Spatio-Temporal Access Methods: A Survey (2017-2025) - Classic, Learned, and Contact Tracing Indexes

SYED MUHAMMED ABUBAKER, Purdue University, USA
NADA M. ELKORDI, Independent Researcher, USA
YEASIR RAYHAN, Purdue University, USA
AHMED R. MAHMOOD, Google LLC., USA
WALID G. AREF, Purdue University, USA

The proliferation of location-aware devices and location-based services continues to generate exceptional volumes of spatio-temporal data, demanding increasingly efficient indexing structures for query processing. This survey presents **Part 4** of our ongoing series on spatio-temporal access methods, building upon our previous works [202], [211], and [192]. We provide a comprehensive overview and broad classification of spatio-temporal access methods published between 2017 and 2025. Recent approaches are classified into the following categories: (1) Historical spatio-temporal indexes, (2) Current-time and recent spatio-temporal indexes, (3) Future prediction spatio-temporal indexes, (4) Unified past-present-future spatio-temporal indexes, (5) Spatio-temporal-textual and multi-attribute indexes, and (6) Scalable indexing for big spatio-temporal data. Furthermore, within the aforementioned categories, we note an increased emphasis on developing indexes that address pandemic response through contact tracing, indexes that leverage Machine Learning for adaptive query processing, and methods optimized for high-velocity streaming data. This survey summarizes recent advances, highlights major innovations, and identifies potential avenues for future research in the dynamic needs of spatio-temporal access methods.

CCS Concepts: • **General and reference → Surveys and overviews**; • **Information systems → Data management systems**; **Spatial-temporal systems**.

Additional Key Words and Phrases: Spatio-temporal · Survey · Indexing · Databases

ACM Reference Format:
Syed Muhammed Abubaker, Nada M. Elkordi, Yeasir Rayhan, Ahmed R. Mahmood, and Walid G. Aref. 2026. Spatio-Temporal Access Methods: A Survey (2017-2025) - Classic, Learned, and Contact Tracing Indexes . In *Woodstock '18: ACM Symposium on Neural Gaze Detection, April 03–05, 2026, Woodstock, NY*. ACM, New York, NY, USA, 58 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The exponential growth of location-aware technologies and location-based services has driven an unprecedented surge in spatio-temporal data generation and utilization. This proliferation has intensified the need for efficient spatio-temporal access methods to manage and query vast volumes of data across various temporal dimensions. Building upon our

Authors' addresses: Syed Muhammed Abubaker, syedm@alumni.purdue.edu, Purdue University, West Lafayette, Indiana, USA, 47907; Nada M. Elkordi, nada.magdi@gmail.com, Independent Researcher, Madison, Wisconsin, USA; Yeasir Rayhan, yrayhan@purdue.edu, Purdue University, West Lafayette, Indiana, USA, 47907; Ahmed R. Mahmood, amahmoo@google.com, Google LLC., Mountain View, CA, USA; Walid G. Aref, aref@purdue.edu, Purdue University, West Lafayette, Indiana, USA, 47907.

Manuscript submitted to ACM

previous surveys [202], [211], and [192], this paper presents **Part 4** of our comprehensive series on spatio-temporal access methods, focusing on developments from 2017 to 2025. In recent years, the landscape of spatio-temporal applications has expanded significantly, encompassing diverse domains such as transportation and ride-sharing, trajectory analytics, location-based social networks, and pandemic contact tracing. These applications generate and process spatio-temporal data at unprecedented scales, presenting new challenges for data management and querying; consequently, researchers have continued to develop and improve spatio-temporal access methods to address these evolving needs. The period from 2017 to 2025 has witnessed several notable trends in the development of spatio-temporal access methods. While traditional approaches for indexing historical, current, and future data remain relevant, new paradigms have emerged. Of particular significance is the integration of Machine Learning techniques [9, 26, 27, 164, 165, 329] to enhance index construction and query processing, exemplified by methods that incorporate probabilistic models and automatic parameter tuning. Also, the COVID-19 pandemic has introduced a new category of indexing requirements, leading to the rapid development of contact tracing methods [15, 40, 51, 87, 164, 165, 199, 269, 279, 327, 328] that balance efficiency while addressing privacy concerns. Furthermore, there has been sustained focus on developing scalable indexing structures capable of handling big spatio-temporal data in distributed environments [43, 51, 52, 127, 162, 163, 177, 199, 270, 302, 303, 308, 328]. Another major development is the increasing attention given to spatio-temporal-textual indexing methods that can efficiently handle additional data types X along with the spatio-temporal data, such as keywords and semantic attributes [11, 12, 14, 22, 58, 58, 59, 59, 64, 65, 71, 88, 89, 109, 214, 215, 218, 219, 223, 235, 266, 284, 329], in addition to other complex spatio-temporal data types, including annotated trajectories [109, 235, 266], moving regions [40, 41], and spatio-temporal events [11, 43, 77]. Figure 1 illustrates the evolution of spatio-temporal access methods during this period, with lines indicating the relationships between new index structures and their predecessors. The rest of this paper is organized as follows. Section 2 surveys methods for indexing historical spatio-temporal data; Section 3 focuses on indexing current and recent spatio-temporal data; Section 4 examines approaches for future prediction indexing; Section 5 discusses unified indexing methods for past, present, and future data; Section 6 explores indexing techniques for spatio-temporal-textual data and multi-attribute objects; Section 7 covers scalable indexing methods for big spatio-temporal data, and finally, Section 8 concludes the survey and outlines potential directions for future research in this dynamic field.

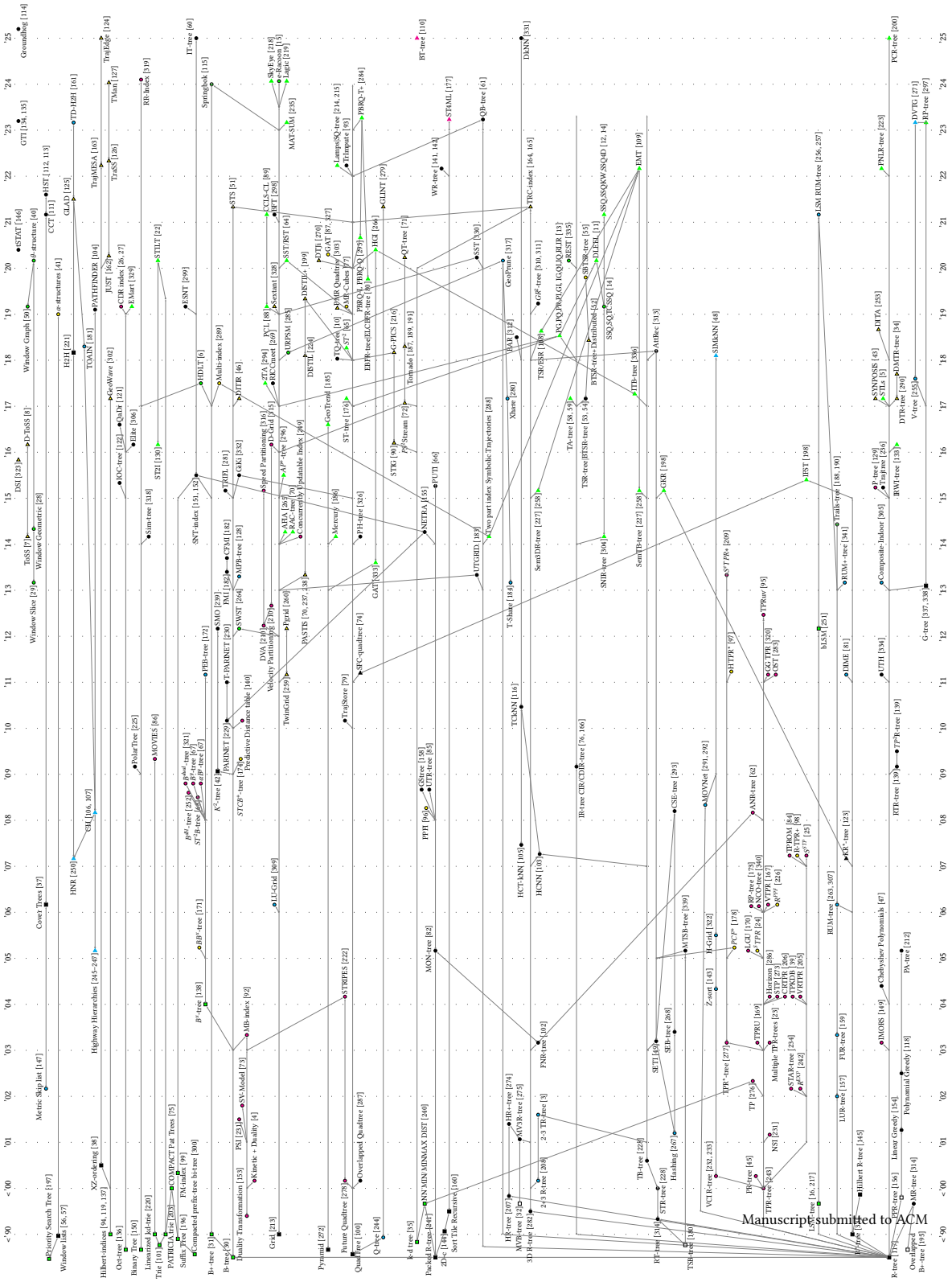

Fig. 1. The evolution of spatio-temporal access methods.

2 INDEXING THE PAST

This section surveys indexing methods designed for historical spatio-temporal data, where the complete dataset is available at index construction time. These structures optimize query performance over archival trajectory collections. They enable efficient retrieval of past movement patterns, spatial distributions, and temporal relationships. We organize the discussion around three major themes. First are trajectory-oriented access methods, which index movement paths and their properties. Second, are AI/ML-powered indexes that use machine learning, such as reinforcement learning, to adaptively optimize partitioning strategies and query processing based on data distribution and workload characteristics. Finally, we examine contact tracing indexes, which emerged in response to the COVID-19 pandemic, to support epidemiological analysis of proximity events.

TSR & BTSR Tree [53, 54]: The TSR-tree extends the R-tree [117] spatial index by augmenting each node with Minimum Bounding Time Series (MBTS) that capture temporal bounds alongside traditional Minimum Bounding Rectangles for spatial bounds. Each node maintains a pair of bounding time series: an upper bound T_{up} and a lower bound T_{lo}, constructed by selecting maximum and minimum values at each timestamp across all time series within the node's sub-tree. During query processing, the index computes lower-bound distances of spatial and temporal domains simultaneously, enabling dual-domain pruning as it traverses the tree. The BTSR-tree optimizes this approach by clustering similar time series within each node using k-means clustering, creating multiple tighter MBTS bundles rather than a single loose bound. The bundling strategy leverages the behaviour that time series with similar patterns yield more refined bounds, thereby significantly improving pruning effectiveness by reducing dead space within bounding regions. To manage node capacity constraints at higher tree levels, the BTSR-tree employs Piecewise Aggregate Approximation to decrease temporal resolution while increasing the number of bundles by a factor c at each level. The hybrid cost function for node insertion balances spatial MBR enlargement with temporal MBTS expansion using the parameter λ. During query evaluation, nodes are pruned when either spatial distance bounds exceed spatial thresholds or when temporal similarity bounds fail to meet time series similarity requirements, with the BTSR-tree checking each bundle's bounds individually for more aggressive pruning. The MBTS verification process computes distances at each timestamp between the query series and the bounding series, accumulating violations to determine pruning feasibility. Query processing employs best-first traversal for top-k queries and depth-first traversal for range queries, with priority queues managing candidates based on lower-bound distances. The index supports boolean range queries that combine spatial proximity and time-series similarity filters, top-k spatial queries with time-series constraints, and hybrid distance queries using exponential decay functions. The notation uses $T_{N,up}$ and $T_{N,lo}$ for the upper and lower bound time series, with *mindist* functions that compute lower-bound distances for pruning decisions.

SBTSR-tree [55]: The SBTSR-tree enhances the BTSR-tree [53, 54] through temporal segmentation of time series data, creating fine-grained bounds that improve pruning effectiveness for local similarity queries. The index segments each time series into fixed intervals, computing separate MBTS for each segment rather than maintaining bounds across entire sequences. This segmentation approach eliminates much of the dead space typically found in global time series bounds, which is highly beneficial when time series show varying patterns throughout different temporal regions. Each segmented MBTS maintains upper and lower bounding series constructed by selecting the maximum and minimum values within the segment boundaries. To preserve continuity of information across segment boundaries, the index employs bit vectors associated with each MBTS, indicating which time series are shared between adjacent segments. These bit-vectors enable traversal during local similarity verification by identifying relevant MBTS in neighboring segments without exhaustive checking. The local similarity scoring mechanism counts consecutive timestamps where

two series remain within the user-defined threshold ϵ, focusing on subsequence matching rather than global similarity measures. Checkpoint optimization decreases computational overhead by verifying similarity at selected timestamps (every δ intervals) rather than an exhaustive timestamp-by-timestamp comparison, with forward and backward scans from qualifying checkpoints to determine the actual similarity extents. Segmented bounds yield tighter pruning, since local patterns within segments are more homogeneous than global time-series characteristics. The SBTSR-tree processes range and k-nearest neighbor queries, including both spatial distance and local time series similarity constraints, with enhanced pruning through segment-specific bounds. The notation uses $\sigma(T, T', \epsilon)$ for local similarity scores and δ for minimum consecutive timestamp thresholds in local similarity queries.

***α*-structure** [41]: The α-structure enables querying of spatio-temporal datasets as storm events by retrieving simplified polygonal representations called α-shapes, which generalize convex hulls by including edges shorter than a threshold α, allowing flexible approximations of the true spatial extent. The structure supports temporal range queries of the form $Q = [t, t']$ to retrieve all points that have timestamps within that interval, and can handle spatio-temporal queries combining both spatial ranges $[x, x'] \times [y, y']$ and temporal ranges. Each edge in the α-shape is active for a specific time range, and the data structure stores activity boxes that define the time intervals during which each edge is included in the shape, where an activity box is defined as the smallest time range within which a particular edge is part of the α-shape. Activity boxes are represented as rectangles in a $2D$ plane where each temporal range query $[t, t']$ corresponds to a point (t, t') in this $2D$ space, with $t \leq t'$ making sure all queries lie above the diagonal. The boundaries of an activity box $\tau = (t_r, t_p, t_q, t_s)$ correspond to the time stamps of points connected by an edge, where t_r is the largest timestamp before t_p of points in the spatial domain of the edge, and t_s is the smallest timestamp after t_q. The box remains valid as long as there is no point within the spatial domain of the edge at which the inclusion of the edge would be invalid. By predefining these time-based boxes, the α-structure allows quick access to edges that should be displayed for any given temporal query. During the preprocessing phase, the structure is computed by aggregating all possible edges across temporal ranges, thus allowing fast responses to new queries. The α-structure utilizes rotational sweeps and balanced binary search trees (BST) [150] to maintain spatial and temporal intervals. The rotational sweeps are used to identify and store points that fall within the spatial domain of an edge for a given α-shape, while the balanced binary search trees are used to store and query the time stamps of points that are within the spatial domain of an edge. The BST helps ensure that only active edges are included in the α-shape during a time-range query. The α-structure aims to grow linearly with the number of points in practical use cases, despite the possibility of worst-case quadratic growth. The extensions include octilinear α-shapes, where edges are constrained to specific orientations, reducing memory consumption, and providing simplified visualizations. The structure also supports order-k α-shapes, which provide robust shape representations by including edges that may contain up to $k - 1$ points within their spatial domain, thereby making it more resistant to outliers.

Groundhog [114]: Groundhog introduces a self-contained and segment-based storage model optimized for in-storage computing on computational storage devices (CSDs) to process spatio-temporal range queries efficiently. The storage model organizes GPS point data, where each point $p = \{oid, lon, lat, t, a_1, a_2, \ldots, a_m\}$ contains an object identifier, spatial coordinates (longitude and latitude), timestamp, and optional attributes, into fixed-size 4KB data blocks aligned with the CSD sector sizes. Each block is designed to be self-contained, ensuring that complete tuples never span multiple blocks, thereby eliminating complex cross-block data management within the storage device. Internally, each block is subdivided into multiple logical data segments, where each segment maintains lightweight metadata in the form of a sketch containing an MBR (Minimal Bounding Rectangle) defined by spatial bounds $(lon_{\min}, lon_{\max}, lat_{\min}, lat_{\max})$, temporal bounds $(time_{\min}, time_{\max})$, segment offset, segment size, and an object identifier filter. The block format

follows the structure $[seg\_count \mid seg\_meta_0 \mid seg\_meta_1 \mid \ldots \mid seg\_meta_n \mid data\_seg_0 \mid data\_seg_1 \mid \ldots \mid data\_seg_n]$, enabling sub-block granularity during query processing. Data within blocks is ordered using Z-curve [220] space-filling curves to ensure minimal overlap between segment MBRs, enhancing pruning effectiveness. The query processing technique uses fine-grained pruning where, for each block, segment metadata is first evaluated against the query predicate P before examining individual tuples, allowing entire segments with non-overlapping MBRs to be skipped. Additionally, Groundhog implements fine-grained offloading, which makes block-level decisions about whether to process each block on the host CPU or offload it to the CSD, rather than offloading entire queries; blocks with low selectivity (small fraction of matching tuples) are offloaded to the CSD where reduced data movement provides benefit, while blocks with high selectivity are processed on the host where faster computation outweighs data transfer costs. This offloading strategy is guided by a cost model that estimates host execution time $T_{\text{host}}(Q) = \frac{(n \times S)}{B_n} + \frac{(n \times S)}{B_p} + \sum_i T^i_{\text{host\_comp}}$ and device execution time $T_{\text{device}}(Q) = \frac{(n \times S)}{B_n} + \sum_i T^i_{\text{device\_comp}} + \frac{(n \times S \times \alpha)}{B_p}$, where n denotes the number of blocks, S represents block size, B_n is the internal SSD bandwidth, B_p is the host interface bandwidth, α indicates result data ratio (selectivity), and T^i_{comp} represents the computation time at processing step i. Groundhog handles spatio-temporal range queries $Q(S, R)$ that search dataset D for points located within spatial range $S = [lon_{\min}, lat_{\min}, lon_{\max}, lat_{\max}]$ during time range $R = [t_s, t_e]$, supporting both time-preferred queries with narrow temporal windows over large spatial regions and space-preferred queries with small spatial regions spanning extended time periods.

2.1 Trajectory-oriented Access Methods

Trajectory-oriented access methods organize historical movement data to support queries over paths, routes, and movement patterns. These indexes address challenges including compression of large trajectory datasets, similarity search across millions of paths, and efficient retrieval under combined spatio-temporal and attribute constraints. The methods presented here extend classical spatial indexes with temporal dimensions, leverage string-processing techniques for path matching, and employ hierarchical partitioning strategies designed for road networks and movement characteristics.

PATHFINDER [104]: PATHFINDER presents an indexing approach for massive trajectory sets in road networks by re-purposing *Contraction Hierarchies* (*CH*) [107], traditionally used for route planning, as both a compression mechanism and a spatial index. The index augments *CH* with two important components: *Path Boxes* (*PB*) and *Downgraph Boxes* (*DB*). For each edge e in *CH*, $PB(e)$ maintains the bounding box of the path it represents in the original graph, while $DB(v)$ for each node v stores the bounding box of all nodes reachable via downward paths in the *CH* hierarchy. The index construction begins by building a *CH* through iterative node contraction, in which nodes are removed, and shortcuts are added to preserve shortest-path distances. Trajectories are then compressed by converting them to *CH* representations and replacing edge sequences with shortcuts, achieving significant compression. The temporal dimension is handled through interval trees storing time information and a periodic time-slice system using bitset (64 slices per week). Query processing uses a top-down traversal of the *CH*, starting from the highest-level nodes and using *DB* boxes for efficient spatial pruning. The process identifies candidate edges whose *PB* intersect the query window, then refines them using actual path geometry and temporal filtering via interval trees. For ambiguous cases, the shortcuts are recursively unpacked to verify the true intersection.

GTI [134, 135]: GTI is a graph-based trajectory imputation framework that reconstructs missing GPS locations and timestamps in sparse movement data without requiring underlying road network maps. The framework uses a Right-of-Way (RoW) concept to build a directed graph $G = (V, E)$ where the vertices V represent unique GPS

samples and the edges E connect samples within the distance threshold d, using a modified distance metric $d_\theta(s_i, s_t) = \sqrt{d(s_i, s_t)^2 + (\theta \frac{d_\theta(\alpha_i, \alpha_t)}{180^\circ})^2}$ that incorporates bearing angle penalties θ to discourage unrealistic direction changes. The system includes graph generation optimization using clustering techniques to decrease computational complexity, and applies density-based edge weighting for road centerline preference during shortest-path computation. GTI can act as a preprocessing step to increase the accuracy of trajectory data management systems and trajectory-based applications such as map inference and travel time estimation. The system supports queries for spatial trajectory imputation (finding plausible paths between consecutive sparse samples using shortest-path algorithms) and temporal trajectory imputation (estimating timestamps based on dynamic travel cost distributions with temporal discretization into weekdays/weekends and hourly slots for contextual speed information). Post-processing trajectory refinements use sliding-window techniques to improve the quality of imputed trajectories. GTI handles range queries for intersection monitoring and trajectory similarity computations across the imputed dense trajectories.

REST [335]: REST (Reference-based Spatio-temporal trajectory compression) is a framework that compresses trajectory data by representing raw trajectories as concatenations of historical reference trajectories within given spatio-temporal deviation thresholds. It maintains bounded spatial and temporal errors through a Dynamic Time Warping (DTW) distance metric, MaxDTW. REST employs greedy and optimal dynamic programming algorithms to select the most efficient combination of reference trajectories for compression, supporting both spatial-only and spatio-temporal compression modes. It incorporates a hybrid indexing structure called IR-tree [166], to support range and trajectory query processing on compressed spatio-temporal trajectories. The IR-tree combines an R-tree [117] with inverted files to create an indexing mechanism. The R-tree component is constructed using the minimum bounding rectangles (MBRs) of reference trajectories, which are used to compress the original trajectory data. Each node in the R-tree is augmented with an inverted file, creating a multi-layered indexing structure. The leaf nodes in the IR-tree contain entries of the form $(T.id, MBR, ifile)$, where $T.id$ is the identifier of a reference trajectory, MBR is its minimum bounding rectangle and $ifile$ points to an inverted file. This inverted file stores mappings from the reference trajectory to the original trajectories that utilize it for compression, represented as triples $(T^{(m,n)}, T^{(m)}.t, T^{(n)}.t)$, where $T^{(m,n)}$ indicates the portion of the original trajectory compressed by the reference, and $T^{(m)}.t$ and $T^{(n)}.t$ are the start and end timestamps. Uncompressed trajectory segments are treated as new reference trajectories within this indexing scheme. Non-leaf nodes maintain similar structures, but with inverted files that index all reference trajectories in their subtrees. The index construction process includes building the R-tree on reference trajectory MBRs, creating inverted files for each reference trajectory, attaching these files to leaf nodes, and then propagating and merging them up the tree hierarchy.

TrImpute [93]: TrImpute is a framework for trajectory imputation that inserts artificial GPS points between real ones, mimicking high sampling rates without requiring a road network. TrImpute has three main components: Preprocessing, Spatial Imputation, and Temporal Imputation. For Preprocessing, TrImpute builds two index structures: a Hierarchical Quadtree for spatial indexing and an Inverted list for organizing trajectory *ID*s, while enriching GPS points with angle and speed data. The Hierarchical Quadtree index organizes GPS points spatially using coordinates (*latitude, longitude*), allowing spatial queries to find nearby candidate points within a specified radius, and the Inverted list index retrieves trajectory points in temporal order using trajectory IDs as keys, allowing temporal queries to access GPS points ordered by timestamp for each trajectory. In the notation, each GPS point P has format ⟨*TrajID, PointID, latitude, longitude, timestamp*⟩, where candidate points P_C are potential imputed locations with attributes (*latitude, longitude, angle, timestamp, speed*). Spatial imputation finds the shortest path between points using crowd wisdom-based candidate

selection with parameters N (number of candidates), α (crowd ratio threshold), δ (angle threshold), and d (distance threshold). Temporal imputation assigns timestamps to imputed points by distributing timing errors evenly across segments while respecting traffic conditions inferred from endpoints.

Springbok [115]: Springbok presents a dual-indexing architecture designed specifically for cloud-based trajectory data management: a grid-based [213] index for active "head chunks" and a B$^+$tree [31] variant for historical "immutable chunks". The head chunk index uses a semi-split strategy that creates new sub-grids without redistributing existing data when a grid cell reaches capacity, maintaining a depth table to track the grid hierarchy. This approach decreases the overhead of handling skewed trajectory distributions while preserving spatial locality. The immutable chunk index operates at chunk granularity rather than individual points, organizing entries in a B$^+$tree structure with the format $([time_s, time_e], [lon_{min}, lat_{min}, lon_{max}, lat_{max}], oid, pointer)$, where entries are temporally ordered and contain logical pointers to handle cloud storage dynamics. Each leaf node maintains both temporal and spatial indexing components, including a spatial bitmap that divides the trajectory's MBR into $n \times n$ grids for precise spatial filtering beyond simple MBR checks. The system optimizes insertions by using an active leaf-node pointer to avoid tree traversals for new entries and delays parent-node updates until leaf nodes reach capacity. Query processing spans both indexes, with spatio-temporal range queries leveraging the grid hierarchy in the head chunk index and the B$^+$tree structure using bitmap filtering in the immutable chunk index. For example, a spatial range query first calculates overlapping top-level grid keys, checks the depth table for sub-grids, and retrieves matching posting lists, while the temporal predicates utilize the ordered nature of the B$^+$tree. The system uses logical pointers rather than physical locations to handle data movement between storage tiers and implements lazy deletion with background merging to maintain efficiency.

QB-tree [61]: The QB-tree is a three-layer hybrid index designed for efficient Max-Influence Trajectory (MIT) queries, combining spatial partitioning with motion pattern-based organization. MIT queries find an optimal location $c \in C$ that maximizes the total weight of influenced trajectories, where a trajectory t is influenced by c if $d(t, c) \leq d(t, f)$ for all existing facilities $f \in F$. At its core is a Quadlike tree structure that recursively divides the space into four disjoint subspaces, grouping spatially similar trajectories. Within each node, trajectories are further classified into hash buckets based on their motion patterns, encoded using Morton-based encoding [204, 220] method. Each bucket contains a rectilinear polygon (RP) representing trajectory subregions, offering a more precise spatial representation than traditional MBRs. The finest granularity is achieved with sorted lists of trajectories within each bucket, ordered by their attractor distance $ad(t) = min_{f \in F}\{d(t, f)\}$, where $d(t, f)$ is the distance between the trajectory t and the facility f. This hierarchical organization enables a three-level pruning approach during query processing: node-level pruning using MBR and attractor-distance bounds, bucket-level pruning leveraging RPs and bucket-specific bounds, and trajectory-level pruning using sorted lists. The index construction process recursively partitions the space, inserting trajectories into the smallest enclosing node and appropriate bucket, updating bounds and weight sums along the path. The nodes are split when they exceed capacity and contain trajectories spanning multiple child nodes. Query processing uses a Branch-and-Bound MIT (BBM) algorithm that recursively partitions candidate facilities into regions, computes influence bounds using the three-level pruning technique, and refines the remaining candidates. This structure extends concepts from Quadtrees [100] and R-trees [117], combining them with hash buckets and sorted lists for a trajectory-specific organization.

SNT-index [151, 152]: The Suffix-Array-based Network-Constrained Trajectory index (SNT-index) adopts string processing techniques to handle spatio-temporal trajectory queries efficiently. The SNT-index incorporates a Spatial FM-index [99], an in-memory structure built on suffix arrays [193, 194] that allows fast pattern matching across trajectory networks. By representing spatial paths as sequences of road segment IDs, the SNT-index efficiently leverages

suffix arrays to quickly and effectively handle spatial queries. Furthermore, temporal information, timestamps, for each road segment is managed in an inverted index based on B$^+$tree-based [31] inverted index, allowing efficient temporal filtering. Each road segment in the road network has an associated posting list that stores trajectory IDs and timestamps. The B$^+$tree supports fast temporal range queries, and each posting list also contains inverse suffix array values (ISA) to link the temporal data with the spatial FM-index. In addition, the inverse suffix array operates as a bridge between the FM-index and the temporal B$^+$trees. It allows the FM-index to handle temporal constraints by associating spatial queries with timestamped records, integrating spatial and temporal dimensions. The SNT-index supports the following queries: Strict Path Query (SPQ), Trajectory Extraction Query (TEQ), and Time-Period-Based All-Path Enumeration Query (TAPEQ). The Strict Path Query (SPQ) finds trajectories that match a specific route and a temporal constraint. The SNT-index reduces disk accesses by enabling a single scan of the FM-index for the spatial pattern, followed by a B$^+$tree query to filter by time. Therefore, the Trajectory Extraction Query (TEQ) allows predicting future positions by retrieving subpaths of trajectories that continue after a given spatial and temporal condition, and the Time-Period-Based All-Path Enumeration Query (TAPEQ) enumerates frequently traveled paths between two points within a particular time frame, helping to identify preferred routes or shortcuts in the network. In addition, the SNT-index reduces disk I/O by minimizing B$^+$tree retrievals, with FM-index operations taking place in-memory, improving query processing speed, especially for longer paths. In addition, the suffix-array-based approach allows data to be appended to the index periodically without reconstructing the entire index, making it well-suited for applications with periodic data updates.

The Extended SNT-index [299]: The Extended SNT-index enables a system that processes network-constrained trajectories (NCT) and strict-path queries (SPQs) to estimate accurate travel times by retrieving trajectories that strictly follow a given path. The system handles travel-time queries consisting of a road network path and start time, returning travel-time histograms based on historical trajectory data. NCT refers to vehicle movement data constrained to road networks, while SPQs retrieve all trajectories that traverse a specific path without detours. The index combines spatial and temporal indexing, with the Spatial index being an FM-index [99] using the Burrows-Wheeler Transform (BWT) [44], which compactly encodes trajectory data, enabling fast, exact path matching and reducing the search space. The temporal Index logs timestamps and traversal durations for road segments, allowing time-specific queries, such as average travel times during peak hours. The system follows a data-oriented architecture in which a Dispatcher coordinates incoming updates and queries via lock-free SPSC (single-producer/single-consumer) and MPMC (multiple-producer/multiple-consumer) queues. An Update Worker processes trajectory updates and maintains batch operations for index refreshing. The system is optimized for Non-Uniform Memory Access (NUMA) architecture to decrease latency by using region and socket-specific partitioning, reducing remote memory access, and improving parallel processing. Therefore, the Strict-Path Query(SPQ) processing retrieves only trajectories that exactly match a queried path, while a convolution thread (CT) consolidates partial travel-time results into a single travel-time histogram. The system combines spatial workers and temporal workers. Spatial workers handle SPQs and partition complex queries for distributed processing. Temporal workers manage segment traversal times and return results to the CT. The extended SNT-index provides real-time update capability by processing trajectory data simultaneously with travel-time queries. It uses batch updates to refresh the index without disrupting query execution.

tSTAT [146]: tSTAT (trajectory-indexing succinct trit-array trie) enables trajectory similarity search in large datasets. The index handles queries to find trajectories within a specified Fréchet distance [17] threshold R from a query trajectory, retrieving all similar trajectories from collections containing millions of trajectories. It leverages Locality-Sensitive Hashing (LSH) to efficiently retrieve similar trajectories using the Fréchet distance. tSTAT uses a trie-based [101] structure for efficient trajectory traversal, implements succinct data structures to decrease memory while ensuring

fast operations, and handles millions of trajectories effectively. During the tSTAT's search process, the trajectories are compressed into sketches, preserving the Fréchet distance as the Hamming distance [120]. Sketches are partitioned into blocks, and separate tries are built for each block position across all sketches, with each node representing a common prefix, enabling fast lookups. Moreover, sketches are split into blocks, each assigned Hamming distance thresholds to improve search efficiency, while queries follow a multi-index approach, filtering candidates block by block; non-matching subtrees are skipped to save computation time. On the other hand, the Fréchet distance is computed for the remaining candidates to guarantee accurate results. In addition, tSTAT's trie is optimized by discarding redundant nodes, reducing memory usage without affecting search speed, where succinct data structures are represented by compressing the trie to store data efficiently while supporting fast queries.

RP-tree [297]: The RP-tree (Road network-aware Partition Tree) introduces a spatio-temporal indexing structure designed for trajectory range query processing. The RP-tree handles spatio-temporal range queries (RQs), which find all trajectories that intersect given rectangular query ranges in space and time. These include single-range queries (finding trajectories that intersect a single spatio-temporal rectangle) and multiple-range queries (finding trajectories that intersect all of multiple query ranges). $G = (V, E)$ represents the road network graph with vertices V (road joints) and edges E (road segments), trajectories $t = \langle p_0, p_1, \ldots, p_k \rangle$ are sequences of vertex-timestamp pairs $p_i = (v_i, \tau_i)$, and query ranges r or $R = \{r_1, r_2, ..., r_m\}$ define spatio-temporal rectangles. The RP-tree utlizes the underlying road network topology to create a hierarchical balanced Binary Search Tree (BST) [150]. Index construction begins by recursively partitioning the road network graph into subgraphs, prioritizing edge-balanced partitions based on trajectory density. Each node in the tree represents a subgraph and maintains its spatial minimum bounding rectangle (MBR), associated trajectories, and linking edges to child nodes. The partitioning process continues until the leaf nodes contain no more than a predefined threshold of vertices. The RP-tree uses a minimum-level threshold for trajectory indexing, therefore optimizing storage for large networks. During query processing, the tree is traversed recursively to identify edges that intersect the query range, and the corresponding trajectories are retrieved directly from the relevant nodes. This road-aware design eliminates the need for a refinement step. For multiple-range queries, the RP-tree uses a minimum multi-way cut algorithm to prune unnecessary edges. The index supports both exact and approximate query processing, with the latter utilizing stratified sampling techniques based on the hierarchical structure. RP-tree also supports dynamic updates via localized repartitioning of affected nodes. The index requires initial map-matching of trajectories and incurs higher indexing time; these trade-offs enable faster query processing by eliminating the refinement step.

