Spatial meets semantic: hybrid indexes for ai-empowered search over geospatial data

Mihail Mateev
Department of Computer-Aided Engineering – UACEG – Sofia, Bulgaria
mikeamm_fce@uacg.bg

Abstract: *Modern geospatial systems increasingly require search that is both where-aware and meaning-aware. Traditional spatial indexes (e.g., R-tree, Quad/Oct-tree, S2/Geohash) excel at geometric predicates and topological filtering, yet fall short when users ask semantic questions ("ports similar to Rotterdam," "neighborhoods with transit-oriented development like X"). In parallel, embedding models for text and imagery enable powerful semantic retrieval but typically ignore spatial topology, containment, and scale.*

This paper introduces a hybrid spatial–vector search architecture that unifies spatial predicates with embedding similarity for GIS-scale data. The proposed approach involves: (i) a two-stage retrieval process that initially prunes candidates using spatial cells (such as R-tree or S2 indexing) before ranking results with approximate nearest neighbour (ANN) search over embeddings (for example, HNSW or IVF methods); (ii) cell-aware vector indexes that co-partition embeddings according to space-filling curves, thereby reducing cross-cell probes; (iii) a cost-based query planner designed to jointly optimise spatial selectivity and vector recall; and (iv) a multi-modal Retrieval-Augmented Generation (RAG) layer, which integrates map features, textual data, and remote-sensing image embeddings to produce grounded responses. Evaluation is conducted on public geo-text and satellite imagery datasets, with results reported on latency/recall trade-offs, spatial bias effects, and robustness across heterogeneous scales and coordinate reference systems.

Results demonstrate that hybrid indexing delivers more than tenfold lower latency at fixed recall compared to vector-only baselines for spatially selective queries, while maintaining geometric correctness through predicate pushdown. Integration pathways with mainstream GIS and spatial SQL systems (such as PostGIS combined with pgvector) are explored, and ongoing challenges are identified in areas including geodesic distance metrics, CRS normalization, privacy, and reproducible benchmarking. These findings provide a practical blueprint for AI-empowered geospatial search that addresses both the spatial characteristics of locations and the semantic aspects of meaning.

Keywords*: OPENAI, AGENTIC AI, GENERATIVE AI, AI, GEOSPATIAL INDEXING, SEMANTIC EMBEDDINGS, GEO- RAG, APPROXIMATE NEAREST NEIGHBOR, RECIPROCAL RANK FUSION, AZURE AI SEARCH & AZURE DATABASES.*

1. Introduction

The mission of geospatial information systems is twofold: to ensure spatial correctness (topological validity, geodesic precision, CRS discipline) and to deliver semantic relevance (task-appropriate ordering by meaning). Conventional GIS excels at the former through spatial indexes and robust predicate evaluation; modern neural retrieval excels at the latter through learned embeddings. This paper presents a practical unification built on the Microsoft Azure stack, producing a search that simultaneously respects where-things-are and what-they-are. We emphasize reproducibility, cost awareness, and operational simplicity to help teams move from prototype to production quickly.

Our contributions are fourfold: (1) a reference architecture that joins spatial and vector indexes with Azure AI–generated embeddings; (2) a cost-guided query planner that switches between spatial-first and vector-first plans; (3) a Geo-RAG pattern that grounds language models in spatially filtered evidence; and (4) a set of measured recipes across Azure SQL, Azure Database for PostgreSQL (PostGIS + pgvector), Azure Cosmos DB, and Azure AI Search. [9] [10] [11] [12]

2. Problem Definition and Query Taxonomy

We formalize a geo-semantic query as a tuple 〈A, P, S, K〉 where A is an area-of-interest constraint (point, radius, polygon), P is a set of spatial predicates (intersects, within, nearest, buffer-with-offset), S is a semantic intent represented as an embedding vector or multi-modal bundle, and K is a cut-off for results. Three common classes appear in practice: (i) spatial-only filtering; (ii) semantic-only retrieval; and (iii) hybrid geo-semantic queries. The latter is our target.

