IT UNIVERSITY OF COPENHAGEN

KISPECI1SE - Master Thesis

Filtered Approximate Nearest-Neighbour Search on Real Multimedia Workloads

Prepared by: Jakob Splidsboel Eg (jaeg@itu.dk)
Student number: 20280
Supervised by: Omar Shahbaz Khan (omsh@itu.dk)
Date: August 31, 2026

Abstract

Filtered approximate nearest-neighbour search (FANNS) combines similarity search over high-dimensional embeddings with structured predicates over metadata. Existing FANNS benchmarks evaluate this problem with single-attribute filters over synthetic labels, queries held out from the corpus, and geometric Recall@k as the sole quality measure. This thesis characterises the query logs of the Video Browser Showdown, a video-retrieval competition over the V3C collection, and finds that real known-item search queries differ from the benchmarked workloads. They are human-authored and cross-modal, express deep heterogeneous conjunctions of predicates, and require predicates evaluated across multiple relations. To measure the effect of these differences, the thesis introduces FRAME, a filtered-ANN benchmark for known-item search that draws queries from real user phrasings, evaluates predicates over a normalised multi-relation schema, and reports the rank of the designated known item alongside Recall@k. Evaluated on pgvector and ChromaDB, the two systems reach geometric recall of approximately 0.9 while mean reciprocal rank over the known target is approximately 0.07.

AI Usage Disclaimer

In this project, Claude Opus 4.8, Opus 4.6, and Sonnet 4.6 were used as coding and writing assistants during the preparation of the thesis. The author takes full responsibility for the content, correctness, and final wording of the manuscript and the accompanying code.

Contents

1 Introduction

Vector databases combine high-dimensional similarity search with structured metadata filtering. Given a query embedding and a predicate over per-item attributes, a filtered approximate nearest-neighbour search (FANNS) returns the top k items from the dataset closest to the query embedding and satisfy the predicate [1, 2]. The algorithmic basis, approximate nearest-neighbour search (ANNS) [3], is well benchmarked in the unfiltered case [4, 5], and a growing body of filtered-ANN benchmarks evaluates how predicate evaluation interacts with the approximate index [1, 6, 7].

Existing FANNS benchmarks are effective at what they measure, but what they measure is narrow relative to real retrieval. Their filters are typically single-attribute predicates over synthetic or randomly assigned labels. Their queries are held out from the corpus rather than drawn from real users, and success is reported almost entirely as geometric correctness (Recall@k). Multimedia search tasks are typically out-of-distribution (OOD) queries typed by users which require multiple filters on the same query. Existing benchmarks therefore give a reliable picture of index geometry but a limited one of how a system behaves under the multi-attribute, human-authored filters of an actual multimedia search session. This thesis characterises real multimedia retrieval workloads from the Video Browser Showdown (VBS), a video-retrieval competition over the V3C collection [8], to understand what query patterns appear in practice, and then evaluates how current vector databases handle these patterns. Concretely, this thesis sets out to answer these research questions:

- ***To what extent do existing FANNS benchmarks reflect the predicate complexity and query patterns found in real multimedia retrieval workloads?***
- ***How do current vector database systems handle the query patterns characteristic of real multimedia retrieval workloads, and what workarounds are required when native support is insufficient?***
- ***Are standard FANNS evaluation metrics sufficient to capture retrieval system utility in real-world multimedia search scenarios?***

To answer these questions, this thesis first characterises the datasets, query construction, and filter workloads of existing ANNS and FANNS benchmarks and examines whether they are representative of the patterns observed in real multimedia retrieval. It then introduces FRAME (Filtered Retrieval Attribute Media Evaluation), a filtered-ANN benchmark for known-item search over V3C. FRAME's query set stems from real user queries extracted from logs published by teams in past years of the Video Browser Showdown (VBS) competition. FRAME evaluates predicates over a normalised multi-relation schema of visual metadata, and reports task success (the rank of the designated known item)

alongside the geometric Recall@k used by existing benchmarks. The benchmark is evaluated on pgvector and ChromaDB. The results show a geometric recall of approximately 0.9 alongside low task success on the same items, indicating that the approximate index faithfully returns the geometric neighbourhood while the item the user is looking for usually does not appear in it. Additionally, the results show a significant discrepancy in query latency, likely attributable to system-specific predicate evaluation approaches.

2 Background

2.1 High-dimensional data

Within the scope of this thesis, "high-dimensional data" refers to dense numerical vectors $x \in \mathbb{R}^d$ where the dimensionality d typically ranges from a few hundred to a few thousand. In contrast to the columns of a relational record, the individual coordinates of such a vector carry no fixed interpretation. The information content lies instead in the geometric relationships between vectors. Which points cluster together, which are far apart, and along which directions variation reflects task-relevant signal [2].

The dominant source of such vectors in modern multimedia systems is the embedding produced by a learned neural model. Early dense representations were obtained from co-occurrence statistics. Bag-of-visual-words for image retrieval [9] and shallow word-embedding models such as GloVe for text [10]. Contemporary embeddings are almost exclusively the output of transformer-based architectures [11]. Sentence- and document-level transformers such as Sentence-BERT [12] and specialised variants [13] produce textual embeddings, while contrastively trained vision–language models such as CLIP [14], supported by web-scale aligned image–text corpora [15], map images and natural-language descriptions into a shared embedding space [16]. The resulting vectors are typically of dimensionality 384, 512, 768, 1024 or higher depending on the encoder.

A defining difficulty of operating in such spaces is the *curse of dimensionality*. As d grows, the volume of the space expands exponentially while any fixed corpus becomes sparse within it, and the distances from a query to its nearest and farthest points concentrate toward a common value, so that under broad conditions the nearest neighbor becomes barely distinguishable from an arbitrary point [17]. The geometric contrast that similarity search relies on therefore erodes, and space-partitioning structures that prune effectively in low dimensions must inspect an ever-larger fraction of the corpus to stay correct.

A retrieval system queries this representation by similarity. Given a query embedding q and a corpus $X \subset \mathbb{R}^d$, it returns the items in X that are closest to q under some chosen distance or similarity measure. The three measures used in essentially all current vector databases are Euclidean (L_2) distance, cosine similarity (the cosine of the angle between two vectors), and inner product. For unit-normalised embeddings these three measures are monotonic transformations of one another and induce identical top-k rankings [2].

2.2 Approximate Nearest Neighbor Search

Given a query vector q and a corpus $X \subset \mathbb{R}^d$, the nearest neighbor problem asks for the k vectors in X closest to q under a distance or similarity. For high-dimensional embeddings, exact k-NN degrades toward linear scan and is impractical at scale. Approximate nearest neighbor search (ANNS) trades a small recall loss for orders-of-magnitude gains in throughput [2]. The standard quality metric is Recall@k at a fixed throughput target, in the form of queries per second for a query workload. Ma et al. [2] group ANNS index families into four categories: graph-based, quantization-based, hash-based, and tree-based. Each of the four families make a different structural compromise to escape the curse of dimensionality.

2.2.1 Graph based

Graph-based methods reach state-of-the-art recall–QPS trade-offs by treating the corpus as a navigable proximity graph and greedy-routing toward the query. The baseline for the family is the Navigable Small World (NSW) graph, which combines short- and long-range edges so that greedy traversal converges on the query in a small number of hops [2]. Hierarchical NSW (HNSW) refines this construction by stacking NSW graphs in layers of decreasing density. Search begins at the sparse top layer and descends, yielding logarithmic expected search cost. Its build is governed by the number of neighbours retained per node M and the build-time beam width $\mathrm{ef}_{\mathrm{construction}}$, while query quality is tuned at runtime through the search beam width $\mathrm{ef}_{\mathrm{search}}$. HNSW is the default in-memory index in pgvector and ChromaDB [18]. Later designs flatten or relocate this structure. The Navigating Spreading-out Graph (NSG) replaces the layered hierarchy with a single sparse monotonic-relative-neighbourhood graph, attaining comparable recall at lower memory [19]. Vamana/DiskANN keeps the graph compact enough to reside on SSD, enabling billion-scale out-of-memory search [19]. Pruning variants such as HNSW-P trim redundant edges to lower traversal cost without compromising recall [2].