TQ-tree [10]: The Trajectory Quadtree (TQ-tree) builds on the traditional Quadtree [100] and supports two new queries in trajectory databases, which are the k Best Facility Trajectory Search (kBFT) and the k Best Coverage Facility Trajectory Search (kBCovFT) query. The $kBFT$ query returns k facilities that maximize a predetermined objective function for a set of user trajectories. The $kBCovFT$ query returns k facilities that combined satisfy a predetermined objective function for the maximum number of user trajectories. The TQ-tree uses a Quadtree to partition the space, with the Quadtree nodes forming the TQ-tree nodes. Unlike a quadtree, both the internal nodes and the leaf nodes of a TQ-tree store data. Each node of the TQ-tree consists of a pointer to its child nodes, a list of stored trajectories, and an upper-bound of the objective function for the trajectories stored within its sub-tree. The leaf nodes of the TQ-tree store only intra-node trajectories, i.e., trajectories whose two endpoints reside within the same node. In contrast, the internal nodes of the TQ-tree consist of intra-node trajectories, whose endpoints reside in the two immediate child nodes of the corresponding internal node. A TQ-tree node is split until all its inter-node trajectories are fully indexed, or the number of intra-node trajectories within the node falls below a specified threshold. To improve search latency, the TQ-tree further organizes the trajectories in each TQ-tree node list into buckets called *z-nodes*, based on Morton ordering [220].

The insertion algorithm of the TQ-tree follows the Quadtree insertion algorithm to locate the appropriate TQ-tree node. The algorithm identifies the corresponding *z-node* within that node's list of stored trajectories, which may trigger node split or reordering of the *z-nodes* within the corresponding index node.

AttRec [313]: This paper presents an indexing technique for multi-attribute trajectories, combining spatial, temporal, and attribute-based filtering in a unified framework. The core structure consists of a 3D R-tree [282] for spatial and temporal indexing, augmented with an *AttRec* (Attributes and Records) structure for efficient attribute-based filtering. The 3D R-tree indexes packed trajectory segments using 3D bounding boxes (x, y, time), with each node containing entries of bounding boxes and pointers to subtrees or trajectories. The *AttRec* structure, built on top of the R-tree [117], maps the attribute values to integers and maintains a relation for each attribute value, storing records of R-tree nodes containing that value. Each record uses a bitmap to mark matching entries within a node, enabling quick filtering during query processing. The index supports Range Queries on Multi-attribute Trajectories (RQMAT) that combine attribute predicates with spatio-temporal constraints, returning trajectories that both contain specific attribute values and intersect a given spatio-temporal window. The query-building process includes packing raw trajectory data, bulk-loading the 3D R-tree, and constructing the *AttRec* structure by traversing the R-tree for each attribute value. Query processing occurs in two steps: select R-tree nodes containing query attribute values using *AttRec*, and then traverse the R-tree using breadth-first search, pruning branches that do not satisfy both attribute and spatio-temporal conditions. This procedure allows for efficient pruning of non-matching attribute combinations early in the query process. The index supports dynamic updates by creating small R-trees and *AttRec* structures for new data and merging them with existing structures. Although the index construction is more complex due to the *AttRec* building, it significantly speeds up query matching through early pruning.

BAR [312]: The BAR (B-tree, Attribute relation, and Record file) index structure presents an approach to indexing multi-attribute trajectories by combining a novel 3D R-tree [282] for spatio-temporal data with a specialized attribute indexing mechanism. Multi-attribute trajectories enhance standard trajectories with descriptive attributes that are independent of location, such as vehicle COLOR (RED, SILVER, BLACK), BRAND (BENZ, VW, TOYOTA), bus route IDs, or taxi company IDs. BAR consists of three key elements: a relation *Att_Rel* storing attribute values, heights (tree levels from leaf = 1 to root = H), and record IDs. A B-tree built on *Att_Rel* for fast lookups, and a record file storing items with format (nid, b, t) where nid is the node ID, b is a bitmap representing entries containing the attribute value, and t is the time interval. The index building process begins by packing small trajectory units into larger ones to create a compact dataset, followed by constructing the 3D R-tree using bulk loading. For each attribute value, the R-tree [117] is traversed to identify the nodes containing that value, creating records with bitmaps indicating which entries have the value, and inserting the corresponding tuples into *Att_Rel*. This structure allows for querying of CkNN_MAT (continuous k nearest neighbor queries over multi-attribute trajectories), which returns k trajectories at each time $t \in T(mq)$ that contain query attribute values $Q_a = (a_1, \cdots, a_d)$ and belong to k nearest neighbors of the query trajectory. To handle frequent updates efficiently, BAR introduces a lightweight update structure called *lw*-BAR (lightweight BAR). Rather than immediately updating the main BAR structure when new trajectories arrive, *lw*-BAR buffers update information in a simplified format that stores only the affected record items. This approach significantly reduces I/O costs during updates by avoiding the need to load and modify entire records for each change. The buffered updates in *lw*-BAR are periodically merged into the main BAR structure, maintaining query performance while supporting dynamic trajectory databases. This hybrid approach allows the index to efficiently process both historical and incoming trajectory data while minimizing update overhead.

WR-tree [141, 142]: The WR-tree is a spatio-temporal index structure designed specifically for trajectory-based entity linking, which is designed to match the same moving objects across separate datasets by comparing their movement patterns. The index handles k-nearest neighbor queries to find the most similar moving objects based on weighted spatial signatures. In the notation, $f(o)$ represents the signature of moving object o, $w(p)$ denotes the TF-IDF weight of spatial point p, MBR(o) is the minimum bounding rectangle of object o's spatial signature, and $\text{sim}(o_1, o_2)$ measures the cosine similarity between two object signatures. The index works with trajectory points that are weighted using TF-IDF scoring, where the weight represents both the commonality (frequency of a location in an object's trace) and unicity (discriminative power of the location across all objects) of each point in an object's movement history. The WR-tree extends the traditional R-tree [117] by incorporating spatial and weight information, storing object signatures as weighted vectors that capture movement patterns. The index structure consists of leaf nodes containing object identifiers, signatures, and Minimum Bounding Rectangles (MBRs), while internal nodes maintain pointers to children, aggregated signatures, and aggregated MBRs. The tree is constructed using a bulk-loading algorithm based on the Sort-Tile-Recursive [160] strategy, which optimizes three criteria: signature enlargement (maximizing common points), MBR area, and overlap minimization. For query processing, the WR-tree uses a "best-first" k-Nearest Neighbor search with two effective pruning strategies: spatial overlap and signature-similarity thresholding. To increase accuracy, two optimization techniques are implemented: re-ranking, which first retrieves top-k candidates using small signatures (e.g., 10 dimensions) and then updates rankings using larger signatures (e.g., 500 dimensions) for better precision, and stable marriage matching, which creates one-to-one mappings between objects from different datasets based on mutual preference lists derived from signature similarities. The combination of these optimizations significantly improves the accuracy of linking with minimal computational overhead. The index supports dynamic updates via incremental insertion, though periodic bulk rebuilding is recommended to maintain optimal performance. This architecture transforms high-dimensional signature similarity searches into efficient 2D spatial searches, making it suitable for large-scale trajectory datasets where classic methods struggle with dimensionality and cardinality challenges.

HST [112, 113]: The Hierarchical Simplification Tree addresses sub-trajectory nearest-neighbor queries by simplifying the input trajectory (P) at multiple resolutions, pruning the search space, and accelerating query processing. Unlike time-based structures, HST represents different granularities of a trajectory. HST works on resolution simplification by reducing P's vertices at multiple levels while preserving shape by grouping close vertices into simplified segments. Its tree structure is represented as a hierarchical tree, where each node represents a trajectory interval at a specific resolution. HST is built top-down using a greedy simplification approach. The nodes are split into segments at each resolution, stopping at the finest resolution (single-line segments). In addition, it has a linear space complexity, outperforming the Cluster Center Tree (CCT), which can be quadratic in sub-trajectory settings. HST's nodes store simplified sub-trajectories and prune irrelevant ones, ensuring efficient nearest-neighbor search under the continuous Fréchet [17] distance. The query processing in HST begins with a top-down search at the coarsest resolution, refining candidates using metric pruning and the triangle inequality. At each resolution for distance computation, candidate sub-trajectories are compared to the query trajectory. Finally, the exact continuous Fréchet distance is computed, returning the closest sub-trajectory at the finest resolution (leaf nodes).

GR^2-tree [310, 311]: The GR^2-tree is an indexing structure designed to efficiently support Continuous Range queries with Attributes (CRA) over multi-attribute trajectories. A multi-attribute trajectory consists of both spatio-temporal components (sequence of time-stamped locations) and descriptive attributes, e.g., vehicle type and status. The index consists of two integrated components: a GR-tree (an adapted 3D R-tree [282]) and R_{att} (an attribute relation). The

indexing technique uses an optimal grid partitioning strategy in which the 2D space is divided into $\delta \times \delta$ equal sized cells and time into K intervals, with the optimal granularity calculated as $\delta = \left\lceil \sqrt{P} \right\rceil$ where $P = \frac{|O| \cdot T_{avg}}{K \cdot f}$ balances the number of nodes and the quality of the approximation. Each spatio-temporal trajectory is decomposed into "cell trajectories" that are segments confined to a single grid cell during a specific time interval. Cell trajectories are then indexed in the *GR*-tree and sorted by time, cell ID, and bounding box. In addition, the R_{att} component stores attribute values as tuples (nid, a_tr, b) where nid is the node ID, a_tr is a transformed attribute value created by interleaving binary representations of the attribute ID and value, and b is a bitmap indicating which entries contain each attribute value. It can process CRA queries of the form (o_q, d, Q_a) that return trajectories containing attribute values Q_a whose distance from the query trajectory o_q is always below the threshold d during overlapping time periods. The query processing algorithm runs in three phases: (1) construct a C^3 structure containing a cell tree (mapping time intervals to cells within query distance), cell set (all cells within distance d), and cell list (marked cells where the maximum distance < d), (2) traversing the GR^2-tree in breadth-first order with spatial, temporal, and attribute-based pruning; and (3) perform exact time-dependent distance calculations for candidates. To accelerate distance computation, the index implements a *BB-array* structure that allows efficient retrieval of minimum bounding boxes. The GR^2-tree supports dynamic updates with complexity $O((f + H) \cdot (1 + \frac{dom(A)}{b}))$, where f is the capacity of the R-tree node, H is the height of the historical R-tree, $dom(A)$ is the total domain size of attributes, and b is the block size.

Bloom filtered Trajectories ($\mathbf{I_{TBF}}$) [298]: This paper presents an index based on the Bloom Filter (I_{TBF}) to process track queries over trajectory data. A track query, denoted as $Q = \{q_1, q_2, \ldots, q_n\}$ where q_i represents query points, retrieves trajectory subsequences, $l = \{p_m, \ldots, p_{m+k}\}$ where each point p_i is represented as $\langle id, timestamp, latitude, longitude \rangle$. The method converts spatial trajectory queries into string queries by employing a geographic grid system and geocoding. Initially, the geographic space is partitioned into a regular grid $G_{geo} = \{T_{msr}, g_{size}\}$, where T_{msr} is the minimum spatial range of coordinates and g_{size} is the cell size in meters. Each cell is assigned a unique geocode. Trajectory data are then segmented into tracks, which are encoded using the geocodes of the grid cells they traverse, effectively converting spatial trajectories into string representations. The core of the technique lies in the construction of a Bloom filter $I_{TBF} = \{\sum_{i=0}^{n} S_i, \sum_{j=0}^{m} hf, B_{arr}\}$ where S_i represents encoded tracks, hf represents m hash functions, and the resulting hash values set corresponding bits in the Bloom filter's bit array. During query processing, the input query track undergoes the same encoding process and is then hashed using the identical hash functions. The Bloom filter is then checked to determine if all corresponding bits are set, indicating the potential existence of the track in the dataset with a false positive rate calculated as $p = \left(1 - e^{\frac{-nk}{m}}\right)^k$, where n is the number of input elements, k is the number of hash functions, m is the size of the bit-array and p is the false positive rate. The technique supports exact-match queries via bit-array operations and hash computations, with query processing performed directly on the bit array without accessing the original trajectory data. The index maintains trajectory information in a compact bit array format, although it requires filter reconstruction for updates and cannot support similarity or range-based queries due to its exact matching nature.

SST [330]: SST (Synchronized Spatial-Temporal trajectory similarity search) presents an efficient top-k trajectory similarity search using a grid-based [213] indexing technique combined with query partitioning. The approach processes queries represented as $Q = \langle l_1, \ldots, l_{|Q|} \rangle$ (a sequence of line segments with spatial-temporal points) to find k trajectories T that maximize the bidirectional similarity measure $Sim(T, Q) = \alpha \cdot Sim(Q \rightarrow T) + (1 - \alpha) \cdot Sim(T \rightarrow Q)$. The indexing method divides the spatial area into grid cells, mapping the trajectories to the cells they intersect. To reduce index size and improve search efficiency, the approach introduces a grid cell merging algorithm that clusters cells with

common trajectory IDs. This clustering creates compact trajectory "skeletons" that serve as a more space-efficient representation of the trajectory database. The merged grid cells are then indexed using an R-tree [117] structure. The main similarity measure used in SST is MPS (Matching Point-Point Based Similarity), which combines spatial and temporal aspects by computing similarity at each matched point pair using an exponential decay function $f(x) = e^{-x}$ to normalize distances. Query processing follows a three-phase approach. First, the framework partitions long query trajectories into smaller segments using a cost model that balances search and candidate maintenance costs. Second, it performs an incremental search using these partitions to retrieve trajectory candidates from the grid index, where each partition's search space is pruned using spatial-temporal bounds. Finally, it refines the candidates by computing the exact similarities in descending order of their upper bounds until the top-k results are confirmed. The approach maintains upper and lower similarity bounds for candidate trajectories and stops the search when the k^{th} greatest lower bound exceeds the upper bound of unexamined trajectories.

IT-tree [60]: The IT-tree (Indoor Trajectory Tree) is a two-layer hybrid spatio-temporal index designed for indoor trajectory data to support Top-k Probabilistic Influence (TkPI) queries, which return k sites with the highest individual influence values, and Collective-k Probabilistic Influence (CkPI) queries, which find k sites collectively maximizing influence over a set of user trajectories, where influence probability $P(t, s) = 1 - \prod_{i=1}^{n}(1 - e^{-\lambda \cdot \text{dist}(p_i, s)})$ measures the likelihood of trajectory t being influenced by site s given threshold θ. Each trajectory $t \in T$ represents a sequence of discrete points $\{p_1, p_2, \ldots, p_n\}$ expressed in a 2D Cartesian coordinate system, where each point $p_i = (x_i, y_i)$ encodes a physical location within the indoor floor plan measured along two perpendicular spatial axes within the indoor space $\mathbb{I}^2$. The first layer, called the indoor partitioning layer L_ρ, adopts a hierarchical spatial organization by combining adjacent indoor partitions P into nodes following a bottom-up construction strategy, where each IP node N_i^ρ is represented as a sextuplet $(G_i, BD_i, \underline{\text{len}}, \overline{\text{len}}, ws, ptr)$ containing: a D2D subgraph G_i with access doors AD_i and pre-computed distance matrix DM_i; border doors BD_i that connect internal partitions to external space; upper and lower bounds $(\underline{\text{len}}, \overline{\text{len}})$ on trajectory lengths; weight sum ws of enclosed trajectories; and pointers ptr to the second layer. The first layer stores each trajectory in the smallest node that completely covers it, enabling $O(\rho^2 \log L)$ distance computations where ρ denotes the average number of access doors per node and L represents leaf node count. The second layer, the trajectory trie layer $\mathcal{L}_\tau$, transforms each trajectory into a check-in trajectory $\check{t}_i = \{P_1 : n_1, P_2 : n_2, \ldots, P_k : n_k\}$, a histogram sketch recording partition visitation frequencies, then compresses trajectories sharing identical partition prefixes using a trie structure where each trie node N_{τ_i} stores $(P_i, \underline{n}, \overline{n}, \overleftarrow{ws}, \overrightarrow{ws}, T_i)$: partition label P_i, point count bounds $(\underline{n}, \overline{n})$ for this partition, cumulative weight sums for paths $(\overrightarrow{ws})$ versus stored trajectories $(\overleftarrow{ws})$, and the actual trajectory set T_i. Query processing for TkPI and CkPI exploits this structure through subtree pruning, which computes probability bounds $P_{lb}(N^\rho, s) = 1 - (1 - e^{-\lambda \cdot \text{dist}^+(N^\rho, s)})^{N^\rho . \underline{\text{len}}}$ and $P_{ub}(N^\rho, s) = 1 - (1 - e^{-\lambda \cdot \text{dist}^-(N^\rho, s)})^{N^\rho . \overline{\text{len}}}$ using indoor distance bounds dist^+ and dist^- derived from border doors, and progressive pruning along trie paths that incrementally tightens influence bounds node by node as more partition-level information is revealed, using trajectory monotonicity (if $t' \subseteq t$ then $P(t, s) \geq P(t', s)$) to compute tighter bounds $\check{P}_{lb}$ and $\check{P}_{ub}$ and prune trajectories early without full probability computation.

2.2 AI/ML-Powered

This subsection presents indexing methods that leverage machine learning to optimize index construction and query processing over historical spatio-temporal data. These approaches use techniques, e.g., Reinforcement Learning (RL), to adapt partitioning strategies based on data distributions and query workloads, thereby improving I/O performance in comparison to traditional fixed-hierarchy approaches.

BT-tree [110]: BT-tree is a reinforcement learning-based index for historical trajectory data, built on top of a disk-based spatio-temporal structure constructed through recursive bi-partitioning. It supports range queries with combined spatial and temporal predicates (e.g., finding all trajectories passing through a spatial region during a specific time interval) and K-Nearest-Neighbor (KNN) queries that return the K spatially nearest trajectory segments within a temporal range. Each node represents a subspace of the 3-dimensional trajectory data space defined by a Minimum Bounding Box (MBB) with six extreme values $(x_{\min}, x_{\max}, y_{\min}, y_{\max}, t_{\min}, t_{\max})$ encompassing spatial coordinates and timestamps. The fundamental innovation lies in its ability to partition each internal node along any dimension (x, y, or t) at any location based on data characteristics and historical query workload, rather than enforcing a rigid spatial-first or temporal-first hierarchy. The RL component uses Deep Q-Networks [201], where the state space comprises impact metrics (*QSred* and *SegInc* values for each dimension) combined with spatial embeddings of node boundaries, the action space contains three candidate cuts (one per dimension) that maximize the cost function, and rewards are computed based on I/O performance relative to a baseline partitioning approach. This state design, using percentage-based ratios in $[0, 1]$, enables training on small datasets (approximately 100,000 points) and subsequent application on large datasets (up to 800 million points). The construction of the underlying index uses a cost function $CF_i(cut) = (1 - \lambda) \cdot (1 - QSred_i(cut)) + \lambda \cdot SegInc_i(cut)$, where $QSred_i$ denotes the percentage reduction in data-aware query skew achieved by a cut along the dimension i, $SegInc_i$ represents the percentage increase in trajectory segments caused by that cut, and λ is a weighting parameter balancing these factors. Data-aware query skew is computed by first constructing n subranges along each dimension such that each subrange contains approximately equal data points, then clustering queries using DBSCAN based on their mass contribution vectors F_q across these subranges, which captures query dimensions, sizes, and spatial locations simultaneously. Leaf nodes store up to M trajectory segments on disk pages, and when partitioning cuts through existing trajectory segments, new smaller segments are created to ensure that each segment belongs to exactly one leaf node. Query processing follows standard tree traversal, in which nodes whose MBBs do not intersect the query region are pruned.

2.3 Contact Tracing Access Methods

Contact tracing access methods support queries that identify when and where moving entities came into proximity, a capability that became critical during the COVID-19 pandemic. These structures have to efficiently compute reachability across contact graphs, aggregate meeting durations between individuals, and visualize the temporal evolution of spatial interactions. The indexes in this category combine spatial partitioning with temporal aggregation to enable real-time or near-real-time analysis of contact events over large populations.

***θ*-structure** [40]: The $\theta-$structure enhances classical density maps with efficient querying capabilities for arbitrarily specified time windows, by encoding and retrieving spatial data which allows users to interactively restrict data to different time windows, thus revealing both short-term and long-term changes in the dataset. The proposed data structure is output-sensitive and achieves query times that are logarithmic in the total number of points and linear in the number of colored cells, which refers to the grid cells that meet the criteria specified in the time-window query, enabling real-time visualization of large point sets. It is also applicable for decision problems like counting, sum, and maximum of point weights. The θ-structure enables efficient data retrieval by organizing spatial points in a tree-based search structure, such as a Quadtree [100]. When a user performs a query specifying a time window, the structure efficiently narrows the relevant data using logarithmic search operations that hierarchically partition the spatial domain into smaller regions. Each cell or node in the tree is associated with a time function that enables rapid determination of whether a given cell contains events within the specified time window. Through experiments on real-world datasets,

such as bird migration and COVID-19 spread, the $\theta-$structure supports interactive exploration of large spatio-temporal data, with response times in the range of milliseconds and immediate feedback, making the system practical for real-time use. The θ-structure incorporates fractional cascading to optimize query time and reduce the time required to search across multiple related data structures. By storing additional pointers and pre-computed values, the θ-structure can quickly traverse its hierarchical nodes to locate and display relevant data for the query while minimizing repeated calculations. Therefore, the combination of hierarchical data partitioning, fractional cascading, precomputed time functions, and output-sensitive querying enables the θ-structure to deliver fast response times, allowing users to explore large spatio-temporal datasets interactively with minimal latency.

TRC-index [164, 165]: The Time Range Count index (TRC) is an indexing structure designed to efficiently support spatio-temporal Nearest Neighbor joins (ST-kNN) in Apache Spark [324] clusters, with each spatio-temporal partition maintaining its own TRC-index that is broadcast across processing nodes. The system incorporates Bayesian optimization and XGBoost [69] models for automatic parameter tuning. The index handles temporal data for objects with time ranges $[t_{min}, t_{max}]$ and can process trajectory data represented as line strings. The spatial component uses a hierarchical quad-tree [100] structure that recursively divides the global spatial domain into equal-sized sub-regions, with objects assigned to partitions based on their Minimum Bounding Rectangle (MBR) intersections, and builds upon the $3D$ R-tree [282] index where spatial coordinates form the first two dimensions and time serves as the third dimension. The purpose of the TRC-index is to help reduce the number of candidate objects to be considered in a join by providing an estimate of how many objects in a partition have overlapping time ranges with a query. It improves the efficiency of temporal filtering by giving a lower bound on the number of objects in a spatio-temporal partition that satisfy the temporal condition of the ST-kNN join. The TRC-index consists of two primary arrays: T_{min} and T_{max}, where T_{min} records the count of objects whose start time falls within a certain time bin, and T_{max} records the count of objects whose end time falls within a certain time bin. The temporal dimension is divided into discrete bins of equal length (controlled by parameter *binNum*), with the index using a sweep-line approach to accumulate counts where T_{min} is accumulated from right to left and T_{max} from left to right. The TRC-index supports temporal overlap queries to find objects whose time ranges intersect with a query time range $[t_{min}, t_{max}]$, using the notation $tr = [t_{min}, t_{max}]$ to compute a lower-bound on intersecting objects by calculating $|S_i| - T_{max}[b_{min}] - T_{min}[b_{max}]$, where b_{min} and b_{max} are bin numbers corresponding to $tr.t_{min}$ and $tr.t_{max}$ respectively. The TRC-index allows filtering objects by time, which is useful in distributed systems where minimizing data shuffling is critical.

RICCmeet [269]: RICCmeet's index constructs a spatio-temporal structure that handles reachability queries while constrained by transfer delays (time needed for actual information exchange during sustained contact) rather than just processing delays (time to process received information before retransmission). The contact graph represents direct object interactions within the threshold distance d_{cont} at each time instant, while the reachability graph links contact graphs over time through valid meetings of duration $m = \tau_f - \tau_s \geq m_q$, forming a directed graph that accounts for intermediary paths at which objects must maintain contact for minimum meeting duration μ. The index partitions the temporal domain into non-overlapping time blocks C and applies spatial grid partitioning with resolution H, creating a two-level hierarchy in which objects are organized by coarse grid cells (side H) and fine cells (side $d_{cc} = 2d_{max} + d_{cont}$). RICCmeetMin precomputes all μ-reachable objects while RICCmeetMax maintains tuples (O_j, m_{max}) indicating maximum meeting durations for each reachable object pair, using a modified depth-first search with a plane-sweep algorithm and priority queue ordered by earliest reach time $\tau_R(O_i)$. The technique can be represented as a three-level disk-based index: *Time Block Index* organizing reachability data by time blocks, *Meetings Index* mapping objects to meeting data pages, and *Reached Index* providing direct access to precomputed reachability

information, all maintaining spatial locality through cell-ordered storage. Query processing iteratively updates reachable objects over time instants and uses *Reached Index* to minimize disk accesses, first pruning unreachable targets, then traversing *Meetings Index* when detailed verification is required. The technique handles reachability queries with meetings $Q_{meet} : O_S, O_T, [\tau_s, \tau_f], m_q$ determining whether the target object O_T is (m_q)-reachable from the source object O_S during time interval I, supporting query variants including one-to-many and many-to-many through efficient boundary meeting management across time blocks for moving objects such as vehicles (100m Bluetooth contact) or people (3m physical contact).

e-Racoon [15]: *e-Racoon* presents a multi-level in-memory spatio-temporal access structure optimized for contact tracing queries through temporal aggregation joins. The structure's architecture uses a three-tiered approach: at its foundation lies a spatial grid[213] that partitions and indexes zones occupied by Moving Objects (MOs), where each spatial grid cell contains a second-level temporal indexing layer with configurable granularity (supporting day, half-day, or hourly divisions), and finally, each temporal bucket maintains a pointer to an Interval tree, implemented using Red-black trees for their self-balancing properties, storing precise time intervals of zone occupancy. The indexing mechanism processes Temporal Aggregation Join (TAJ) queries by efficiently matching two trajectories (T_l for local and T_q for query) to compute cumulative contact durations within specified spatial (ϵ meters) and temporal (W_t seconds) windows.

DkNN [331]: This paper introduces an indexing approach to handle Durable k-Nearest Neighbor ($DkNN$) queries over trajectory databases, which finds all trajectories that remain within the k-nearest neighbors of a query point p for at least r percent of a specified time window W. For a trajectory, o is a temporally ordered sequence of points $o = \langle (p_1, t_1), (p_2, t_2), \ldots, (p_n, t_n) \rangle$ where each $p_i = (x_i, y_i)$ represents spatial coordinates and t_i is the corresponding timestamp. The snapshot distance $d_t(o) = \|o(t) - p\|$ measures the Euclidean distance between the trajectory o and the query point p at time t. The durability $\rho_{k,W}(o) \stackrel{def}{=} \frac{|T_{k,W}(o)|}{|W|}$ quantifies the fraction of time interval W during which o belongs to the snapshot k-NN set $N_{k,t}$; and a $DkNN$ query $q = (p, k, W, r)$ returns all objects o where $\rho_{k,W}(o) \geq r$. The proposed technique augments a standard 3D R-tree [282], which indexes trajectory line segments $s_{ij} = \langle p_i, t_i; p_{i+1}, t_{i+1}; o_j \rangle$ within Minimum Bounding Rectangles $M = X_m \times Y_m \times T_m$, with auxiliary data structures attached to each MBR: a coverage function $C_m(t)$ implemented as a binary search tree that counts active line segments at any time t, and four envelope functions $x_{\min}(t), x_{\max}(t), y_{\min}(t), y_{\max}(t)$ that define temporal slices $M(t)$ providing tighter spatial bounds than the full MBR at each timestamp. The index uses a filter-and-refine framework where the filtering phase performs a prioritized breadth-first traversal, maintaining a working structure Δ containing an interval tree of entries $E = (d_{\min}(M), d_{\max}(M), C_m(t), I_m)$, applying pruning based on the lemma that an MBR M can be eliminated if for all $t \in T_m \cap W$ the accumulated coverage of visited MBRs with smaller maximum distances exceeds k. The refinement phase executes a sweep line algorithm over candidate segments, processing left endpoints, right endpoints, and curve intersections through an event queue Q, maintaining snapshot rankings in a BST L, and tracking durability accumulation in counter D, with early termination when $\max\{D(o) \mid D(o) < r\} + (1 - \rho) < r$ where ρ represents the swept fraction of W.

3 INDEXING THE CURRENT AND RECENT POSITIONS

This section surveys indexing methods optimized for querying the current and recent positions of moving objects, where real-time performance and low update latency take precedence over temporal evolution patterns. These structures address scenarios ranging from maintaining only current positions to preserving a sliding window of recent locations for

continuous query evaluation. Major challenges include handling high-frequency position updates, supporting continuous nearest-neighbor queries, and adapting to dynamically changing object distributions. The methods presented here use techniques such as hierarchical grids, LSM-tree architectures, and road-network-aware decompositions designed to balance update efficiency with query performance.

GeoPrune [317]: GeoPrune introduces a query processing technique for ride-sharing that utilizes geometric properties combined with R-trees [117] for efficient vehicle matching. The core mechanism uses two primary R-tree indices: T_{seg} to index the detour ellipses of trip schedule segments, and T_{end} to index the stop locations of vehicles. For each new trip request r_n, the system constructs a waiting circle ($r_n.wc$) centered at the source with a radius equal to the maximum waiting time multiplied by vehicle speed, and a detour ellipse ($r_n.rd$) with the source and destination as focal points where the length of the major axis equals the maximum allowed travel time multiplied by speed. The pruning process executes four critical queries: two point queries on T_{seg} to find segments whose detour ellipses contain the request's source and destination, and two range queries on T_{end} using the waiting circle and detour ellipse. The technique processes three cases: insert-insert (both the source and destination are inserted into schedule segments), insert-append (the source is inserted, the destination is appended), and append-append (both are appended). To efficiently update when vehicles move or receive new assignments, the system maintains dynamic ellipses that bound the reachable areas based on the maximum allowed travel times between schedule stops. These ellipses are indexed using minimum bounding rectangles (MBRs) in the R-trees for practical efficiency. The index supports vehicle movement updates in $O(|S|log(|S||C|) + |C|log^2|C|)$ time and new request updates in $O(|S|log^2(|S||C|))$ time, where $|S|$ is the maximum number of stops and $|C|$ is the vehicle count.

DVTG [271]: The DVTG-index (Dynamic V-tree Double-Layer Grid Index) handles continuous k-nearest neighbor (CkNN) queries on large road networks with moving objects using a query processing algorithm called DVTG-CkNN. The index supports two primary query types: kNN and CkNN. The kNN query, denoted as $k\text{NN}(q, k)$, finds k moving objects closest to the query point q at a specific time instance, where the distance is measured along the road network rather than the Euclidean distance. The CkNN query, formalized as $\text{C}k\text{NN}(q, k, \Delta t)$, continuously updates k-nearest moving objects to query point q as objects move over time intervals Δt, maintaining temporal consistency of results. The DVTG-index operates as a multilayer grid index, dividing the road network recursively into grid subgraphs utilizing a Quadtree [100] structure, followed by the construction of vertex subgraphs for each leaf node grid comprising of active vertices (linked with moving objects) and boundary vertices (connecting distinct grids). It maintains a vertex-distance matrix for each vertex subgraph, storing the shortest distances between active and boundary vertices, enabling dynamic grid merging or splitting based on the active vertex count to balance computational load. The index-building process involves a Quadtree-based division of the road network into layers, construction of vertex subgraphs for leaf-node grids containing only active and boundary vertices, parallel computation of shortest distances within each vertex subgraph, and dynamic grid adjustment based on the active vertex count. For index matching, upon addition or removal of a moving object, the index updates the corresponding leaf node grids, active vertices, and vertex distance matrix, with grid splitting or merging as needed. The DVTG-CkNN algorithm starts by associating the query point with its nearest vertex, then performs a Dijkstra-like [83] search on the elevated graph of the vertex, expanding from the query vertex to neighboring vertices, maintaining a candidate moving object queue F_q and distance threshold ζ (maximum distance to the kth object), terminating upon finding k nearest objects or exploring all reachable vertices, and incrementally expanding the search for continuous queries. The DVTG-index supports dynamic updates by mapping changes in moving objects to index updates by associating objects with active vertices, enabling real-time tracking as they move

along the road network. This approach efficiently handles position updates by only modifying the relevant parts of the index rather than completely rebuilding it.

LSM RUM-tree [256, 257]: The LSM RUM-tree is a spatial indexing structure designed to handle update-intensive spatial workloads. It combines the Log-Structured Merge (LSM) tree [16] with an R-tree [117] structure and incorporates an in-memory Update Memo (UM) [263, 307] component. The index comprises an in-memory R-tree for storing recent spatial data, multiple disk-based R-trees organized in levels like the LSM-tree structure, and the UM, which is a hash map tracking deletes and updates. Each object in the index is represented as a triplet $\langle Loc, O_{id}, ts \rangle$, where Loc is the location, O_{id} is the object identifier, and ts is a global timestamp. The UM entries take the form $\langle O_{id}, ts, cnt \rangle$, where ts denotes the most recent update timestamp and cnt tracks the number of obsolete versions in the index. During insert operations, new objects are added to the in-memory R-tree without modifying the UM. For delete and update operations, the UM is updated to reflect changes, and in the case of updates, new object versions are inserted into the in-memory R-tree. When the in-memory R-tree reaches capacity, it is flushed to disk as a new R-tree, with objects ordered using space-filling curves for efficient storage. Periodic merging of disk-based R-trees maintains the LSM structure. Query processing involves searching all R-trees and using the UM to validate result freshness. To optimize performance and manage UM size, the index employs four cleaning strategies: Buffered Cleaning tracks update frequency per node and cleans hot-spots when a threshold is reached; Vacuum Cleaning systematically processes cold-spot nodes based on a global counter to ensure fair cleaning distribution; Clean Upon Flush filters obsolete objects during the flush operation to avoid writing them to disk; and Clean Upon Merge removes obsolete objects during disk component merges. These strategies remove obsolete objects and reduce the UM size.

