb{R}^2$:

$$P_t = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

In a traditional (Snapshot) approach, any minimal change produces a full duplication of the polygon, resulting in $O(n)$ redundancy per change [3][4].

2 Solution: Enhanced Temporal Topological Decomposition

We separate the polygon into **topological structure** and **temporal states**, applying compression and redundancy elimination improvements.

2.1 Topological Structure (Immutable Polygon)

We represent P as an ordered sequence of vertex identifiers [4]:

$$P_{\text{struct}} = \langle id_{v1}, id_{v2}, \ldots, id_{vn} \rangle$$

- The topology is stored **only once** as long as the vertices do not change.
- As a **physical storage optimization**, consecutive identifiers can be encoded using *range encoding* or equivalent dictionary compression techniques.

Conceptual Storage Architecture

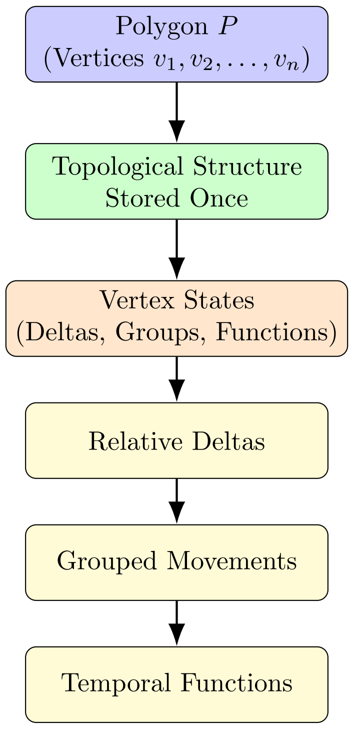

Figure 1: Conceptual storage model for spatio-temporal polygons: separation between topological structure and temporal vertex states, with deltas, grouping, and function-based trajectories.

2.2 Vertex State (Temporality and Deltas)

Each vertex maintains a history of positions [4]:

$$S(v) = \{(\Delta x_k, \Delta y_k, [t_{\text{start}}, t_{\text{end}}))\}_k$$

- **Relative deltas** are stored when movements are small. This technique constitutes an **adaptation of classical delta encoding** to spatial geometric coordinates, proposed in this work to reduce redundancy when displacements are local [2]. [5]:
  - Example: $[v1..v50] \to$ same in $[t1, t2)$

Error Control and Absolute State Recalculation

When using relative deltas, small rounding or approximation errors can accumulate over time. To maintain precision, we define an **error threshold** ϵ:

$$\epsilon = \text{maximum allowable deviation per vertex from true position}$$

Let $s_0 = (x_0, y_0)$ be the last absolute state at time t_0, and let $\Delta s_1, \ldots, \Delta s_m$ be consecutive relative deltas. The reconstructed position at time t_m is:

$$s_m^{\text{recon}} = s_0 + \sum_{k=1}^{m} \Delta s_k$$

If the deviation from the true position exceeds the threshold:

$$\|s_m^{\text{recon}} - s_m^{\text{true}}\| > \epsilon,$$

a new absolute state s_m^{abs} is stored:

$$s_m^{\text{abs}} = s_0 + \sum_{k=1}^{m} \Delta s_k$$

and the delta sequence is reset, starting from s_m^{abs}.

Notes:

- ϵ can be defined globally or per vertex depending on precision requirements.
- This ensures that cumulative rounding or approximation errors do not propagate indefinitely.
- Combined with periodic recalculation (e.g., every $m_{\max}$ deltas), this guarantees numerical stability while preserving storage efficiency.