2.2.2 Quantization based

Quantization-based methods compress vectors, restrict search to query-relevant partitions, or both, trading a bounded distance error for large reductions in memory and latency. The partitioning idea is embodied by the Inverted File Index (IVF), which clusters the corpus with k-means and, at query time, scans only the n_{probe} posting lists whose centroids lie closest to q; a small n_{probe} is fast but low-recall, whereas a large one approaches brute-force scan [2]. Product Quantization (PQ) instead targets memory. It splits a d-dimensional vector into m sub-vectors and trains a k-centroid codebook per subspace, so that each vector is stored as m codebook indices and distances are computed asymmetrically from precomputed sub-distance tables without ever decompressing the codes [20]. Optimized PQ (OPQ) lowers the resulting quantization error by jointly learning an orthogonal rotation R alongside the codebooks so that variance is balanced

across subspaces [2]. The two ideas compose in IVF_PQ, which uses IVF to select candidate clusters and PQ to supply cheap approximate distances over their contents. This is the pipeline that underpins FAISS's billion-scale indexes [21]. Further variants specialise the scheme along particular axes. ScaNN applies an anisotropic vector quantization tuned for maximum inner product search, weighting the loss toward the error components that most distort relevance while tolerating error orthogonal to the query [2]. Online PQ updates codebooks and codes incrementally as vectors arrive, serving streaming corpora without offline retraining [2]. In practice, quantization is typically paired with a full-precision re-ranking pass over the top-N candidates to recover the recall lost to compression [21].

2.2.3 Hash based

Hash-based methods reduce ANNS to bucket lookup by mapping similar vectors to similar binary codes. Hash-based methods tend to plateau on recall in high dimensions and have largely been displaced by graph- and quantization-based methods in production vector databases [4]. The canonical instance is Locality-Sensitive Hashing (LSH), which draws from hash families whose collision probability $\Pr[h(x)=h(y)]$ rises with the similarity of x and y. Multiple hash tables amplify recall, and distance-specific families exist for L_2 (p-stable projections) and cosine similarity (random hyperplanes) [3]. Data-dependent variants attempt to sharpen these codes. Spectral hashing derives them from the eigenvectors of the data similarity graph, preserving local structure at the cost of assuming a particular data distribution [2]. Spherical hashing partitions the space with hyperspheres rather than hyperplanes, yielding tighter regions and more candidate neighbours per bucket [2]. Deep hashing learns the hash functions end-to-end with a neural network, producing more discriminative but training-data-dependent codes [2]. Polysemous codes bridge hashing and quantization outright, using Hamming distance over the codes as a fast filter and PQ for accurate re-ranking of the survivors [22].

2.2.4 Tree based

Tree-based methods recursively partition the vector space and prune whole branches at query time, but the partitioning loses effectiveness as dimensionality grows, which has motivated the shift toward graph- and quantization-based approaches for modern embedding workloads. The classical structures are KD-trees, which split on one axis-aligned dimension per level and are effective only for low dimensionality ($d \lesssim 20$) unless extended by randomised or multi-tree variants [2]. Ball-trees bound subsets with nested hyperspheres rather than axis-aligned boxes and so handle non-axis-aligned data better while still degrading as d grows [2]. Randomisation makes the tree based approaches practical at scale. Annoy (Spotify) builds a forest of random-projection trees, each splitting on a random hyperplane between two sampled points, and answers a query by traversing every tree and merging the candidate sets in a file-backed index

used in production [2]. Other variants change the partitioning primitive or the traversal order. The k-means tree applies hierarchical k-means so that each level induces centroid Voronoi cells [2], while Best Bin First reorders traversal with a priority queue so that the most promising bins are visited first under a fixed budget, keeping KD-tree-style search tractable at higher dimensionality [2].

The practical choice between families is driven less by raw recall and more by deployment constraints. Pure graph indices are preferred when the corpus fits in RAM, recall targets are aggressive, and queries are read-heavy. Quantization-based indices are preferred when memory is the bottleneck or when build cost must be amortized across very large corpora. Disk-resident graph variants such as Vamana bridge the two regimes when the working set exceeds RAM but a high-recall navigational structure is still desirable. Update patterns matter as well. HNSW supports incremental insertion well but is comparatively expensive to delete from. IVF requires periodic re-clustering as the distribution drifts, and online PQ variants exist precisely for streaming corpora [2].

2.3 ANNS Benchmarking

The proliferation of ANNS algorithm families described above has been accompanied by an equally large variation in implementation quality, parameter conventions, and self-reported evaluation protocols, which makes algorithm-to-algorithm comparison from primary sources unreliable [4]. Independent benchmarking studies have emerged in response, and the protocol they have converged on is now broadly standard. Ground truth is recomputed from scratch by exhaustive linear scan in the query metric, so all algorithms are evaluated against the same definition of correctness [4, 23, 19]. The primary quality measure is Recall@k, formalised distance-based to remain well-defined when several points share the threshold distance [4]. Efficiency is reported not as a single operating point but as a recall–QPS Pareto front. Each algorithm is run across a grid of parameter settings and the best attainable QPS at each recall is plotted [4, 19, 23]. Index build time and memory footprint are reported as secondary measures, and experiments are pinned to a single thread on fixed hardware to control for parallelism differences across implementations. The datasets used in this family of benchmarks are dense float (or, less commonly, binary Hamming) vector files. Correctness is defined purely as geometric proximity, with no associated metadata, predicates, or relevance judgements.

Among these studies, ANN-Benchmarks [4] has become the de facto standard against which new ANNS algorithms position themselves. The framework provides Docker-isolated wrappers for algorithm implementations, automated dataset download (HDF5 files bundling the corpus, query set, and precomputed k-NN ground truth at $k = 100$), and a YAML-driven experiment loop that traverses a parameter grid under a wall-clock budget per setting. The initial published evaluation covered fourteen implementations spanning all four ANNS families: graph-based (KGraph, SWGraph, HNSW [18], PyNNDescent, PANNG), tree-based (FLANN, BallTree, Annoy, RPForest, MRPT), LSH-based (FALCONN, MPLSH), and a residual category containing FAISS-IVF [21] and

Multi-Index Hashing. The canonical dataset suite (SIFT, GIST, GLOVE, NY-Times, Rand-Euclidean, SIFT-Hamming, and Word2Bits) covers Euclidean, angular/cosine, and Hamming metrics and is reused by essentially every subsequent ANN paper to give its results a place on the ANN-Benchmarks Pareto front. The companion website `ann-benchmarks.com` maintains a continuously updated leaderboard of community-contributed results, and the typical pattern in newer ANN work is to compare against the published Pareto front rather than re-run baselines from scratch.