HDLT [6]: The Hierarchical Dense Location Time Index (HDLT-Index) is a spatio-temporal indexing technique designed for efficient processing of dense location queries on indoor tracking data of moving entities. The HDLT-index maintains a separate tree structure (HDLT-tree) for each indoor location, resembling a modified B$^+$-tree [31]. The leaf nodes of the HDLT-tree store time points together with aggregate information tuples $C_i = \langle c_{total}, c_{enter}, c_{exit} \rangle$, representing the total number of objects present, objects entering, and objects exiting at each time point, respectively. Non-leaf nodes contain time points and two types of aggregate tuples: $C_{nl} = \langle c'_{total}, c'_{enter} \rangle$ for all entries except the rightmost, and $C_r = \langle c''_{total}, c''_{enter} \rangle$ for the rightmost entry, encapsulating aggregate information for the subtree below. The index construction process involves pre-computing aggregate information for each distinct time point from mapping records, inserting time points into HDLT-trees using a B$^+$-tree-like insertion mechanism, computing C_{nl} and C_r tuples for non-leaf entries based on child node information, and linking nodes at intermediate levels. Query processing leverages this hierarchical structure, with point queries using a B$^+$-tree-like search to find leaf entries, interval queries summing c_{enter} values across relevant leaf entries, and dense location queries employing pruning strategies to avoid full tree traversal. These pruning techniques convert density thresholds into object-count thresholds and skip subtrees or terminate early based on aggregate counts. The HDLT-index supports count-based and duration-based density queries, extending temporal indexing concepts to symbolic indoor spaces. Although it offers significant performance improvements for queries over relational databases, the index has limitations, such as increased storage overhead due to separate trees per location and the need to recompute aggregates during updates.

MR-Cubes [77]: MR-Cubes presents a spatio-temporal indexing structure designed for efficient computation of *location popularity* from streaming check-in data. The index supports queries of the form $q = \langle box, r \rangle$, where box is the spatial bounding box of interest and r is the desired resolution level, returning a collection of cells at resolution r that intersect with the query region, along with their popularity values. *Location popularity* is a measurement that ranks locations based on their importance, expressed in three ways: *frequency*, *diversity* (Count of unique visits), or

location entropy (LE) (combination of frequency and diversity using Shannon entropy [254]). Key notation includes U_{c_i} as the set of distinct users who visited the cell c_i, $|v_{u_j}, c_i|$ as the number of visits to the cell c_i by the user u_j, and the entropy formula $H_{c_i} = -\sum_{u_j \in U_{c_i}} p_{u_j,c_i} \cdot \log p_{u_j,c_i}$ where p_{u_j,c_i} represents the probability of visit. The index uses a hierarchical quadtree-like structure, but with crucial augmentations to support multi-resolution popularity calculations. MR-Cubes divides the geographical area into a grid of fixed-size base cells, each represented by a leaf node in the index. These leaf nodes maintain specialized data structures, such as *Space Saving* summaries, to track user visits and compute approximate location entropy. As new check-ins arrive in the data stream, the corresponding leaf nodes are dynamically updated by incrementing user counters in hash tables or using the *Space Saving* algorithm to maintain the most frequent users while evicting low-frequency counters when memory limits are reached. For temporal decay of old data, MR-Cubes implements exponential time decay where new check-ins are weighted by $e^{\lambda t_i}$ and query-time values are computed as $|v_{u_j}, c_i| = cr_{u_j}^{double} * e^{-\lambda t}$, effectively decreasing the influence of older check-ins without explicit deletion. The index's novelty lies in its on-the-fly construction of intermediate nodes that represent different spatial resolutions. These nodes are created during query execution by merging child nodes, supporting flexible multi-resolution analysis. Query processing in MR-Cubes follows a three-phase approach: SCAN to traverse the index and identify relevant nodes, MERGE to combine nodes at the desired resolution, and CALCULATE to compute final popularity values. The MERGE phase uses one of three strategies: Top-Down, Bottom-Up, or Hybrid, each offering different trade-offs between performance and accuracy. To address the challenges of real-time processing and scalability, MR-Cubes introduces several approximation heuristics, including *TopKF*, *TopKUE*, and *TopKF+UEC*, which intelligently balance memory usage, update speed, and result accuracy. These heuristics adopt techniques from streaming algorithms to maintain bounded-size summaries of user activities, enabling efficient merging operations across different resolutions.

SSQ [12, 14]: The Spatial-Social Quadtree (*SSQ*) supports real-time personalized geo-social queries over streaming data, specifically handling Spatial-Social Temporal Range Query (*SSTRQ*) and Spatial-Social Temporal kNN Query (*SSTkQ*). The *SSTRQ* retrieves the most recent k objects within a spatial range posted by a user's friends or friends-of-friends, while *SSTkQ* finds the top-k closest objects based on spatio-temporal distance. Its framework handles the high volume and velocity of geo-social data that combines temporal, spatial, and social aspects. Therefore, this indexing framework is designed as follows: an in-memory index, an in-disk index, and an in-memory buffer. While the in-memory index is used to ingest highly dynamic geo-social data in real time, it stores recent geo-social objects in memory for fast access, focusing on objects that arrive in a continuous stream. The in-disk index stores relatively stable data, specifically the social graph, which does not change frequently. This data is used for efficient query processing when required and is optimized for low-overhead storage and retrieval. On the other hand, the in-memory buffer caches frequently accessed social graph data in memory, which reduces the number of disk reads when retrieving social information, such as friends or friends-of-friends, for the queries. The Spatial-Social Quadtree (*SSQ*) index uses a Quadtree [100] to partition the spatial domain and includes social information in the leaf nodes. The Quadtree organizes geo-social objects in the in-memory index based on their spatial location. Each leaf node holds a hash structure that organizes objects by the posting user (notation: uid represents the user identifier, loc represents spatial coordinates, $keywords$ represents textual content, and $timestamp$ represents posting time), allowing efficient access to data for social pruning of unnecessary objects during query processing by filtering based on the user's social connections. In addition, the index supports three-dimensional pruning during query processing: spatial retrieval, social filtering, and temporal pruning, which improve query performance by narrowing the search space to only relevant objects based on the user's friends, recent posts, and nearby locations.

SIMkNN [48]: SIMkNN is an in-memory spatial indexing and query processing approach for k-nearest neighbor (kNN) searches on moving objects in road networks. SIMkNN uses a dual index structure: a hierarchical grid (H-grid) [322] to manage moving objects and an R-tree [117] to store the topology of the road network. The H-grid adapts dynamically to skewed object distributions by starting with a uniform grid and recursively splitting cells that exceed a capacity threshold N_c into $m \times m$ smaller sub-cells, while merging cells when occupancy falls below N_c. This adaptive structure results in smaller cells in dense areas and larger cells in sparse regions, optimizing space utilization and query performance. A key component is the "cell communication" technique, which maintains up-to-date neighbor information for each cell in eight directions, facilitating efficient updates during cell splits and merges. kNN query processing in SIMkNN occurs in two phases: result space estimation and kNN retrieval. The estimation phase begins at the query point's cell and expands the search space incrementally based on the smallest neighbor cell size until k objects are found using the Euclidean distance. This estimated area is then used to extract a partial road network from the R-tree for the retrieval phase, where the network expansion computes actual network distances to refine results. This approach allows for gradual expansion that adapts to data distribution and provides tight bounds for network search. SIMkNN supports dynamic updates through cell splits/merges and object movements between cells, making it suitable for frequently changing object locations. The index structure is more complex and memory-intensive than uniform grids. However, it offers improved query performance, especially for skewed distributions. The efficiency of the technique depends on properly adjusting parameters such as cell capacity N_c and split factor m.

TD-H2H [161]: TD-H2H (Time-Dependent Highway to Highway) is an indexing technique that extends the H2H [221] labeling framework to efficiently process k-nearest neighbor queries on time-dependent road networks, where edge weights are functions of time rather than static values. The index structure is built on a function-preserved tree decomposition (FPTD) of the road network, which hierarchically decomposes the network while preserving time-dependent weight functions during vertex elimination. FPTD works by creating a rooted tree where each node contains a subset of vertices, and when eliminating a vertex v, it preserves path information by either adding new edges between v's neighbors with aggregated weight functions $f(u \rightarrow w) = f(u,v) \bigoplus f(v,w)$ or updating existing edges with minimized functions $f(u \rightarrow w) = Min\{f(u,w), f(u,v) \bigoplus f(v,w)\}$. For each vertex v, the index maintains two critical function arrays: $X(v)^o.f$ storing weight functions from v to its ancestors, and $X(v)^i.f$ containing functions from ancestors to v, along with position arrays $X(v).pos$ tracking vertex positions in ancestor lists. The index construction proceeds in a top-down manner through the decomposition tree, computing and storing aggregated weight functions for each vertex and its ancestors using a Min operation on weight functions. TD-H2H handles time-dependent costs through piecewise linear functions, where function aggregation is performed during index construction to pre-compute travel costs. For query processing, the index uses the lowest common ancestor (LCA) concept: when computing the time-dependent fastest travel cost between vertices s and d, it identifies their LCA in the decomposition tree and uses precomputed function arrays to efficiently compute the minimal travel cost. The index incorporates two pruning strategies: lower-bound cost pruning using static costs and label pruning that reuses common-ancestor computations.

4 INDEXING THE FUTURE

This section examines indexing methods that manage predicted or planned future positions of moving objects, enabling proactive query processing over anticipated trajectories. Unlike historical indexes that operate on recorded data, future-oriented structures must handle inherent uncertainty in position predictions while supporting time-sensitive applications, e.g., traffic management and collision avoidance. These methods integrate predictive models, traffic-aware calibration, and sliding window mechanisms to maintain accurate representations of expected object locations. The

indexes presented here address challenges such as dynamic route updates, trajectory uncertainty measurement, and real-time conflict detection in domains ranging from road networks to airspace management.

RR-index [319]: The RR-index (Route Record index) is a spatio-temporal index structure designed to manage future route trajectories with a focus on traffic-aware time calibration. The index maintains route records as triplets $\langle r, in/out, f \rangle$ where r represents the route ID, in/out indicates entry/exit status, and f tracks traffic flow modifications (+1 for entry, -1 for exit). The temporal dimension is organized through a sliding window mechanism that partitions time into fixed-length slices, with each slice containing route records structured within a Binary Search Tree (BST) [150] for temporal querying. The index uses a two-tier organization: the upper tier manages time slices within a sliding window (4-6 hours into the future), while the lower tier maintains route records ordered by timestamps within each slice. Spatially, records are organized by road segments, with every segment maintaining sets of timestamps corresponding to traffic flow changes along with their associated route records. The index implements sophisticated update mechanisms employing lazy updates, a strategy that postpones actual temporal updates until necessary to minimize computational overhead. For route insertions, the index processes records road-by-road, updating traffic flows and recalculating travel times for affected routes using a Bureau of Public Roads (BPR) [168] function that accounts for road capacity and current traffic conditions. The index supports dynamic updates through a propagation mechanism that identifies affected routes using a termination condition and updates their temporal information accordingly. Additionally, "road-oriented update operation" processes multiple route updates on the same road segment collectively, reducing redundant computations. Query processing uses BST organization within time slices to retrieve affected routes and update their temporal information.

4.1 AI/ML-Powered

Machine Learning (ML, for short) techniques offer powerful capabilities for predicting future trajectories and enhancing index performance through learned models of movement patterns. The method in this subsection integrates probabilistic models, e.g., Hidden Markov Models, with spatial indexing structures, to support both trajectory prediction and conflict resolution. These hybrid approaches leverage historical movement data to inform predictions while maintaining efficient spatial query capabilities for real-time decision support.

OCTtrees for Aircraft Conflict Detection and Resolution (CDR) [26, 27]: CDR is a framework for a prescriptive analytics system for long-range aircraft conflict detection and resolution. It is an advanced decision-support tool developed to address critical safety and efficiency challenges in modern air traffic management (ATM). The integrated indexing system for aircraft conflict detection and resolution combines two complementary indexing mechanisms: Octree-based [136] spatial-temporal indexing for conflict detection and Hidden Markov Model state-space indexing for trajectory prediction and resolution. The Octree component uses temporal partitioning where each discrete time instance t maps to a dedicated Octree data structure, creating time-indexed spatial trees that manage a four-dimensional problem space (latitude, longitude, altitude, time) using hierarchical tree traversal with $O(1)$ temporal lookup followed by $O(\log n)$ spatial traversal. Aircraft protected zones are represented as expandable cube formations composed of atomic 4D spatio-temporal data cubes, identified by centroid coordinates, that support precise conflict detection through dynamic constraint lists CL_{PZ} containing existing trajectory reservations with automatic temporal expiration mechanisms. Simultaneously, the HMM indexing system maintains state-space representations in which N states, $S = S_1, S_2, \ldots, S_N$, correspond to reference point coordinates forming aligned trajectories on a 3D grid network with 6km × 6km spatial resolution, which processes 4D trajectories. The HMM framework indexes transition probability distributions $A = a_{ij}$ representing aircraft movement patterns between states, observation symbol probability distributions $B = b_j(k)$

mapping weather parameters to trajectory segments, and conflict-free probability distributions $C = c_j(k)$ indicating spatial regions free from conflicts. State transitions are indexed via probability matrices, enabling efficient Viterbi algorithm traversal for optimal path computation. Conflict resolution combines both indexing systems by assigning minimal conflict-free probabilities (1×10^{-100}) to Octree-occupied cubes while maintaining HMM state accessibility for alternative route generation. The combined system achieves $O(k \times \log n)$ complexity for conflict detection through Octree queries and $O(k(2 \log n + |N| - 1) + N)$ for HMM-based resolution, where k represents trajectory segments, n denotes spatial occupancy, and N indicates HMM state space cardinality. This dual-indexing approach enables real-time strategic conflict avoidance by efficiently managing both spatial constraints via Octree structures and probabilistic trajectory optimization via HMM state indexing, supporting scalable multi-aircraft conflict detection and resolution in complex airspace environments.

5 INDEXING THE PAST, PRESENT, AND THE FUTURE POSITIONS

This section presents unified indexing structures capable of managing spatio-temporal data across all temporal dimensions simultaneously. The methods maintain historical trajectories, track current positions, and support predictive queries within a single coherent framework. Such integrated approaches eliminate the need for separate index structures while enabling complex queries that span multiple time horizons.

PASTIS [238]: PASTIS (PArallel Spatio-Temporal Indexing System) is an in-memory spatio-temporal indexing structure which uses a versioned grid, dividing the spatial domain into regular cells ordered by Z-curve [220]. Each cell maintains partial temporal indexes comprising an interval lookup table *Itab*, compressed bitmaps *CBmap* identifying objects present during specific time intervals, and hashmaps *Hm-RIDList* linking objects to their record IDs. PASTIS supports three main types of range queries: historical queries (e.g., "Find all objects in region R between timestamps t_1 and t_2"), present queries (e.g., "Find all objects currently in region R"), and predictive queries (e.g., "Find all objects expected to be in region R at future time t_f"). The query window is defined by a spatial extent *QRect* and temporal bounds [*StartDS, EndDS*]. For predictive capabilities, PASTIS maintains two specialized structures: a hashmap *PHm* storing predicted ⟨latitude, longitude⟩ tuples for each object, and an array of hashmaps *ArHm* mapping grid cells to compressed bitmaps that encode predicted object status for a configurable future interval F. Object positions are predicted using linear interpolation based on current location and velocity, though the system supports pluggable prediction models. The index building process involves computing the grid cell ID for each location update, updating the corresponding temporal index structures, and modifying the prediction data as needed. PASTIS employs atomic operations and fine-grained locking to enable concurrent updates, along with load-balancing techniques to distribute updates across threads. Query processing exploits multi-threaded execution and in-memory bitmap operations for rapid results. Historical queries use bitwise OR operations on bitmaps to identify relevant objects, with additional checks on partially covered cells. Present queries focus on the most recent time interval, while predictive queries utilize the dedicated prediction structures. Its design prioritizes insert efficiency and query speed in 2D space.

6 SPATIO-TEMPORAL-TEXTUAL INDEXING

This section examines indexing methods that jointly manage spatial, temporal, and textual dimensions, enabling queries that combine location constraints with keyword matching and time-based filtering. These multi-dimensional indexes support applications such as location-based social networks, geo-tagged document retrieval, and streaming content analysis, where all three dimensions are essential for relevance ranking. The methods presented here employ diverse strategies, including interleaved tries, hybrid tree structures,and grid-based partitioning, designed to balance the

competing demands of spatial proximity, textual similarity, and temporal recency. Main challenges include efficiently pruning across heterogeneous dimensions and supporting continuous queries over high-velocity data streams.

STILT [22]: STILT (Spatio-temporal Textual InterLeaved Trie) presents an indexing approach built upon a binary Patricia trie [203] structure that fundamentally interleaves bits from spatial, temporal, and textual components according to predefined integers of which the bits are concatenated to produce the interleaved bit-string called path schedules. The index can handle range queries (finding all documents within specified spatial boundaries, time ranges, and containing certain keywords) and top-k queries (retrieving the k highest-ranked documents based on spatial proximity, temporal recency, and textual relevance). The index uses a hierarchical node structure with varying capacities ($Node0$, $Node8$, $Node16$, $Node32$, $Node64$) where interior nodes store common bit prefixes and leaf nodes maintain document keys (id, location, word, timestamp). The core indexing mechanism converts indexable components into integers using dimension-specific mapping functions: linear mapping for spatial and temporal dimensions, and a case-insensitive hash function for textual components. The integers are then bit-interleaved according to a path schedule that dictates the concatenation order, producing an interleaved bit string that determines the path through the trie. The index supports three variants: STILT$_{stx}$ for full spatio-temporal-textual queries (64-bit paths), STILT$_{sx}$ for spatio-textual queries (63-bit paths), and STILT$_{st}$ for spatio-temporal queries (63-bit paths). To optimize storage efficiency, STILT implements a deduplication strategy using *SlimCommon* objects for repeating fields and *SlimKey* objects for unique components, significantly reducing the memory footprint. The index construction process is parallelizable, supporting concurrent insertions through a progressive building mechanism. Query processing uses a unified ranking scheme that combines spatial scores (based on distance), temporal scores (based on recency), and textual scores (using cosine similarity) for top-k queries, while range queries utilize efficient pruning based on the interleaved bit patterns. The index structure achieves efficient pruning through its binary trie organization, where each node's path represents a specific region in the multi-dimensional space.

SST/RST [64]: This paper introduces a hybrid approach to organizing and processing geo-textual data-stream subscriptions over spatio-temporal documents. The system handles two types of continuous top-k term queries: SST subscriptions that find locally popular terms based on spatial proximity, frequency, and recency without requiring users to specify regions; and RST subscriptions that maintain trending terms within user-defined spatial regions. The solution combines multiple indexing components: a Grid [213]/Quadtree [100] based spatial organization for subscription regions, inverted files for term-based lookups, and min-heaps for maintaining top-k results. For SST (Similarity-based Spatial-Temporal Term) subscriptions, the system uses a hierarchical summarization structure where subscriptions are recursively subdivided into quadrants until each cell contains at most M subscriptions (where M is a system parameter controlling granularity). Each cell maintains a summary label derived from the min-heap elements of contained subscriptions, enabling efficient group-based filtering. The RST (Region-based Spatial-Temporal Term) subscriptions are organized using a Temporal Popularity (TP) index that consists of a hashmap for term-score pairs and a min-heap for current top-k terms. The system implements a Group-Constrained Online Clustering (GCOC) algorithm for document grouping, where documents are clustered based on spatial proximity with a constraint on the maximum diagonal length of the cluster's MBR (Minimum Bounding Rectangle). For efficient score computation, the system maintains aggregate partial LTP scores for document groups, where LTP represents the Local Term Popularity combining frequency, spatial proximity, and temporal decay rate. The subscription matching process uses these index structures using a filtering-and-refinement approach: first, using spatial indexes to identify potentially affected subscriptions; then applying group-based filtering using summary labels; and finally performing detailed score computations only for

the remaining candidates. Document updates are handled through lazy removal strategies with error bounds, while subscription updates trigger incremental maintenance of the hierarchical organization.

TA-tree [58, 59]: The TA-tree is a spatio-temporal-textual indexing structure that efficiently handles Time-Aware Boolean Spatial Keyword Queries (TABSKQ). TABSKQ retrieves k objects that contain all query keywords, which are within a specified search radius, and maximizes both temporal overlap with a query time interval and spatial proximity to a query location. It extends the CIR-tree [76], combining elements of the vEB-tree [78] and signature files for temporal indexing and signature files for textual filtering. The index categorizes objects based on keyword frequency, with the f most frequent keywords defining $f + 1$ categories. The objects in each category are then assigned to different time periods within the tree structure. The TA-tree consists of a root node containing metadata, non-leaf nodes with pointers to child nodes and temporal range information, and leaf nodes storing object data and signature files. The key innovation is the computation of a "keyword sensitive valid time" for each object, which combines both temporal and textual information into a single dimension. This allows for simultaneous filtering on both time and keywords during query processing. Index construction proceeds top-down, with objects inserted based on their keyword-sensitive valid time. The tree dynamically updates the time ranges of non-leaf nodes during insertion. Query processing leverages this structure by computing a keyword-sensitive query interval and traversing the tree to find matching time ranges. Signature files in leaf nodes enable efficient textual filtering, followed by spatial refinement of candidates. The TA-tree supports both on-disk and in-memory implementations and allows dynamic updates through insert and delete operations. Although it offers superior query performance for frequent keywords, it incurs higher index construction costs and increased storage overhead due to object replication across categories. The index particularly excels in scenarios with skewed keyword distributions and time-constrained queries.

DLEEL [11]: DLEEL presents a system that adapts and extends existing spatial indexing techniques to efficiently handle multi-predicate spatial queries on high-velocity streaming data. The core innovation lies in its hybrid indexing approach, which combines in-memory and disk-based components to balance update speed, memory consumption, and query latency. *DLEEL* supports four query types: (1) Spatial-social Temporal Range Query (*SSTRQ*) ⟨spatial range R, keywords W, integer k, user u⟩ retrieving k recent records within R containing W posted by u's social connections; (2) Spatial-social Temporal kNN Query (*SSTkQ*) ⟨spatial location L, keywords W, integer k, timestamp T, user u⟩ retrieving top-k records containing W from u's social network ranked by spatio-temporal distance; (3) Spatial-keyword Temporal Range Query (*SKTRQ*) ⟨spatial range R, keywords W, integer k⟩ retrieving k recent records within R containing W; and (4) Spatial-keyword Temporal kNN Query (*SKTkQ*) ⟨spatial location L, keywords W, integer k, timestamp T⟩ retrieving top-k records containing W ranked by spatio-temporal distance. For spatial-social queries, *DLEEL* employs an in-memory spatial Quadtree [100] where each node contains a hash index organizing data by user, coupled with a disk-based social graph stored as friend lists and an in-memory buffer for caching recent social information. This structure allows for rapid ingestion of streaming spatial data while maintaining efficient access to relatively stable social network information. For spatial-keyword queries, *DLEEL* builds upon traditional spatial-keyword indexes (combinations of spatial indexes such as Quadtrees [100], R-trees [117], and grids [213] with inverted indexes) by incorporating streaming-friendly components and lightweight batch insertion techniques. The system's query processing uses these index structures to efficiently prune the search space across multiple dimensions simultaneously. It begins by using the spatial index to identify relevant areas, then refines the results using temporal, keyword, and social attributes as applicable. *DLEEL* employs an incremental pruning strategy, initially retrieving k results and progressively tightening bounds to eliminate irrelevant records. This approach allows for efficient processing of complex queries with millisecond-level latency while ingesting hundreds of thousands of records per second. The system's architecture

is designed to handle the dynamic nature of streaming data, with all indexing components except the social graph residing in main memory for rapid updates and query processing.

PCR-tree [200]: The PCR-tree (Personalized and Context-Aware R-tree) enriches standard R-tree [117] nodes with bitmap-based metadata structures to allow selective node traversal during spatial search for Points of Interest (POIs). The PCR-tree supports queries comprising a spatial query window Q_S, a list of category and sub-category preferences, and a keyword vector with respective weights representing user context. Each non-leaf node is structured as $\{\text{ptr}, \text{mbr}, B_{\text{cat}}\}$, where ptr denotes a pointer to a child node, mbr represents the Minimum Bounding Rectangle enclosing all child MBRs, and B_{cat} is a fixed-length category bitmap indicating the presence (1) or absence (0) of POI categories within the subtree rooted at that node. The category bitmap is computed through bitwise OR operations across all descendant nodes, formally expressed as $B_{\text{cat}}(N) = \bigcup B_{\text{cat}}(N_i)$ for all $N_i \in T(N)$, where $T(N)$ represents the subtree rooted at node N. Each active bit in B_{cat} points to a corresponding sub-category bitmap B_{subcat}, which provides finer-grained filtering by indicating which sub-categories exist within a category. These sub-category bitmap entries further point to sorted linked lists containing top-K POIs for that sub-category within the region, where each linked list entry stores: PID (unique POI identifier), (x, y) GPS coordinates, **K** (a context-keyword vector containing weighted terms representing situational relevance factors), **D** (POI descriptions and metadata), and ptr_s (pointer to sequentially visited POIs). Context matching between POI keyword vectors and query keyword vectors employs cosine similarity to compute matching scores MS_{PQ}, enabling semantic ranking beyond spatial proximity. For spatio-temporal event indexing, the sub-category bitmaps are replaced with time-slot buckets that partition time into discrete intervals and link to event linked-lists containing temporal metadata such as duration and recurrence patterns. During search, when a spatial query window Q_S completely overlaps an internal node's MBR, the index retrieves POIs directly from that node's linked lists without traversing to leaf nodes, significantly reducing disk I/O; partial overlaps trigger recursive descent with bitmap-guided pruning of irrelevant subtrees.

6.1 Current and Recent Location Indexing

This subsection presents indexes optimized for querying the current and recent positions of objects that carry both spatial coordinates and textual attributes. These structures support continuous top-k queries, range queries, and subscription-based monitoring, where results must be updated as objects move and their keywords change. The methods use grid-based partitioning with auxiliary structures, e.g., safe regions and cell links, to minimize recomputation when objects update.

PCL [88]: This paper presents a query processing technique for continuously searching dynamic spatial keyword objects, using a grid [213]-based index coupled with a buffer called the Partial Cell List (`PCL`). The grid-based index manages both dynamic objects and queries, with objects indexed in cells based on their location, and queries indexed in cells overlapping their "influential circle." Each cell maintains maximum and minimum keyword weights to efficiently bound similarity scores. Objects are defined as $o = (o.\rho, o.\psi, t)$, where $o.\rho$ represents location coordinates, $o.\psi$ is a set of keywords, and t is the timestamp. Both location and keywords change over time, with the system tracking the m most recent states of each object. Queries are defined as $q = (q.\rho, q.\psi, q.k, q.\alpha)$ where $q.\rho$ is the location, $q.\psi$ is the set of keywords, $q.k$ specifies the number of results needed, and $q.\alpha$ is a smoothing parameter that balances the importance of spatial and textual similarity. The `PCL` buffer, a subset of relevant cells for each query, is designed to expedite top-k result updates. It contains cells with $minscore < PCL.up$ and $maxscore > PCL.low$, where $PCL.up$ represents the current kth score, and $PCL.low$ is a lower-bound. The query processing mechanism handles object changes by identifying affected queries (`OutQ` and `InQ`) and updating the top-k results accordingly. The system uses a `PCL` maintenance strategy,

checking if changed cells need addition or removal from the PCL, trimming the PCL when the score ranges change, and recreating it if it becomes empty. The top-k refiller module efficiently updates results using the PCL, using different strategies for various cases (L2L, S2L, L2S, S2S) (L2L (large to large), S2L (small to large), L2S (large to small), and S2S (small to small)) based on how object changes impact queries. For example, in the L2S case, it searches for the top-1 object from PCL to potentially refill the top-k list. The approach builds on concepts from continuous top-k queries and leverages grid-based spatial indexing. The technique's ability to handle dynamic updates through its grid structure and PCL maintenance strategies makes it particularly suited for applications involving frequently changing spatial keyword data, such as location-based services and real-time social media analysis.

CCLS-CL [89]: This paper introduces a hybrid indexing approach for continuous top-k spatial-keyword queries on dynamic objects, combining a grid [213] based spatial index with auxiliary data structures. The objects can change their location and keywords at any time. The core spatial indexing uses a uniform grid partitioning where each cell maintains two critical components: cell-cell links (*CC links*) that form a directed graph between cells based on score upper bounds, and l-signatures that represent combinations of up to l keywords to optimize keyword-based pruning. The grid structure maintains both objects and queries, where each cell c stores distinct keywords $c^t.\psi$ appearing in its objects, with $c^t.\psi_{max}$ and $c^t.\psi_{min}$ representing the same keyword set but weighted by the maximum and minimum weights, respectively, among all objects in the cell containing each keyword. The *CC links* are constructed by computing score upper bounds between cells using both spatial and keyword similarities, and by creating directed edges from an object's cell to cells that might contain queries in Q_{next} based on spatial proximity. For keyword indexing, the system builds indices on l-signatures (keyword combinations) to reduce posting list traversal overhead. These l-signatures are selected using a greedy approximation algorithm that minimizes the total cost of posting list access while ensuring complete coverage of potential variants. Additionally, each query maintains a cell list (*CL*) that tracks cells guaranteed to contain its $(k+1^{th})$ result, using cell-based score bounds (`maxscore` and `minscore`) to minimize unnecessary cell access during result maintenance. The index supports dynamic updates via a lazy update strategy for *CC links* and cell lists, modifying structures only when score bounds change significantly. The memory-resident index achieves efficient query processing by avoiding expensive Quadtree [100] traversals via direct cell access via *CC links*, reducing inverted index lookups via l-signatures, and minimizing result maintenance overhead via bounded cell lists.

Lagic [219]: Lagic is a parallelized in-memory layered grid [213] structure designed for continuous spatial-keyword range queries over moving objects. These queries continuously monitor moving objects to find those whose combined spatial proximity and textual relevance scores fall below a specified threshold, returning all qualifying objects rather than a fixed number. The query follows the form $q = (q.\lambda, q.\psi, q.T, q.\alpha, q.R)$, where $q.\lambda$ represents the query location, $q.\psi$ contains the query keywords, $q.T$ is the score threshold, $q.\alpha$ balances spatial versus textual importance, and $q.R$ defines the spatial impact radius. The index's architecture comprises multiple independent layers, where each layer corresponds to a unique moving object, enabling parallel processing across different processors. Each object's layer maintains the object's primary information along with a grid index that handles frequent location updates, efficiently retrieves information about the object's Short-term Safe Region (SSR), and tracks the Conditional Circles (CCs) that can be affected by location updates. The grid index within each layer partitions the space into cells that serve as Buffer Regions (BRs), which are fundamental to localizing required computations. Each cell contains information about the affected CCs, while the SSR - a circular region centered at the object's current location with a radius determined by the minimum distance to any CC's circumference - provides a filtering mechanism before checking individual CCs. The index employs two key optimization techniques: SSRs reduce processing cost by requiring only one circle to be monitored instead of all CCs at every timestamp, while BRs minimize the number of comparisons needed when an

object moves by limiting the scope of affected queries to those whose CCs intersect the current BR. Complementing the layered structure is an auxiliary grid index exclusively for object locations and SSRs, which optimizes access patterns and enables lazy updates to the main layered grid index. The index processes updates through a dual-phase approach: initialization and continuous monitoring. During initialization, it computes textual thresholds for pruning irrelevant queries, constructs the auxiliary grid index, creates index layers by computing Safe Regions and SSRs, and builds initial answer sets. In the continuous monitoring phase, when an object updates its location, the index first updates the indices with the new location, checks if the object left its BR, and based on this check, either reconstructs the SSR and examines CCs in both old and new BRs. If the object remains within its SSR, no updates are needed; if it moves outside, a new SSR is constructed, and CCs in the current BR are checked.

SkyEye [218]: SkyEye uses a grid [213] based spatial indexing structure combined with in-memory indices to support moving continuous top-k spatial-keyword queries over moving objects (MTSKQ-MO) in directed street networks. The spatial index partitions the region into a grid, where each cell maintains information about objects/queries within its boundaries and stores precomputed shortest-path distances using the Dijkstra [83] algorithm. The index structure uses a two-layer architecture: the edge node layer for mobile devices and the cloud layer for centralized processing. The grid index maps object/query locations to cells and maintains dynamic updates as entities move between cells, with experimental results suggesting an optimal grid granularity of 10×10 cells balancing query processing efficiency with maintenance overhead. The index supports two critical spatial pruning regions: C_r (initial pruning circle with radius $r = q.R + len_{edge_q} + \Delta X_{max}; (q = Query; \Delta X_{max} = Max\ speed \times time)$) and C_a (refined pruning circle based on answer set objects), which are used to identify spatially relevant objects. Index maintenance is triggered when objects cross cell boundaries, requiring updates to both spatial and auxiliary indices. The auxiliary indices in the cloud layer store additional information, including query-pruning bounds, impacted objects, and textual distances, to reduce computational overhead. The index structure supports continuous updates through a "safe interval" approach, where answer sets remain valid for calculated time intervals rather than requiring updates at every timestamp. An optimization is to maintain the shortest path distances indexed by grid cells, eliminating the need for frequent path computations. The structure builds on traditional grid indexing by incorporating temporal aspects and textual similarity measures, although its performance is particularly sensitive to object movement speeds and grid granularity.