Simultaneous Vertex Deltas

In this model, each vertex maintains its **own independent delta history**, even if multiple vertices move simultaneously:

$$S(v_i) = \{(\Delta x_k, \Delta y_k, [t_{\text{start}}, t_{\text{end}}))\}_k$$

- **Independent storage:** Each vertex's movement is recorded separately. Simultaneous changes do not require synchronization between vertices.
- **Reconstruction:** For a target time t_{target}, each vertex is reconstructed individually:

  $$\text{Pos}(v_i, t_{\text{target}}) = s_0(v_i) + \sum_k \Delta s_k(v_i)$$

  If a temporal function $f_i(t)$ exists for v_i, it is evaluated instead.
- **Topological integrity:** Even with independent deltas, all edges E_t of the polygon are checked for intersections:

  $$\forall e_i, e_j \in E_t, \quad e_i \cap e_j = \emptyset \quad \text{(excluding endpoints)}$$

  If any violation occurs, a *Topology Break* event forces a full structural update for the affected vertices.
- **Advantages:**
  - Fine-grained tracking: minimal changes are recorded without duplicating the entire polygon.
  - Dynamic grouping: vertices that move identically can be temporarily grouped but remain independent.
  - Error control: Each vertex can reset its delta sequence independently if the error threshold ϵ is exceeded.

Example: Independent Vertex Deltas

Consider three vertices v_1, v_2, v_3 with initial positions at t_0:

$$s_0(v_1) = (0,0), \quad s_0(v_2) = (1,0), \quad s_0(v_3) = (0.5,1)$$

They receive deltas over three time steps:

Time	Δv_1	Δv_2	Δv_3
t_1	$(0.1, 0.05)$	$(-0.05, 0.1)$	$(0, -0.1)$
t_2	$(0.08, 0.07)$	$(-0.03, 0.12)$	$(0.01, -0.12)$
t_3	$(0.05, 0.1)$	$(-0.02, 0.15)$	$(0.02, -0.15)$

Reconstructed positions:

- v_1: $(0,0) + (0.1, 0.05) + (0.08, 0.07) + (0.05, 0.1) = (0.23, 0.22)$
- v_2: $(1,0) + (-0.05, 0.1) + (-0.03, 0.12) + (-0.02, 0.15) = (0.9, 0.37)$
- v_3: $(0.5,1) + (0, -0.1) + (0.01, -0.12) + (0.02, -0.15) = (0.53, 0.63)$

Observation: Each vertex is reconstructed independently, so simultaneous deltas do not interfere with each other.

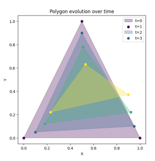

Figure 2: Polygon evolution over time

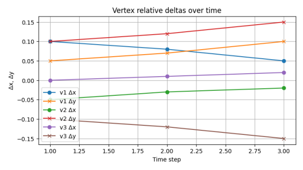

Figure 3: Delta evolution over time

Grouping of Identical Vertex Movements

When multiple vertices follow the same displacement pattern, either via relative deltas or temporal functions, we can store a single movement record and reference it from all corresponding vertices.

Identification of identical movements:

- Two vertices v_i and v_j are considered identical over an interval $[t_s, t_e)$ if their displacement vectors match exactly or within a small tolerance ϵ for all states:

$$\forall t \in [t_s, t_e), \quad \|\mathrm{Pos}(v_i, t) - \mathrm{Pos}(v_j, t)\| < \epsilon$$

Storage:

- Store a single movement record ΔS_{group} or temporal function $f_{\text{group}}(t)$.
- Maintain a reference list of vertex identifiers associated with this record:

$$V_{\text{group}} = \{v_{i_1}, v_{i_2}, \dots, v_{i_k}\}$$

Reconstruction:

- When reconstructing positions, iterate over V_{group} and apply the shared movement record or function to each vertex:

$$\mathrm{Pos}(v, t) = \mathrm{Pos}_{\text{base}}(v) + \Delta S_{\text{group}}(t), \quad \forall v \in V_{\text{group}}$$

- This guarantees that each vertex retains its identity and correct absolute position, while reducing storage duplication.

Notes:

- Tolerance ϵ allows grouping of vertices with near-identical movements.
- If vertices later diverge, the group is split and new movement records are created.