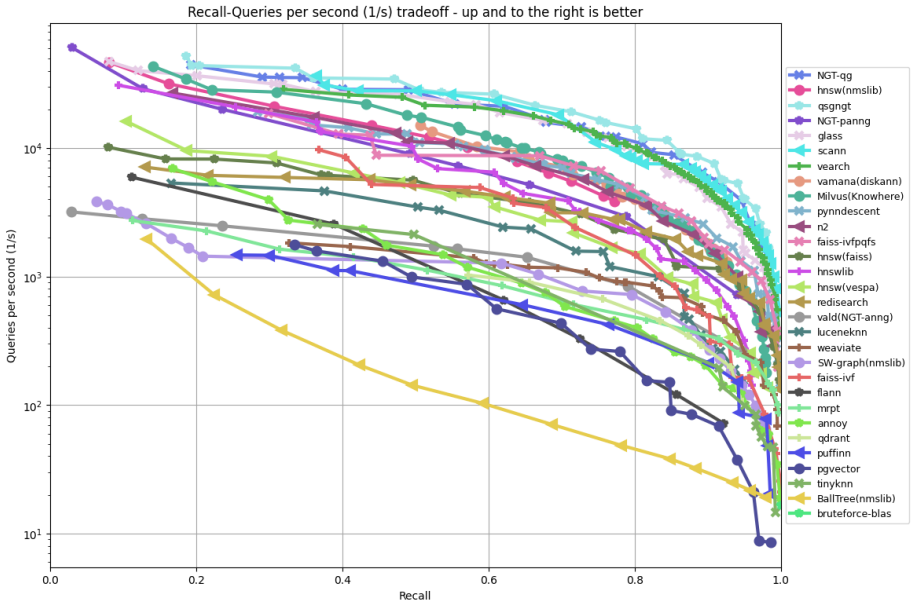


Figure 1: Recall–queries-per-second Pareto front on the `glove-100-angular` dataset. Retrieved from the live ANN-Benchmarks leaderboard `ann-benchmarks.com` [4] (retrieved 2026-08-27).

The picture these benchmarks paint of the algorithm landscape is consistent across the three independent studies [4, 19, 23]. Graph-based methods, and HNSW [18] in particular, consistently lead the recall–QPS trade-off on in-memory workloads across the canonical datasets. ANN-Benchmarks reports HNSW as the fastest algorithm at every recall level on GLOVE, with KGraph closing the gap only at near-perfect recall, and as the fastest of the tested algorithms on SIFT for 100-nearest-neighbour queries [4]. FAISS-IVF is the strongest non-graph competitor and remains close at moderate recall before falling away above approximately 0.9. Tree-based methods (Annoy, BallTree, FLANN) and LSH variants are consistently slower or unable to reach high recall within the experimental budget [4, 23]. Graph-based dominance is, however, not universal. On a synthetic dataset engineered to expose only local rather than global structure, HNSW and SWGraph fail above recall 0.86 while Annoy and

the k-NN-graph variants win, illustrating that the small-world property HNSW exploits depends on dataset geometry [4]. Two larger-scale studies extend this account in complementary directions. Wang et al. [19] re-implement thirteen graph-based algorithms in a unified C++ framework (WEAVESS) that decomposes each algorithm into seven components, and find that the empirical advantage of the family is attributable to a small number of design choices: HNSW's neighbour-selection strategy and NSG's seed-acquisition routine rather than to whole-algorithm novelty. Li et al. [23] broaden the comparison across all four ANNS families on twenty datasets and stratify hardness using *local intrinsic dimensionality* (LID) and *relative contrast* (RC). Their headline observation is that algorithm rankings on low-LID datasets (SIFT, MNIST, Audio) generalise poorly to high-LID datasets (GloVe, Crawl, GIST), so a single Pareto front rarely captures the full picture.

While ANN-Benchmarks has fixed its scope at single-machine, in-memory, dense, unfiltered, static workloads, the more recent Big ANN Challenge at NeurIPS'23 [5] reframes evaluation as a competition over fixed datasets with public and private query splits, ranked by maximum QPS subject to recall@10 $\geq$ 0.9. The *out-of-distribution* track uses Yandex Text-to-Image (10M vectors, 200-dim) where database vectors come from an image encoder and queries from a text encoder, so the query distribution is no longer a random holdout of the database and the standard ground-truth-by-Euclidean-distance assumption is tested against genuinely cross-modal retrieval.

VIBE [24] addresses a different limitation of ANN-Benchmarks, namely the age of its datasets. SIFT, GIST, and the MNIST variants consist of raw-pixel images or hand-engineered descriptors that predate the dense neural embeddings now standard in retrieval, and a benchmark built on them no longer reflects the data distributions encountered in practice. VIBE constructs its corpora from contemporary text and vision encoders, among them CLIP and DINO, and evaluates twenty-two open-source index implementations over eleven such datasets under the established recall–QPS Pareto-front protocol. The frontier is occupied by graph-based methods (Glass, SymphonyQG, NGT-QG) and clustering-based methods (LoRANN, ScaNN, IVF-PQ), each employing some form of vector quantization, while tree- and hash-based methods remain uncompetitive at high recall. A second component of the benchmark evaluates the out-of-distribution setting, in which queries and corpus are drawn from different distributions, over eight further datasets covering multimodal text-to-image retrieval and maximum-inner-product workloads.

2.4 FANNS

Filtered Approximate Nearest Neighbor Search (FANNS) augments the ANNS primitive with a structured predicate S over per-vector metadata. Given a query embedding q and a corpus $X \subset \mathbb{R}^d$, the answer is the top-k vectors that are closest to q *among those whose metadata satisfies* S [1, 25]. The parameter that controls the difficulty of a filtered query is the *filter selectivity* $\sigma_S = |\{v \in X : f_v \models S\}|/|X|$, the fraction of the corpus that satisfies the

predicate. The predicate vocabulary actually exercised in the literature spans equality, set containment, set overlap, fixed-length equality on categorical label sets, and range or inequality on numeric attributes.

The strategies that systems and algorithms use to evaluate a FANNS query fall along a workflow taxonomy. Pre-filter (filter first, then search), post-filter (search first, then filter), in-filter (push the predicate into the index traversal), and partition or unified indexes that jointly organise the corpus by both metadata and geometry [1, 26].

2.4.1 Pre-filtering

Pre-filtering evaluates the predicate first, typically against an inverted or scalar index, materialises the matching subset of identifiers, and then runs a search restricted to that subset. The simplest realisation is brute-force linear scan over the filtered set with full-precision distances, which serves as the recall-1.0 baseline in every FANNS benchmark [1, 5]. It is trivial to implement, guarantees perfect recall, and is fastest at very small σ_S where the filtered set itself is the bottleneck. Above roughly 1–5% selectivity at million-scale corpora, however, the linear scan dominates query latency, and the strategy degrades steeply. The matching subset changes per query, so no general-purpose ANN index can be precomputed over it. Substituting a smarter search for the linear scan does little to change this. A pre-filter can hand its materialised identifier set to any search routine, but because that set is query-specific the routine cannot amortize a prebuilt ANN index over it, which is precisely what confines the strategy's usefulness to the low-σ_S regime.

2.4.2 Post-filtering

Post-filtering takes the opposite stance: an unmodified ANNS index is queried over the full corpus, and the predicate is checked on candidates as they are produced. It requires no index modification and supports arbitrary predicates. In its graph realisation, standard HNSW search runs and each candidate is tested against the predicate before it is admitted to the result heap. If fewer than k valid hits survive, the query is re-issued with a larger $\mathrm{ef}_{\mathrm{search}}$. This makes post-filtering competitive when the predicate is permissive (high σ_S), but it degrades sharply as selectivity falls. Gollapudi et al. [25] report up to three orders of magnitude lower QPS than in-filter approaches once the matching fraction reaches roughly 1% or below, because the traversal then spends almost all of its distance computations on non-matching vectors. The retry-with-larger-ef termination is moreover heuristic, with no a-priori bound on how many escalations a query may require [1]. The same construction applies to a quantization-based index. Post-filter IVF_PQ lets IVF select the n_{probe} nearest cells, PQ supply asymmetric distances over their contents, and the predicate act at result-list construction [21]. It inherits PQ's memory advantage but suffers the same selectivity collapse as post-filter HNSW, now compounded by quantization error over an under-filled candidate set.