Lamps [214, 215]: The LAMPS system introduces the SQ-tree and *IQF* (*I*nverted file, and *Q*uad-tree [100] for each *F*requent keyword) indexing structures for processing moving top-k spatio-textual subscriptions. The SQ-tree extends traditional Quadtree architecture by incorporating safe regions and maintaining subscriptions at both leaf and non-leaf levels. Each SQ-tree node n stores: S_n (a subscription set), I_n (an inverted file for subscription keywords), and $\Lambda_n[\cdot]$ (spatial proximity thresholds for filtering). Non-leaf nodes, rooted at n additionally maintain T_n^r (keyword sets) and $\Lambda_n^r[\cdot]$ (thresholds) for their entire subtrees. The SQ-tree construction process uses a probability-based metric $pnum(s, n)$ that determines subscription placement by calculating the expected number of common keywords requiring result updates. This metric considers both $kscore_{ub}(s)$ (upper-bound score) and $E[i_n]$ (expected keyword intersection). The *IQF* structure combines a global inverted file with specialized Quadtrees (Q_w) for frequent keywords, where the frequency is determined by a configurable threshold x. For query processing, the system handles three primary operations: object generation, object expiration, and subscription movement. During object generation, the SQ-tree employs a two-step filtering process: node filtering using $\Lambda_n^r[i_n]$ thresholds and subscription filtering using $\Lambda_n[i_{min}]$ thresholds. For object expiration and subscription movement, the *IQF* facilitates efficient retrieval of new k-th and $(k+1)$-th objects through a score-based search strategy. Safe regions are computed using approximate techniques that consider spatial proximity and textual-similarity bounds. The system maintains these regions dynamically, updating them when subscriptions move or

when top-k results change due to object updates. The index processes continuous monitoring queries by evaluating spatial proximity thresholds $\lambda(s, o|i)$ and textual similarity bounds $text_{lb}(s.t, o.t|i)$ to determine result updates, while maintaining approximate safe regions to reduce computational overhead in continuous monitoring scenarios.

PNLR-tree [223]: The PNLR-tree (Positive Negative List R-tree) is a spatio-temporal-textual indexing technique designed for efficient processing of Multi-Keyword Spatial Boolean Queries (MKBSQ) on streaming data. It combines an inverted file structure with an R-tree [117] to handle both textual and spatial components effectively. At its core is a hashmap that maps each keyword to its own R-tree, enabling rapid access to spatial data associated with specific terms. This structure allows efficient pruning of the search space based on both textual and spatial constraints. The index supports a sliding window model for streaming data, continuously updating to retain only the most recent records. During index construction, incoming data points are processed by extracting their keywords, locating or creating corresponding R-trees in the hashmap, and inserting the spatial information into these trees. The PNLR-tree supports both single and batch insertions, with the latter grouping records by keywords for improved efficiency. Query processing leverages this structure by first identifying relevant R-trees through the hashmap based on positive keywords, then applying spatial filtering within these trees, and finally refining results by excluding objects containing negative keywords. This approach enables efficient handling of complex boolean queries, including 'But-Not' and 'XOR' operations. The index also incorporates an update mechanism that marks and removes expired objects during query processing, ensuring data freshness without excessive overhead. This design choice reflects a trade-off between update efficiency and memory usage, optimized for high-velocity streaming environments where rapid data ingestion and query processing are crucial.

QT-tree [71]: The QuadText-tree (QT-tree) extends the Quadtree [100] and serves as the primary index of the Streaming Spatio-Textual Data (SSTD) system. SSTD is a distributed in-memory spatio-textual streaming system for answering continuous and snapshot queries with spatial, textual, and temporal constraints over data streams. The main idea of the QT-tree is to split an index node based on the spatial or textual properties of the indexed objects to minimize the total workload, i.e., the number of retrieved objects needed to answer a query. In this regard, each QT-tree node maintains a set of statistics that include the probability distribution of the supported queries, the probability distribution of the query keywords, and the number of incoming objects within the node's region. In SSTD, the QT-tree is built offline on a set of historical queries and objects. The QT-tree node split algorithm follows two partitioning schemes. The *spatial* partitioning scheme follows the traditional Quadtree split algorithm, while the *textual* partitioning scheme groups objects with similar textual attributes into the same child node. Unlike the spatial scheme, the MBR of the child nodes in the textual scheme is the same as the parent MBR. During the split, the QT-tree selects the partitioning scheme that minimizes the total workload using a custom cost function. The nodes in the QT-tree are partitioned until the number of leaf nodes reaches a certain threshold.

ST^2 [65]: ST^2 (Top-k Spatial-Temporal Term Subscription) is a location-based term publish/subscribe system that continually updates the top-k most popular terms across multiple subscriptions within a stream of spatio-temporal documents. Term popularity is determined by both frequency and recency, calculated as $TP(s, w, t) = \sum_{d \in s.r} F(d, w) \times D^{-(t-d.t_c)}$, where $F(d, w)$ represents term frequency in document d and $D^{-(t-d.t_c)}$ applies exponential decay based on the document age. The Temporal Popularity (TP) index manages the top-k list using a hash table that stores term-score pairs with update timestamps, and a min-heap indexes the current top-k terms with the highest scores. When a new document arrives, the system avoids full min-heap reconstruction by using equation $TP(s, w_i, t_{cur}) = TP(s, w_i, t_u(w_i)) \times D^{-(t_{cur}-t_u(w_i))} + F(d_n, w_i)$, allowing $O(log\ k)$ updates instead of $O(k\ log\ k)$. Subscription grouping further optimizes the computation through Subscription Groups (SG) characterized by their Intersection Region

$I(SG)$ and Union Bounding Region $U(SG)$, with a compactness metric $C(SG) = I(SG)/U(SG)$ determining group similarity. The Threshold-based Online Group Generation (TOG) algorithm dynamically assigns new subscriptions to existing groups or creates new ones when $C(SG) >$ threshold θ. Spatial indexing uses a Quadtree [100] to associate subscription groups with cells through recursive traversal until no single cell covers $U(SG)$, facilitating retrieval of relevant subscriptions. When processing documents, the system performs three-stage filtering: spatial filtering through Quadtree traversal, group filtering using the condition $TP(SG, w, t_{cur}) + TP(s_{max}(w, SG), w, t_{cur}) \leq SG.currentMinTop$ to eliminate entire subscription groups for certain terms, and individual subscription evaluation for remaining candidates.

6.2 Historical Spatio-Temporal-Textual Indexing

This subsection covers indexes designed for querying archived collections of geo-tagged, time-stamped documents and trajectories with associated textual content. These structures support pattern matching over semantic trajectories, collective spatial keyword queries, and multi-aspect trajectory analysis. These methods combine spatial hierarchies with inverted indexes and episode-based representations to enable efficient retrieval across all three dimensions.

HGI [266]: The Hybrid Grid Index (HGI) is an indexing technique designed to process collective spatial keyword queries (CSKQ) on activity trajectories. At its core, HGI integrates spatial, textual, and popularity information through a dual-component structure. The first component, a Popularity-aware Quad-tree (PQ-tree), hierarchically organizes the spatial domain using a d-Grid structure reminiscent of a Quad-tree [100]. Each node in the PQ-tree encapsulates a grid cell and maintains an inverted file ($ifile$) implemented as a hash table. This $ifile$ maps activities to their maximum popularity scores within the cell and lists of sub-grid IDs containing the activity, enabling efficient pruning during query processing. The second component, the Inverted Activity Length List (IALL), stored on disk, for each trajectory stores an activity sketch summarizing the trajectory's activities, activity posting lists detailing the trajectory points for each activity, and cumulative length information. This structure facilitates rapid filtering and linear scanning of trajectories during query evaluation. The index construction process involves recursively dividing the spatial area, inserting trajectory points into appropriate grid cells, building inverted files for each cell, and constructing the PQ-tree bottom-up. Concurrently, IALLs are created for individual trajectories. During query processing, the system uses a best-first search strategy on the PQ-tree, prioritizing grid cells based on a best-match similarity score that considers both spatial proximity and activity relevance. Candidate trajectories are retrieved in batches from promising cells, and their IALLs are scanned to identify matching sub-trajectories. The technique maintains upper-bound scores to enable early termination, significantly reducing computational overhead.

MAT-SUM [235]: MAT (Multiple Aspect Trajectory Summarization) is a technique for summarizing mobility data by reducing large trajectory datasets while preserving spatio-temporal and semantic details. It processes historical trajectory data that are enriched with user activities, visited places, and geographic context. The method can handle queries for analyzing movement patterns across semantically similar locations, retrieving summarized trajectories within specific geographic regions, and identifying frequent mobility behaviors while maintaining semantic context. In the notation, T represents raw trajectories, $\tilde{T}$ denotes weighted sequences of semantic locations, and $\hat{T}$ indicates summarized semantic trajectories. The main components of MAT-SUM are location-centric enrichment, in which geographic areas are enriched with semantic details (e.g., points of interest, land use), and tessellation, a process that divides these areas into smaller cells (e.g., squares or hexagons). In semantic trajectory summarization, each trajectory is mapped into a sequence of semantically enriched locations, tracking spatial paths, and associated information. Similar semantic locations are then grouped using the similarity condition $sl_j \sim sl_i$ if $sim(S_j, S_i) \geq \tau$, where S_j and S_i are semantic contexts, $sim(\cdot, \cdot)$ is a similarity function, and τ is the similarity threshold. MAT-SUM operates in two phases:

map semantic enrichment, where geographic areas are divided and enriched with semantic attributes, and *semantic trajectory summarization*, where trajectories are converted into weighted sequences of semantic regions, merging similar regions for summarization. The method optimizes the dual objective: $\arg\max_{\text{Summarize}} \sum_{T \in D_{\text{GeoSem}}} \text{SumRate}(T, \hat{T}) + \text{SemSim}(T, \hat{T})$, balancing the summarization rate with the preservation of semantic similarity. This approach reduces data volume while retaining essential spatial and semantic information.

EMT [109]: The EMT (Episode-based Multiple aspect Trajectory) Dual Index presents an indexing approach for multi-aspect (spatial, temporal, textual, and social) trajectories by implementing two interlinked k-d tree [35] based structures that share the same nodes but maintain distinct properties for different aspects of the data. The index handles Social Spatio-temporal Keyword Pattern (S^2KP) queries, which are sequences of episode abstractions containing spatial constraints (MBR intersections), temporal constraints (timeframe matching), textual constraints (keyword containment), and social ranking constraints (numerical range filtering). The notation S^2KP represents queries where episodes can include wildcards (*) for any-value matching and Kleene star operators for zero-or-more episode patterns. The first component, the Spatial Box Sort (*SBS*) tree, extends the Box Sort [131] algorithm to handle spatial dimensions by using a balanced binary search tree in which each node contains MBR coordinates and maintains minimum/maximum subtree coordinates for efficient spatial filtering. The second component, the Episode Sort (*ES*) tree, uses the same physical nodes but organizes them by social ranking values, while maintaining temporal bounds and textual indexes via inverted files. The dual index construction process begins with the *SBS* tree, which employs recursive partitioning based on alternating spatial dimensions and establishes index edges between nodes, followed by the ES tree construction, which sorts based on social rankings and builds text indexes via depth-first traversal. The index incorporates visit edges to maintain trajectory sequence information, enabling efficient pattern matching queries. Query processing uses a hybrid approach in which the search algorithm dynamically chooses between spatial-first (using the *SBS* tree) and social-first (using the *ES* tree) traversal based on keyword frequency in the dataset, followed by constraint verification and sequence matching via visit edges. This structure is implemented in Neo4j, using its graph traversal capabilities and Apache Lucene [2] for text indexing, enabling both disk and memory operations. The dual structure enables efficient filtering on multiple dimensions while maintaining trajectory sequence information through the graph structure.

6.3 Privacy-Preserving Spatio-Temporal-Textual Indexing

This subsection addresses indexing techniques that enable spatial and keyword queries over encrypted data while enforcing temporal access control policies. These methods integrate cryptographic primitives, such as attribute-based encryption, with spatial index structures to secure sensitive location and textual information. This approach balances privacy guarantees with query functionality, supporting secure range and keyword queries without exposing plaintext data.

PBRQ-T$^+$ [284]: This paper proposes *P*rivacy-preserving *B*oolean *R*ange *Q*uery with *T*emporal access control on encrypted spatial data, utilizing a modified Quadtree [100] index structure in its enhanced scheme (PBRQ-T$^+$). The indexing technique combines cryptographic methods with spatial indexing to enable secure querying. The Quadtree structure is adapted to organize encrypted spatial objects, where each leaf node contains an encrypted object and its access policy, while non-leaf nodes store an area covering child nodes, including keywords, and a unified access policy. The index building process involves recursively dividing the space into quadrants, with each non-leaf node assigned a unique secret value. Areas and bounding points are encoded using an adapted Gray code, while keywords are encoded with Bloom filters. These encodings are then encrypted with the secret value of the node using Katz-Sahai-Waters

(KSW) [148] encryption, which is an asymmetric predicate encryption method that evaluates whether the inner product of two vectors is equal to zero without leaking confidentiality. Access policies are encrypted using Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [36] with the same secret. The index matching process starts from the root, traversing the tree by checking if user attributes satisfy each node's access policy, if the query range intersects the node's area (using secure inner product on encrypted Gray code vectors), and if the node contains query keywords (using secure inner product on encrypted Bloom filters). The matching leaf nodes are returned as results. The technique supports Boolean range and keyword queries, balancing privacy preservation with query functionality.

6.4 AI/ML-Powered Spatio-Temporal-Textual Indexing

This subsection presents indexing methods that incorporate machine learning models to enhance query processing over spatio-temporal-textual data. These approaches embed trained topic models within spatial index structures to support complex analytical queries such as market competition analysis across regions and time periods. The integration of learned models enables semantic analysis that goes beyond keyword matching to capture latent topical relationships.

EMart [329]: This paper introduces an enhanced Octree [136] based indexing technique for spatio-temporal-textual data that addresses *M*arket *C*ompetition w.r.t. *T*opics within a specified *R*egion and a *T*ime span query (MCTRT Query). The MCTRT query seeks to mine K topics and determine the probabilities of brand competition for each topic, based on documents within user-specified spatial and temporal bounds. The index structure uses a three dimensional Octree where dimensions represent latitude, longitude, and time, with each cell containing both documents and novel pre-trained `CompLDA` models, which assume a topic distribution for each document and assigns a topic labels to each word tokens and brand mentions, addressing the challenge that topic distribution θ_d , word distribution of topics, and the market competition parameters are initially unknown. The key differentiation lies in its cell division strategy, which, unlike traditional Octrees [136] that split cells equally, optimizes divisions by minimizing word overlap between child cells while maintaining a balanced document distribution through a threshold σ. The index building process begins with document insertion at the root node, followed by recursive subdivision based on an optimization function that is subject to document balance constraints. This function effectively considers the textual similarity between cells during partitioning, using document locations as candidate division points. For each cell, the index maintains pre-computed `CompLDA` models governed by space threshold τ_s and the accuracy threshold τ_e . During query processing for a region R and time interval $[t_b, t_e]$, the index traverses the Octree to identify overlapping cells, then uses a novel combining algorithm utilizing Metropolis-Hastings sampling to efficiently merge topic assignments with bounded error guarantees. The query processing achieves a complexity of $O((|S_2| + |B|)K)$ per iteration, where $|S_2|$ represents topic sets, B denotes brands, and K indicates topics, enabling exploration of market competition dynamics across different spatial regions and temporal periods.

7 PARALLEL AND DISTRIBUTED SPATIO-TEMPORAL ACCESS METHODS

The exponential growth of spatio-temporal data has rendered centralized indexing approaches insufficient for modern application demands, as single-machine resources impose fundamental scalability limits. Parallel and distributed spatio-temporal access methods address these limitations through partitioning data and computation over multiple nodes, enabling horizontal scaling of both storage capacity and query throughput. These systems apply concepts such as MapReduce and GPU architectures to process massive trajectory collections, streaming location data, and complex spatio-temporal joins. The methods presented here employ multiple strategies, including space-filling curves

for distributed key-value stores, hierarchical partitioning schemes, and GPU-optimized data structures, to achieve orders-of-magnitude performance improvements over centralized techniques.

The Distributed BTSR-tree [52]: The Distributed BTSR-tree addresses the scalability of the BTSR-tree [53, 54] (described in Section 2 using MapReduce-based parallel processing, distributing query evaluation over multiple nodes while maintaining hybrid pruning capabilities. This implementation supports similarity joins over large geolocated datasets, handling queries that combine spatial proximity with time series similarity constraints in distributed environments.

DT$\mathcal{J}$i [270]: The DT$\mathcal{J}$i index enables spatial-temporal subtrajectory joins by quickly filtering and accessing trajectory segments within specified temporal and spatial thresholds. It has a three-level indexing scheme: Level 1 (Temporal Partitioning), Level 2 (Spatial Partitioning and Spatial Index), and Level 3 (Trajectory Index). In Level 1, the dataset is partitioned into uniform time intervals. Each partition is expanded by a temporal tolerance parameter, ϵ_t, which ensures that all data points within the specified time tolerance for join operations are included. In Level 2, within each temporal partition, the data points are spatially divided into cells using a QuadTree [100] structure to maintain load balancing. This spatial partitioning is captured in a spatial index (*SpI*), which helps quickly filter points that are spatially close within a specified threshold ϵ_{sp}. In Level 3, this layer includes a trajectory-specific index (*TrI*) that keeps track of the sequence of points within each trajectory, facilitating quick access to temporally adjacent points. The *SpI* allows efficient access to points within the spatially expanded region of a query point, reducing the need for a full scan within each time window. Furthermore, the *TrI* enables quick access to temporally adjacent points, helping to identify subtrajectory portions that may form a join candidate. In addition, the indexing scheme reduces the number of spatial and temporal comparisons required, leading to faster query processing, and the DT$\mathcal{J}$i algorithm, which utilizes this index, achieves an order-of-magnitude improvement over non-indexed approaches in join processing.

JUST [162]: This paper introduces Z2T and XZ2T, which are distributed spatio-temporal indexing strategies that improve the existing Z2/Z3 [220] and XZ2/XZ3 [38] approaches. JUST handles main spatial and spatio-temporal queries, e.g., spatio-temporal range and spatial kNN queries. These indexes address the challenge of different scales between spatial and temporal dimensions in spatio-temporal queries. Z2T, designed for point data, divides the time dimension into disjoint periods and constructs a Z2 spatial index within each period. Its key structure concatenates the time period number with the Z2-encoded spatial coordinates. XZ2T, tailored for non-point data such as polygons and lines, follows a similar principle but uses XZ2 encoding for spatial indexing. The index building process for Z2T involves calculating the time period number for each record, computing the Z2 spatial code for its coordinates, and combining these elements. XZ2T's indexing process first determines the minimum bounding rectangle (MBR) of the non-point object, calculates the time period using the start time, computes the XZ2 code for the MBR coordinates, and concatenates these components. During query processing, the system identifies relevant time periods that intersect the query range, generates key ranges using Z2/XZ2 encoding on the spatial query bounds for each qualified period, and performs parallel SCAN operations on the underlying NoSQL key-value store (HBase) [20]. This approach prunes the search space in both spatial and temporal dimensions. The indexes support dynamic updates as each record's spatio-temporal information is independently encoded in its key. The system uses Apache Spark [324, 325] for complex post-processing of query results, supporting scalable management of large spatio-temporal datasets. By combining time partitioning with space-filling curve techniques, Z2T and XZ2T achieve a balance between index construction efficiency and query performance, demonstrating improved scalability and query speed compared to several state-of-the-art systems, particularly for spatio-temporal range queries and k-NN queries.

TrajMESA [163]: TrajMesa introduces a distributed NoSQL storage engine for trajectory data, building upon GeoMesa's framework [1]. It uses a dual-index approach with an ID Temporal Indexing Table and a Spatial Range

Indexing Table, both using a horizontal storage schema (H-Store) that encapsulates entire trajectories in single rows. The ID Temporal index key structure *(shard+oid+BinNum+ElementCode+tid)* incorporates a novel *XZT* encoding technique that efficiently represents time ranges within predefined bins, using a binary search-like approach to generate compact element codes. The Spatial Range index key *(shard+XZ2+PosCode+tid)* extends GeoMesa's *XZ2* [38] spatial encoding with a novel *PosCode*, which divides extended areas into grids and sets bits for occupied cells, providing finer-grained spatial representation. Both indexes leverage sharding for load-balancing and include trajectory IDs for precise retrieval. The H-Store schema compresses GPS point lists and includes a spatial signature, dramatically reducing storage requirements and I/O operations. Query processing exploits these structures, with ID Temporal queries generating multi-level query windows based on time bins and offsets, while Spatial Range queries combine *XZ2* ranges with matching *PosCodes*. TrajMesa introduces advanced pruning strategies for 'Similarity' and 'k-NN' queries, including MBR Pruning, *SEP_LB* (start/end-point lower bound), and *SIG_LB* (signature-based lower bound), significantly enhancing query efficiency. For trajectory similarity specifically, it implements a two-phase filtering-refinement approach that combines spatial indexing with progressive distance-based pruning. The filtering phase uses spatial range queries around the trajectory endpoints with distance threshold ε, while the refinement phase applies the three pruning strategies before computing expensive Fréchet distances [17]. The k-NN algorithm employs an iterative spatial expansion technique that leverages these pruning methods to minimize unnecessary trajectory comparisons. This indexing approach enables TrajMesa to efficiently support a wide range of trajectory queries, including ID Temporal, Spatial Range, Similarity, and k-NN queries, while maintaining scalability in a distributed environment. The system demonstrates linear growth in storage requirements and query times as data size increases, highlighting its effectiveness in managing massive trajectory datasets.

SYNPOSIS [43]: SYNOPSIS presents a distributed spatiotemporal indexing technique designed for high-velocity streaming data. The core of SYNOPSIS is the SIFT (Summarization Involving a Forest of Trees) structure, which organizes data hierarchically across distributed nodes called sketchlets. Each sketchlet is responsible for a specific geospatial region, determined by geohash prefixes. Within a sketchlet, data is organized in a forest of trees, where each tree represents a subregion and incorporates multiple levels of temporal and feature-based organization. The upper levels of these trees represent coarser temporal granularity, while the lower levels represent finer temporal resolution and individual features. Leaf nodes store summary statistics and metadata about the data points, which are updated incrementally using Welford's method [301] as new data streams in. To optimize memory usage, SYNOPSIS employs dynamic reconfiguration techniques, including moving high fan-out features lower in the tree hierarchy, applying quantization to merge similar value ranges, and compacting older data by merging nodes. Query processing in SYNOPSIS involves routing queries to relevant sketchlets based on spatial scope, then traversing the trees within sketchlets using temporal and feature constraints to prune irrelevant subtrees. Summary statistics from leaf nodes are used to compute approximate query results, with partial results from distributed sketchlets merged for global queries. SYNOPSIS employs density-driven quantization and temporal dimensionality reduction to manage memory usage, allowing it to maintain a compact representation of large-scale spatiotemporal data streams.

GeoWave [302]: GeoWave introduces a distributed spatio-temporal indexing technique that supports multiple query types, including spatial range queries (bounding boxes), spatiotemporal queries combining both space and time constraints, multi-dimensional queries (up to 4D with X, Y, Z, and time), selective queries targeting small data portions, and complex geometric relationship queries. The index uses space-filling curves (SFC), specifically the compact Hilbert SFC [119], to transform n-dimensional spatial and temporal data into a single dimension while preserving data locality in distributed key-value stores. For spatial range queries, the core indexing mechanism uses a tiered

hierarchical structure, such as a Quadtree [100], where each tier corresponds to different precision levels of the Hilbert curve, enabling multi-resolution querying. For spatiotemporal queries, the system implements a binning strategy with a configurable periodicity (default: 1 year for temporal data) for unbounded dimensions like time, where each bin represents a bounded period. The index structure combines multiple components: an Index ID containing the SFC tier, bin information, and Hilbert value; an 'Adapter ID' for data type identification; a 'Data ID' unique within each adapter; a duplicate count; and extended metadata. During index construction, the dimensional values are normalized to fit the SFC bounds, mapped to the values of the Hilbert curve, and composite keys are generated. To process multi-dimensional and selective queries efficiently, the query processing mechanism decomposes query bounding boxes into minimal continuous quadrants through a recursive process, where the system first identifies the deepest node of the Quadtree that fully contains the queried range by performing XOR operations on Hilbert values. These quadrants are then recursively decomposed into sub-regions until reaching optimal granularity, and the resulting regions are converted into SFC ranges for efficient retrieval. For complex geometric queries, GeoWave implements server-side filtering by evaluating fine-grained geometric relationships using the Java Topology Suite (JTS) [179] library. For visualization queries, the system uses a unique pixel-based spatial subsampling technique that transforms the pixel space into the SFC context and implements server-side row skipping to avoid processing unnecessary data. The index supports both disk and memory operations, handles dynamic updates, and works across multiple distributed datastores, including Accumulo [21], DynamoDB [18], HBase [20], and Cassandra [19].

TMan [127]: TMan is a distributed trajectory management system with two novel indexing structures: TR (Temporal Range) and TShape indexes, designed for key-value stores. TMan handles multiple trajectory queries, including temporal-range, spatial-range, and trajectory-similarity queries. The TR index is used for temporal indexing, dividing the timeline into fixed-length periods and representing trajectory time ranges using k consecutive periods as time bins. It uses an encoding formula $TR(TB_{i,j}) = i * N + j - i$, where i and j represent start and end periods respectively, and N is the maximum periods a bin can cover, ensuring that adjacent time ranges get adjacent index values without redundant storage. The TShape index addresses spatial representation using a Quadtree-based [100] variable-shaped index. It uses a composite structure that combines quadrant sequences (location encoding) with shape codes ($\alpha * \beta$ bits representing cell usage patterns), merged into a 64-bit index value. The shape encoding is optimized using greedy algorithms and genetic algorithms to maintain spatial proximity in index values, with an index cache mechanism storing shape-to-code mappings. For query processing, TMan implements a push-down strategy in which filter conditions are pushed to the storage layer and executed in parallel across distributed nodes. The system maintains primary and secondary tables to reduce storage redundancy: the primary table stores complete trajectories, and the secondary tables maintain index mappings. The system supports dynamic updates through a buffer cache mechanism for new shape codes and periodic re-encoding.

PMR Quadtree [303]: The PMR Quadtree is used to support spatial and spatio-temporal joins efficiently on large geospatial datasets in Apache Spark [324, 325] using both regular grids and PMR Quadtrees for spatial indexing, which are more adaptive to data distribution, helping to manage data skew more effectively. In addition to spatial indexing, temporal indexing is necessary for spatio-temporal joins, in which time intervals are treated as extrusions of spatial bins to create a three-dimensional structure for time and space. Moreover, the Quadtree [100] index has a binning strategy. This strategy grouped data points into spatial and temporal bins to facilitate the join process, thereby dividing the data into manageable pieces for processing. In addition, there are two main join strategies based on data size: Broadcast and Bin Joins. For smaller datasets, a Broadcast Join is used to load one dataset entirely into memory on each node, while larger datasets employ a Bin Join, partitioning data across nodes based on spatial bins, and can utilize a regular

Grid [213] or a Quadtree to manage data skew. The group join finds all matching pairs for a given feature, outputting one feature per match, and the summary join aggregates data for each feature, resulting in a single summary result per target feature. For optimization, the Quadtree uses two techniques: Direct Summarization with Pre-Aggregation technique that aggregates coincident points within small bins, micro-cells, to handle highly skewed data, improving performance and reducing memory usage, and the adaptive Binning technique, which adjusts bin sizes based on data density, using smaller bins in dense areas to avoid memory overload. To handle skew and coincidence, the system uses strategies such as reference-point de-duplication to remove duplicate entries from highly skewed data. For each candidate feature pair, a single reference point is chosen to determine the specific bin in which the candidate pair will be processed. If the candidate pair appears in multiple bins, due to overlaps or near distances, only the bin containing the chosen reference point will retain this pair, and it will be ignored in other bins. So, the system avoids duplicate processing of the same pair across multiple bins, thus reducing memory usage and computation time. For spatio-temporal joins, reference point de-duplication is extended to both spatial and temporal dimensions, where the reference point de-duplication method is applied separately in space and time, handling both dimensions to remove duplicates effectively. This method minimizes cross-node communication and prevents unnecessary recomputation of pairs.

ST4ML [177]: ST4ML is a distributed spatio-temporal data processing system designed for machine learning applications, employing optimized indexing techniques to support efficient data loading and instance conversion. The system supports various ML applications, including traffic flow prediction, urban anomaly detection, travel recommendation, and air quality forecasting. ST4ML integrates and optimizes existing methods to enable scalable processing of large-scale spatio-temporal data. The system utilizes a partition-based on-disk indexing approach, where data is partitioned using the novel T-STR partitioner. T-STR first segments the data along the temporal dimension into n_t partitions, and then splits each temporal partition into n_s spatial partitions, i.e., segmenting into $n_t \times n_s$ partitions using the STR algorithm, effectively extending STR to be spatio-temporally aware. A metadata file is maintained to index the partitions, recording the boundaries of all indexing dimensions, such as the Minimum Bounding Rectangle (MBR) for spatial dimensions and endpoints for temporal dimensions. During data loading, only partitions that overlap the query range are selected and loaded into memory, reducing memory consumption and improving query performance by pruning irrelevant partitions. For efficient conversion of singular spatio-temporal instances (e.g., events, trajectories) to collective instances (e.g., time series, spatial maps, rasters), ST4ML employs in-memory R-tree [117] indexing. This indexing scheme particularly benefits ML applications such as trajectory classification, where stay points need to be extracted, and traffic speed prediction, which requires efficient aggregation of vehicle trajectories into grid-based features. The cells of the collective structure are indexed using R-trees of different dimensions: 1-d for time series (supporting demand prediction), 2-d for spatial maps (enabling site selection and tourism planning), and 3-d for rasters (facilitating traffic forecasting and pollution prediction). The R-tree is constructed on the master node and is broadcast to all worker nodes to avoid duplicate computations. Each singular instance then traverses the R-tree to find the intersecting cells for allocation, reducing the average time complexity from $O(mn)$ to $O(mlog(n))$, where m is the number of singular instances and n is the number of cells.

TrajEdge [124]: TrajEdge introduces a spatiotemporal trie-based peer-to-peer (P2P) index designed for distributed trajectory data analysis in edge computing environments, where trajectory data is stored across multiple edge servers, requiring efficient cross-device querying. The index uses a dual encoding scheme that combines XZ2 [38] for spatial encoding and XZT [132] for temporal encoding, transforming multi-dimensional spatiotemporal data into one-dimensional keys that preserve ordering and locality. For spatial encoding, the system identifies the minimum bounding rectangle (MBR) of a trajectory τ, locates the spatial region containing the trajectory's start point, expands this region

to twice its original size, and assigns the resulting region code as the S-key. Temporal encoding uses the formula $\text{BinNum}(t_s) = \lfloor \frac{t_s - \text{RefTime}}{\text{BinLen}} \rfloor$ to segment the timeline into bins, then iteratively performs binary search on a line segment $L = [tl_s, tl_e]$ within each bin, appending '0' if the relative start time t'_s falls in the left half of L or '1' if in the right half, continuing until the extended region $[tl_s, 2tl_e - tl_s]$ cannot contain the interval $[t'_s, t'_e]$. The complete index key concatenates these components as T-key = S-key | BinNum | eCode | τ.id, where eCode represents the element code from temporal encoding and τ.id denotes the trajectory identifier. The trie structure comprises three node types: external nodes forming the P2P overlay network, each maintaining a routing label ω, a routing table Z tracking the parent, child, and neighboring nodes, and segmented bloom filters for query pruning; internal nodes within local tries holding T-keys; and leaf nodes that store trajectory identifiers. The index construction optimizes the mapping $f : O \rightarrow B$ from external nodes O to edge servers B by minimizing $\sum df(\omega(O_i), \omega(f(O_i)))$ subject to capacity constraints C_j, where d_f denotes shortest-path hops between the trie labels, ensuring spatially proximate data resides on physically adjacent servers to minimize query latency $q_t(P) \propto \sum_i^{|P|} \|\text{loc}_{B_i} - \text{loc}_{B_{i+1}}\|_2$ across routing paths. $R_s = [\text{loc}_{\min}, \text{loc}_{\max}]$ for spatial query ranges, $R_t = [t_s, t_e]$ for temporal ranges, and $E(\hat{\tau})$, $D(\hat{\tau})$, $T(\hat{\tau})$ representing edge sequences, distance sequences, and timestamp sequences of road-constrained trajectories respectively. The index supports four query types: ID temporal queries, spatial range queries, spatio-temporal range queries, and k-NN queries. An ID temporal query retrieves all trajectories whose moving object identifier matches a given id and contains at least one point whose timestamp falls within a specified time range $R_t = [t_s, t_e]$, formally returning $\mathcal{R} \subseteq \mathcal{D}_s$ where $\forall\, \hat{\tau} \in \mathcal{R},\ \hat{\tau}.id = id\ \&\ \exists\, \hat{p} \in \hat{\tau},\ \hat{p}.t \in R_t$. Because such queries carry no spatial component, they constitute an index-mismatch scenario in which the trie's spatial routing key cannot be used to navigate directly to the relevant external node; instead, segmented bloom filters maintained at each external node encode both T-keys from internal nodes and $\tau.id$ values from leaf nodes prune the broadcast space and route the query only to candidate nodes, avoiding a full network-wide broadcast. The index achieves $O(\log N)$ complexity for spatial queries through trie traversal and $O(\frac{N}{r})$ for index-mismatch scenarios using bloom filter optimization, where r represents the false positive rate.