Temporal Patterns: Function-Based Vertex Trajectories

For vertices following predictable movements, we store a temporal function instead of individual states. This reduces redundancy while allowing precise reconstruction. We consider three main types of functions:

- **Linear:** constant velocity motion

$$x(t) = x_0 + v_x(t - t_0), \quad y(t) = y_0 + v_y(t - t_0)$$

where (x_0, y_0) is the reference point, t_0 the reference time, and (v_x, v_y) the velocity vector.

- **Polynomial:** smooth curved motion, modeled as a polynomial in time

$$x(t) = \sum_{i=0}^{n} a_i (t - t_0)^i, \quad y(t) = \sum_{i=0}^{n} b_i (t - t_0)^i$$

where n is the polynomial degree and a_i, b_i are coefficients fitted to the vertex trajectory.

- **Circular:** periodic or circular motion

$$x(t) = c_x + r\cos(\omega(t - t_0) + \phi),$$
$$y(t) = c_y + r\sin(\omega(t - t_0) + \phi)$$

where (c_x, c_y) is the circle center, r the radius, ω the angular velocity, and ϕ the phase offset.

Fallback to Delta Encoding: If the vertex deviates from the assumed function beyond a predefined error threshold, the temporal function is terminated and subsequent positions are stored either as relative deltas or a new absolute state:

3 Storage Logic (Enhanced Persistence)

A validity-time versioning scheme is followed, where each state change generates a new row with its validity interval. This is equivalent to the **Slowly Changing Dimensions (Type 2)** model in data warehousing [1] and aligns with classical temporal database models [6].

Transition Example

- **State t_0:** v_1 at $(10, 10)$ valid during $[2023 - 01 - 01, \infty)$
- **Change t_1:** v_1 moves to $(12, 15)$ on $2024 - 01 - 01$

Operations:

1. Close the previous interval $[2023 - 01 - 01, 2024 - 01 - 01)$
2. Insert new state $(\Delta x = 2, \Delta y = 5)$ valid during $[2024 - 01 - 01, \infty)$

Other vertices remain untouched.

Storage Optimization

- Delta encoding: reduces space for small changes.
- Grouping vertices with identical changes: minimizes repeated rows.
- Functions for regular patterns: avoids storing every instant.

---

4 Topological Integrity and Validation

A challenge in vertex-level temporal models is maintaining geometric validity. Since vertices move independently via deltas or functions, there is a risk of self-intersection (creating non-simple polygons).

To ensure integrity, the model incorporates a **Validation Layer** based on the Jordan Curve Theorem. Before committing a sequence of deltas Δs, the system pre-calculates the resulting edges E_t and verifies:

$$\forall e_i, e_j \in E_t, \quad e_i \cap e_j = \emptyset \quad \text{(exclusive of endpoints)} \tag{1}$$

If a movement violates this constraint, the system forces a "Topology Break" event, requiring a full structural update rather than a simple delta.

5 Reconstruction Algorithm (Query)

To reconstruct the position of a vertex v at a target time t_{target}, we follow:

5.1 Vertex Position

To reconstruct the position of a vertex v at a target time t_{target}, we follow:

1. Identify the last absolute state $s_0 = (x_0, y_0)$ valid before t_{target}.
2. Retrieve all consecutive delta states $\Delta s_k = (\Delta x_k, \Delta y_k)$ such that $t_{\text{start}_k} < t_{\text{target}}$.
3. Compute the cumulative sum:
$$(x, y) = (x_0, y_0) + \sum_k (\Delta x_k, \Delta y_k)$$
4. If a temporal function $f(t)$ exists for v, evaluate:
$$(x, y) = (x_0 + f_x(t_{\text{target}}),\ y_0 + f_y(t_{\text{target}}))$$

If a valid state exists for t_{target}, the vertex position is reconstructed by summing the relevant deltas or evaluating the function. Otherwise, the vertex is considered inactive (null) for that instant.

$$\text{Pos}(v, t_{\text{target}}) = \begin{cases} (x, y), & \text{if } \exists k : t_{\text{start}_k} \le t_{\text{target}} < t_{\text{end}_k} \\ \text{null}, & \text{else} \end{cases} \tag{2}$$

- If deltas are used: cumulative sums are applied over the last absolute state.
- If a temporal function exists: evaluate $f(t_{\text{target}})$.