2.4.3 In-filtering

In-filtering integrates the predicate into the index itself either at construction time, by making the graph filter-aware, or at search time, by masking traversal with the predicate bitset. This way distance computations are spent overwhelmingly on satisfying vectors. This is the most active area of recent FANNS research and the regime where the algorithm-level gap to the pre- and post-filter baselines is largest.

The most developed in-filter algorithm is Filtered-DiskANN [25], which makes the Vamana graph label-aware in both of its core routines. `FilteredGreedySearch` follows only those out-neighbours whose label set intersects the query filter F_q, and `FilteredRobustPrune` retains an edge (a, c) only when a and c share one of the labels that c carries, so that pruning itself respects the filter. Two construction strategies build on these routines. `FilteredVamana` grows a single incremental graph covering the full label set, whereas `StitchedVamana` builds a separate subgraph per label, takes their union, and re-prunes. On the Turing dataset both variants sustain recall@10 above 90% as filter specificity falls from 10^{-1} to 10^{-6}. Here, the IVF inline and Vamana post-processing baselines fail to reach comparable accuracy and run up to three orders of magnitude ($\sim$1000$\times$) slower. The design is SSD-resident and scales to thousands of distinct labels. The implementation does carry limitations in that each point is assigned a single *primary* filter. The algorithm targets singleton F_q queries (multi-filter conjunctions are left to future work, while disjunctions are handled by per-filter union). Because label awareness is fixed at build time, any schema change forces a re-index.

ACORN [27] occupies the opposite design point. The index is constructed with no knowledge of the predicate, and filter-awareness is deferred entirely to search time. Its motivating observation is that HNSW's relative-neighbourhood-graph (RNG) pruning removes an edge $v \to b$ whenever some intermediate node a is closer to b than v is. Should a fail the predicate, the removed edge is precisely the one needed to keep the predicate-satisfying subgraph connected. ACORN-γ therefore over-builds an HNSW with its degree inflated to $M \cdot \gamma$, retains the M_β nearest neighbours explicitly, and replaces RNG pruning with a metadata-blind two-hop heuristic. At query time the predicate bitset masks the traversal, with an optional compression-aware lookup that expands to two-hop neighbours to recover pruned edges. The lighter ACORN-1 keeps a standard HNSW and performs the two-hop expansion only at search time, cutting time-to-index by 9–53$\times$ at the cost of 1.5–5$\times$ lower QPS. ACORN handles equality, range, regex, and set-membership predicates, including OR semantics, and the authors report 2–10$\times$ higher QPS than Filtered-DiskANN on low-cardinality predicate sets (SIFT1M, Paper), 30–50$\times$ on high-cardinality sets (TripClick, LAION-1M), and over 1,000$\times$ on LAION-25M. The ACORN-γ index is up to 11$\times$ more expensive to build than a vanilla HNSW. Predicate-subgraph connectivity is not guaranteed in theory but rests on a user-specified minimum selectivity $s_{\min}$, below which the system falls back to pre-filtering. The metadata-blind graph incurs a documented 36–53% overhead in distance computations relative

to an oracle partition index.

2.4.4 Partition and range-specialised structures

Another family neither pre- nor post-filters but builds structures specialised to the shape of the filter itself: range filters over a totally ordered attribute, or tag bags in which the filter's cardinality structure becomes the primary index key. SeRF [28] targets range filters on a single numeric attribute. Inserting points in attribute-sorted order yields a sequence of n prefix-HNSW graphs, and annotating each HNSW edge with the interval $[b, e]$ of prefix-graphs that contain it collapses all n of them into a single *segment graph* of size $O(nM)$ for half-bounded ranges. A two-dimensional segment graph generalises to arbitrary $[x, y]$ ranges at an expected $O(nM \log n)$ size. SeRF thereby serves any range filter on the attribute without per-query rebuilding, and reports substantial recall–QPS gains over both ANNS-first and range-first baselines on real datasets. Its limitations follow directly from its specialisation. A single totally ordered numeric attribute, no native categorical or multi-attribute support, and a per-edge memory footprint larger than vanilla HNSW. UNIFY [29] attacks the same range-filtered problem with a unified index. Its Segmented Inclusive Graph (SIG) partitions the dataset by attribute value so that the proximity graph induced by any combination of segments is itself a subgraph of SIG. This lets a single index serve pre-, post-, and hybrid-filter strategies without the consistency and maintenance costs of keeping separate indexes in step. A hierarchical variant, HSIG, layers an HNSW-style structure over SIG to attain logarithmic hybrid-filter complexity. UNIFY reports state-of-the-art range-filtered ANNS performance across small, medium, and large query ranges but, like SeRF, remains confined to single-attribute range filters. Tag-structured filters are addressed by the two winners of the BigANN'23 filtered track [5], run on YFCC CLIP image embeddings with conjunctive tag predicates. ParlayANN indexes by tag rather than by vector. High-cardinality tags receive a Vamana subgraph and a spatial inverted index, low-cardinality tags receive plain lists, and a two-tag query chooses among bitvector, candidate-list, and linear-scan intersection strategies according to the cardinality pair. This achieves roughly 11× the track baseline's QPS at recall@10 of 90%. The runner-up, Baidu Puck, builds a four-level hierarchical cluster index (vector quantization at the upper levels, product quantization at the lowest) and labels each cluster with the union of its members' tags, so that whole centroids can be pruned by the query predicate. Both designs are tuned explicitly to the track's restriction to one- and two-tag conjunctions, and the organisers caution that such specialised structures may transfer poorly to richer predicate shapes.

Across these families the choice of strategy is principally a function of selectivity and predicate shape. Pre-filter wins at very small σ_S, post-filter at very large σ_S, and in-filter or partition-based approaches occupy the middle and dominate on structured predicates such as ranges, label-set containment, and tag bags. The workflow taxonomy used above (pre-filter, post-filter, in-filter, partition) is the one adopted by the Shi et al. unified benchmark [1]. Lin et al.

[26] organise the same space by which side prunes (scalar-side, vector-side, or joint pruning), and the two views are largely equivalent rephrasings of the same design space. How these algorithm-level strategies are exposed by production vector database systems, including pgvector and ChromaDB is taken up in the Vector Databases subsection below.

2.5 Vector Databases

The algorithm-level strategies of the previous subsections are rarely what an application developer works with directly. They are exposed through vector database systems, which wrap an ANN index in the machinery needed to store vectors, attach metadata, and evaluate combined similarity-plus-predicate queries. Such a system is organised in three layers. Durable vector storage, one or more of the high-dimensional indexes surveyed above (in practice HNSW or an IVF/PQ variant), and a metadata-and-query layer that holds per-vector attributes, evaluates filters, and coordinates the interaction between filtering and vector search [2]. It is this last layer that decides which of the pre-, post-, and in-filter strategies a given predicate actually receives, and how much of the algorithmic performance surveyed above survives the surrounding system.

Two architectural philosophies divide the field. *DB-first* systems embed vector search into an existing database engine, inheriting its storage manager, transactions, and query language. *Vector-first* systems are built around the ANN index and add metadata and query facilities on top. pgvector an example of a DB-first design. A PostgreSQL extension that adds a `vector` column type, the standard distance operators, and HNSW and IVFFlat indexes, while leaving predicate evaluation to PostgreSQL's own scalar indexes and cost-based planner.

ChromaDB sits on the vector-first side but at different scales. ChromaDB stores metadata as key–value pairs alongside each embedding and evaluates filters inside its own retrieval pipeline rather than through SQL, applying the predicate as part of the search rather than as a separate query stage.