7.1 Distributed Contact Tracing

This subsection presents distributed indexing methods specifically designed for contact-tracing applications, where the scale of population-level mobility data demands parallel processing. These systems must efficiently identify proximity events between moving objects while meeting real-time latency requirements for public health response. The methods use GPU acceleration, distributed memory architectures, and hierarchical spatial partitioning to process millions of trajectories with sub-second query response times.

STS-index [51]: This paper presents a privacy-aware distributed spatio-temporal indexing structure designed for efficient sub-trajectory similarity joins, utilizing a two-tier architecture that separates data storage between clients and a central server. At the client level, the original trajectories are stored and indexed by local IDs, where the trajectories undergo a modified Douglas-Peucker [91] algorithm that maintains the accuracy of the points within a threshold θ_{sp}, followed by a bounded Laplace mechanism that adds controlled noise (bounded by the θ_{ob} obfuscated shifting distance) for the preservation of privacy. The server tier implements a hierarchical hybrid structure that combines temporal and spatial indexing components. The temporal component is organized as a B-tree [30] structure indexing time intervals hierarchically, where leaf entries point to spatial indices covering 3-minute intervals. The spatial component uses a quasi-Quadtree structure that maintains non-overlapping minimum bounding rectangles (MBRs), built by recursively partitioning the space into quadrants when a node's capacity is exceeded. A distinctive feature is the addition of bitmap tables in leaf nodes, corresponding to disjoint time intervals, where each bit indicates whether a segment overlaps

the interval. This enhances query efficiency for densely populated nodes. The index supports dynamic updates via insertion and deletion operations, in which new segments traverse the temporal B-tree to locate appropriate time intervals, then are inserted into the corresponding quasi-Quadtrees. For query processing, the index uses a sophisticated backtracking-based algorithm that expands query segment MBRs by $\delta_d + \theta_{sp} + \theta_{ob}$ to guarantee no false dismissals, utilizing pivot points (vertices of expanded MBRs) to efficiently locate adjacent segments. The structure achieves efficient pruning through both its non-overlapping MBR property and a Close-distance Duration Similarity (CDDS) upper-bound computation, significantly reducing the number of candidate pairs sent to clients for final verification.

Sextant [328]: Sextant introduces a distributed spatial indexing approach to efficiently handle real-time K-nearest neighbor (KNN) queries on moving objects. The system supports KNN queries of the form (loc, ts, k), where loc is the query location coordinate, ts is the timestamp, and k is the number of nearest neighbors to return. It also handles frequent location update operations in the form of data tuples $(id, loc, ts, metadata)$, where id uniquely identifies an object, loc is its current location, ts is the timestamp, and $metadata$ specifies object status (e.g., car driver vs. motorcycle driver). The system partitions the $2D$ geographical space into fixed-size grid cells called *shards*, which are further subdivided into smaller cells. This two-level hierarchical partitioning technique forms the basis of the Sextant spatial indexing strategy. Objects (e.g., drivers) are dynamically assigned to cells based on their current location using the indexing function index $(lat, lon) = (\lfloor \frac{lon+180}{l} \rfloor, \lfloor \frac{lat+90}{l} \rfloor)$, where l is the size of the grid and $(-180, -90)$ serves as the origin. All data is stored in memory for rapid updates and queries. The indexing technique uses a hybrid distribution method that combines consistent hashing with a custom "ShardTable" for load balancing across nodes. ShardTable allows dedicated nodes for high-demand areas, addressing the challenge of the uneven spatial distribution of objects and queries. For KNN queries, Sextant uses a breadth-first search (BFS) algorithm that starts from the cell that contains the query point and expands outward. To optimize performance in the distributed setting, the system aggregates queries at the shard level, reducing the number of remote calls and enabling parallelization across shards and cells. This approach enables Sextant to efficiently handle millions of objects and billions of daily queries. The indexing technique supports dynamic updates, necessary for tracking moving objects, and uses a TTL (Time To Live) mechanism to manage object expiration. To ensure fault tolerance, Sextant implements replica sets and periodically takes snapshots of the in-memory data to external storage.

GLINT [279]: GLINT (GPU-based ReaL-tIme CoNtact Tracing) implements a hierarchical space-partitioning index utilizing a Quadtree [100] structure optimized for GPU-accelerated processing of moving objects in contact tracing scenarios. The index supports "within queries" (spatial proximity queries with temporal constraints) to identify all objects within a specified contact distance from query objects, with additional validation requiring contacts to persist for a minimum contact time to be considered valid. The core indexing mechanism uses a recursive spatial decomposition in which each internal node represents a spatial region that subdivides into four quadrants when the number of contained objects exceeds the partition granularity threshold *part_gran*, while leaf nodes maintain partition buffers that store object identifiers within their spatial boundaries. The system uses pre-allocated memory structures, including *node_lst* (tree nodes), *part_lst* (partition buffers), *node_stk*, and *part_stk* (free-space management stacks), to eliminate dynamic memory allocation overhead during GPU execution. Object indexing is performed via parallel tree traversal, where each GPU thread traverses from the root node to the appropriate leaf partition, collecting filtering information to optimize subsequent range queries for boundary objects whose contact regions span multiple partitions. The filtering phase uses a block-shared stack mechanism enabling up to 1,024 threads per block to collaboratively perform depth-first traversal while maintaining *filter_root* nodes that minimize search space for objects requiring multi-partition queries. Refinement processing implements distance calculations between object pairs using an unrolling technique parameterized by

refinement granularity to maximize GPU utilization by decomposing variable-sized partition evaluations into fixed-size computational units. Spatial-temporal contact evaluation uses a GPU-based hash table with Cantor pairing [175] functions to map object pairs to unique keys, while maintaining contact start/end timestamps and pruning contacts that fail to meet temporal duration requirements. Dynamic index maintenance operates at the leaf level through split operations when partitions exceed capacity and merge operations when sibling partitions collectively fall below the granularity threshold, avoiding expensive full-tree reconstruction while adapting to evolving object distributions via overflow buffers that handle exceptional cases where spatial constraints prevent further subdivision.

DISTIL$^+$ [199]: DISTIL$^+$ presents a distributed in-memory spatio-temporal indexing technique designed for high throughput location updates and many concurrent spatio-temporal range queries (STRQ) and kNN queries (STkNNQ). The index uses a multilevel hierarchical structure comprising three levels: a global Quadtree [100] index for spatial partitioning, a per-node local spatial index, and a per-tile partial temporal index (PTI). The spatial domain is discretized into a grid of tiles, each of which maintains a PTI for discrete time intervals. The PTI uses a bit vector to efficiently track object presence within a tile during specific time intervals, complemented by a hash table that maps object IDs to record ID lists for full location information. Index construction begins with tile assignment to nodes using placement policies, e.g., Row-wise Round-Robin partitioning (RRR) and Multi-Dimensional Range partitioning (MDR). As location updates arrive, they are batched and routed to appropriate nodes based on tile ID. At each node, updates are inserted into an in-memory table, a local persistent store, and the spatio-temporal index, updating the bit vector and hash table for the relevant tile and time interval. Query processing leverages this structure to efficiently prune and parallelize execution. Range queries use the global index to identify overlapping tiles, then process fully contained tiles using bitwise OR operations on bit-vectors, while partial tiles require precise location checks. kNN queries start at the query point's tile, expand the search area using the global index, and process tiles in parallel across nodes. The index supports dynamic updates through its multi-level structure and batching approach, balancing memory usage with query performance. DISTIL$^+$ extends concepts from grid-based [213] spatial indexing and bitmap-based temporal indexing, implementing them in a distributed, in-memory framework using the APGAS model (asynchronous partitioned global address space) [248]. This solution enables high-throughput location updates and concurrent query processing, addressing scalability challenges in location-based services and large-scale spatio-temporal data management.

GAT [87, 327]: The GAT framework introduces GTIDX, a GPU-optimized indexing scheme for efficient batch trajectory query processing. GTIDX uses a three-part structure: a cell-based trajectory table (S), a trajectory index (I_T), and a Quadtree-like [100] spatial index (I_Q). The spatial region is partitioned into 4^n cells using a Quadtree-like approach, with trajectory points mapped to these cells and stored in S. This cell-based organization facilitates rapid pruning for both range and similarity queries [261, 262]. I_T enables efficient trajectory reconstruction by mapping trajectory IDs to vectors of $(cell_{id}, offset, length)$ triplets, representing trajectory subsequences within cells. I_Q groups cells into balanced blocks, each containing a similar number of trajectory points, which is necessary for load-balancing during GPU processing. The index construction process involves splitting the trajectories into cell-based subsequences, updating S and I_T accordingly, building a complete Quadtree, and then recursively pruning them to form I_Q. A key idea is the use of Morton-based encoding [204] for cell identifiers, ensuring that cells in the same block are stored contiguously in memory and enabling coalesced data access on the GPU. For query processing, range queries employ CPU-based filtering by traversing I_Q to identify overlapping blocks, followed by parallel point verification on the GPU. Similarity queries utilize a frequency distance (*FD*) lower-bound computed on the CPU, iteratively selecting candidates with the smallest *FD* for parallel *Edit Distance on Real sequence* (*EDR*) [63] computation on the GPU. The framework incorporates several optimizations, including a Memory Allocated Table (*MAT*) to prevent redundant data transfers

and a dynamic batch query partitioning strategy to balance workloads across GPU resources. This indexing approach effectively leverages the massive parallelism of GPUs while addressing challenges such as limited global memory bandwidth and load-balancing, resulting in significant performance improvements over CPU-based methods and other GPU-accelerated approaches.

8 CONCLUSION

This survey is Part 4 of a series of surveys [192, 202, 211] that collectively cover the spatio-temporal access methods developed from the earliest approaches through 2025. In the years 2017 to 2025, several transformative categories of spatio-temporal access methods have emerged and matured, namely: (1) Machine Learning enhanced indexing structures that learn from data distributions and query patterns, (2) specialized contact tracing methods developed in response to the COVID-19 pandemic, (3) advanced spatio-temporal-textual indexes that seamlessly integrate semantic information, (4) GPU-accelerated and distributed indexing techniques that leverage massive parallelism, (5) privacy-preserving indexing methods that support encrypted queries, and (6) trajectory-oriented methods that exploit road network topology and enable efficient similarity search at scale.

Trajectory data management has seen significant advances through methods that exploit the road network structure. PATHFINDER [104] repurposes Contraction Hierarchies for trajectory compression and indexing, while the SNT-index [151, 152] applies string-processing techniques for efficient path matching. The RP-tree [297] leverages road network topology for range query processing, and TD-H2H [161] extends hierarchical decomposition to time-dependent networks. Trajectory similarity search has matured using methods such as tSTAT [146] using locality-sensitive hashing, HST [112, 113] employing hierarchical simplification, and WR-tree [141, 142] supporting trajectory-based entity linking. Hybrid indexing techniques that combine multiple structures, e.g., the IR-tree in REST [335] and BAR [312], demonstrate the value of integrating spatial indexes with inverted files for multi-attribute trajectories.

The integration of Machine Learning into spatio-temporal indexing represents a paradigm shift from static, one-size-fits-all structures to adaptive systems that optimize themselves based on workload characteristics. Methods like the CDR index [26, 27] for aircraft conflict detection demonstrate how Hidden Markov Models can be embedded within indexing structures to provide predictive capabilities. The TRC-index [164, 165] incorporates XGBoost [69] for automatic parameter tuning in distributed environments. This trend suggests future indexes will increasingly incorporate learned components, potentially replacing traditional heuristics with data-driven decisions.

Contact tracing methods, while developed under urgent pandemic conditions, have contributed important insights into privacy-preserving indexing, efficient proximity detection, and temporal aggregation techniques. Structures like the θ-structure [40] for interactive density map visualization, RICCmeet [269] for reachability queries with transfer delays, and e-Racoon [15] for temporal aggregation joins have shown how to balance analytical demands with efficiency concerns, techniques that will likely influence future location-based services beyond pandemic applications.

The evolution of spatio-temporal-textual indexes reflects the reality that modern data is inherently multi-dimensional. Social media streams and mobile application logs naturally combine location, time, and semantic information. Indexes such as STILT [22], DLEEL [11], and subscription-based methods SST/RST [64] and Lamps [214, 215] demonstrate sophisticated approaches to handling this complexity, from interleaved trie structures to parallel processing of continuous queries. Streaming and real-time workloads have driven innovations in methods like SSQ [12, 14] for geo-social queries, MR-Cubes [77] for location popularity computation, and SYNOPSIS [43] for high-velocity streaming data summarization.

Update-intensive workloads have motivated new indexing architectures. The LSM RUM-tree [256, 257] adapts log-structured merge principles to spatial indexing with an in-memory update memo for efficient modifications. The

Springbok system [115] introduces dual-indexing for cloud-based trajectory management with separate structures for active and historical data. Methods for indexing geolocated time series, including TSR-tree, BTSR-tree [53, 54], and SBTSR-tree [55], demonstrate how temporal patterns can be indexed alongside spatial coordinates through minimum bounding time series.

GPU acceleration and distributed processing have moved from experimental methods to production-ready solutions. Systems like GLINT [279], GAT [87, 327], and Sextant [328] show how massive parallelism can be harnessed for spatio-temporal operations, while distributed frameworks like JUST [162], TrajMESA [163], and GeoWave [302] demonstrate how to scale across clusters. The ST4ML [177] system specifically targets Machine Learning applications, recognizing that spatio-temporal data increasingly feed AI/ML pipelines. The PASTIS system [238] uniquely addresses past, present, and future queries within a unified versioned grid structure.

Privacy and security considerations have become first-class concerns in index design, as evidenced by methods such as PBRQ-T$^+$ [284] for encrypted queries combining Quadtree[100] structures with attribute-based encryption, and the STS-index [51] for privacy-aware similarity joins using trajectory simplification and bounded noise injection. This trend will likely accelerate as regulations impose stricter requirements on the handling of location data.

Looking ahead, several trends are likely to shape the next generation of spatio-temporal access methods. The continued integration of learned components into index structures may enable automatic adaptation to changing data distributions and query patterns. Real-time applications in transportation, including ride-sharing systems, will demand ever-lower latency for location updates and queries, supported by methods such as GeoPrune [317] and DVTG [271]. Trajectory imputation methods such as GTI [134, 135] and TrImpute [93] suggest a growing interest in reconstructing complete movement patterns from sparse observations. As spatial computing matures, we anticipate new indexing challenges for higher-dimensional spatio-temporal data at unprecedented scales and update rates.

REFERENCES

[1] 2021. GeoMesa. https://www.geomesa.org/ [Online; accessed 09-February-2025].

[2] 2022. *Apache Lucene.* https://lucene.apache.org/ Retrieved from https://lucene.apache.org/.

[3] Mahdi Abdelguerfi, Julie Givaudan, Kevin Shaw, and Roy Ladner. 2002. The 2-3TR-tree, a trajectory-oriented index structure for fully evolving valid-time spatio-temporal datasets. In *ACM-GIS 2002, Proceedings of the Tenth ACM International Symposium on Advances in Geographic Information Systems, McLean, VA (near Washington, DC), USA, USA, November 8-9, 2002*, Agnès Voisard and Shu-Ching Chen (Eds.). ACM, 29–34. https://doi.org/10.1145/585147.585155

[4] Pankaj K. Agarwal, Lars Arge, and Jeff Erickson. 2000. Indexing Moving Points. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, May 15-17, 2000, Dallas, Texas, USA*, Victor Vianu and Georg Gottlob (Eds.). ACM, 175–186. https://doi.org/10.1145/335168.335220

[5] Pritom Ahmed, Mahbub Hasan, Abhijith Kashyap, Vagelis Hristidis, and Vassilis J. Tsotras. 2017. Efficient Computation of Top-k Frequent Terms over Spatio-temporal Ranges. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*, Semih Salihoglu, Wenchao Zhou, Rada Chirkova, Jun Yang, and Dan Suciu (Eds.). ACM, 1227–1241. https://doi.org/10.1145/3035918.3064032

[6] Tanvir Ahmed, Torben Bach Pedersen, and Hua Lu. 2017. Finding dense locations in symbolic indoor tracking data: modeling, indexing, and processing. *GeoInformatica* 21, 1 (2017), 119–150. https://doi.org/10.1007/S10707-016-0276-8

[7] Afsin Akdogan, Cyrus Shahabi, and Ugur Demiryurek. 2014. ToSS-it: A Cloud-Based Throwaway Spatial Index Structure for Dynamic Location Data. In *IEEE 15th International Conference on Mobile Data Management, MDM 2014, Brisbane, Australia, July 14-18, 2014 - Volume 1*, Arkady B. Zaslavsky, Panos K. Chrysanthis, Christian Becker, Jadwiga Indulska, Mohamed F. Mokbel, Daniela Nicklas, and Chi-Yin Chow (Eds.). IEEE Computer Society, 249–258. https://doi.org/10.1109/MDM.2014.37

[8] Afsin Akdogan, Cyrus Shahabi, and Ugur Demiryurek. 2016. D-ToSS: A Distributed Throwaway Spatial Index Structure for Dynamic Location Data. *IEEE Trans. Knowl. Data Eng.* 28, 9 (2016), 2334–2348. https://doi.org/10.1109/TKDE.2016.2572697

[9] Abdullah Al-Mamun, Hao Wu, Qiyang He, Jianguo Wang, and Walid G. Aref. 2025. A Survey of Learned Indexes for the Multi-dimensional Space. *ACM Comput. Surv.* 58, 4, Article 96 (Oct. 2025), 37 pages. https://doi.org/10.1145/3768575

[10] Mohammed Eunus Ali, Shadman Saqib Eusuf, Kaysar Abdullah, Farhana Murtaza Choudhury, J. Shane Culpepper, and Timos Sellis. 2018. The Maximum Trajectory Coverage Query in Spatial Databases. *Proc. VLDB Endow.* 12, 3 (2018), 197–209. https://doi.org/10.14778/3291264.3291266

[11] Abdulaziz Almaslukh, Laila Abdelhafeez, and Amr Magdy. 2020. DLEEL: Multi-Predicate Spatial Queries on User-generated Streaming Data. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 1770–1773. https://doi.org/10.1109/ICDE48307.2020.00166

[12] Abdulaziz Almaslukh, Yunfan Kang, and Amr Magdy. 2021. Temporal Geo-Social Personalized Keyword Search Over Streaming Data. *ACM Trans. Spatial Algorithms Syst.* 7, 4 (2021), 20:1–20:28. https://doi.org/10.1145/3473006

[13] Abdulaziz Almaslukh and Amr Magdy. 2018. Evaluating spatial-keyword queries on streaming data. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018*, Farnoush Banaei Kashani, Erik G. Hoel, Ralf Hartmut Güting, Roberto Tamassia, and Li Xiong (Eds.). ACM, 209–218. https://doi.org/10.1145/3274895.3274936

[14] Abdulaziz Almaslukh and Amr Magdy. 2019. Temporal Geo-Social Personalized Search Over Streaming Data. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*, Farnoush Banaei Kashani, Goce Trajcevski, Ralf Hartmut Güting, Lars Kulik, and Shawn D. Newsam (Eds.). ACM, 189–198. https://doi.org/10.1145/3347146.3359073

[15] Rakan A. Alseghayer. 2024. Optimizing Operators for Temporal and Spatiotemporal Data. In *25th IEEE International Conference on Mobile Data Management, MDM 2024, Brussels, Belgium, June 24-27, 2024*. IEEE, 331–333. https://doi.org/10.1109/MDM61037.2024.00068

[16] Sattam Alsubaiee, Alexander Behm, Vinayak R. Borkar, Zachary Heilbron, Young-Seok Kim, Michael J. Carey, Markus Dreseler, and Chen Li. 2014. Storage Management in AsterixDB. *Proc. VLDB Endow.* 7, 10 (2014), 841–852. https://doi.org/10.14778/2732951.2732958

[17] Helmut Alt and Michael Godau. 1995. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.* 5 (1995), 75–91. https://doi.org/10.1142/S0218195995000064

[18] Amazon Web Services. 2012. *Amazon DynamoDB*. Amazon. https://aws.amazon.com/dynamodb/ A fully managed NoSQL database service.

[19] Apache Software Foundation. 2008. *Apache Cassandra*. Apache Software Foundation. http://cassandra.apache.org/ A highly-scalable partitioned row store database.

[20] Apache Software Foundation. 2008. *Apache HBase*. Apache Software Foundation. https://hbase.apache.org/ A distributed, scalable, big data store.

[21] Apache Software Foundation. 2011. *Apache Accumulo*. Apache Software Foundation. https://accumulo.apache.org/ A sorted, distributed key/value store based on Google's BigTable.

[22] Yoann Arseneau, Saransh Gautam, Bradford G. Nickerson, and Suprio Ray. 2020. STILT: Unifying Spatial, Temporal and Textual Search using a Generalized Multi-dimensional Index. In *SSDBM 2020: 32nd International Conference on Scientific and Statistical Database Management, Vienna, Austria, July 7-9, 2020*, Elaheh Pourabbas, Dimitris Sacharidis, Kurt Stockinger, and Thanasis Vergoulis (Eds.). ACM, 11:1–11:12. https://doi.org/10.1145/3400903.3400927

[23] Vijayalakshmi Atluri, Nabil R Adam, and Mahmoud Youssef. 2003. Towards a unified index scheme for mobile data and customer profiles in a location-based service environment. In *Workshop on Next Generation Geospatial Information (NG2I'03)*.

[24] Vijayalakshmi Atluri and Qi Guo. 2005. Unified Index for Mobile Object Data and Authorizations. In *Computer Security - ESORICS 2005, 10th European Symposium on Research in Computer Security, Milan, Italy, September 12-14, 2005, Proceedings (Lecture Notes in Computer Science, Vol. 3679)*, Sabrina De Capitani di Vimercati, Paul F. Syverson, and Dieter Gollmann (Eds.). Springer, 80–97. https://doi.org/10.1007/11555827_6

[25] Vijayalakshmi Atluri and Heechang Shin. 2007. Efficient Security Policy Enforcement in a Location Based Service Environment. In *Data and Applications Security XXI, 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Redondo Beach, CA, USA, July 8-11, 2007, Proceedings (Lecture Notes in Computer Science, Vol. 4602)*, Steve Barker and Gail-Joon Ahn (Eds.). Springer, 61–76. https://doi.org/10.1007/978-3-540-73538-0_5

[26] Samet Ayhan, Pablo Costas, and Hanan Samet. 2018. Prescriptive analytics system for long-range aircraft conflict detection and resolution. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018*, Farnoush Banaei Kashani, Erik G. Hoel, Ralf Hartmut Güting, Roberto Tamassia, and Li Xiong (Eds.). ACM, 239–248. https://doi.org/10.1145/3274895.3274947

[27] Samet Ayhan, Pablo Costas, and Hanan Samet. 2019. A Data-driven Framework for Long-Range Aircraft Conflict Detection and Resolution. *ACM Trans. Spatial Algorithms Syst.* 5, 4 (2019), 24:1–24:23. https://doi.org/10.1145/3328832

[28] Michael J. Bannister, William E. Devanny, Michael T. Goodrich, Joseph A. Simons, and Lowell Trott. 2014. Windows into Geometric Events: Data Structures for Time-Windowed Querying of Temporal Point Sets. In *Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG 2014, Halifax, Nova Scotia, Canada, 2014*. Carleton University, Ottawa, Canada. http://www.cccg.ca/proceedings/2014/papers/paper02.pdf

[29] Michael J. Bannister, Christopher DuBois, David Eppstein, and Padhraic Smyth. 2013. Windows into Relational Events: Data Structures for Contiguous Subsequences of Edges. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, Sanjeev Khanna (Ed.). SIAM, 856–864. https://doi.org/10.1137/1.9781611973105.61

[30] Rudolf Bayer and Edward M. McCreight. 1970. Organization and Maintenance of Large Ordered Indexes. In *Record of the 1970 ACM SIGFIDET Workshop on Data Description and Access, November 15-16, 1970, Rice University, Houston, Texas, USA (Second Edition with an Appendix)*, Edgar F. Codd (Ed.). ACM, 107–141. https://doi.org/10.1145/1734663.1734671

[31] Rudolf Bayer and Karl Unterauer. 1977. Prefix B-Trees. *ACM Trans. Database Syst.* 2, 1 (1977), 11–26. https://doi.org/10.1145/320521.320530

[32] Bruno Becker, Stephan Gschwind, Thomas Ohler, Bernhard Seeger, and Peter Widmayer. 1996. An Asymptotically Optimal Multiversion B-Tree. *VLDB J.* 5, 4 (1996), 264–275. https://doi.org/10.1007/S007780050028

[33] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1990. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, USA, May 23-25, 1990*, Hector Garcia-Molina and H. V. Jagadish (Eds.). ACM Press, 322–331. https://doi.org/10.1145/93597.98741

[34] Amina Belhassena and Hongzhi Wang. 2017. Distributed skyline trajectory query processing. In *Proceedings of the ACM Turing 50th Celebration Conference - China, TUR-C 2017, Shanghai, China, May 12-14, 2017*, John C. S. Lui, Xinbing Wang, Alexander Wolf, Yunhao Liu, and Chuanping Hu (Eds.). ACM, 19:1–19:7. https://doi.org/10.1145/3063955.3063974

[35] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18, 9 (1975), 509–517. https://doi.org/10.1145/361002.361007

[36] John Bethencourt, Amit Sahai, and Brent Waters. 2007. Ciphertext-Policy Attribute-Based Encryption. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*. IEEE Computer Society, 321–334. https://doi.org/10.1109/SP.2007.11

[37] Alina Beygelzimer, Sham M. Kakade, and John Langford. 2006. Cover trees for nearest neighbor. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006 (ACM International Conference Proceeding Series, Vol. 148)*, William W. Cohen and Andrew W. Moore (Eds.). ACM, 97–104. https://doi.org/10.1145/1143844.1143857

[38] Christian Böhm, Gerald Klump, and Hans-Peter Kriegel. 1999. XZ-Ordering: A Space-Filling Curve for Objects with Spatial Extension. In *Advances in Spatial Databases, 6th International Symposium, SSD'99, Hong Kong, China, July 20-23, 1999, Proceedings (Lecture Notes in Computer Science, Vol. 1651)*, Ralf Hartmut Güting, Dimitris Papadias, and Frederick H. Lochovsky (Eds.). Springer, 75–90. https://doi.org/10.1007/3-540-48482-5_7

[39] Kyoung Soo Bok, Dong Min Seo, Seung Soo Shin, and Jae Soo Yoo. 2004. TPKDB-Tree: An Index Structure for Efficient Retrieval of Future Positions of Moving Objects. In *Conceptual Modeling for Advanced Application Domains, ER 2004 Workshops CoMoGIS, COMWIM, ECDM, CoMoA, DGOV, and ECOMO, Shanghai, China, November 8-12, 2004, Proceedings (Lecture Notes in Computer Science, Vol. 3289)*. Springer, 67–78. https://doi.org/10.1007/978-3-540-30466-1_7

[40] Annika Bonerath, Benjamin Niedermann, Jim Diederich, Yannick Orgeig, Johannes Oehrlein, and Jan-Henrik Haunert. 2020. A Time-Windowed Data Structure for Spatial Density Maps. In *SIGSPATIAL '20: 28th International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, November 3-6, 2020*, Chang-Tien Lu, Fusheng Wang, Goce Trajcevski, Yan Huang, Shawn D. Newsam, and Li Xiong (Eds.). ACM, 15–24. https://doi.org/10.1145/3397536.3422242

[41] Annika Bonerath, Benjamin Niedermann, and Jan-Henrik Haunert. 2019. Retrieving α-Shapes and Schematic Polygonal Approximations for Sets of Points within Queried Temporal Ranges. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*, Farnoush Banaei Kashani, Goce Trajcevski, Ralf Hartmut Güting, Lars Kulik, and Shawn D. Newsam (Eds.). ACM, 249–258. https://doi.org/10.1145/3347146.3359087

[42] Nieves R. Brisaboa, Susana Ladra, and Gonzalo Navarro. 2009. k2-Trees for Compact Web Graph Representation. In *String Processing and Information Retrieval, 16th International Symposium, SPIRE 2009, Saariselkä, Finland, August 25-27, 2009, Proceedings (Lecture Notes in Computer Science, Vol. 5721)*. Springer, 18–30. https://doi.org/10.1007/978-3-642-03784-9_3

[43] Thilina Buddhika, Matthew Malensek, Sangmi Lee Pallickara, and Shrideep Pallickara. 2017. Synopsis: A Distributed Sketch over Voluminous Spatiotemporal Observational Streams. *IEEE Trans. Knowl. Data Eng.* 29, 11 (2017), 2552–2566. https://doi.org/10.1109/TKDE.2017.2734661

[44] Michael Burrows. 1994. A block-sorting lossless data compression algorithm. *SRS Research Report* 124 (1994).

[45] M. Cai and P. Revesz. 2000. Parametric R-Tree: An Index Structure for Moving Objects. In *Proceedings of the International Conference on Management of Data (COMAD)*, Krithi Ramamritham and T.M. Vijayaraman (Eds.). Tata McGraw-Hill Publishing Company Ltd., 57–64. In Advances in Data Management 2000.