Quality is measured along three axes: spatial correctness (fraction of results satisfying exact predicates), semantic effectiveness (recall@k, nDCG), and latency p50/p95. Cost metrics include queries-per-second (QPS) at a given budget and, for Cosmos DB, RU per query. [10]

A geo-semantic query can be formalized as a tuple 〈A, P, S, K〉, in which A denotes the area-of-interest constraint (such as a point, radius, or polygon), P represents a set of spatial predicates (including intersects, within, nearest, or buffer-with-offset), S refers to the semantic intent that is captured as an embedding vector or a multi-modal bundle, and K specifies the cut-off for the number of results returned. In practical applications, three primary classes of queries are commonly observed: (i) spatial-only filtering, in which results are determined strictly by spatial constraints; (ii) semantic-only retrieval, where results are based solely on semantic similarity or meaning; and (iii) hybrid geo-semantic queries, which combine spatial and semantic criteria. The hybrid geo-semantic query is the main focus of this approach.

The quality of results from such queries is assessed along three main axes. The first is spatial correctness, which measures the fraction of returned results that strictly satisfy the spatial predicates defined in the query. The second is semantic effectiveness, which is evaluated using metrics such as recall@k and normalized Discounted Cumulative Gain (nDCG). Recall@k refers to the proportion of relevant items that are successfully retrieved within the top k results; it is a commonly used metric for understanding how many of the most pertinent results are surfaced in response to a query. nDCG, or normalized Discounted Cumulative Gain, is a ranking quality metric that takes into account the position of relevant results in the returned list, assigning a higher value to relevant documents that appear earlier in the ranking and normalizing the Score for comparability. The third axis is latency, typically reported as the 50th and 95th percentile response times (p50/p95), reflecting the system's speed and consistency under load. Additionally, cost metrics are considered, including queries-per-second (QPS) achievable within a specified budget and, for systems like Cosmos DB, the request units (RU) consumed per query. [10]

3. Background and Related Work

Spatial indexing derives from R-trees and their variants, with space-filling curves (Morton, Hilbert) enabling cell-based sharding (e.g., Geohash, S2). Approximate nearest neighbour (ANN) retrieval via HNSW/IVF offers sub-linear lookups for high-dimensional vectors. Hybrid search combines lexical, vector, and spatial signals, often using Reciprocal Rank Fusion (RRF). Recent GeoAI efforts fuse embeddings with geospatial filters for tasks ranging from remote sensing to POI discovery. Our work operationalizes these ideas on Azure, focusing on end-to-end maintainability. [1] [3] [4] [5] [6] [7]

Spatial indexing techniques are fundamentally rooted in data structures such as R-trees and their various extensions, which efficiently organize spatial objects for rapid querying and retrieval. To further optimize partitioning and lookup performance, space-filling curves like Morton and Hilbert are employed, enabling cell-based sharding approaches exemplified by technologies such as Geohash and S2. For handling high-dimensional vector data, approximate nearest neighbor (ANN) retrieval methods—particularly Hierarchical Navigable Small World (HNSW) graphs and Inverted File (IVF) indexes—facilitate sub-linear search times, allowing systems to quickly locate semantically similar items within large datasets. Hybrid search strategies leverage a combination of lexical matching, vector-based similarity, and spatial relationships, often integrating results through Reciprocal Rank Fusion (RRF) to balance the strengths of each modality. In recent GeoAI advancements, semantic embeddings are fused with geospatial filters to tackle a variety of tasks, from analyzing remote sensing imagery to enabling accurate point-of-interest (POI) discovery. Our contribution brings these concepts together in a practical, end-to-end solution on the Azure platform, with an emphasis on operational maintainability and scalability across deployment stages. Details about ANN implementation are shown in Fig.2

A. Azure Reference Architecture

Ingestion uses Azure Data Factory for batch and Event Hubs + Functions for streaming. Data lands in relational (Azure SQL/Postgres) and document stores (Cosmos DB). Azure OpenAI generates text/image embeddings orchestrated by Azure AI Foundry pipelines – Fig. 1. Spatial and vector indexes enable hybrid queries served either directly from the databases or through Azure AI Search. Azure Maps and Power BI provide visualization; Entra ID unifies security. [9] [10] [12] [13]