3 Literature Review

3.1 The pure-ANN template

Approximate nearest-neighbour benchmarking has converged on a common evaluation template, which the filtered benchmarks examined later largely inherit. ANN-Benchmarks [4] established the arrangement that is now standard. A corpus of flat floating-point vectors carrying no accompanying metadata, a query set drawn either as a random holdout from the corpus or from a predefined split shipped with the dataset, brute-force exact k-nearest-neighbours as ground truth, and Recall@k reported against throughput (queries per second) as the principal trade-off.

The template fits the pure-ANN problem because that problem is defined entirely within the geometry of the vector space. The correct answer to a

				Filters exercised					
Benchmark	Corpus	Labels	Queries	Cat.	Rng.	Set	Cross	JOIN	Schema
Lu et al. [30]†	mixed	synthetic	holdout	*bitmap (predicate-agnostic)*					single
Li et al. [7]	mixed	synthetic	holdout	✓	✓		✓		single
Iff et al. [6]	modern	real	LLM	✓	✓	✓	○		single
Shi et al. [1]	mixed	real	holdout	✓		✓			single
BigVectorBench [31]	modern	real	holdout	✓	✓				single
BigANN'23 [5]	modern	real	holdout			✓			single
Amanbayev et al. [32]	modern	real	holdout		✓			○	**multi**

Table 1: The reviewed filtered-ANN benchmarks along the axes of this section.

query is, by construction, the k corpus points closest to it under the chosen distance metric. There is no schema to model, no relevance to judge, and no task beyond proximity, so the ground truth is exactly computable by a linear scan and authoritative at the same time. Nothing outside the vector space bears on which answer is correct. Recall@k measures the fraction of the true neighbours an approximate index recovers, and the setup is reproducible because no part of it depends on outside judgement.

3.2 What filtering breaks

Attaching a predicate to a query is a minor change in form, but it adds substantially to what a benchmark must commit to. In form, the question shifts only from "the k vectors in the corpus closest to q" to "the k vectors closest to q among those satisfying a predicate ϕ". The added clause requires the benchmark to take positions on several questions the pure-ANN template did not face. Which attributes each corpus item carries, how those attributes are distributed, which predicates over them are well-formed, whether the attributes correlate with the embedding geometry or vary independently of it, and how the attributes are organised, whether within a single flat relation or across several tables joined by key. None of these commitments appears in the inherited template, because none arises in the pure-ANN problem. Each is a degree of freedom along which a filtered benchmark can sit closer to or further from a realistic workload, and the subsections that follow examine, one axis at a time, where the reviewed benchmarks fall.

3.3 Compromises in the current literature

The reviewed benchmarks match the pure-ANN template on some axes and depart from it on others. Table 1 summarises where each sits, one row per benchmark and one column per axis of comparison, and the subsections that follow take up these axes in turn. Two entries need qualification. The BigANN'23 row [5] describes only that competition's filtered track, setting aside its pure-ANN

out-of-distribution, sparse, and streaming tracks, which lie outside the filtered problem. The Lin survey [26] is left out of the table altogether, because it charts the design space of filtered-ANN methods without running a benchmark of its own and so has no position to record on these axes. Its observations still inform the discussion below.

3.3.1 Corpus realism

The *Corpus* column separates the benchmarks by the age of their embeddings. These range from pre-neural descriptors such as SIFT, which summarise local image texture around an interest point rather than image content [7, 30], and frozen Inception features from YouTube-8M [1], to modern contrastive image-to-text and transformer embeddings over LAION, YFCC, RedCaps, and arXiv [1, 7, 6]. Three of the reviewed benchmarks (Lu et al., Li et al., Shi et al.) still fold descriptor types no longer used in production systems into corpora that are otherwise contemporary. The column marks these *mixed* to set them apart from the fully *modern* suites.

The *Labels* column describes how benchmarks choose their metadata. Where a corpus carries rich native metadata that will not fit a single-attribute, fixed-selectivity filter scheme, the benchmark tends to set that metadata aside. Li et al. [7] replace the Knowledge-Graph tags of YouTube-8M and the subreddit labels of RedCaps with a uniform-random vocabulary of 500 labels, retaining only a single harvested scalar per corpus (view count, post timestamp) for their range filters, and pair SIFT and SpaceV, which carry no side information at all, with synthetic attributes outright. Where real labels are kept, they are commonly pruned for usability, as when Shi et al. [1] reduce TripClick to its 27 most frequent clinical areas and arXiv to 26 equality labels. Because the substituted attributes are drawn independently of the vectors, any correlation between an item's metadata and its position in the embedding space is lost, a property of real collections that Li et al. place out of scope and record as an open problem [7].

3.3.2 Query construction

The *Queries* column is dominated by holdout construction, and this holds even where an alternative is available. The pure-ANN studies hold out random points even on SIFT, GIST, and BIGANN, datasets that ship purpose-built query files [23]. Among the filtered benchmarks Shi et al. [1] curate the holdout so that every query keeps at least 10 ground-truth results after filtering. Iff et al. [6] use search terms generated by GPT-4o , and the BigANN'23 filtered track, though it draws its tag predicates from genuine image metadata, caps each query at one or two tags. The organizers note that this leaves specialised structures as "less interesting for a more general setting" [5]. No reviewed benchmark draws its queries from a real session log.

When the query distribution diverges from that of the base corpus, as in cross-modal retrieval where a text query is issued against an image corpus, graph

indexes built and tuned on in-distribution holdout queries lose substantial recall and throughput. This is because the neighbours a real query seeks are poorly connected within a graph organised around the base distribution. RoarGraph [33] addresses this by reconnecting the graph through a query-distribution-aware projected bipartite structure, and Yue et al. [34] restore graph monotonicity across the two distributions. Both report large gains over distribution-agnostic HNSW. The BigANN'23 out-of-distribution track (Yandex Text-to-Image) is the benchmark-level acknowledgement of the same effect [5]. VIBE [24] also acknowledge this effect, reporting across eight out-of-distribution datasets, several of them text-to-image, that general-purpose indexes lose measurable recall and throughput once the query distribution departs from that of the corpus. Holdout-from-corpus queries therefore under-sample the real workload, and they fall on the easier side of a distribution gap that this literature has shown can alter which method performs best.

3.3.3 Filter complexity

The filters the benchmarks actually exercise are built from a small set of operators over three attribute types: equality on categorical attributes, range and inequality on ordered numeric ones, and containment, overlap, and fixed-length equality on set-valued label fields. Iff et al. [6] organise their workload along this structure, anchoring one operator to each attribute type (exact match on a category count, range on a publication date, exact-match-in-set on a category set), while Shi et al. [1] sweep four set-semantics scenarios over real label sets. Where conjunctions appear, they stay shallow, and with a single exception they stay within one attribute type. Li et al. [7] pair one categorical with one numeric attribute while BigVectorBench conjoins up to five categorical labels [31] and the BigANN'23 track admits at most two tags [5].

Query difficulty is controlled principally through *selectivity*, the fraction of the corpus a predicate admits. Li et al. [7] construct exact-selectivity windows and sweep the full range from 0.1% to 100%; Shi et al. [1] bucket queries by selectivity percentile and also vary label-set length and k. Iff et al. [6] rely on the differing selectivity distributions of their attributes (bimodal, uniform, and long-tailed) to span the range. In each case the filter workload serves mainly to place an algorithm on a selectivity axis, along which the relative performance of the pre-filter, post-filter, in-filter, and hybrid strategies changes.