[46] Ruichu Cai, Zijie Lu, Li Wang, Zhenjie Zhang, Tom Z. J. Fu, and Marianne Winslett. 2017. DITIR: Distributed Index for High Throughput Trajectory Insertion and Real-time Temporal Range Query. *Proc. VLDB Endow.* 10, 12 (2017), 1865–1868. https://doi.org/10.14778/3137765.3137795

[47] Yuhan Cai and Raymond T. Ng. 2004. Indexing Spatio-Temporal Trajectories with Chebyshev Polynomials. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, Gerhard Weikum, Arnd Christian K¨onig, and Stefan Deßloch (Eds.). ACM, 599–610. https://doi.org/10.1145/1007568.1007636

[48] Bin Cao, Chenyu Hou, Suifei Li, Jing Fan, Jianwei Yin, Baihua Zheng, and Jie Bao. 2018. $\sf {SIMkNN}$: A Scalable Method for in-MemorykNN Search over Moving Objects in Road Networks. *IEEE Trans. Knowl. Data Eng.* 30, 10 (2018), 1957–1970. https://doi.org/10.1109/TKDE.2018.2808971

[49] V. Prasad Chakka, Adam Everspaugh, and Jignesh M. Patel. 2003. Indexing Large Trajectory Data Sets With SETI. In *First Biennial Conference on Innovative Data Systems Research, CIDR 2003, Asilomar, CA, USA, January 5-8, 2003, Online Proceedings*. www.cidrdb.org. http://www-db.cs.wisc.edu/cidr/cidr2003/program/p15.pdf

[50] Farah Chanchary and Anil Maheshwari. 2019. Time Windowed Data Structures for Graphs. *J. Graph Algorithms Appl.* 23, 2 (2019), 191–226. https://doi.org/10.7155/JGAA.00489

[51] Yanchuan Chang, Jianzhong Qi, Egemen Tanin, Xingjun Ma, and Hanan Samet. 2021. Sub-trajectory Similarity Join with Obfuscation. In *SSDBM 2021: 33rd International Conference on Scientific and Statistical Database Management, Tampa, FL, USA, July 6-7, 2021*, Qiang Zhu, Xingquan Zhu, Yicheng Tu, Zichen Xu, and Anand Kumar (Eds.). ACM, 181–192. https://doi.org/10.1145/3468791.3468822

[52] Georgios Chatzigeorgakidis, Kostas Patroumpas, Dimitrios Skoutas, Spiros Athanasiou, and Spiros Skiadopoulos. 2018. Scalable hybrid similarity join over geolocated time series. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information*

Systems, SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018, Farnoush Banaei Kashani, Erik G. Hoel, Ralf Hartmut Güting, Roberto Tamassia, and Li Xiong (Eds.). ACM, 119–128. https://doi.org/10.1145/3274895.3274949

[53] Georgios Chatzigeorgakidis, Kostas Patroumpas, Dimitrios Skoutas, Spiros Athanasiou, and Spiros Skiadopoulos. 2019. Visual Exploration of Geolocated Time Series with Hybrid Indexing. *Big Data Res.* 15 (2019), 12–28. https://doi.org/10.1016/J.BDR.2019.02.001

[54] Georgios Chatzigeorgakidis, Dimitrios Skoutas, Kostas Patroumpas, Spiros Athanasiou, and Spiros Skiadopoulos. 2017. Indexing Geolocated Time Series Data. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2017, Redondo Beach, CA, USA, November 7-10, 2017*, Erik G. Hoel, Shawn D. Newsam, Siva Ravada, Roberto Tamassia, and Goce Trajcevski (Eds.). ACM, 19:1–19:10. https://doi.org/10.1145/3139958.3140003

[55] Georgios Chatzigeorgakidis, Dimitrios Skoutas, Kostas Patroumpas, Themis Palpanas, Spiros Athanasiou, and Spiros Skiadopoulos. 2019. Local Similarity Search on Geolocated Time Series Using Hybrid Indexing. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*, Farnoush Banaei Kashani, Goce Trajcevski, Ralf Hartmut Güting, Lars Kulik, and Shawn D. Newsam (Eds.). ACM, 179–188. https://doi.org/10.1145/3347146.3359349

[56] Bernard Chazelle. 1983. Filtering Search: A New Approach to Query-Answering. In *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*. IEEE Computer Society, 122–132. https://doi.org/10.1109/SFCS.1983.17

[57] Bernard Chazelle. 1986. Filtering Search: A New Approach to Query-Answering. *SIAM J. Comput.* 15, 3 (1986), 703–724. https://doi.org/10.1137/0215051

[58] Gang Chen, Jingwen Zhao, Yunjun Gao, Lei Chen, and Rui Chen. 2017. Time-Aware Boolean Spatial Keyword Queries. *IEEE Trans. Knowl. Data Eng.* 29, 11 (2017), 2601–2614. https://doi.org/10.1109/TKDE.2017.2742956

[59] Gang Chen, Jingwen Zhao, Yunjun Gao, Lei Chen, and Rui Chen. 2018. Time-Aware Boolean Spatial Keyword Queries (Extended Abstract). In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*. IEEE Computer Society, 1781–1782. https://doi.org/10.1109/ICDE.2018.00248

[60] Jian Chen, Hong Gao, Yuhong Shi, Junle Chen, Donghua Yang, and Jianzhong Li. 2025. Maximizing Influence Query Over Indoor Trajectories. *IEEE Trans. Knowl. Data Eng.* 37, 3 (2025), 1294–1310. https://doi.org/10.1109/TKDE.2024.3514323

[61] Jian Chen, Hong Gao, Kaiqi Zhang, Jiachi Wang, Yubo Luo, Zhenqing Wu, and Jianzhong Li. 2023. Towards Efficient MIT query in Trajectory Data. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 2194–2206. https://doi.org/10.1109/ICDE55515.2023.00170

[62] Ji-Dong Chen and Xiao-Feng Meng. 2007. Indexing Future Trajectories of Moving Objects in a Constrained Network. *J. Comput. Sci. Technol.* 22, 2 (2007), 245–251. https://doi.org/10.1007/S11390-007-9031-9

[63] Lei Chen, M. Tamer Özsu, and Vincent Oria. 2005. Robust and Fast Similarity Search for Moving Object Trajectories. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005*, Fatma Özcan (Ed.). ACM, 491–502. https://doi.org/10.1145/1066157.1066213

[64] Lisi Chen, Shuo Shang, Christian S. Jensen, Jianliang Xu, Panos Kalnis, Bin Yao, and Ling Shao. 2020. Top-k term publish/subscribe for geo-textual data streams. *VLDB J.* 29, 5 (2020), 1101–1128. https://doi.org/10.1007/S00778-020-00607-8

[65] Lisi Chen, Shuo Shang, Zhiwei Zhang, Xin Cao, Christian S. Jensen, and Panos Kalnis. 2018. Location-Aware Top-k Term Publish/Subscribe. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*. IEEE Computer Society, 749–760. https://doi.org/10.1109/ICDE.2018.00073

[66] Ling Chen, Yanlin Tang, Mingqi Lv, and Gencai Chen. 2015. Partition-based range query for uncertain trajectories in road networks. *GeoInformatica* 19, 1 (2015), 61–84. https://doi.org/10.1007/S10707-014-0206-6

[67] Nan Chen, Lidan Shou, Gang Chen, and Jinxiang Dong. 2008. Adaptive Indexing of Moving Objects with Highly Variable Update Frequencies. *J. Comput. Sci. Technol.* 23, 6 (2008), 998–1014. https://doi.org/10.1007/S11390-008-9185-0

[68] Su Chen, Beng Chin Ooi, Kian-Lee Tan, and Mario A. Nascimento. 2008. ST^2B-tree: a self-tunable spatio-temporal b^+-tree index for moving objects. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, Jason Tsong-Li Wang (Ed.). ACM, 29–42. https://doi.org/10.1145/1376616.1376622

[69] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi (Eds.). ACM, 785–794. https://doi.org/10.1145/2939672.2939785

[70] Wei Chen, Lei Zhao, Jiajie Xu, Kai Zheng, and Xiaofang Zhou. 2014. Ranking Based Activity Trajectory Search. In *Web Information Systems Engineering - WISE 2014 - 15th International Conference, Thessaloniki, Greece, October 12-14, 2014, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 8786)*, Boualem Benatallah, Azer Bestavros, Yannis Manolopoulos, Athena Vakali, and Yanchun Zhang (Eds.). Springer, 170–185. https://doi.org/10.1007/978-3-319-11749-2_14

[71] Yue Chen, Zhida Chen, Gao Cong, Ahmed R. Mahmood, and Walid G. Aref. 2020. SSTD: A Distributed System on Streaming Spatio-Textual Data. *Proc. VLDB Endow.* 13, 11 (2020), 2284–2296. http://www.vldb.org/pvldb/vol13/p2284-chen.pdf

[72] Zhida Chen, Gao Cong, Zhenjie Zhang, Tom Z. J. Fu, and Lisi Chen. 2017. Distributed Publish/Subscribe Query Processing on the Spatio-Textual Data Stream. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*. IEEE Computer Society, 1095–1106. https://doi.org/10.1109/ICDE.2017.154

[73] Hae Don Chon, Divyakant Agrawal, and Amr El Abbadi. 2001. Storage and Retrieval of Moving Objects. In *Mobile Data Management, Second International Conference, MDM 2001, Hong Kong, China, January 8-10, 2001, Proceedings (Lecture Notes in Computer Science, Vol. 1987)*, Kian-Lee Tan, Michael J. Franklin, and John C. S. Lui (Eds.). Springer, 173–184. https://doi.org/10.1007/3-540-44498-X_14

[74] Maria Christoforaki, Jinru He, Constantinos Dimopoulos, Alexander Markowetz, and Torsten Suel. 2011. Text vs. space: efficient geo-search query processing. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, Craig Macdonald, Iadh Ounis, and Ian Ruthven (Eds.). ACM, 423–432. https://doi.org/10.1145/2063576.2063641

[75] David R. Clark and J. Ian Munro. 1996. Efficient Suffix Trees on Secondary Storage (extended Abstract). In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, 28-30 January 1996, Atlanta, Georgia, USA*, Éva Tardos (Ed.). ACM/SIAM, 383–391. http://dl.acm.org/citation.cfm?id=313852.314087

[76] Gao Cong, Christian S. Jensen, and Dingming Wu. 2009. Efficient Retrieval of the Top-k Most Relevant Spatial Web Objects. *Proc. VLDB Endow.* 2, 1 (2009), 337–348. https://doi.org/10.14778/1687627.1687666

[77] George Constantinou, Chrysovalantis Anastasiou, Dimitris Stripelis, and Cyrus Shahabi. 2019. MR-Cubes: On-the-Fly Computation of Location Popularity from Check-in Data Streams. In *20th IEEE International Conference on Mobile Data Management, MDM 2019, Hong Kong, SAR, China, June 10-13, 2019*. IEEE, 27–36. https://doi.org/10.1109/MDM.2019.00-77

[78] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms, 3rd Edition*. MIT Press. http://mitpress.mit.edu/books/introduction-algorithms

[79] Philippe Cudré-Mauroux, Eugene Wu, and Samuel Madden. 2010. TrajStore: An adaptive storage system for very large trajectory data sets. In *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA*, Feifei Li, Mirella M. Moro, Shahram Ghandeharizadeh, Jayant R. Haritsa, Gerhard Weikum, Michael J. Carey, Fabio Casati, Edward Y. Chang, Ioana Manolescu, Sharad Mehrotra, Umeshwar Dayal, and Vassilis J. Tsotras (Eds.). IEEE Computer Society, 109–120. https://doi.org/10.1109/ICDE.2010.5447829

[80] Ningning Cui, Jianxin Li, Xiaochun Yang, Bin Wang, Mark Reynolds, and Yong Xiang. 2019. When Geo-Text Meets Security: Privacy-Preserving Boolean Spatial Keyword Queries. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. IEEE, 1046–1057. https://doi.org/10.1109/ICDE.2019.00097

[81] Jing Dai and Chang-Tien Lu. 2011. DIME: Disposable Index for Moving Objects. In *12th IEEE International Conference on Mobile Data Management, MDM 2011, Luleå, Sweden, June 6-9, 2011, Volume 1*, Arkady B. Zaslavsky, Panos K. Chrysanthis, Dik Lun Lee, Dipanjan Chakraborty, Vana Kalogeraki, Mohamed F. Mokbel, and Chi-Yin Chow (Eds.). IEEE Computer Society, 68–77. https://doi.org/10.1109/MDM.2011.69

[82] Victor Teixeira de Almeida and Ralf Hartmut G¨uting. 2005. Indexing the Trajectories of Moving Objects in Networks. *GeoInformatica* 9, 1 (2005), 33–60. https://doi.org/10.1007/S10707-004-5621-7

[83] Edsger W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numer. Math.* 1 (1959), 269–271. https://doi.org/10.1007/BF01386390

[84] Xiaofeng Ding, Yansheng Lu, Xiaochao Ding, Na Zhao, and Qiong Wei. 2007. An Efficient Index for Moving Objects with Frequent Updates. In *2007 International Conference on Wireless Communications, Networking and Mobile Computing*. 5951–5954. https://doi.org/10.1109/WICOM.2007.1459

[85] Zhiming Ding. 2008. UTR-Tree: An Index Structure for the Full Uncertain Trajectories of Network-Constrained Moving Objects. In *9th International Conference on Mobile Data Management (MDM 2008), Beijing, China, April 27-30, 2008*, Xiaofeng Meng, Hui Lei, Stéphane Grumbach, and Hong Va Leong (Eds.). IEEE, 33–40. https://doi.org/10.1109/MDM.2008.8

[86] Jens Dittrich, Lukas Blunschi, and Marcos Antonio Vaz Salles. 2009. Indexing Moving Objects Using Short-Lived Throwaway Indexes. In *Advances in Spatial and Temporal Databases, 11th International Symposium, SSTD 2009, Aalborg, Denmark, July 8-10, 2009, Proceedings (Lecture Notes in Computer Science, Vol. 5644)*, Nikos Mamoulis, Thomas Seidl, Torben Bach Pedersen, Kristian Torp, and Ira Assent (Eds.). Springer, 189–207. https://doi.org/10.1007/978-3-642-02982-0_14

[87] Kaixing Dong, Bowen Zhang, Yanyan Shen, Yanmin Zhu, and Jiadi Yu. 2020. GAT: A Unified GPU-Accelerated Framework for Processing Batch Trajectory Queries. *IEEE Trans. Knowl. Data Eng.* 32, 1 (2020), 92–107. https://doi.org/10.1109/TKDE.2018.2879862

[88] Yuyang Dong, Hanxiong Chen, and Hiroyuki Kitagawa. 2019. Continuous Search on Dynamic Spatial Keyword Objects. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. IEEE, 1578–1581. https://doi.org/10.1109/ICDE.2019.00146

[89] Yuyang Dong, Chuan Xiao, Hanxiong Chen, Jeffrey Xu Yu, Kunihiro Takeoka, Masafumi Oyamada, and Hiroyuki Kitagawa. 2021. Continuous top-k spatial-keyword search on dynamic objects. *VLDB J.* 30, 2 (2021), 141–161. https://doi.org/10.1007/S00778-020-00627-4

[90] Harish Doraiswamy, Huy T. Vo, Cláudio T. Silva, and Juliana Freire. 2016. A GPU-based index to support interactive spatio-temporal queries over historical data. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*. IEEE Computer Society, 1086–1097. https://doi.org/10.1109/ICDE.2016.7498315

[91] David H Douglas and Thomas K Peucker. 1973. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization* 10, 2 (1973), 112–122.

[92] Khaled M. Elbassioni, Amr Elmasry, and Ibrahim Kamel. 2003. An Efficient Indexing Scheme for Multi-dimensional Moving Objects. In *Database Theory - ICDT 2003, 9th International Conference, Siena, Italy, January 8-10, 2003, Proceedings (Lecture Notes in Computer Science, Vol. 2572)*, Diego Calvanese, Maurizio Lenzerini, and Rajeev Motwani (Eds.). Springer, 422–436. https://doi.org/10.1007/3-540-36285-1_28

[93] Mohamed M. Elshrif, Keivin Isufaj, and Mohamed F. Mokbel. 2022. Network-less trajectory imputation. In *Proceedings of the 30th International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2022, Seattle, Washington, November 1-4, 2022*, Matthias Renz and Mohamed Sarwat (Eds.). ACM, 8:1–8:10. https://doi.org/10.1145/3557915.3560942

[94] Christos Faloutsos and Shari Roseman. 1989. Fractals for Secondary Key Retrieval. In *Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, March 29-31, 1989, Philadelphia, Pennsylvania, USA*, Avi Silberschatz (Ed.). ACM Press, 247–252. https://doi.org/10.1145/73721.73746

[95] Ping Fan, Guohui Li, Ling Yuan, and Yanhong Li. 2012. Vague continuous K-nearest neighbor queries over moving objects with uncertain velocity in road networks. *Inf. Syst.* 37, 1 (2012), 13–32. https://doi.org/10.1016/J.IS.2011.08.002

[96] Ying Fang, Jiaheng Cao, Yuwei Peng, and Liwei Wang. 2008. Indexing the Past, Present and Future Positions of Moving Objects on Fixed Networks. In *International Conference on Computer Science and Software Engineering, CSSE 2008, Volume 4: Embedded Programming / Database Technology / Neural Networks and Applications / Other Applications, December 12-14, 2008, Wuhan, China*. IEEE Computer Society, 524–527. https://doi.org/10.1109/CSSE.2008.1449

[97] Ying Fang, Jiaheng Cao, Junzhou Wang, Yuwei Peng, and Wei Song. 2011. HTPR*-Tree: An Efficient Index for Moving Objects to Support Predictive Query and Partial History Query. In *Web-Age Information Management - WAIM 2011 International Workshops: WGIM 2011, XMLDM 2011, SNA 2011, Wuhan, China, September 14-16, 2011, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 7142)*, Liwei Wang, Jingjue Jiang, Jiaheng Lu, Liang Hong, and Bin Liu (Eds.). Springer, 26–39. https://doi.org/10.1007/978-3-642-28635-3_3

[98] Jun Feng, Jiamin Lu, Yuelong Zhu, Naoto Mukai, and Toyohide Watanabe. 2007. Indexing of Moving Objects on Road Network Using Composite Structure. In *Knowledge-Based Intelligent Information and Engineering Systems, 11th International Conference, KES 2007, XVII Italian Workshop on Neural Networks, Vietri sul Mare, Italy, September 12-14, 2007. Proceedings, Part II (Lecture Notes in Computer Science, Vol. 4693)*, Bruno Apolloni, Robert J. Howlett, and Lakhmi C. Jain (Eds.). Springer, 1097–1104. https://doi.org/10.1007/978-3-540-74827-4_137

[99] Paolo Ferragina and Giovanni Manzini. 2000. Opportunistic Data Structures with Applications. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*. IEEE Computer Society, 390–398. https://doi.org/10.1109/SFCS.2000.892127

[100] Raphael A. Finkel and Jon Louis Bentley. 1974. Quad Trees: A Data Structure for Retrieval on Composite Keys. *Acta Informatica* 4 (1974), 1–9. https://doi.org/10.1007/BF00288933

[101] Edward Fredkin. 1960. Trie memory. *Commun. ACM* 3, 9 (1960), 490–499. https://doi.org/10.1145/367390.367400

[102] Elias Frentzos. 2003. Indexing Objects Moving on Fixed Networks. In *Advances in Spatial and Temporal Databases, 8th International Symposium, SSTD 2003, Santorini Island, Greece, July 24-27, 2003, Proceedings (Lecture Notes in Computer Science, Vol. 2750)*, Thanasis Hadzilacos, Yannis Manolopoulos, John F. Roddick, and Yannis Theodoridis (Eds.). Springer, 289–305. https://doi.org/10.1007/978-3-540-45072-6_17

[103] Elias Frentzos, Kostas Gratsias, Nikos Pelekis, and Yannis Theodoridis. 2007. Algorithms for Nearest Neighbor Search on Moving Object Trajectories. *GeoInformatica* 11, 2 (2007), 159–193. https://doi.org/10.1007/S10707-006-0007-7

[104] Stefan Funke, Tobias Rupp, André Nusser, and Sabine Storandt. 2019. PATHFINDER: Storage and Indexing of Massive Trajectory Sets. In *Proceedings of the 16th International Symposium on Spatial and Temporal Databases, SSTD 2019, Vienna, Austria, August 19-21, 2019*, Walid G. Aref, Michela Bertolotto, Panagiotis Bouros, Christian S. Jensen, Ahmed R. Mahmood, Kjetil Nørvåg, Dimitris Sacharidis, and Mohamed Sarwat (Eds.). ACM, 90–99. https://doi.org/10.1145/3340964.3340978

[105] Yunjun Gao, Chun Li, Gencai Chen, Qing Li, and Chun Chen. 2007. Efficient Algorithms for Historical Continuous k NN Query Processing over Moving Object Trajectories. In *Advances in Data and Web Management, Joint 9th Asia-Pacific Web Conference, APWeb 2007, and 8th International Conference, on Web-Age Information Management, WAIM 2007, Huang Shan, China, June 16-18, 2007, Proceedings (Lecture Notes in Computer Science, Vol. 4505)*, Guozhu Dong, Xuemin Lin, Wei Wang, Yun Yang, and Jeffrey Xu Yu (Eds.). Springer, 188–199. https://doi.org/10.1007/978-3-540-72524-4_22

[106] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. 2008. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In *Experimental Algorithms, 7th International Workshop, WEA 2008, Provincetown, MA, USA, May 30-June 1, 2008, Proceedings (Lecture Notes in Computer Science, Vol. 5038)*, Catherine C. McGeoch (Ed.). Springer, 319–333. https://doi.org/10.1007/978-3-540-68552-4_24

[107] Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. 2012. Exact Routing in Large Road Networks Using Contraction Hierarchies. *Transp. Sci.* 46, 3 (2012), 388–404. https://doi.org/10.1287/TRSC.1110.0401

[108] Fragkiskos Gryllakis, Nikos Pelekis, Christos Doulkeridis, Stylianos Sideridis, and Yannis Theodoridis. 2018. Spatio-Temporal-Keyword Pattern Queries over Semantic Trajectories with Hermes@Neo4j. In *Proceedings of the 21st International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018*, Michael H. Böhlen, Reinhard Pichler, Norman May, Erhard Rahm, Shan-Hung Wu, and Katja Hose (Eds.). OpenProceedings.org, 678–681. https://doi.org/10.5441/002/EDBT.2018.83

[109] Fragkiskos Gryllakis, Nikos Pelekis, Christos Doulkeridis, Iraklis Varlamis, and Yannis Theodoridis. 2022. Social Spatio-temporal Keyword Pattern (S^2KP) Queries in Multiple Aspect Trajectories Databases. In *SSDBM 2022: 34th International Conference on Scientific and Statistical Database Management, Copenhagen, Denmark, July 6 - 8, 2022*, Elaheh Pourabbas, Yongluan Zhou, Yuchen Li, and Bin Yang (Eds.). ACM, 8:1–8:12. https://doi.org/10.1145/3538712.3538716

[110] Tu Gu, Kaiyu Feng, Jingyi Yang, Gao Cong, Cheng Long, and Rui Zhang. 2024. BT-Tree: A Reinforcement Learning Based Index for Big Trajectory Data. *Proc. ACM Manag. Data* 2, 4 (2024), 194:1–194:27. https://doi.org/10.1145/3677130

[111] Joachim Gudmundsson, Michael Horton, John Pfeifer, and Martin P. Seybold. 2021. A Practical Index Structure Supporting Fréchet Proximity Queries among Trajectories. *ACM Trans. Spatial Algorithms Syst.* 7, 3 (2021), 15:1–15:33. https://doi.org/10.1145/3460121

[112] Joachim Gudmundsson, John Pfeifer, and Martin P. Seybold. 2023. On Practical Nearest Sub-Trajectory Queries under the Fréchet Distance. *ACM Trans. Spatial Algorithms Syst.* 9, 2 (2023), 14:1–14:24. https://doi.org/10.1145/3587426

[113] Joachim Gudmundsson, Martin P. Seybold, and John Pfeifer. 2021. On Practical Nearest Sub-Trajectory Queries under the Fréchet Distance. In *SIGSPATIAL '21: 29th International Conference on Advances in Geographic Information Systems, Virtual Event / Beijing, China, November 2-5, 2021*, Xiaofeng Meng, Fusheng Wang, Chang-Tien Lu, Yan Huang, Shashi Shekhar, and Xing Xie (Eds.). ACM, 596–605. https://doi.org/10.1145/3474717.3484264

[114] Yang Guo, Tianyu Wang, Zizhan Chen, and Zili Shao. 2025. A Storage Model with Fine-Grained In-Storage Query Processing for Spatio-Temporal Data. In *41st IEEE International Conference on Data Engineering, ICDE 2025, Hong Kong, May 19-23, 2025*. IEEE, 669–682. https://doi.org/10.1109/ICDE65448.2025.00056

[115] Yang Guo, Zhiqi Wang, Jin Xue, and Zili Shao. 2024. A Spatio-Temporal Series Data Model with Efficient Indexing and Layout for Cloud-Based Trajectory Data Management. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*. IEEE, 1171–1184. https://doi.org/10.1109/ICDE60146.2024.00313

[116] Ralf Hartmut Güting, Thomas Behr, and Jianqiu Xu. 2010. Efficient k-nearest neighbor search on moving object trajectories. *VLDB J.* 19, 5 (2010), 687–714. https://doi.org/10.1007/S00778-010-0185-7

[117] Antonin Guttman. 1984. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, USA, June 18-21, 1984*, Beatrice Yormark (Ed.). ACM Press, 47–57. https://doi.org/10.1145/602259.602266

[118] Marios Hadjieleftheriou, George Kollios, Vassilis J. Tsotras, and Dimitrios Gunopulos. 2002. Efficient Indexing of Spatiotemporal Objects. In *Advances in Database Technology - EDBT 2002, 8th International Conference on Extending Database Technology, Prague, Czech Republic, March 25-27, Proceedings (Lecture Notes in Computer Science, Vol. 2287)*. Springer, 251–268. https://doi.org/10.1007/3-540-45876-X_17

[119] Chris H. Hamilton and Andrew Rau-Chaplin. 2008. Compact Hilbert indices: Space-filling curves for domains with unequal side lengths. *Inf. Process. Lett.* 105, 5 (2008), 155–163. https://doi.org/10.1016/J.IPL.2007.08.034

[120] R. W. Hamming. 1950. Error detecting and error correcting codes. *The Bell System Technical Journal* 29, 2 (1950), 147–160. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x

[121] Laipeng Han, Lan Huang, Xueyi Yang, Wei Pang, and Kangping Wang. 2016. A Novel Spatio-Temporal Data Storage and Index Method for ARM-Based Hadoop Server. In *Cloud Computing and Security - Second International Conference, ICCCS 2016, Nanjing, China, July 29-31, 2016, Revised Selected Papers, Part I (Lecture Notes in Computer Science, Vol. 10039)*, Xingming Sun, Alex X. Liu, Han-Chieh Chao, and Elisa Bertino (Eds.). 206–216. https://doi.org/10.1007/978-3-319-48671-0_19

[122] Yuxing Han, Liping Wang, Ying Zhang, Wenjie Zhang, and Xuemin Lin. 2015. Spatial Keyword Range Search on Trajectories. In *Database Systems for Advanced Applications - 20th International Conference, DASFAA 2015, Hanoi, Vietnam, April 20-23, 2015, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 9050)*, Matthias Renz, Cyrus Shahabi, Xiaofang Zhou, and Muhammad Aamir Cheema (Eds.). Springer, 223–240. https://doi.org/10.1007/978-3-319-18123-3_14

[123] Ramaswamy Hariharan, Bijit Hore, Chen Li, and Sharad Mehrotra. 2007. Processing Spatial-Keyword (SK) Queries in Geographic Information Retrieval (GIR) Systems. In *19th International Conference on Scientific and Statistical Database Management, SSDBM 2007, 9-11 July 2007, Banff, Canada, Proceedings*. IEEE Computer Society, 16. https://doi.org/10.1109/SSDBM.2007.22

[124] Changhao He, Ziquan Fang, Linsen Li, and Yunjun Gao. 2025. TrajEdge: An Efficient and Lightweight Trajectory Data Analysis Framework in Edge Environments. In *41st IEEE International Conference on Data Engineering, ICDE 2025, Hong Kong, May 19-23, 2025*. IEEE, 2894–2907. https://doi.org/10.1109/ICDE65448.2025.00217

[125] Dan He, Sibo Wang, Xiaofang Zhou, and Reynold Cheng. 2021. GLAD: A Grid and Labeling Framework with Scheduling for Conflict-Aware kNN Queries. *IEEE Trans. Knowl. Data Eng.* 33, 4 (2021), 1554–1566. https://doi.org/10.1109/TKDE.2019.2942585

[126] Huajun He, Ruiyuan Li, Sijie Ruan, Tianfu He, Jie Bao, Tianrui Li, and Yu Zheng. 2022. TraSS: Efficient Trajectory Similarity Search Based on Key-Value Data Stores. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*. IEEE, 2306–2318. https://doi.org/10.1109/ICDE53745.2022.00218

[127] Huajun He, Zihang Xu, Ruiyuan Li, Jie Bao, Tianrui Li, and Yu Zheng. 2024. TMan: A High-Performance Trajectory Data Management System Based on Key-Value Stores. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*. IEEE, 4951–4964. https://doi.org/10.1109/ICDE60146.2024.00376

[128] Zhenwen He, Menno-Jan Kraak, Otto Huisman, Xiaogang Ma, and Jing Xiao. 2013. Parallel indexing technique for spatio-temporal data. *ISPRS Journal of Photogrammetry and Remote Sensing* 78 (2013), 116–128. https://doi.org/10.1016/j.isprsjprs.2013.01.014

[129] Abdeltawab M. Hendawi, Jie Bao, Mohamed F. Mokbel, and Mohamed H. Ali. 2015. Predictive tree: An efficient index for predictive queries on road networks. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, Johannes Gehrke, Wolfgang Lehner, Kyuseok Shim, Sang Kyun Cha, and Guy M. Lohman (Eds.). IEEE Computer Society, 1215–1226. https://doi.org/10.1109/ICDE.2015.7113369

[130] Tuan-Anh Hoang-Vu, Huy T. Vo, and Juliana Freire. 2016. A Unified Index for Spatio-Temporal Keyword Queries. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, Snehasis Mukhopadhyay, ChengXiang Zhai, Elisa Bertino, Fabio Crestani, Javed Mostafa, Jie Tang, Luo Si, Xiaofang Zhou, Yi Chang, Yunyao Li, and Parikshit Sondhi (Eds.). ACM, 135–144. https://doi.org/10.1145/2983323.2983751

[131] Piet Houthuys. 1987. Box Sort, a multidimensional binary sorting method for rectangular boxes, used for quick range searching. *Vis. Comput.* 3, 4 (1987), 236–249. https://doi.org/10.1007/BF01952830

[132] James N. Hughes, Andrew Annex, Christopher N. Eichelberger, Anthony Fox, Andrew Hulbert, and Michael Ronquest. 2015. GeoMesa: a distributed architecture for spatio-temporal fusion. In *Geospatial Informatics, Fusion, and Motion Video Analytics V*, Matthew F. Pellechia, Kannappan

Palaniappan, Peter J. Doucette, Shiloh L. Dockstader, Gunasekaran Seetharaman, and Paul B. Deignan (Eds.), Vol. 9473. International Society for Optics and Photonics, SPIE, 94730F. https://doi.org/10.1117/12.2177233

[133] Hamza Issa and Maria Luisa Damiani. 2016. Efficient Access to Temporally Overlaying Spatial and Textual Trajectories. In *IEEE 17th International Conference on Mobile Data Management, MDM 2016, Porto, Portugal, June 13-16, 2016*, Chi-Yin Chow, Prem Prakash Jayaraman, and Wei Wu (Eds.). IEEE Computer Society, 262–271. https://doi.org/10.1109/MDM.2016.47

[134] Keivin Isufaj, Jade Choghari, and Mohamed Mokhtar Elshrif. 2023. A Demonstration of GTI: A Scalable Graph-based Trajectory Imputation. In *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2023, Hamburg, Germany, November 13-16, 2023*, Matthias Renz and Mario A. Nascimento (Eds.). ACM, 83:1–83:4. https://doi.org/10.1145/3589132.3625633

[135] Keivin Isufaj, Mohamed Mokhtar Elshrif, Sofiane Abbar, and Mohamed F. Mokbel. 2023. GTI: A Scalable Graph-based Trajectory Imputation. In *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2023, Hamburg, Germany, November 13-16, 2023*, Matthias Renz and Mario A. Nascimento (Eds.). ACM, 70:1–70:10. https://doi.org/10.1145/3589132.3625620

[136] Chris L. Jackins and Steven L. Tanimoto. 1980. Oct-trees and their use in representing three-dimensional objects. *Computer Graphics and Image Processing* 14, 3 (1980), 249–270. https://doi.org/10.1016/0146-664X(80)90055-6

[137] H. V. Jagadish. 1990. Linear Clustering of Objects with Multiple Atributes. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, USA, May 23-25, 1990*, Hector Garcia-Molina and H. V. Jagadish (Eds.). ACM Press, 332–342. https://doi.org/10.1145/93597.98742

[138] Christian S. Jensen, Dan Lin, and Beng Chin Ooi. 2004. Query and Update Efficient B+-Tree Based Indexing of Moving Objects. In *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004, Toronto, Canada, August 31 - September 3 2004*, Mario A. Nascimento, M. Tamer ¨Ozsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer (Eds.). Morgan Kaufmann, 768–779. https://doi.org/10.1016/B978-012088469-8.50068-1

[139] Christian S. Jensen, Hua Lu, and Bin Yang. 2009. Indexing the Trajectories of Moving Objects in Symbolic Indoor Space. In *Advances in Spatial and Temporal Databases, 11th International Symposium, SSTD 2009, Aalborg, Denmark, July 8-10, 2009, Proceedings (Lecture Notes in Computer Science, Vol. 5644)*, Nikos Mamoulis, Thomas Seidl, Torben Bach Pedersen, Kristian Torp, and Ira Assent (Eds.). Springer, 208–227. https://doi.org/10.1007/978-3-642-02982-0_15

[140] Hoyoung Jeung, Man Lung Yiu, Xiaofang Zhou, and Christian S. Jensen. 2010. Path prediction and predictive range querying in road network databases. *VLDB J.* 19, 4 (2010), 585–602. https://doi.org/10.1007/S00778-010-0181-Y

[141] Fengmei Jin, Wen Hua, Jiajie Xu, and Xiaofang Zhou. 2019. Moving Object Linking Based on Historical Trace. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. IEEE, 1058–1069. https://doi.org/10.1109/ICDE.2019.00098

[142] Fengmei Jin, Wen Hua, Thomas Zhou, Jiajie Xu, Matteo Francia, Maria E. Orlowska, and Xiaofang Zhou. 2022. Trajectory-Based Spatiotemporal Entity Linking. *IEEE Trans. Knowl. Data Eng.* 34, 9 (2022), 4499–4513. https://doi.org/10.1109/TKDE.2020.3036633

[143] Dmitri V. Kalashnikov, Sunil Prabhakar, and Susanne E. Hambrusch. 2004. Main Memory Evaluation of Monitoring Queries Over Moving Objects. *Distributed Parallel Databases* 15, 2 (2004), 117–135. https://doi.org/10.1023/B:DAPD.0000013068.25976.88

[144] Ibrahim Kamel and Christos Faloutsos. 1993. On Packing R-trees. In *CIKM 93, Proceedings of the Second International Conference on Information and Knowledge Management, Washington, DC, USA, November 1-5, 1993*, Bharat K. Bhargava, Timothy W. Finin, and Yelena Yesha (Eds.). ACM, 490–499. https://doi.org/10.1145/170088.170403

[145] Ibrahim Kamel and Christos Faloutsos. 1994. Hilbert R-tree: An Improved R-tree using Fractals. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo (Eds.). Morgan Kaufmann, 500–509. http://www.vldb.org/conf/1994/P500.PDF

[146] Shunsuke Kanda, Koh Takeuchi, Keisuke Fujii, and Yasuo Tabei. 2020. Succinct Trit-array Trie for Scalable Trajectory Similarity Search. In *SIGSPATIAL '20: 28th International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, November 3-6, 2020*, Chang-Tien Lu, Fusheng Wang, Goce Trajcevski, Yan Huang, Shawn D. Newsam, and Li Xiong (Eds.). ACM, 518–529. https://doi.org/10.1145/3397536.3422210

[147] David R. Karger and Matthias Ruhl. 2002. Finding nearest neighbors in growth-restricted metrics. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, John H. Reif (Ed.). ACM, 741–750. https://doi.org/10.1145/509907.510013

[148] Jonathan Katz, Amit Sahai, and Brent Waters. 2007. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. *IACR Cryptol. ePrint Arch.* (2007), 404. http://eprint.iacr.org/2007/404

[149] Kyoung-Sook Kim, Si Wan Kim, Tae-wan Kim, and Ki-Joune Li. 2003. Fast indexing and updating method for moving objects on road networks. In *4th International Conference on Web Information Systems Engineering Workshops, WISE 2003 Workshops, Rome, Italy, December 13, 2003*, Giuseppe Santucci, Wolfgang Klas, Michela Bertolotto, Coral Calero, and Luciano Baresi (Eds.). IEEE Computer Society, 34–42. https://doi.org/10.1109/WISEW.2003.1286784

[150] Donald E. Knuth. 1973. *The Art of Computer Programming, Volume I: Fundamental Algorithms, 2nd Edition*. Addison-Wesley. https://www.worldcat.org/oclc/310903895

[151] Satoshi Koide, Yukihiro Tadokoro, and Takayoshi Yoshimura. 2015. SNT-index: Spatio-temporal index for vehicular trajectories on a road network based on substring matching. In *Proceedings of the 1st International ACM SIGSPATIAL Workshop on Smart Cities and Urban Analytics, UrbanGIS@SIGSPATIAL 2015, Bellevue, WA, USA, November 3-6, 2015*. ACM, 1–8. https://doi.org/10.1145/2835022.2835023

[152] Satoshi Koide, Yukihiro Tadokoro, Takayoshi Yoshimura, Chuan Xiao, and Yoshiharu Ishikawa. 2018. Enhanced Indexing and Querying of Trajectories in Road Networks via String Algorithms. *ACM Trans. Spatial Algorithms Syst.* 4, 1 (2018), 3:1–3:41. https://doi.org/10.1145/3200200