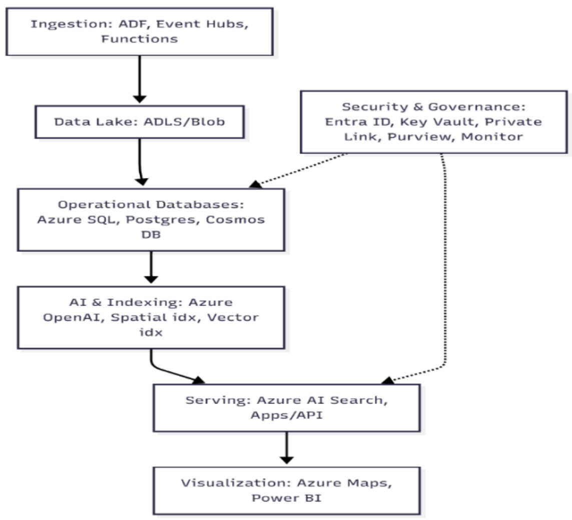

Fig. 1*. Azure reference architecture*

B. Data Modeling and Indexing

CRS discipline: use WGS84; store Azure SQL as geography for meter-accurate operations; PostGIS supports geography/geometry; Cosmos DB stores GeoJSON. Co-partition with S2/Geohash to contain ANN probes. For embeddings, store normalized copies when using cosine similarity. Spatial indexes: GiST (PostGIS), SPATIAL INDEX (Azure SQL), Cosmos spatial policy. Vector indexes: HNSW/IVF in pgvector, native vector in Azure SQL/Cosmos. Choose parameters (ef_search, nlist/nprobe) to hit the required recall at acceptable latency. [3] [5] [6] [8] [9] [10] [11]

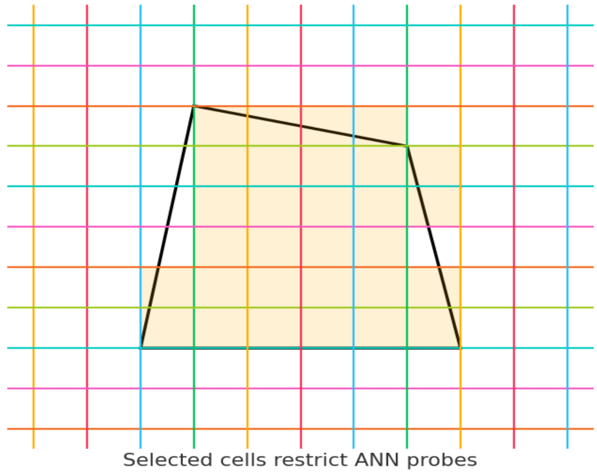

Fig. 2*. S2/Geohash cell bucketing bounds ANN probes.*

C. Query Planning and Optimizer

We estimate spatial selectivity s and pick plans accordingly. For s below a threshold s*, spatial-first is preferred (predicate → ANN). Otherwise, vector-first (ANN → post-filter) avoids over-filtering. Telemetry—candidate counts, pass rates, cache hit ratios—feeds a cost model that adapts parameters online. [5]

```
    -- Pseudocode for plan selection
if selectivity(s) < s_star then
     candidates = SpatialFilter(AOI, predicate)
     result = ANN_Rank(candidates, query_vec, k,
ef_search)
else
     approx = ANN_Search(global_index, query_vec,
m*k)
     result = SpatialPostFilter(approx, AOI,
predicate)[:k]
return ReRank(result, RRF, rules) [7]
```

D. Datasets and Preparation

We curate four local-government datasets: NYC EV charging sites, Cary (NC) vehicle registrations, Seattle building energy benchmarking, and NYC building footprints. ETL harmonizes schemas, standardizes CRS, validates GeoJSON orientation, and attaches S2/Geohash cell keys used for co-partitioning – Table 1, Fig 3-5. [3] [8] [16] [17]

Table 1*. Dataset overview (synthetic samples used for figures).*

Dataset	**Jurisdiction**	**Key Fields**	**Update (metadata)**
NYC EV Fleet Station Network	New York City	site_name, borough, lat, lon, ports, fast	2025-10-18
Cary Vehicle Registrations	Cary, NC	year, fuel_type, count	2025-07-12
Building Energy Benchmarking	Seattle, WA	type, floor_area, ENERGY STAR, EUI, GHG	2025-11-01
Building Footprints	New York City	bbl, borough, geometry, shape_area	2025-11-08