Genuine heterogeneity of filter conjunctions is barely touched. The only conjunction across attribute types in any reviewed benchmark is Li et al.'s synthetic categorical-plus-numeric filter, which combines two fabricated attributes rather than the categorical, numeric, and string predicates a real schema mixes. No benchmark exercises a string-pattern predicate, and none spans a JOIN, even though several of the schemas define such predicates [6, 32]. The single-attribute predicate, optionally conjoined with others of its own type and swept across selectivity, marks the limit of the workloads these benchmarks exercise.

3.3.4 Schema

The *Schema* column is the axis on which the reviewed benchmarks depart least from the pure-ANN template: it reads "single" for every benchmark but one. Relational structure is not treated as a design variable at all. The corpus is a single flat relation of vectors and their attached attributes, and a predicate ranges over it. Amanbayev et al. [32] are the lone exception, but even they do not utilize the multi-table structure they build. Their benchmark uses a two-table schema (Movies and Reviews, joined on a `mid` foreign key) that anticipates JOIN-plus-ANN workloads, yet runs no JOIN predicate, because pgvector's optimiser reverts to a sequential scan on a cross-table predicate and bypasses the vector index. The schema is set up and then left unused, so the relational case is one the reviewed literature describes but does not evaluate.

3.3.5 Ground truth and metrics

The final axis is one the reviewed benchmarks hold constant. Every reviewed benchmark constructs ground truth as exact k-NN over the filter-restricted subset, meaning the k vectors satisfying the predicate that lie closest to the query, and scores an index by Recall@k against throughput. This uniformity reflects an unmodified inheritance from the pure-ANN template, one the filtered benchmarking literature does not question. BigVectorBench extends the *set* of metrics, reporting end-to-end embedding latency alongside Recall@k and throughput on the argument that embedding cost dominates retrieval cost in a deployed system [31]. Whether geometric correctness is an adequate proxy for success at a real retrieval task is a question this literature does not raise. It is taken up, as a contribution of this thesis, in the VBS workload characterisation and the experiments that follow.

3.4 Systems versus algorithms

A final distinction cuts across these axes: whether a benchmark evaluates an algorithm or the system that hosts it. Most of the reviewed benchmarks take the algorithm as their object. Iff et al., Shi et al., and Li et al. [6, 1, 7] evaluate research implementations of filtered-ANN algorithms, among them ACORN, Filtered-DiskANN, UNG, SeRF, NHQ, and CAPS, under unified harnesses in which the surrounding system is not under study.

Two of the reviewed works look at the system instead, and both find that the system materially alters the algorithm's behaviour. Amanbayev et al. [32] benchmark pgvector, Milvus, and FAISS as integrated systems and report that pgvector's cost-based optimiser routinely selects an approximate index scan where an exact sequential scan would return perfect recall at comparable latency. Lu et al. [30] measure latency gaps of roughly 5–10× between filtered-ANN algorithms running standalone and the same algorithms running inside pgvector, and attribute the gap to buffer locking, the translation between index and heap tuple identifiers, page access, and transactional consistency.

3.5 What remains uninvestigated

The compromises traced across these axes follow from carrying the pure-ANN template into a problem it was not designed for, and together they define what the current benchmarking literature leaves uninvestigated. That remainder maps onto the thesis's three research questions.

The workloads these benchmarks exercise are single-attribute, single-table, and drawn either from the corpus or from a language model rather than from users. Whether they resemble the workloads a real multimedia retrieval system faces is not something the benchmarks can settle, because none of them observes a real workload. Establishing what such a workload looks like, and measuring its distance from the benchmarked workloads, is the object of the VBS workload characterisation that follows and the first question this thesis sets out to answer.

The systems-versus-algorithms discussion isolates a second gap. The optimiser fallbacks that Amanbayev et al. exclude from their own experiments [32] and the system-level overheads that Lu et al. quantify [30] indicate that native support for a filtered workload is uneven across systems, and that a practitioner will at times have to work around it. How pgvector and ChromaDB handle the query patterns of a real multimedia workload, and what workarounds are required where native support falls short, is the second question.

The ground-truth axis points to the third. The reviewed literature evaluates filtered retrieval only against geometric ground truth and never asks whether geometric correctness coincides with success at the underlying task. Whether the standard Recall@k evaluation suffices to capture the utility of a real multimedia retrieval system, or whether a benchmark grounded in a real workload must instead measure task success directly, is the third question this thesis takes up.

4 VBS Workload Characterisation

The literature established that the reviewed benchmarks exercise single-attribute, single-table workloads drawn from the corpus or from a language model, and that whether these resemble a real multimedia workload cannot be settled from within the benchmarks. This section serves to answer that by analyzing the queries logged during the Video Browser Showdown, a video-retrieval competition, over its V3C video collection. Firstly, the geometry of the V3C corpus is compared against corpora used in existing benchmarks. Secondly, query origin, filter complexity, success metrics, and schema structure is analyzed. The aim is to establish what a real workload looks like and how far it sits from the benchmarked ones. The workload throughout is competition known-item search over a fixed video collection.

4.1 Competition workload

The Video Browser Showdown (VBS) is a live video-retrieval competition held over the V3C collection [8]. Participating teams issue text queries against the

collection under time pressure to locate a designated target segment. Every property the pure-ANN template removes is present here: the queries are authored by people rather than held out from the corpus, the collection is fixed and shared across teams, and success is defined by the competition rather than by proximity in the embedding space. The competition runs several kinds of tasks. The queries used here are those from its known-item search (KIS) tasks, where the target is a portion of a video spanning one to a few available shots. The workload characterization done in this section was on 529 known-item queries, issued by two competing systems (PraK, 421; Exquisitor, 108) and joined to the official server run log by timestamp across 20 tasks. Each task carries a ground-truth target, given as a video identifier and a temporal range, together with the progressive hint text released to the teams over the course of the task.

These queries are shaped by the two systems that produced them. They reflect how expert users formulate text queries at these two systems on a known-item task, rather than how users search in general.

4.2 Corpus geometry

Wang et al. [19] and Li et al. [23] establish that a corpus's geometric hardness, captured by high local intrinsic dimensionality (LID) and low relative contrast (RC), determines how an approximate index performs and can reorder which algorithm wins. A concern that benchmark results fail to transfer to a real corpus borrows this finding. A different corpus geometry would imply different results. The hypothesis under test is thus: real multimedia-retrieval corpora differ from the benchmark corpora on LID and RC, the properties the field uses to predict difficulty.

LID is reported using the maximum-likelihood estimator at $k = 100$, so the reference values are those of Aumüller and Ceccarello [35], computed with the same estimator, and not the values published by Wang and Li, whose figures use a different estimator. RC, defined as $D_{\text{mean}}/D_{\text{min}}$, is estimator-independent and is compared directly. The result, in Table 2, does not support the hypothesis. V3C lands in the centre of the canonical cluster, with a mean LID near 20 and an RC in the middle of the pool, and it does so despite a nominal dimension of 768 against the 100 to 128 of the reference corpora. The higher nominal dimension is therefore not the operative variable. On the geometry that existing literature predicts search difficulty, V3C is not distinguishable from the corpora the benchmarks already use.

V3C is of course not geometrically identical to the reference corpora. Its SigLIP embeddings are anisotropic, occupying a narrow cone ($D_{\text{mean}} \approx 0.97 < \sqrt{2}$) in the manner characteristic of contrastive image-to-text models, and the corpus contains near-duplicate keyframes drawn from adjacent video frames. Neither of these properties registers on LID or RC.