5.2 Rendered Polygon

$$P_{\text{render}}(t_{\text{target}}) = \bigcup_{i=1}^{n} \text{Pos}(v_i, t_{\text{target}})$$

- Temporal indexing (B-Tree, GiST) ensures efficient search $O(\log H)$.
- Compression and grouping reduce the number of rows to scan.

---

6 Efficiency and Trade-off Analysis

The proposed model prioritizes storage density over instantaneous reconstruction speed. The following table summarizes the conceptual trade-offs between the traditional snapshot model and the proposed delta-function model. Complexity measures are given in terms of: V = number of vertices, S = number of snapshots, Δ = number of deltas stored, Φ = number of temporal functions applied, and D_{avg} = average number of deltas since the last absolute state.

Metric	**Snapshot Model**	**Delta-Function Model**
Storage Complexity	$O(V \cdot S)$	$O(V + \Delta + \Phi)$
Write Complexity	$O(V)$	$O(1)$ per vertex change
Read Complexity	$O(V)$	$O(V \cdot D_{avg})$
Redundancy	High ($> 99\%$)	Minimal ($< 1\%$)
Ideal Use Case	Real-time rendering	High-frequency historical archiving

Table 1: Comparative analysis: V (vertices), S (snapshots), Δ (deltas), Φ (functions), D_{avg} (average deltas since last checkpoint).

7 Conclusion

This work presents a storage model for spatio-temporal polygons that optimizes efficiency by separating topology and temporal vertex-level states. The proposed approach integrates **delta encoding**, **grouping of vertices with identical movements**, and **function-based temporal representation**, while ensuring **geometric integrity** and **reducing redundancy**.

The main contributions of the model are:

1. **Vertex-level temporal granularity**, allowing representation of minimal changes without duplicating the entire geometry.
2. **Storage efficiency**, achieved through the combination of relative deltas, grouping, and predictive functions.
3. **Topological integrity**, enforced via geometric validations that prevent self-intersections and preserve simple polygons.
4. **Compatibility with classical temporal models**, such as Slowly Changing Dimensions (Type 2), facilitating integration with temporal databases and GIS systems.

The model is particularly suitable for scenarios requiring **historical tracking of large polygons with frequent localized changes**, such as environmental monitoring, urban planning, and mobile object trajectory analysis.

Limitations include potential computational overhead during reconstruction in cases of highly irregular or divergent vertex movements, and the dependency of grouping and temporal functions on repetitive data patterns. Future work includes empirical evaluation of storage efficiency, automated detection of movement patterns, and exploration of advanced compression and parallelized reconstruction techniques.

This model provides a **robust and flexible conceptual framework** for temporal polygon storage, balancing efficiency, integrity, and precision, and laying the foundation for practical implementations in GIS and spatial database systems.

References

[1] Ralph Kimball and Margy Ross, *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling*, 3rd edition, Wiley, Chapter 5, section "Type 2: Add New Row", pp. 151–154, 2002.

[2] Amit Chavan, *Delta-Based Storage and Querying for Versioned Datasets*, University of Maryland, 2018, Chapter 6: Delta Format, pp. 143–153.

[3] May Yuan, *Temporal GIS and Spatio-Temporal Modeling*, University of Oklahoma.

[4] M. F. Worboys,

A Unified Model for Spatial and Temporal Information, The Computer Journal, Vol. 37, No. 1, pp. 26–34, 1994.

[5] M. H. Böhlen, R. T. Snodgrass, and M. D. Soo, *Coalescing in Temporal Databases*, Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB), pp. 180–191, 1996.

[6] Richard T. Snodgrass, *Temporal Databases*, ACM Computing Surveys, Vol. 23, No. 4, pp. 501–549, 1992.