[153] George Kollios, Dimitrios Gunopulos, and Vassilis J. Tsotras. 1999. On Indexing Mobile Objects. In *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania, USA*, Victor Vianu and Christos H. Papadimitriou (Eds.). ACM Press, 261–272. https://doi.org/10.1145/303976.304002

[154] George Kollios, Vassilis J. Tsotras, Dimitrios Gunopulos, Alex Delis, and Marios Hadjieleftheriou. 2001. Indexing Animated Objects Using Spatiotemporal Access Methods. *IEEE Trans. Knowl. Data Eng.* 13, 5 (2001), 758–777. https://doi.org/10.1109/69.956099

[155] Benjamin B. Krogh, Nikos Pelekis, Yannis Theodoridis, and Kristian Torp. 2014. Path-based queries on trajectory data. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Dallas/Fort Worth, TX, USA, November 4-7, 2014*, Yan Huang, Markus Schneider, Michael Gertz, John Krumm, and Jagan Sankaranarayanan (Eds.). ACM, 341–350. https://doi.org/10.1145/2666310.2666413

[156] Anil Kumar, Vassilis J. Tsotras, and Christos Faloutsos. 1998. Designing Access Methods for Bitemporal Databases. *IEEE Trans. Knowl. Data Eng.* 10, 1 (1998), 1–20. https://doi.org/10.1109/69.667079

[157] Dongseop Kwon, Sangjun Lee, and Sukho Lee. 2002. Indexing the Current Positions of Moving Objects Using the Lazy Update R-tree. In *Proceedings of the Third International Conference on Mobile Data Management (MDM 2002), Singapore, January 8-11, 2002*. IEEE Computer Society, 113–120. https://doi.org/10.1109/MDM.2002.994387

[158] Thuy Thi Thu Le and Bradford G. Nickerson. 2008. Efficient search of moving objects on a planar graph. In *16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2008, November 5-7, 2008, Irvine, California, USA, Proceedings*, Walid G. Aref, Mohamed F. Mokbel, and Markus Schneider (Eds.). ACM, 41. https://doi.org/10.1145/1463434.1463486

[159] Mong-Li Lee, Wynne Hsu, Christian S. Jensen, Bin Cui, and Keng Lik Teo. 2003. Supporting Frequent Updates in R-Trees: A Bottom-Up Approach. In *Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003, Berlin, Germany, September 9-12, 2003*, Johann Christoph Freytag, Peter C. Lockemann, Serge Abiteboul, Michael J. Carey, Patricia G. Selinger, and Andreas Heuer (Eds.). Morgan Kaufmann, 608–619. https://doi.org/10.1016/B978-012722442-8/50060-4

[160] Scott T. Leutenegger, J. M. Edgington, and Mario Alberto López. 1997. STR: A Simple and Efficient Algorithm for R-Tree Packing. In *Proceedings of the Thirteenth International Conference on Data Engineering, April 7-11, 1997, Birmingham, UK*, W. A. Gray and Per-Åke Larson (Eds.). IEEE Computer Society, 497–506. https://doi.org/10.1109/ICDE.1997.582015

[161] Jiajia Li, Cancan Ni, Dan He, Lei Li, Xiufeng Xia, and Xiaofang Zhou. 2023. Efficient kNN query for moving objects on time-dependent road networks. *VLDB J.* 32, 3 (2023), 575–594. https://doi.org/10.1007/S00778-022-00758-W

[162] Ruiyuan Li, Huajun He, Rubin Wang, Yuchuan Huang, Junwen Liu, Sijie Ruan, Tianfu He, Jie Bao, and Yu Zheng. 2020. JUST: JD Urban Spatio-Temporal Data Engine. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 1558–1569. https://doi.org/10.1109/ICDE48307.2020.00138

[163] Ruiyuan Li, Huajun He, Rubin Wang, Sijie Ruan, Tianfu He, Jie Bao, Junbo Zhang, Liang Hong, and Yu Zheng. 2023. TrajMesa: A Distributed NoSQL-Based Trajectory Data Management System. *IEEE Trans. Knowl. Data Eng.* 35, 1 (2023), 1013–1027. https://doi.org/10.1109/TKDE.2021.3079880

[164] Ruiyuan Li, Jiajun Li, Minxin Zhou, Rubin Wang, Huajun He, Chao Chen, Jie Bao, and Yu Zheng. 2025. Learning-Based Distributed Spatio-Temporal kk Nearest Neighbors Join. *IEEE Trans. Big Data* 11, 2 (2025), 861–878. https://doi.org/10.1109/TBDATA.2024.3442539

[165] Ruiyuan Li, Rubin Wang, Junwen Liu, Zisheng Yu, Huajun He, Tianfu He, Sijie Ruan, Jie Bao, Chao Chen, Fuqiang Gu, Liang Hong, and Yu Zheng. 2021. Distributed Spatio-Temporal k Nearest Neighbors Join. In *SIGSPATIAL '21: 29th International Conference on Advances in Geographic Information Systems, Virtual Event / Beijing, China, November 2-5, 2021*, Xiaofeng Meng, Fusheng Wang, Chang-Tien Lu, Yan Huang, Shashi Shekhar, and Xing Xie (Eds.). ACM, 435–445. https://doi.org/10.1145/3474717.3484209

[166] Zhisheng Li, Ken C. K. Lee, Baihua Zheng, Wang-Chien Lee, Dik Lun Lee, and Xufa Wang. 2011. IR-Tree: An Efficient Index for Geographic Document Search. *IEEE Trans. Knowl. Data Eng.* 23, 4 (2011), 585–599. https://doi.org/10.1109/TKDE.2010.149

[167] Wei Liao, Guifen Tang, Ning Jing, and Zhinong Zhong. 2006. VTPR-Tree: An Efficient Indexing Method for Moving Objects with Frequent Updates. In *Advances in Conceptual Modeling - Theory and Practice, ER 2006 Workshops BP-UML, CoMoGIS, COSS, ECDM, OIS, QoIS, SemWAT, Tucson, AZ, USA, November 6-9, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 4231)*, John F. Roddick, V. Richard Benjamins, Samira Si-Said Cherfi, Roger H. L. Chiang, Christophe Claramunt, Ramez Elmasri, Fabio Grandi, Hyoil Han, Martin Hepp, Miltiadis D. Lytras, Vojislav B. Misic, Geert Poels, Il-Yeol Song, Juan Trujillo, and Christelle Vangenot (Eds.). Springer, 120–129. https://doi.org/10.1007/11908883_15

[168] Sejoon Lim and Daniela Rus. 2012. Stochastic distributed multi-agent planning and applications to traffic. In *IEEE International Conference on Robotics and Automation, ICRA 2012, 14-18 May, 2012, St. Paul, Minnesota, USA*. IEEE, 2873–2879. https://doi.org/10.1109/ICRA.2012.6224710

[169] B. Lin, H. Mokhtar, R. Pelaez-Aguilera, and J. Su. 2003. Querying moving objects with uncertainty. In *2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No.03CH37484)*, Vol. 4. 2783–2787 Vol.4. https://doi.org/10.1109/VETECF.2003.1286090

[170] Bin Lin and Jianwen Su. 2005. Handling frequent updates of moving objects. In *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*, Otthein Herzog, Hans-Jörg Schek, Norbert Fuhr, Abdur Chowdhury, and Wilfried Teiken (Eds.). ACM, 493–500. https://doi.org/10.1145/1099554.1099691

[171] Dan Lin, Christian S. Jensen, Beng Chin Ooi, and Simonas Saltenis. 2005. Efficient indexing of the historical, present, and future positions of moving objects. In *6th International Conference on Mobile Data Management (MDM 2005), Ayia Napa, Cyprus, May 9-13, 2005*, Panos K. Chrysanthis and George Samaras (Eds.). ACM, 59–66. https://doi.org/10.1145/1071246.1071256

[172] Dan Lin, Christian S. Jensen, Rui Zhang, Lu Xiao, and Jiaheng Lu. 2011. A MovingObject Index for Efficient Query Processing with Peer-Wise Location Privacy. *Proc. VLDB Endow.* 5, 1 (2011), 37–48. https://doi.org/10.14778/2047485.2047489

[173] Dan Lin, Rui Zhang, and Aoying Zhou. 2006. Indexing Fast Moving Objects for kNN Queries Based on Nearest Landmarks. *GeoInformatica* 10, 4 (2006), 423–445. https://doi.org/10.1007/S10707-006-0341-9

[174] Hung Yi Lin. 2009. Indexing the Trajectories of Moving Objects. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Vol. 2. IAENG, Hong Kong, 732–737.

[175] Meri Lisi. 2007. Some remarks on the Cantor pairing function. *Le Matematiche* 62, 1 (2007), 55–65.

[176] Huiwen Liu, Jiajie Xu, Kai Zheng, Chengfei Liu, Lan Du, and Xian Wu. 2017. Semantic-aware Query Processing for Activity Trajectories. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, Maarten de Rijke, Milad Shokouhi, Andrew Tomkins, and Min Zhang (Eds.). ACM, 283–292. https://doi.org/10.1145/3018661.3018678

[177] Kaiqi Liu, Panrong Tong, Mo Li, Yue Wu, and Jianqiang Huang. 2023. ST4ML: Machine Learning Oriented Spatio-Temporal Data Processing at Scale. *Proc. ACM Manag. Data* 1, 1 (2023), 87:1–87:28. https://doi.org/10.1145/3588941

[178] Zhao-Hong Liu, Xiao-Li Liu, Jun-Wei Ge, and Hae-Young Bae. 2005. Indexing Large Moving Objects from Past to Future with PCFI+-Index. In *Advances in Data Management 2005, Proceedings of the Eleventh International Conference on Management of Data, January 6, 7, and 8, 2005, Goa, India*, Jayant R. Haritsa and T. M. Vijayaraman (Eds.). Computer Society of India, 131–137. http://comad2005.persistent.co.in/COMAD2005Proc/pages131-137.pdf

[179] LocationTech. 2001. *JTS Topology Suite*. Eclipse Foundation. https://github.com/locationtech/jts An open source Java library for creating and manipulating vector geometry.

[180] David B. Lomet and Betty Salzberg. 1989. Access Methods for Multiversion Data. In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon, USA, May 31 - June 2, 1989*, James Clifford, Bruce G. Lindsay, and David Maier (Eds.). ACM Press, 315–324. https://doi.org/10.1145/67544.66956

[181] Siqiang Luo, Ben Kao, Guoliang Li, Jiafeng Hu, Reynold Cheng, and Yudian Zheng. 2018. TOAIN: A Throughput Optimizing Adaptive Index for Answering Dynamic kNN Queries on Road Networks. *Proc. VLDB Endow.* 11, 5 (2018), 594–606. https://doi.org/10.1145/3187009.3177736

[182] Wuman Luo, Haoyu Tan, Lei Chen, and Lionel M. Ni. 2013. Finding time period-based most frequent path in big trajectory data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*, Kenneth A. Ross, Divesh Srivastava, and Dimitris Papadias (Eds.). ACM, 713–724. https://doi.org/10.1145/2463676.2465287

[183] Chunyang Ma, Hua Lu, Lidan Shou, and Gang Chen. 2013. KSQ: Top-(k) Similarity Query on Uncertain Trajectories. *IEEE Trans. Knowl. Data Eng.* 25, 9 (2013), 2049–2062. https://doi.org/10.1109/TKDE.2012.152

[184] Shuo Ma, Yu Zheng, and Ouri Wolfson. 2013. T-share: A large-scale dynamic taxi ridesharing service. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, Christian S. Jensen, Christopher M. Jermaine, and Xiaofang Zhou (Eds.). IEEE Computer Society, 410–421. https://doi.org/10.1109/ICDE.2013.6544843

[185] Amr Magdy, Ahmed M. Aly, Mohamed F. Mokbel, Sameh Elnikety, Yuxiong He, Suman Nath, and Walid G. Aref. 2016. GeoTrend: spatial trending queries on real-time microblogs. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2016, Burlingame, California, USA, October 31 - November 3, 2016*, Siva Ravada, Mohammed Eunus Ali, Shawn D. Newsam, Matthias Renz, and Goce Trajcevski (Eds.). ACM, 7:1–7:10. https://doi.org/10.1145/2996913.2996986

[186] Amr Magdy, Mohamed F. Mokbel, Sameh Elnikety, Suman Nath, and Yuxiong He. 2014. Mercury: A memory-constrained spatio-temporal real-time search on microblogs. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, Isabel F. Cruz, Elena Ferrari, Yufei Tao, Elisa Bertino, and Goce Trajcevski (Eds.). IEEE Computer Society, 172–183. https://doi.org/10.1109/ICDE.2014.6816649

[187] Ahmed R. Mahmood. 2017. SRC: tornado: a distributed spatio-textual stream processing system. *ACM SIGSPATIAL Special* 9, 3 (2017), 8–9. https://doi.org/10.1145/3178392.3178397

[188] Ahmed R. Mahmood, Ahmed M. Aly, Tatiana Kuznetsova, Saleh M. Basalamah, and Walid G. Aref. 2018. Disk-Based Indexing of Recent Trajectories. *ACM Trans. Spatial Algorithms Syst.* 4, 3 (2018), 7:1–7:27. https://doi.org/10.1145/3234941

[189] Ahmed R. Mahmood, Ahmed M. Aly, Thamir Qadah, El Kindi Rezig, Anas Daghistani, Amgad Madkour, Ahmed S. Abdelhamid, Mohamed S. Hassan, Walid G. Aref, and Saleh M. Basalamah. 2015. Tornado: A Distributed Spatio-Textual Stream Processing System. *Proc. VLDB Endow.* 8, 12 (2015), 2020–2023. https://doi.org/10.14778/2824032.2824126

[190] Ahmed R. Mahmood, Walid G. Aref, Ahmed M. Aly, and Saleh M. Basalamah. 2014. Indexing recent trajectories of moving objects. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Dallas/Fort Worth, TX, USA, November 4-7, 2014*, Yan Huang, Markus Schneider, Michael Gertz, John Krumm, and Jagan Sankaranarayanan (Eds.). ACM, 393–396. https://doi.org/10.1145/2666310.2666427

[191] Ahmed R. Mahmood, Anas Daghistani, Ahmed M. Aly, MingJie Tang, Saleh M. Basalamah, Sunil Prabhakar, and Walid G. Aref. 2018. Adaptive processing of spatial-keyword data over a distributed streaming cluster. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018*, Farnoush Banaei Kashani, Erik G. Hoel, Ralf Hartmut Güting, Roberto Tamassia, and Li Xiong (Eds.). ACM, 219–228. https://doi.org/10.1145/3274895.3274932

[192] Ahmed R. Mahmood, Sri Punni, and Walid G. Aref. 2019. Spatio-temporal access methods: a survey (2010 - 2017). *GeoInformatica* 23, 1 (2019), 1–36. https://doi.org/10.1007/S10707-018-0329-2

[193] Udi Manber and Eugene W. Myers. 1993. Suffix Arrays: A New Method for On-Line String Searches. *SIAM J. Comput.* 22, 5 (1993), 935–948. https://doi.org/10.1137/0222058

[194] Udi Manber and Gene Myers. 1990. Suffix Arrays: A New Method for On-Line String Searches. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, 22-24 January 1990, San Francisco, California, USA*, David S. Johnson (Ed.). SIAM, 319–327. http://dl.acm.org/

citation.cfm?id=320176.320218

[195] Yannis Manolopoulos and G. Kapetanakis. 1990. Overlapping B+trees for temporal data. In *Next Decade in Information Technology: Proceedings of the 5th Jerusalem Conference on Information Technology 1990, Jerusalem, October 22-25, 1990*, Joshua Maor and Abraham Peled (Eds.). IEEE Computer Society, 491–498. https://doi.org/10.1109/JCIT.1990.128320

[196] Edward M. McCreight. 1976. A Space-Economical Suffix Tree Construction Algorithm. *J. ACM* 23, 2 (1976), 262–272. https://doi.org/10.1145/321941.321946

[197] Edward M. McCreight. 1985. Priority Search Trees. *SIAM J. Comput.* 14, 2 (1985), 257–276. https://doi.org/10.1137/0214021

[198] Paras Mehta, Dimitrios Skoutas, and Agnès Voisard. 2015. Spatio-temporal keyword queries for moving objects. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, Bellevue, WA, USA, November 3-6, 2015*, Jie Bao, Christian Sengstock, Mohammed Eunus Ali, Yan Huang, Michael Gertz, Matthias Renz, and Jagan Sankaranarayanan (Eds.). ACM, 55:1–55:4. https://doi.org/10.1145/2820783.2820845

[199] Puya Memarzia, Maria Patrou, Md. Mahbub Alam, Suprio Ray, Virendra C. Bhavsar, and Kenneth B. Kent. 2019. Toward Efficient Processing of Spatio-Temporal Workloads in a Distributed In-Memory System. In *20th IEEE International Conference on Mobile Data Management, MDM 2019, Hong Kong, SAR, China, June 10-13, 2019*. IEEE, 118–127. https://doi.org/10.1109/MDM.2019.00-66

[200] Raghav Mittal, Ayaan Kakkar, Mukesh K. Mohania, Ladjel Bellatreche, and Yoshiharu Ishikawa. 2025. Enriching Spatial Indexes For User-Centric And Context-Aware Points Of Interest Search. In *Proceedings of the 37th International Conference on Scalable Scientific Data Management, SSDBM 2025, Columbus, OH, USA, June 23-25, 2025*, Suren Byna, Anthony Kougkas, Sarah Neuwirth, Venkat Vishwanath, Jalil Boukhobza, Alfredo Cuzzocrea, Dong Dai, and Jean Bez (Eds.). ACM, 6:1–6:12. https://doi.org/10.1145/3733723.3733727

[201] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nat.* 518, 7540 (2015), 529–533. https://doi.org/10.1038/NATURE14236

[202] Mohamed F. Mokbel, Thanaa M. Ghanem, and Walid G. Aref. 2003. Spatio-Temporal Access Methods. *IEEE Data Eng. Bull.* 26, 2 (2003), 40–49. http://sites.computer.org/debull/A03june/arefF.ps

[203] Donald R. Morrison. 1968. PATRICIA - Practical Algorithm To Retrieve Information Coded in Alphanumeric. *J. ACM* 15, 4 (1968), 514–534. https://doi.org/10.1145/321479.321481

[204] Guy M Morton. 1966. A computer oriented geodetic data base and a new technique in file sequencing. (1966).

[205] Naoto Mukai, Jun Feng, and Toyohide Watanabe. 2004. Heuristic Approach Based on Lambda-Interchange for VRTPR-Tree on Specific Vehicle Routing Problem with Time Windows. In *Innovations in Applied Artificial Intelligence, 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2004, Ottawa, Canada, May 17-20, 2004. Proceedings (Lecture Notes in Computer Science, Vol. 3029)*. Springer, 229–238. https://doi.org/10.1007/978-3-540-24677-0_25

[206] Naoto Mukai, Jun Feng, and Toyohide Watanabe. 2004. Indexing Approach for Delivery Demands with Time Constraints. In *PRICAI 2004: Trends in Artificial Intelligence, 8th Pacific Rim International Conference on Artificial Intelligence, Auckland, New Zealand, August 9-13, 2004, Proceedings (Lecture Notes in Computer Science, Vol. 3157)*, Chengqi Zhang, Hans W. Guesgen, and Wai-Kiang Yeap (Eds.). Springer, 95–103. https://doi.org/10.1007/978-3-540-28633-2_12

[207] Mario A. Nascimento and Jefferson R. O. Silva. 1998. Towards historical R-trees. In *Proceedings of the 1998 ACM symposium on Applied Computing, SAC'98, Atlanta, GA, USA, February 27 - March 1, 1998*, K. M. George and Gary B. Lamont (Eds.). ACM, 235–240. https://doi.org/10.1145/330560.330692

[208] Mario A. Nascimento, Jefferson R. O. Silva, and Yannis Theodoridis. 1999. Evaluation of Access Structures for Discretely Moving Points. In *Spatio-Temporal Database Management, International Workshop STDBM'99, Edinburgh, Scotland, September 10-11, 1999, Proceedings (Lecture Notes in Computer Science, Vol. 1678)*, Michael H. B¨ohlen, Christian S. Jensen, and Michel Scholl (Eds.). Springer, 171–188. https://doi.org/10.1007/3-540-48344-6_10

[209] Thi Nguyen, Zhen He, and Yi-Ping Phoebe Chen. 2013. S^{e}TPR*-tree: Efficient Buffering for Spatiotemporal Indexes Via Shared Execution. *Comput. J.* 56, 1 (2013), 115–137. https://doi.org/10.1093/COMJNL/BXS020

[210] Thi Nguyen, Zhen He, Rui Zhang, and Phillip Ward. 2012. Boosting Moving Object Indexing through Velocity Partitioning. *Proc. VLDB Endow.* 5, 9 (2012), 860–871. https://doi.org/10.14778/2311906.2311913

[211] Long-Van Nguyen-Dinh, Walid G. Aref, and Mohamed F. Mokbel. 2010. Spatio-Temporal Access Methods: Part 2 (2003 - 2010). *IEEE Data Eng. Bull.* 33, 2 (2010), 46–55. http://sites.computer.org/debull/A10june/Aref.pdf

[212] Jinfeng Ni and Chinya V. Ravishankar. 2005. PA-Tree: A Parametric Indexing Scheme for Spatio-temporal Trajectories. In *Advances in Spatial and Temporal Databases, 9th International Symposium, SSTD 2005, Angra dos Reis, Brazil, August 22-24, 2005, Proceedings (Lecture Notes in Computer Science, Vol. 3633)*, Claudia Bauzer Medeiros, Max J. Egenhofer, and Elisa Bertino (Eds.). Springer, 254–272. https://doi.org/10.1007/11535331_15

[213] Jürg Nievergelt, Hans Hinterberger, and Kenneth C. Sevcik. 1984. The Grid File: An Adaptable, Symmetric Multikey File Structure. *ACM Trans. Database Syst.* 9, 1 (1984), 38–71. https://doi.org/10.1145/348.318586

[214] Shunya Nishio, Daichi Amagata, and Takahiro Hara. 2022. Lamps: Location-Aware Moving Top-k Pub/Sub. *IEEE Trans. Knowl. Data Eng.* 34, 1 (2022), 352–364. https://doi.org/10.1109/TKDE.2020.2979176

[215] Shunya Nishio, Daichi Amagata, and Takahiro Hara. 2023. Lamps: Location-Aware Moving Top-k Pub/Sub (Extended abstract). In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. IEEE, 3809–3810. https://doi.org/10.1109/ICDE55515.2023.00331

[216] Zhila Nouri and Yi-Cheng Tu. 2018. GPU-based parallel indexing for concurrent spatial query processing. In *Proceedings of the 30th International Conference on Scientific and Statistical Database Management, SSDBM 2018, Bozen-Bolzano, Italy, July 09-11, 2018*, Dimitris Sacharidis, Johann Gamper, and Michael H. Böhlen (Eds.). ACM, 23:1–23:12. https://doi.org/10.1145/3221269.3221296

[217] Patrick E. O'Neil, Edward Cheng, Dieter Gawlick, and Elizabeth J. O'Neil. 1996. The Log-Structured Merge-Tree (LSM-Tree). *Acta Informatica* 33, 4 (1996), 351–385. https://doi.org/10.1007/S002360050048

[218] Mariam Orabi, Zaher Al Aghbari, and Ibrahim Kamel. 2024. SkyEye: continuous processing of moving spatial-keyword queries over moving objects. *GeoInformatica* 28, 4 (2024), 559–603. https://doi.org/10.1007/S10707-024-00512-0

[219] Mariam Orabi, Zaher Al Aghbari, Ibrahim Kamel, and Djedjiga Mouheb. 2024. Keeping an eye on moving objects: processing continuous spatial-keyword range queries. *GeoInformatica* 28, 1 (2024), 117–143. https://doi.org/10.1007/S10707-023-00499-0

[220] Jack A. Orenstein and T. H. Merrett. 1984. A Class of Data Structures for Associative Searching. In *Proceedings of the Third ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, April 2-4, 1984, Waterloo, Ontario, Canada*, Daniel J. Rosenkrantz and Ronald Fagin (Eds.). ACM, 181–190. https://doi.org/10.1145/588011.588037

[221] Dian Ouyang, Lu Qin, Lijun Chang, Xuemin Lin, Ying Zhang, and Qing Zhu. 2018. When Hierarchy Meets 2-Hop-Labeling: Efficient Shortest Distance Queries on Road Networks. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 709–724. https://doi.org/10.1145/3183713.3196913

[222] Jignesh M. Patel, Yun Chen, and V. Prasad Chakka. 2004. STRIPES: An Efficient Index for Predicted Trajectories. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, Gerhard Weikum, Arnd Christian K¨onig, and Stefan Deßloch (Eds.). ACM, 637–646. https://doi.org/10.1145/1007568.1007639

[223] Mayur Patil. 2022. Advanced Conjunctive Boolean Streaming Spatial Keyword Processing. In *23rd IEEE International Conference on Mobile Data Management, MDM 2022, Paphos, Cyprus, June 6-9, 2022*. IEEE, 41–43. https://doi.org/10.1109/MDM55031.2022.00027

[224] Maria Patrou, Md. Mahbub Alam, Puya Memarzia, Suprio Ray, Virendra C. Bhavsar, Kenneth B. Kent, and Gerhard W. Dueck. 2018. DISTIL: a distributed in-memory data processing system for location-based services. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018*, Farnoush Banaei Kashani, Erik G. Hoel, Ralf Hartmut Güting, Roberto Tamassia, and Li Xiong (Eds.). ACM, 496–499. https://doi.org/10.1145/3274895.3274961

[225] Kostas Patroumpas and Timos K. Sellis. 2009. Monitoring Orientation of Moving Objects around Focal Points. In *Advances in Spatial and Temporal Databases, 11th International Symposium, SSTD 2009, Aalborg, Denmark, July 8-10, 2009, Proceedings (Lecture Notes in Computer Science, Vol. 5644)*, Nikos Mamoulis, Thomas Seidl, Torben Bach Pedersen, Kristian Torp, and Ira Assent (Eds.). Springer, 228–246. https://doi.org/10.1007/978-3-642-02982-0_16

[226] Mindaugas Pelanis, Simonas Saltenis, and Christian S. Jensen. 2006. Indexing the past, present, and anticipated future positions of moving objects. *ACM Trans. Database Syst.* 31, 1 (2006), 255–298. https://doi.org/10.1145/1132863.1132870

[227] Nikos Pelekis, Elias Frentzos, Nikos Giatrakos, and Yannis Theodoridis. 2015. HERMES: A Trajectory DB Engine for Mobility-Centric Applications. *Int. J. Knowl. Based Organ.* 5, 2 (2015), 19–41. https://doi.org/10.4018/IJKBO.2015040102

[228] Dieter Pfoser, Christian S. Jensen, and Yannis Theodoridis. 2000. Novel Approaches to the Indexing of Moving Object Trajectories. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang (Eds.). Morgan Kaufmann, 395–406. http://www.vldb.org/conf/2000/P395.pdf

[229] Iulian Sandu Popa, Karine Zeitouni, Vincent Oria, Dominique Barth, and Sandrine Vial. 2010. PARINET: A tunable access method for in-network trajectories. In *Proceedings of the 26th International Conference on Data Engineering, ICDE 2010, March 1-6, 2010, Long Beach, California, USA*, Feifei Li, Mirella M. Moro, Shahram Ghandeharizadeh, Jayant R. Haritsa, Gerhard Weikum, Michael J. Carey, Fabio Casati, Edward Y. Chang, Ioana Manolescu, Sharad Mehrotra, Umeshwar Dayal, and Vassilis J. Tsotras (Eds.). IEEE Computer Society, 177–188. https://doi.org/10.1109/ICDE.2010.5447885

[230] Iulian Sandu Popa, Karine Zeitouni, Vincent Oria, Dominique Barth, and Sandrine Vial. 2011. Indexing in-network trajectory flows. *VLDB J.* 20, 5 (2011), 643–669. https://doi.org/10.1007/S00778-011-0236-8

[231] Kriengkrai Porkaew, Iosif Lazaridis, and Sharad Mehrotra. 2001. Querying Mobile Objects in Spatio-Temporal Databases. In *Advances in Spatial and Temporal Databases, 7th International Symposium, SSTD 2001, Redondo Beach, CA, USA, July 12-15, 2001, Proceedings (Lecture Notes in Computer Science, Vol. 2121)*, Christian S. Jensen, Markus Schneider, Bernhard Seeger, and Vassilis J. Tsotras (Eds.). Springer, 59–78. https://doi.org/10.1007/3-540-47724-1_4

[232] S Prabhakar, Y Xia, D Kalashnikov, WG Aref, and S Hambrusch. 2000. Queries as data and expanding indexes: techniques for continuous queries on moving objects. *Purdue University* (2000).

[233] Sunil Prabhakar, Yuni Xia, Dmitri V. Kalashnikov, Walid G. Aref, and Susanne E. Hambrusch. 2002. Query Indexing and Velocity Constrained Indexing: Scalable Techniques for Continuous Queries on Moving Objects. *IEEE Trans. Computers* 51, 10 (2002), 1124–1140. https://doi.org/10.1109/TC.2002.1039840

[234] Cecilia Magdalena Procopiuc, Pankaj K. Agarwal, and Sariel Har-Peled. 2002. STAR-Tree: An Efficient Self-Adjusting Index for Moving Objects. In *Algorithm Engineering and Experiments, 4th International Workshop, ALENEX 2002, San Francisco, CA, USA, January 4-5, 2002, Revised Papers (Lecture Notes in Computer Science, Vol. 2409)*, David M. Mount and Clifford Stein (Eds.). Springer, 178–193. https://doi.org/10.1007/3-540-45643-0_14

[235] Chiara Pugliese, Francesco Lettich, Fabio Pinelli, and Chiara Renso. 2023. Summarizing Trajectories Using Semantically Enriched Geographical Context. In *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2023, Hamburg,*

Germany, November 13-16, 2023, Matthias Renz and Mario A. Nascimento (Eds.). ACM, 44:1–44:10. https://doi.org/10.1145/3589132.3625587

[236] Sayan Ranu, Deepak P, Aditya D. Telang, Prasad Deshpande, and Sriram Raghavan. 2015. Indexing and matching trajectories under inconsistent sampling rates. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, Johannes Gehrke, Wolfgang Lehner, Kyuseok Shim, Sang Kyun Cha, and Guy M. Lohman (Eds.). IEEE Computer Society, 999–1010. https://doi.org/10.1109/ICDE.2015.7113351

[237] Suprio Ray, Rolando Blanco, and Anil K. Goel. 2013. Enhanced database support for location-based services. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on GeoStreaming, IWGS 2013, November 5, 2013, Orlando, FL, USA*, Farnoush Banaei Kashani, Anas Basalamah, and Chengyang Zhang (Eds.). ACM, 22–25. https://doi.org/10.1145/2534303.2534308

[238] Suprio Ray, Rolando Blanco, and Anil K. Goel. 2017. High performance location-based services in a main-memory database. *GeoInformatica* 21, 2 (2017), 293–322. https://doi.org/10.1007/S10707-016-0278-6

[239] Miguel Romero, Nieves R. Brisaboa, and M. Andrea Rodríguez. 2012. The SMO-index: a succinct moving object structure for timestamp and interval queries. In *SIGSPATIAL 2012 International Conference on Advances in Geographic Information Systems (formerly known as GIS), SIGSPATIAL'12, Redondo Beach, CA, USA, November 7-9, 2012*, Isabel F. Cruz, Craig A. Knoblock, Peer Kr¨oger, Egemen Tanin, and Peter Widmayer (Eds.). ACM, 498–501. https://doi.org/10.1145/2424321.2424399

[240] Nick Roussopoulos, Stephen Kelley, and Frédéic Vincent. 1995. Nearest Neighbor Queries. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, USA, May 22-25, 1995*, Michael J. Carey and Donovan A. Schneider (Eds.). ACM Press, 71–79. https://doi.org/10.1145/223784.223794

[241] Nick Roussopoulos and Daniel Leifker. 1985. Direct Spatial Search on Pictorial Databases Using Packed R-Trees. In *Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data, Austin, Texas, USA, May 28-31, 1985*, Shamkant B. Navathe (Ed.). ACM Press, 17–31. https://doi.org/10.1145/318898.318900

[242] Simonas Saltenis and Christian S. Jensen. 2002. Indexing of Moving Objects for Location-Based Services. In *Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26 - March 1, 2002*, Rakesh Agrawal and Klaus R. Dittrich (Eds.). IEEE Computer Society, 463–472. https://doi.org/10.1109/ICDE.2002.994759

[243] Simonas Saltenis, Christian S. Jensen, Scott T. Leutenegger, and Mario Alberto López. 2000. Indexing the Positions of Continuously Moving Objects. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein (Eds.). ACM, 331–342. https://doi.org/10.1145/342009.335427

[244] Hanan Samet. 1984. The Quadtree and Related Hierarchical Data Structures. *ACM Comput. Surv.* 16, 2 (1984), 187–260. https://doi.org/10.1145/356924.356930

[245] Peter Sanders and Dominik Schultes. 2005. Highway Hierarchies Hasten Exact Shortest Path Queries. In *Algorithms - ESA 2005, 13th Annual European Symposium, Palma de Mallorca, Spain, October 3-6, 2005, Proceedings (Lecture Notes in Computer Science, Vol. 3669)*, Gerth Stølting Brodal and Stefano Leonardi (Eds.). Springer, 568–579. https://doi.org/10.1007/11561071_51

[246] Peter Sanders and Dominik Schultes. 2006. Engineering Highway Hierarchies. In *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings (Lecture Notes in Computer Science, Vol. 4168)*, Yossi Azar and Thomas Erlebach (Eds.). Springer, 804–816. https://doi.org/10.1007/11841036_71

[247] Peter Sanders and Dominik Schultes. 2012. Engineering highway hierarchies. *ACM J. Exp. Algorithmics* 17, 1 (2012). https://doi.org/10.1145/2133803.2330080

[248] Vijay Saraswat, George Almasi, Ganesh Bikshandi, Calin Cascaval, David Cunningham, David Grove, Sreedhar Kodali, Igor Peshansky, and Olivier Tardieu. 2010. The asynchronous partitioned global address space model. In *The First Workshop on Advances in Message Passing*. 1–8.