Whatever gap exists between the benchmark workloads and a real multimedia workload does not lie in base-corpus geometry, and so must lie on the other axes: the queries, the filters, and the definition of success. The canonical benchmark pool is itself geometrically homogeneous, with algorithm rankings

Corpus	LID (mean)	LID (median)	RC
GLOVE (cosine)	18.0	17.8	1.82
SIFT	21.9	19.2	3.50
GNEWS (cosine)	21.1	20.1	—
V3C SigLIP (cosine)	**20.3**	**18.7**	**2.50**

Table 2: V3C against the canonical benchmark corpora on the geometry the ANN literature uses to predict difficulty. LID by the MLE estimator at $k = 100$ [35]; RC is $D_{\text{mean}}/D_{\text{min}}$. V3C is indistinguishable from the pool despite a much higher nominal dimension (768 vs 100–128). Measured on V3C1 keyframes ($n = 1{,}082{,}566$).

stable across its members [35], so a new real-modality corpus falling inside that cluster is unremarkable. What distinguishes a real workload has been found to lie in its metadata, largely independent of the embedding geometry [30].

4.3 Query source

The queries differ first in where they come from. All 529 known-item queries are text that real users typed at a live system while attempting a task. No reviewed benchmark draws its queries from a session log but rather hold out vectors from the corpus, or, in one case, generate the search terms with a language model [6].

As established in the literature review, holdout queries under-sample real difficulty because they sit in-distribution with the corpus the graph is organized around. A logged text query issued against an image corpus is exactly this out-of-distribution case.

4.4 Filter complexity and schema

Each of the 529 known-item queries was annotated against a fixed predicate taxonomy. The labels are produced by a language model rather than by human annotators, so the distributions below are best read as estimates of the underlying frequencies.

The most direct contrast with the benchmarks concerns conjunction depth. Of the 529 queries, 97.2% express a conjunction of predicates, and only 5 carry a single predicate. The modal query states four or five predicates, and the distribution has a tail reaching sixteen. The reviewed benchmarks assume close to the opposite. They cap conjunctions at one or two clauses of a single attribute type and vary difficulty through selectivity, so conjunction depth is a dimension their workloads hold fixed. The real workload ranges widely across it, the largest single gap between the two.

The conjunctions are also heterogeneous in a way the benchmarks' are not. A single real query routinely combines predicates of different attribute types, conjoining an object with a colour, a spatial relation, and an action, whereas the

Predicate type	Count	Filterable today
object	1026	yes
colour	452	no
action	450	no
person	268	no
scene	250	yes
spatial	211	no
count	84	no
camera	62	no
temporal	24	no
OCR	5	yes

Table 3: Predicate-type frequency across the 529 known-item queries, and whether each type is expressible in the scene/object/OCR metadata available today. LLM-annotated.

conjunctions the benchmarks admit are shallow and, with a single exception, same-typed, such as several categorical equalities.

The visual content of a keyframe is not filterable on its own, so this thesis runs a set of automated extraction passes over the V3C keyframes to derive structured metadata from them. Three dimensions result: scene labels, object detections, and on-screen text recognised by optical character recognition (OCR). The passes that produce them, and the taxonomies they use, are described in the Methods. Here they matter only as the set of attributes a query can currently be filtered on. Only about half of a typical query can be expressed in this metadata. Across the 529 queries, the mean fraction of a query's predicates filterable in these three dimensions is 0.46. Table 3 shows where the other half falls. Colour, action, person, and spatial predicates are among the most frequently expressed types, and none is filterable in those dimensions. The gap is concentrated in a small set of high-frequency attribute types, which names concretely what a benchmark closer to this workload would need to add. Negation, by contrast, is uncommon, appearing in 4.9% of the queries.

Expressing these predicates is a multi-relational problem. Scene labels, object detections, and OCR spans each associate many rows with a single keyframe, so a schema that represents them faithfully spreads a query's predicates across several relations joined by key, and evaluating the query requires combining them.

Real filter workloads are, on this evidence, deep by conjunction count, heterogeneous across attribute types, and multi-relational in the schema they require.

4.5 Task success versus geometric correctness

Success in VBS is defined at the level of the task. For a known-item task, a submission counts as correct when it names the target video and a temporal

range within it, entered before the task's time limit expires, and the competition server records each submission as correct or wrong together with the time it was made. The win condition is identity with a single designated item, which geometric proximity in the embedding space approximates but does not define. A result can be among a query's nearest neighbours without being the target, and the target can sit outside the top k a system returns.

The reviewed benchmarks measure only the geometric side of this relation. Each constructs ground truth as exact k-NN over the filter-restricted subset and scores an index by Recall@k against that ground truth, and none asks whether geometric correctness coincides with success at an underlying task. On a workload with a task-level notion of success, the two come apart in principle, and adjacent evidence suggests they do so in practice. Averaged Recall@k has been shown to conceal per-query tail failures [36], and in retrieval-augmented generation, conventional retrieval metrics correlate only weakly with downstream task success [37].

Because the ground truth of a VBS known-item task is a specific item rather than a neighbourhood, a benchmark built on it can report task success directly and place it beside Recall@k, which a benchmark whose ground truth is geometric k-NN cannot. How to report such a task-level measure will be taken up in the Methods section. Whether it diverges from Recall@k on this workload, and by how much, is measured in the experiments.

4.6 Summary

Across the before mentioned axes, the real workload matches the benchmark pool on one and departs from it on the rest. On corpus geometry the two are matched, and a gap on representativeness is ruled out. On the source of the queries it diverges, being authored by real users at a live system rather than held out from the corpus or generated by a language model. On filter complexity and schema it diverges further, being deep in conjunction, heterogeneous across attribute types, and multi-relational in the schema it requires. On the definition of success it diverges again, admitting a known-item task win condition that geometric recall does not measure.

5 Methods

This thesis introduces FRAME, a filtered-ANN benchmark for known-item search over the V3C collection. FRAME retains some structural properties from existing ANN and FANN benchmarking (a fixed real corpus, a single shared embedding model, exact brute-force ground truth, a thin per-system adapter, and scoring logic written once) while changing two elements of the standard template. First, queries are phrasings real users typed at a live retrieval system, decomposed into a semantic part and a structured predicate, rather than vectors held out from the corpus. Second, the filter attributes are set-valued visual metadata distributed across multiple relations, rather than a single synthetic

label per vector. These two departures allow the suite to measure whether pushing an attribute out of the embedding and into a predicate helps a user locate a specific item, and what a system must give up to answer such a predicate at all. The remainder of this section defines the corpus, the query set, the ground truth, the system interface, the measurement procedure, and the named workloads that the experimental setup consumes.

5.1 What FRAME measures

On the axes the literature review uses to compare filtered-ANN benchmarks (Table 1), FRAME matches the pool on corpus geometry and departs from it on every other. Its corpus is a contemporary neural embedding (SigLIP) shown in the workload characterisation to be geometrically indistinguishable from the benchmark pool. Its filter labels are neither the pre-existing metadata nor the synthetic labels many reviewed benchmarks use, but attributes derived from image content, so they correlate with the embedding and carry extraction error. Its queries are text real users typed at a live system, decomposed into a semantic part and a predicate, rather than vectors held out from the corpus. Its filters are set-membership predicates over a normalised multi-relation schema, evaluated as cross-relation semi-joins against the label tables. Finally, where every reviewed benchmark holds ground truth constant at exact filtered k-NN scored by Recall@k, FRAME reports that geometric measure and, beside it, a task-level one keyed to the designated known item.

The retrieval task is known-item search. Each query has a correct answer, designated in advance by the competition, and a user succeeds only if that answer appears high enough in the returned ranking to be found. FRAME reports two notions of correctness: Recall@k against the exact k-nearest neighbours under the predicate, and the rank of the designated known item. The two can disagree. A system may retrieve the exact neighbourhood the oracle identifies while placing the known item below any practical cutoff, or rank the known item first while missing much of the neighbourhood.