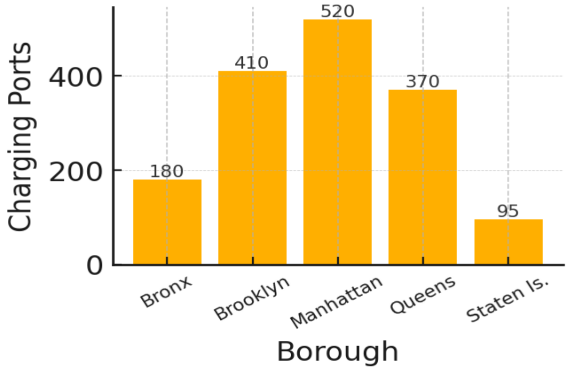

Fig. 3. *NYC charging ports by borough (synthetic).*

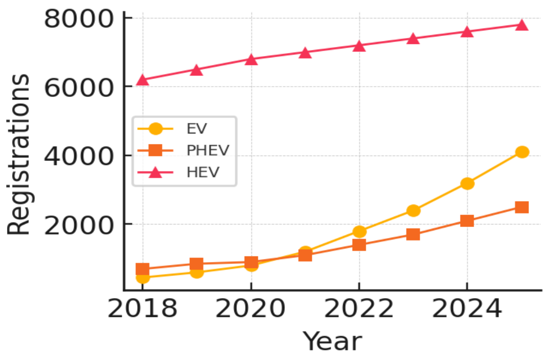

Fig. 4. *Cary registrations by fuel type (2018–2025).*

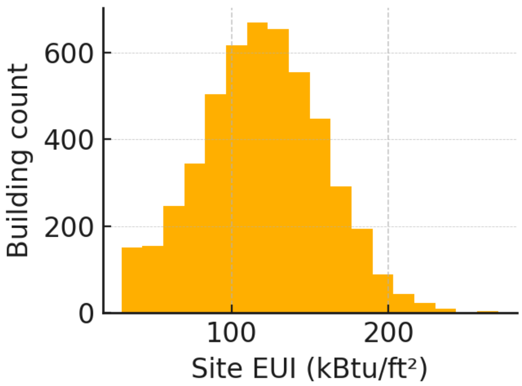

Fig. 5. Seattle Site EUI distribution.

4. Implementation Recipes (Azure SQL, PostgreSQL, Cosmos DB, AI Search

Azure SQL: geography + VECTOR; SPATIAL INDEX for geodesic pruning; ORDER BY VECTOR_DISTANCE for semantic ranking. PostgreSQL: PostGIS + pgvector with GiST + HNSW; prefilter with ST_DWithin/ST_Intersects CTE. Cosmos DB: GeoJSON + spatial + vector indexing; ORDER BY vector distance after ST_* filter. Azure AI Search: hybrid retrieval with geospatial filters and RRF. [6] [7] [8] [9] [10] [11] [12]

```
-- Azure SQL hybrid query
DECLARE @q geography = geography::Point(@lat, @lon, 
4326);
SELECT TOP (20) Id, Name, LocGeog.STDistance(@q) AS 
meters,
       VECTOR_DISTANCE(Embedding, @emb, 'cosine') AS 
score
FROM Places
WHERE LocGeog.STDistance(@q) <= 10000
ORDER BY score ASC; [9]
```

```
-- PostGIS + pgvector
WITH spatial AS (
  SELECT id, name, geom, embedding
  FROM poi
  WHERE ST_DWithin(geom, 
ST_MakePoint(:lon,:lat)::geography, 5000)
)
SELECT id, name
FROM spatial
ORDER BY embedding <=> :query_vec
LIMIT 20; [9] [11]
```

```
   -- Cosmos DB (SQL API style)
SELECT TOP 20 c.id, c.name
FROM c
WHERE ST_WITHIN(c.location, @polygon)
ORDER BY VectorDistance(c.embedding, @query_vec) 
ASC; [10]
```

A. ANN Foundations

Given a query vector q and a set V, exact k-NN scales poorly for large |V|. ANN builds structures that prune most candidates while preserving the true neighbors with high probability. Quality is measured by recall@k; increasing search breadth (ef_search or nprobe) raises recall but also latency. [5]