[249] Philip Schmiegelt, Andreas Behrend, Bernhard Seeger, and Wolfgang Koch. 2014. A concurrently updatable index structure for predicted paths of moving objects. *Data Knowl. Eng.* 93 (2014), 80–96. https://doi.org/10.1016/J.DATAK.2014.07.007

[250] Dominik Schultes and Peter Sanders. 2007. Dynamic highway-node routing. In *International Workshop on Experimental and Efficient Algorithms*. Springer, 66–79.

[251] Russell Sears and Raghu Ramakrishnan. 2012. bLSM: a general purpose log structured merge tree. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, K. Selçuk Candan, Yi Chen, Richard T. Snodgrass, Luis Gravano, and Ariel Fuxman (Eds.). ACM, 217–228. https://doi.org/10.1145/2213836.2213862

[252] Dong-Min Seo, Seok-Il Song, Yong-Hun Park, Jae-Soo Yoo, and Myoung-Ho Kim. 2008. BdH-Tree: A B+-Tree Based Indexing Method for Very Frequent Updates of Moving Objects. In *Proceedings of the International Symposium on Computer Science and Its Applications (CSA '08)*. IEEE Computer Society, USA, 314–319. https://doi.org/10.1109/CSA.2008.51

[253] Zeyuan Shang, Guoliang Li, and Zhifeng Bao. 2018. DITA: Distributed In-Memory Trajectory Analytics. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 725–740. https://doi.org/10.1145/3183713.3183743

[254] Claude E. Shannon. 1948. A mathematical theory of communication. *Bell Syst. Tech. J.* 27, 3 (1948), 379–423. https://doi.org/10.1002/J.1538-7305.1948.TB01338.X

[255] Bilong Shen, Ying Zhao, Guoliang Li, Weimin Zheng, Yue Qin, Bo Yuan, and Yongming Rao. 2017. V-Tree: Efficient kNN Search on Moving Objects with Road-Network Constraints. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*. IEEE Computer Society, 609–620. https://doi.org/10.1109/ICDE.2017.115

[256] Jaewoo Shin, Jianguo Wang, and Walid G. Aref. 2021. The LSM RUM-Tree: A Log Structured Merge R-Tree for Update-intensive Spatial Workloads. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 2285–2290. https://doi.org/10.1109/ICDE51399.2021.00238

[257] Jaewoo Shin, Libin Zhou, Jianguo Wang, and Walid G. Aref. 2025. An update-intensive LSM-based R-tree index. *VLDB J.* 34, 1 (2025), 7. https://doi.org/10.1007/S00778-024-00876-7

[258] Stylianos Sideridis, Nikos Pelekis, and Yannis Theodoridis. 2016. On querying and mining semantic-aware mobility timelines. *Int. J. Data Sci. Anal.* 2, 1-2 (2016), 29–44. https://doi.org/10.1007/S41060-016-0030-1

[259] Darius Sidlauskas, Kenneth A. Ross, Christian S. Jensen, and Simonas Saltenis. 2011. Thread-Level Parallel Indexing of Update Intensive Moving-Object Workloads. In *Advances in Spatial and Temporal Databases - 12th International Symposium, SSTD 2011, Minneapolis, MN, USA, August 24-26, 2011, Proceedings (Lecture Notes in Computer Science, Vol. 6849)*, Dieter Pfoser, Yufei Tao, Kyriakos Mouratidis, Mario A. Nascimento, Mohamed F. Mokbel, Shashi Shekhar, and Yan Huang (Eds.). Springer, 186–204. https://doi.org/10.1007/978-3-642-22922-0_12

[260] Darius Sidlauskas, Simonas Saltenis, and Christian S. Jensen. 2012. Parallel main-memory indexing for moving-object query and update workloads. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, K. Selçuk Candan, Yi Chen, Richard T. Snodgrass, Luis Gravano, and Ariel Fuxman (Eds.). ACM, 37–48. https://doi.org/10.1145/2213836.2213842

[261] Yasin N. Silva, Ahmed M. Aly, Walid G. Aref, and Per-Ake Larson. 2010. SimDB: a similarity-aware database system. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data* (Indianapolis, Indiana, USA) *(SIGMOD '10)*. Association for Computing Machinery, New York, NY, USA, 1243–1246. https://doi.org/10.1145/1807167.1807330

[262] Yasin N. Silva, Walid G. Aref, Per-Ake Larson, Spencer S. Pearson, and Mohamed H. Ali. 2013. Similarity queries: their conceptual evaluation, transformations, and processing. *The VLDB Journal* 22 (2013), 395–420. https://doi.org/10.1007/s00778-012-0296-4

[263] Yasin N. Silva, Xiaopeng Xiong, and Walid G. Aref. 2009. The RUM-tree: supporting frequent updates in R-trees using memos. *VLDB J.* 18, 3 (2009), 719–738. https://doi.org/10.1007/S00778-008-0120-3

[264] Manish Singh, Qiang Zhu, and H. V. Jagadish. 2012. SWST: A Disk Based Index for Sliding Window Spatio-Temporal Data. In *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, Anastasios Kementsietsidis and Marcos Antonio Vaz Salles (Eds.). IEEE Computer Society, 342–353. https://doi.org/10.1109/ICDE.2012.98

[265] Anders Skovsgaard, Darius Sidlauskas, and Christian S. Jensen. 2014. Scalable top-k spatio-temporal term querying. In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, Isabel F. Cruz, Elena Ferrari, Yufei Tao, Elisa Bertino, and Goce Trajcevski (Eds.). IEEE Computer Society, 148–159. https://doi.org/10.1109/ICDE.2014.6816647

[266] Xiaozhao Song, Jiajie Xu, Rui Zhou, Chengfei Liu, Kai Zheng, Pengpeng Zhao, and Nickolas J. G. Falkner. 2020. Collective spatial keyword search on activity trajectories. *GeoInformatica* 24, 1 (2020), 61–84. https://doi.org/10.1007/S10707-019-00358-X

[267] Zhexuan Song and Nick Roussopoulos. 2001. Hashing Moving Objects. In *Mobile Data Management, Second International Conference, MDM 2001, Hong Kong, China, January 8-10, 2001, Proceedings (Lecture Notes in Computer Science, Vol. 1987)*, Kian-Lee Tan, Michael J. Franklin, and John C. S. Lui (Eds.). Springer, 161–172. https://doi.org/10.1007/3-540-44498-X_13

[268] Zhexuan Song and Nick Roussopoulos. 2003. SEB-tree: An Approach to Index Continuously Moving Objects. In *Mobile Data Management, 4th International Conference, MDM 2003, Melbourne, Australia, January 21-24, 2003, Proceedings (Lecture Notes in Computer Science, Vol. 2574)*, Ming-Syan Chen, Panos K. Chrysanthis, Morris Sloman, and Arkady B. Zaslavsky (Eds.). Springer, 340–344. https://doi.org/10.1007/3-540-36389-0_25

[269] Elena Vladislavivna Strzheletska and Vassilis J. Tsotras. 2017. Efficient Processing of Reachability Queries with Meetings. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2017, Redondo Beach, CA, USA, November 7-10, 2017*, Erik G. Hoel, Shawn D. Newsam, Siva Ravada, Roberto Tamassia, and Goce Trajcevski (Eds.). ACM, 22:1–22:10. https://doi.org/10.1145/3139958.3139982

[270] Panagiotis Tampakis, Christos Doulkeridis, Nikos Pelekis, and Yannis Theodoridis. 2020. Distributed Subtrajectory Join on Massive Datasets. *ACM Trans. Spatial Algorithms Syst.* 6, 2 (2020), 8:1–8:29. https://doi.org/10.1145/3373642

[271] Kailei Tang, Zhiyan Dong, Wenxiang Shi, and Zhongxue Gan. 2023. A Dynamic Grid Index for CkNN Queries on Large-Scale Road Networks with Moving Objects. *Applied Sciences* 13, 8 (2023). https://doi.org/10.3390/app13084946

[272] S. Tanimoto and T. Pavlidis. 1975. A hierarchical data structure for picture processing. *Computer Graphics and Image Processing* 4, 2 (1975), 104–119. https://doi.org/10.1016/S0146-664X(75)80003-7

[273] Yufei Tao, Christos Faloutsos, Dimitris Papadias, and Bin Liu. 2004. Prediction and Indexing of Moving Objects with Unknown Motion Patterns. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, Gerhard Weikum, Arnd Christian K¨onig, and Stefan Deßloch (Eds.). ACM, 611–622. https://doi.org/10.1145/1007568.1007637

[274] Yufei Tao and Dimitris Papadias. 2001. Efficient Historical R-trees. In *Proceedings of the 13th International Conference on Scientific and Statistical Database Management, July 18-20, 2001, George Mason University, Fairfax, Virginia, USA*. IEEE Computer Society, 223–232. https://doi.org/10.1109/SSDM.2001.938554

[275] Yufei Tao and Dimitris Papadias. 2001. MV3R-Tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries. In *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, Peter M. G. Apers, Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Kotagiri Ramamohanarao, and Richard T. Snodgrass (Eds.). Morgan Kaufmann, 431–440. http://www.vldb.org/conf/2001/P431.pdf

[276] Yufei Tao and Dimitris Papadias. 2002. Time-parameterized queries in spatio-temporal databases. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, USA, June 3-6, 2002*, Michael J. Franklin, Bongki Moon, and Anastassia

Ailamaki (Eds.). ACM, 334–345. https://doi.org/10.1145/564691.564730

[277] Yufei Tao, Dimitris Papadias, and Jimeng Sun. 2003. The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries. In *Proceedings of 29th International Conference on Very Large Data Bases, VLDB 2003, Berlin, Germany, September 9-12, 2003*, Johann Christoph Freytag, Peter C. Lockemann, Serge Abiteboul, Michael J. Carey, Patricia G. Selinger, and Andreas Heuer (Eds.). Morgan Kaufmann, 790–801. https://doi.org/10.1016/B978-012722442-8/50075-6

[278] Jamel Tayeb, ¨Ozg¨ur Ulusoy, and Ouri Wolfson. 1998. A Quadtree-Based Dynamic Attribute Indexing Method. *Comput. J.* 41, 3 (1998), 185–200. https://doi.org/10.1093/COMJNL/41.3.185

[279] Dejun Teng, Akshay Nehe, Prajeeth Emanuel, Furqan Baig, Jun Kong, and Fusheng Wang. 2021. GPU-based Real-time Contact Tracing at Scale. In *SIGSPATIAL '21: 29th International Conference on Advances in Geographic Information Systems, Virtual Event / Beijing, China, November 2-5, 2021*, Xiaofeng Meng, Fusheng Wang, Chang-Tien Lu, Yan Huang, Shashi Shekhar, and Xing Xie (Eds.). ACM, 1–10. https://doi.org/10.1145/3474717.3483627

[280] Raja Subramaniam Thangaraj, Koyel Mukherjee, Gurulingesh Raravi, Asmita Metrewar, Narendra Annamaneni, and Koushik Chattopadhyay. 2017. Xhare-a-Ride: A Search Optimized Dynamic Ride Sharing System with Approximation Guarantee. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*. IEEE Computer Society, 1117–1128. https://doi.org/10.1109/ICDE.2017.156

[281] Dai Hai Ton That, Iulian Sandu Popa, and Karine Zeitouni. 2015. TRIFL: A Generic Trajectory Index for Flash Storage. *ACM Trans. Spatial Algorithms Syst.* 1, 2 (2015), 6:1–6:44. https://doi.org/10.1145/2786758

[282] Yannis Theodoridis, Michalis Vazirgiannis, and Timos K. Sellis. 1996. Spatio-Temporal Indexing for Large Multimedia Applications. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems, ICMCS 1996, Hiroshima, Japan, June 17-23, 1996*. IEEE Computer Society, 441–448. https://doi.org/10.1109/MMCS.1996.535011

[283] Quoc Cuong To, Tran Khanh Dang, and Josef K¨ung. 2011. OST-Tree: An Access Method for Obfuscating Spatio-Temporal Data in Location Based Services. In *4th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2011, Paris, France, February 7-10, 2011*. IEEE, 1–5. https://doi.org/10.1109/NTMS.2011.5720620

[284] Qiuyun Tong, Xinghua Li, Yinbin Miao, Ximeng Liu, Jian Weng, and Robert H. Deng. 2023. Privacy-Preserving Boolean Range Query With Temporal Access Control in Mobile Computing. *IEEE Trans. Knowl. Data Eng.* 35, 5 (2023), 5159–5172. https://doi.org/10.1109/TKDE.2022.3152168

[285] Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Lei Chen, Jieping Ye, and Ke Xu. 2018. A Unified Approach to Route Planning for Shared Mobility. *Proc. VLDB Endow.* 11, 11 (2018), 1633–1646. https://doi.org/10.14778/3236187.3236211

[286] Hoang Do Thanh Tung, Young Jin Jung, Eung-Jae Lee, and Keun Ho Ryu. 2004. Moving Point Indexing for Future Location Query. In *Conceptual Modeling for Advanced Application Domains, ER 2004 Workshops CoMoGIS, COMWIM, ECDM, CoMoA, DGOV, and ECOMO, Shanghai, China, November 8-12, 2004, Proceedings (Lecture Notes in Computer Science, Vol. 3289)*. Springer, 79–90. https://doi.org/10.1007/978-3-540-30466-1_8

[287] Theodoros Tzouramanis, Michael Vassilakopoulos, and Yannis Manolopoulos. 1998. Overlapping Linear Quadtrees: A Spatio-Temporal Access Method. In *ACM-GIS '98, Proceedings of the 6th international symposium on Advances in Geographic Information Systems, November 6-7, 1998, Washington, DC, USA*, Robert Laurini, Kia Makki, and Niki Pissinou (Eds.). ACM, 1–7. https://doi.org/10.1145/288692.288695

[288] Fabio Valdés and Ralf Hartmut Güting. 2014. Index-supported pattern matching on symbolic trajectories. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Dallas/Fort Worth, TX, USA, November 4-7, 2014*, Yan Huang, Markus Schneider, Michael Gertz, John Krumm, and Jagan Sankaranarayanan (Eds.). ACM, 53–62. https://doi.org/10.1145/2666310.2666402

[289] Fabio Valdés and Ralf Hartmut G¨uting. 2017. Index-supported pattern matching on tuples of time-dependent values. *GeoInformatica* 21, 3 (2017), 429–458. https://doi.org/10.1007/S10707-016-0286-6

[290] Hongzhi Wang and Amina Belhassena. 2017. Parallel trajectory search based on distributed index. *Inf. Sci.* 388 (2017), 62–83. https://doi.org/10.1016/J.INS.2017.01.016

[291] Haojun Wang and Roger Zimmermann. 2008. Snapshot location-based query processing on moving objects in road networks. In *16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2008, November 5-7, 2008, Irvine, California, USA, Proceedings*, Walid G. Aref, Mohamed F. Mokbel, and Markus Schneider (Eds.). ACM, 50. https://doi.org/10.1145/1463434.1463495

[292] Haojun Wang and Roger Zimmermann. 2010. A Novel Dual-Index Design to Efficiently Support Snapshot Location-Based Query Processing in Mobile Environments. *IEEE Trans. Mob. Comput.* 9, 9 (2010), 1280–1292. https://doi.org/10.1109/TMC.2010.63

[293] Longhao Wang, Yu Zheng, Xing Xie, and Wei-Ying Ma. 2008. A Flexible Spatio-Temporal Indexing Scheme for Large-Scale GPS Track Retrieval. In *9th International Conference on Mobile Data Management (MDM 2008), Beijing, China, April 27-30, 2008*, Xiaofeng Meng, Hui Lei, Stéphane Grumbach, and Hong Va Leong (Eds.). IEEE, 1–8. https://doi.org/10.1109/MDM.2008.24

[294] Sheng Wang, Zhifeng Bao, J. Shane Culpepper, Timos Sellis, Mark Sanderson, and Xiaolin Qin. 2017. Answering Top-k Exemplar Trajectory Queries. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*. IEEE Computer Society, 597–608. https://doi.org/10.1109/ICDE.2017.114

[295] Xiangyu Wang, Jianfeng Ma, Ximeng Liu, Robert H. Deng, Yinbin Miao, Dan Zhu, and Zhuoran Ma. 2020. Search Me in the Dark: Privacy-preserving Boolean Range Query over Encrypted Spatial Data. In *39th IEEE Conference on Computer Communications, INFOCOM 2020, Toronto, ON, Canada, July 6-9, 2020*. IEEE, 2253–2262. https://doi.org/10.1109/INFOCOM41043.2020.9155505

[296] Xiang Wang, Ying Zhang, Wenjie Zhang, Xuemin Lin, and Wei Wang. 2015. AP-Tree: efficiently support location-aware Publish/Subscribe. *VLDB J.* 24, 6 (2015), 823–848. https://doi.org/10.1007/S00778-015-0403-4

[297] Yong Wang, Kaiyu Li, Guoliang Li, and Nan Tang. 2023. Road-Aware Indexing for Trajectory Range Queries. *IEEE Trans. Knowl. Data Eng.* 35, 8 (2023), 8476–8489. https://doi.org/10.1109/TKDE.2022.3220822

[298] Zengjie Wang, Wen Luo, Linwang Yuan, Hong Gao, Fan Wu, Xu Hu, and Zhaoyuan Yu. 2021. Query the trajectory based on the precise track: a Bloom filter-based approach. *GeoInformatica* 25, 2 (2021), 397–416. https://doi.org/10.1007/S10707-021-00433-2

[299] Robert Waury, Christian S. Jensen, and Kristian Torp. 2019. A NUMA-aware Trajectory Store for Travel-Time Estimation. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*, Farnoush Banaei Kashani, Goce Trajcevski, Ralf Hartmut Güting, Lars Kulik, and Shawn D. Newsam (Eds.). ACM, 209–218. https://doi.org/10.1145/3347146.3359371

[300] Peter Weiner. 1973. Linear Pattern Matching Algorithms. In *14th Annual Symposium on Switching and Automata Theory, Iowa City, Iowa, USA, October 15-17, 1973*. IEEE Computer Society, 1–11. https://doi.org/10.1109/SWAT.1973.13

[301] Barry Payne Welford. 1962. Note on a method for calculating corrected sums of squares and products. *Technometrics* 4, 3 (1962), 419–420.

[302] Michael A. Whitby, Rich Fecher, and Chris Bennight. 2017. GeoWave: Utilizing Distributed Key-Value Stores for Multidimensional Data. In *Advances in Spatial and Temporal Databases - 15th International Symposium, SSTD 2017, Arlington, VA, USA, August 21-23, 2017, Proceedings (Lecture Notes in Computer Science, Vol. 10411)*, Michael Gertz, Matthias Renz, Xiaofang Zhou, Erik G. Hoel, Wei-Shinn Ku, Agnès Voisard, Chengyang Zhang, Haiquan Chen, Liang Tang, Yan Huang, Chang-Tien Lu, and Siva Ravada (Eds.). Springer, 105–122. https://doi.org/10.1007/978-3-319-64367-0_6

[303] Randall T. Whitman, Bryan G. Marsh, Michael B. Park, and Erik G. Hoel. 2019. Distributed Spatial and Spatio-Temporal Join on Apache Spark. *ACM Trans. Spatial Algorithms Syst.* 5, 1 (2019), 6:1–6:28. https://doi.org/10.1145/3325135

[304] Dingming Wu, Yafei Li, Byron Choi, and Jianliang Xu. 2014. Social-Aware Top-k Spatial Keyword Search. In *IEEE 15th International Conference on Mobile Data Management, MDM 2014, Brisbane, Australia, July 14-18, 2014 - Volume 1*, Arkady B. Zaslavsky, Panos K. Chrysanthis, Christian Becker, Jadwiga Indulska, Mohamed F. Mokbel, Daniela Nicklas, and Chi-Yin Chow (Eds.). IEEE Computer Society, 235–244. https://doi.org/10.1109/MDM.2014.35

[305] Xike Xie, Hua Lu, and Torben Bach Pedersen. 2013. Efficient distance-aware query evaluation on indoor moving objects. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, Christian S. Jensen, Christopher M. Jermaine, and Xiaofang Zhou (Eds.). IEEE Computer Society, 434–445. https://doi.org/10.1109/ICDE.2013.6544845

[306] Xike Xie, Benjin Mei, Jinchuan Chen, Xiaoyong Du, and Christian S. Jensen. 2016. Elite: an elastic infrastructure for big spatiotemporal trajectories. *VLDB J.* 25, 4 (2016), 473–493. https://doi.org/10.1007/S00778-016-0425-6

[307] Xiaopeng Xiong and Walid G. Aref. 2006. R-trees with Update Memos. In *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*, Ling Liu, Andreas Reuter, Kyu-Young Whang, and Jianjun Zhang (Eds.). IEEE Computer Society, 22. https://doi.org/10.1109/ICDE.2006.125

[308] Xiaopeng Xiong, M.F. Mokbel, W.C. Aref, S.E. Hambrusch, and S. Prabhakar. 2004. Scalable spatio-temporal continuous query processing for location-aware services. In *Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004.* 317–326. https://doi.org/10.1109/SSDM.2004.1311223

[309] Xiaopeng Xiong, Mohamed F. Mokbel, and Walid G. Aref. 2006. LUGrid: Update-tolerant Grid-based Indexing for Moving Objects. In *7th International Conference on Mobile Data Management (MDM 2006), Nara, Japan, May 9-13, 2006*. IEEE Computer Society, 13. https://doi.org/10.1109/MDM.2006.102

[310] Jianqiu Xu, Zhifeng Bao, and Hua Lu. 2019. Continuous Range Queries Over Multi-attribute Trajectories. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. IEEE, 1610–1613. https://doi.org/10.1109/ICDE.2019.00154

[311] Jianqiu Xu, Zhifeng Bao, and Hua Lu. 2023. A Framework to Support Continuous Range Queries Over Multi-Attribute Trajectories. *IEEE Trans. Knowl. Data Eng.* 35, 2 (2023), 1119–1133. https://doi.org/10.1109/TKDE.2021.3100650

[312] Jianqiu Xu, Ralf Hartmut Güting, and Yunjun Gao. 2018. Continuous k nearest neighbor queries over large multi-attribute trajectories: a systematic approach. *GeoInformatica* 22, 4 (2018), 723–766. https://doi.org/10.1007/S10707-018-0326-5

[313] Jianqiu Xu, Hua Lu, and Ralf Hartmut Güting. 2018. Range Queries on Multi-Attribute Trajectories. *IEEE Trans. Knowl. Data Eng.* 30, 6 (2018), 1206–1211. https://doi.org/10.1109/TKDE.2017.2787711

[314] X Xu, J Han, and W Lu. 1990. RT-tree: An improved R-tree indexing structure for temporal spatial databases [C]. In *The International Symposium on Spatial Data Handling (SDH). Zurich.* 1040–1049.

[315] Xiaofeng Xu, Li Xiong, and Vaidy S. Sunderam. 2016. D-Grid: An In-Memory Dual Space Grid Index for Moving Object Databases. In *IEEE 17th International Conference on Mobile Data Management, MDM 2016, Porto, Portugal, June 13-16, 2016*, Chi-Yin Chow, Prem Prakash Jayaraman, and Wei Wu (Eds.). IEEE Computer Society, 252–261. https://doi.org/10.1109/MDM.2016.46

[316] Xiaofeng Xu, Li Xiong, Vaidy S. Sunderam, Jinfei Liu, and Jun Luo. 2015. Speed Partitioning for Indexing Moving Objects. In *Advances in Spatial and Temporal Databases - 14th International Symposium, SSTD 2015, Hong Kong, China, August 26-28, 2015. Proceedings (Lecture Notes in Computer Science, Vol. 9239)*, Christophe Claramunt, Markus Schneider, Raymond Chi-Wing Wong, Li Xiong, Woong-Kee Loh, Cyrus Shahabi, and Ki-Joune Li (Eds.). Springer, 216–234. https://doi.org/10.1007/978-3-319-22363-6_12

[317] Yixin Xu, Jianzhong Qi, Renata Borovica-Gajic, and Lars Kulik. 2020. GeoPrune: Efficiently Matching Trips in Ride-sharing Through Geometric Properties. In *SSDBM 2020: 32nd International Conference on Scientific and Statistical Database Management, Vienna, Austria, July 7-9, 2020*, Elaheh Pourabbas, Dimitris Sacharidis, Kurt Stockinger, and Thanasis Vergoulis (Eds.). ACM, 5:1–5:12. https://doi.org/10.1145/3400903.3400912

[318] Yan Xu and Gary Tan. 2014. Sim-Tree: indexing moving objects in large-scale parallel microscopic traffic simulation. In *SIGSIM Principles of Advanced Discrete Simulation, SIGSIM-PADS '14, Denver, CO, USA, May 18-21, 2014*, John A. Hamilton Jr., George F. Riley, and Richard M. Fujimoto

(Eds.). ACM, 51–62. https://doi.org/10.1145/2601381.2601388

[319] Zizhuo Xu, Lei Li, Mengxuan Zhang, Yehong Xu, and Xiaofang Zhou. 2024. Managing the Future: Route Planning Influence Evaluation in Transportation Systems. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*. IEEE, 4558–4572. https://doi.org/10.1109/ICDE60146.2024.00347

[320] Liang Ye. 2011. A Efficient Indexing Maintenance Method for Grouping Moving Objects with Grid. *Procedia Environmental Sciences* 11 (2011), 486–492. https://doi.org/10.1016/j.proenv.2011.12.077 2011 2nd International Conference on Challenges in Environmental Science and Computer Engineering (CESCE 2011).

[321] Man Lung Yiu, Yufei Tao, and Nikos Mamoulis. 2008. The B^{dual}-Tree: indexing moving objects by space filling curves in the dual space. *VLDB J.* 17, 3 (2008), 379–400. https://doi.org/10.1007/S00778-006-0013-2

[322] Xiaohui Yu, Ken Q. Pu, and Nick Koudas. 2005. Monitoring K-Nearest Neighbor Queries Over Moving Objects. In *Proceedings of the 21st International Conference on Data Engineering, ICDE 2005, 5-8 April 2005, Tokyo, Japan*, Karl Aberer, Michael J. Franklin, and Shojiro Nishio (Eds.). IEEE Computer Society, 631–642. https://doi.org/10.1109/ICDE.2005.92

[323] Ziqiang Yu, Yang Liu, Xiaohui Yu, and Ken Q. Pu. 2015. Scalable Distributed Processing of K Nearest Neighbor Queries over Moving Objects. *IEEE Trans. Knowl. Data Eng.* 27, 5 (2015), 1383–1396. https://doi.org/10.1109/TKDE.2014.2364046

[324] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2012. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. In *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2012, San Jose, CA, USA, April 25-27, 2012*, Steven D. Gribble and Dina Katabi (Eds.). USENIX Association, 15–28. https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/zaharia

[325] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: Cluster Computing with Working Sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud'10, Boston, MA, USA, June 22, 2010*, Erich M. Nahum and Dongyan Xu (Eds.). USENIX Association. https://www.usenix.org/conference/hotcloud-10/spark-cluster-computing-working-sets

[326] Tilmann Z¨aschke, Christoph Zimmerli, and Moira C. Norrie. 2014. The PH-tree: a space-efficient storage structure and multi-dimensional index. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, Curtis E. Dyreson, Feifei Li, and M. Tamer ¨Ozsu (Eds.). ACM, 397–408. https://doi.org/10.1145/2588555.2588564

[327] Bowen Zhang, Yanyan Shen, Yanmin Zhu, and Jiadi Yu. 2018. A GPU-Accelerated Framework for Processing Trajectory Queries. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*. IEEE Computer Society, 1037–1048. https://doi.org/10.1109/ICDE.2018.00097

[328] Zhiyin Zhang, Xiaocheng Huang, Chaotang Sun, Shaolin Zheng, Bo Hu, Jagannadan Varadarajan, Yifang Yin, Roger Zimmermann, and Guanfeng Wang. 2019. Sextant: Grab's Scalable In-Memory Spatial Data Store for Real-Time K-Nearest Neighbour Search. In *20th IEEE International Conference on Mobile Data Management, MDM 2019, Hong Kong, SAR, China, June 10-13, 2019*. IEEE, 243–251. https://doi.org/10.1109/MDM.2019.00-51

[329] Kaiqi Zhao, Gao Cong, Jin Yao Chin, and Rong Wen. 2019. Exploring market competition over topics in spatio-temporal document collections. *VLDB J.* 28, 1 (2019), 123–145. https://doi.org/10.1007/S00778-018-0522-9

[330] Peng Zhao, Weixiong Rao, Chengxi Zhang, Gong Su, and Qi Zhang. 2020. SST: Synchronized Spatial-Temporal Trajectory Similarity Search. *GeoInformatica* 24, 4 (2020), 777–800. https://doi.org/10.1007/S10707-020-00405-Y

[331] Wei Zhao, Hao Wang, and Wei Yu. 2025. Efficient nearest-neighbor search on moving objects with durability constraints. *GeoInformatica* 29, 3 (2025), 665–690. https://doi.org/10.1007/S10707-025-00543-1

[332] Bolong Zheng, Nicholas Jing Yuan, Kai Zheng, Xing Xie, Shazia Sadiq, and Xiaofang Zhou. 2015. Approximate keyword search in semantic trajectory database. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, Johannes Gehrke, Wolfgang Lehner, Kyuseok Shim, Sang Kyun Cha, and Guy M. Lohman (Eds.). IEEE Computer Society, 975–986. https://doi.org/10.1109/ICDE.2015.7113349

[333] Kai Zheng, Shuo Shang, Nicholas Jing Yuan, and Yi Yang. 2013. Towards efficient search for activity trajectories. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, Christian S. Jensen, Christopher M. Jermaine, and Xiaofang Zhou (Eds.). IEEE Computer Society, 230–241. https://doi.org/10.1109/ICDE.2013.6544828

[334] Kai Zheng, Goce Trajcevski, Xiaofang Zhou, and Peter Scheuermann. 2011. Probabilistic range queries for uncertain trajectories on road networks. In *EDBT 2011, 14th International Conference on Extending Database Technology, Uppsala, Sweden, March 21-24, 2011, Proceedings*, Anastasia Ailamaki, Sihem Amer-Yahia, Jignesh M. Patel, Tore Risch, Pierre Senellart, and Julia Stoyanovich (Eds.). ACM, 283–294. https://doi.org/10.1145/1951365.1951400

[335] Kai Zheng, Yan Zhao, Defu Lian, Bolong Zheng, Guanfeng Liu, and Xiaofang Zhou. 2020. Reference-Based Framework for Spatio-Temporal Trajectory Compression and Query Processing. *IEEE Trans. Knowl. Data Eng.* 32, 11 (2020), 2227–2240. https://doi.org/10.1109/TKDE.2019.2914449

[336] Kai Zheng, Bolong Zheng, Jiajie Xu, Guanfeng Liu, An Liu, and Zhixu Li. 2017. Popularity-aware spatial keyword search on activity trajectories. *World Wide Web* 20, 4 (2017), 749–773. https://doi.org/10.1007/S11280-016-0414-0

[337] Ruicheng Zhong, Guoliang Li, Kian-Lee Tan, and Lizhu Zhou. 2013. G-tree: an efficient index for KNN search on road networks. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, Qi He, Arun Iyengar, Wolfgang Nejdl, Jian Pei, and Rajeev Rastogi (Eds.). ACM, 39–48. https://doi.org/10.1145/2505515.2505749

[338] Ruicheng Zhong, Guoliang Li, Kian-Lee Tan, Lizhu Zhou, and Zhiguo Gong. 2015. G-Tree: An Efficient and Scalable Index for Spatial Search on Road Networks. *IEEE Trans. Knowl. Data Eng.* 27, 8 (2015), 2175–2189. https://doi.org/10.1109/TKDE.2015.2399306

[339] Panfeng Zhou, Donghui Zhang, Betty Salzberg, Gene Cooperman, and George Kollios. 2005. Close pair queries in moving object databases. In *13th ACM International Workshop on Geographic Information Systems, ACM-GIS 2005, November 4-5, 2005, Bremen, Germany, Proceedings*, Cyrus Shahabi and Omar Boucelma (Eds.). ACM, 2–11. https://doi.org/10.1145/1097064.1097067

[340] Yuelong Zhu, Xiang Ren, and Jun Feng. 2006. NCO-Tree: A Spatio-temporal Access Method for Segment-Based Tracking of Moving Objects. In *Knowledge-Based Intelligent Information and Engineering Systems, 10th International Conference, KES 2006, Bournemouth, UK, October 9-11, 2006, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 4252)*, Bogdan Gabrys, Robert J. Howlett, and Lakhmi C. Jain (Eds.). Springer, 1191–1198. https://doi.org/10.1007/11893004_151

[341] Yuean Zhu, Shan Wang, Xuan Zhou, and Yansong Zhang. 2013. RUM+-tree: A New Multidimensional Index Supporting Frequent Updates. In *Web-Age Information Management - 14th International Conference, WAIM 2013, Beidaihe, China, June 14-16, 2013. Proceedings (Lecture Notes in Computer Science, Vol. 7923)*, Jianyong Wang, Hui Xiong, Yoshiharu Ishikawa, Jianliang Xu, and Junfeng Zhou (Eds.). Springer, 235–240. https://doi.org/10.1007/978-3-642-38562-9_24