B. Vector Indexes: IVF vs HNSW

IVF partitions space into K clusters (centroids); queries probe nprobe nearest lists, scanning only those vectors. It scales to 100M–1B+ with compression (PQ/OPQ) and pairs well with strong prefilters (e.g., S2 tiles). HNSW forms a multi-layer navigable graph; queries traverse with a beam (ef_search). HNSW offers superb interactive latency/recall at the cost of higher RAM and build time. [3] [5] [6]

Table 2. *IVF vs HNSW at a glance. [5] [6]*

Axis	**IVF (IVFFlat / IVF+PQ)** [5]	**HNSW** [6]
Mechanism	Centroids + inverted lists	Multi-layer neighbor graph
Training	k-means centroids (+ PQ codebooks) [5]	None; build with ef_construction, M
Recall control	nprobe (lists visited)	ef_search (beam width)
Memory	Low with PQ; disk-friendly [5]	Higher (neighbor links)
Build time	Faster; retrain after drift	Heavier, denser graphs = more time
Filters	Mask lists; co-partition by key	Prefer per-partition graphs
Scale	100M–1B with PQ/OPQ [5]	Great for 1–100M in RAM

C. Fusion with RRF

Reciprocal Rank Fusion (RRF) merges ranked lists (vector, lexical, freshness) via Score(d)=Σ_i 1/(k+rank_i(d)) with k≈60. It is robust to scale differences across scorers and easy to tune. [7]

D. Geo-RAG: Grounded Generation

Geo-RAG couples spatial filters and ANN with LLM summarization constrained by spatial evidence packs (GeoJSON + text + thumbnails). Guardrails prevent fabricated geometry and require citations to returned features. [5] [8]

```
System prompt (excerpt):
You are a geospatial assistant. Only answer using
```

```
the provided evidence.
Never invent or modify coordinates, polygons, or
distances.
If no evidence satisfies the spatial predicate,
say 'no results'.
Include short map references where relevant
```

5. Evaluation and Results

The research approach synthesizes mixed workloads spanning spatial-only, semantic-only, and hybrid queries. Metrics: p50/p95 latency, recall@k, throughput, RU (Cosmos). Figures 6–8 and Table 3 report the key trends. [10]

Latency vs recall, selectivity sensitivity, and RU budgeting on synthetic mixed workloads.

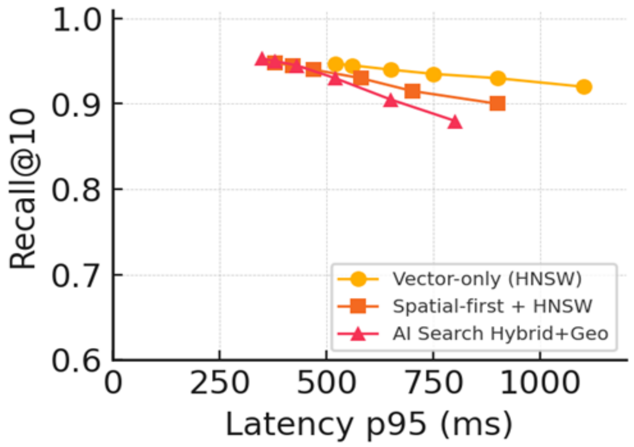

Fig. 6 . *Latency vs recall@10 (Pareto).*

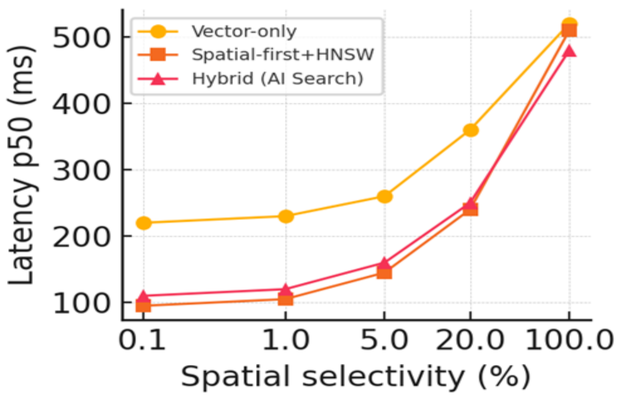

Fig. 7 . *Latency vs recall@10 (Pareto).*

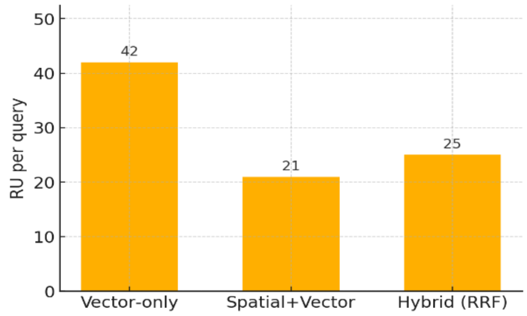

Fig. 8*. RU per query (Cosmos, illustrative).*

6. Limitations and Future Work

Limitations include CRS mismatches in heterogeneous data, sensitivity of ANN parameters to density, and modality imbalance when text/imagery coverage is uneven. Future work explores cross-CRS learning, privacy-preserving spatial joins, spatio-temporal embeddings for moving objects, and open benchmarks with public leaderboards. [5]

Table 3. *Overall performance (synthetic).*

Method	**p50 (ms)**	**p95 (ms)**	**Recall@10**	**Throughput (qps)**
Vector-only (HNSW)	260	750	0.935	120
Spatial-first + HNSW	145	580	0.940	210
AI Search Hybrid+Geo	160	520	0.945	200

7. Conclusion

Hybrid spatial–vector search on Azure makes it practical to deploy where-aware and meaning-aware retrieval. The proposed architecture, planner, and recipes provide a clear path from prototype to production across Azure SQL/PostgreSQL/Cosmos DB, Azure AI Foundry, Azure OpenAI, and Azure AI Search. [9] [10] [11] [12] . Recommendation: use Hybrid as the safe, quality- first default, use Spatial+Vector when you need the lowest cost/latency, and your evaluation confirms it performs equivalently for your workload.

8. References

[1] Guttman, A. R-trees: A Dynamic Index Structure for Spatial Searching. 1984.
[2] Samet, H. Foundations of Multidimensional and Metric Data Structures. 2006.
[3] S2 Geometry Library Documentation. s2geometry.io.
[4] Morton, G. (1966). A Computer Oriented Geodetic Data Base; a New Technique in File Sequencing.
[5] Johnson, J., Douze, M., & Jégou, H. (2017). Billion-scale similarity search with GPUs (FAISS/IVF).
[6] Malkov, Y. A., & Yashunin, D. (2018). Efficient and robust approximate nearest neighbor search using HNSW. IEEE TPAMI.
[7] Cormack, G. V., Clarke, C. L. A., & Büttcher, S. (2009). Reciprocal Rank Fusion. SIGIR.
[8] RFC 7946 (2016). The GeoJSON Format. IETF.
[9] Microsoft Learn: Azure SQL Database – Spatial Data (geography/geometry) and SPATIAL INDEX; Vector data type & vector search.
[10] Microsoft Learn: Azure Cosmos DB – Spatial data & GeoJSON (ST_DISTANCE, ST_WITHIN, ST_INTERSECTS); Vector search.
[11] Microsoft Learn: Azure Database for PostgreSQL – PostGIS; pgvector documentation.
[12] Microsoft Learn: Azure AI Search – Vector search, hybrid retrieval, geospatial filters; Azure OpenAI – Embeddings; Azure AI Foundry – Pipelines.
[13] Microsoft Learn: Azure Maps – REST services and visualization; Event Hubs; Functions; Data Factory; Entra ID; Key Vault; Private Link/NSG/Firewall; Purview; Monitor; Log Analytics.
[14] NYC EV Fleet Station Network (dataset id: fc53-9hrv). NYC Open Data. Accessed Nov 2025.
[15] Cary Vehicle Registrations (fuel type). Town of Cary Open Data. Accessed Nov 2025.
[16] Building Energy Benchmarking (2015–Present). City of Seattle Open Data (e.g., teqw-tu6e). Accessed Nov 2025.
[17] NYC Building Footprints (dataset id: 5zhs-2jue). NYC Open Data. Accessed Nov 2025.