The suite follows an adapter-plus-shared-harness architecture in which a system is integrated by implementing three methods: ingesting the canonical corpus into the system's physical layout, preparing a run, and answering a single query. Everything downstream of the returned identifiers (scoring, ground-truth joining, aggregation, and visualisation) is written once and inherited unchanged. This split follows ANN-Benchmarks [4]. The notion of a published logical specification that independent implementers answer follows the TPC benchmarks [38].

Figure 2 traces the end-to-end path of a query. A query enters as text a user typed together with a structured predicate over corpus metadata. The shared embedding model encodes the text into a query vector. The adapter translates the predicate into its system's native filter syntax and executes a filtered nearest-neighbour search, returning a ranked list of corpus identifiers ordered by estimated similarity. The analyzer joins the returned identifiers against ground truth computed independently of every system under test and produces Recall@k, the rank of the known item, MRR, and query latency.

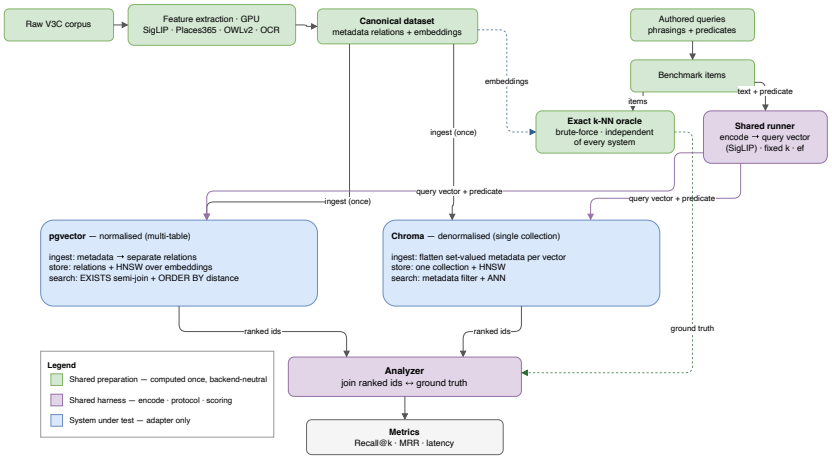


Figure 2: The FRAME pipeline.

5.2 Corpus and filterable metadata

The corpus is V3C [8], the standard collection of the Video Browser Showdown: web video, shot-segmented, with one keyframe extracted per shot. V3C is distributed as three partitions, V3C1 through V3C3. The query set developed for this work targets V3C1 (7,475 videos segmented into 1,082,657 shots, yielding 1,082,566 keyframes), but the retrieval corpus is the V3C union, with all three partitions loaded (4,143,591 keyframes in total), so that each query retrieves against the full distractor pool. A query set spanning V3C2 or V3C3 targets is future work. Each keyframe carries a single 768-dimensional embedding from a SigLIP vision–language model. The keyframe is the unit of retrieval, the unit a predicate selects, and the unit to which the schema of Figure 3 attaches metadata. A shot contributes one keyframe and therefore one vector; shot and video structure exist in the schema for provenance rather than as retrieval units.

The filterable attributes are the output of content-extraction passes run over the keyframes. Because they are derived from image content, they are correlated with the embeddings by construction, and, unlike synthetic labels, they carry extraction error. Three passes supply the metadata the query set draws on. Scene classification uses the 365-category Places365 taxonomy [39], retaining the top five categories per keyframe with their associated probabilities. Open-vocabulary object detection via OWLv2 [40] runs over a fixed 143-term vocabulary, of which 142 terms occur in the corpus. Each detection carries a label, a confidence score, and a bounding box. Optical character recognition yields in-frame text spans indexed for substring matching. OCR is present in the schema but used by only a single query, where extraction error defeats the filter entirely, making it an instance of the excluded-answer problem discussed in the query-set subsection rather than an exercised filter type.

The extraction thresholds that determine which labels are retained are pinned

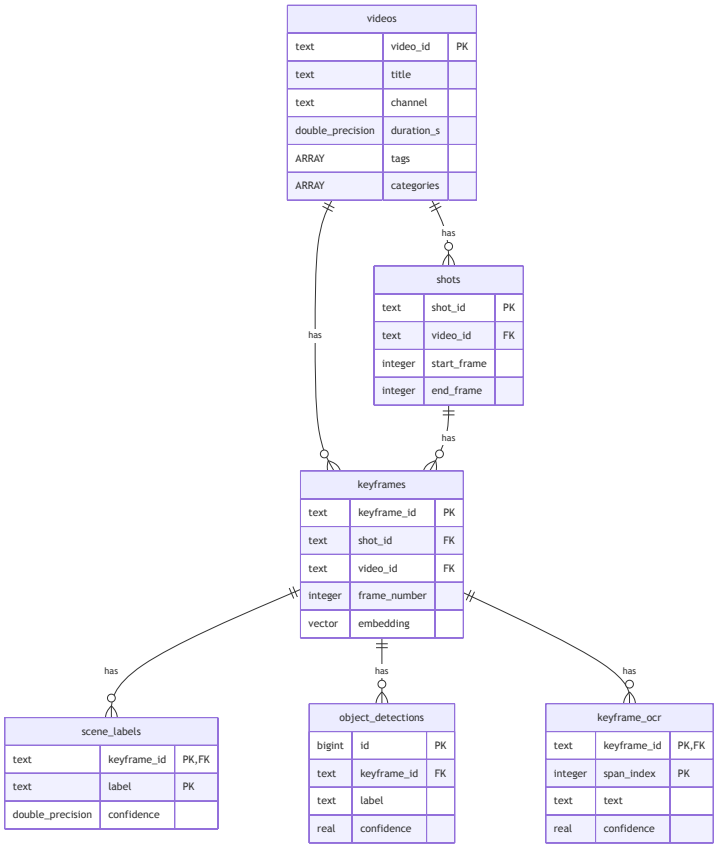


Figure 3: The logical schema every system must answer against. Scene labels, object detections, and OCR spans are many-rows-per-keyframe; the embedding is stored once, on the keyframe. This shape, not any particular physical layout, is what the suite fixes.

at ingest rather than applied at query time. The scene threshold is set at a probability of 0.10 (approximately 37 times the uniform baseline of 1/365 in the Places365 softmax distribution), so that a retained category represents a non-trivial classification signal rather than noise. The object threshold is set at a confidence of 0.30, chosen empirically from target-confidence-versus-selectivity diagnostic curves so that known target detections pass with margin while pruning the candidate set to approximately 0.4% of keyframes per label. Figure 4 shows where each threshold falls on the detector's own output distribution. Both values are then frozen across all systems and all runs.

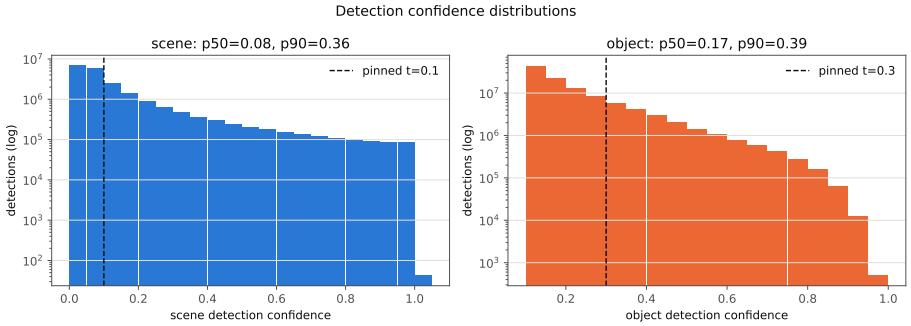
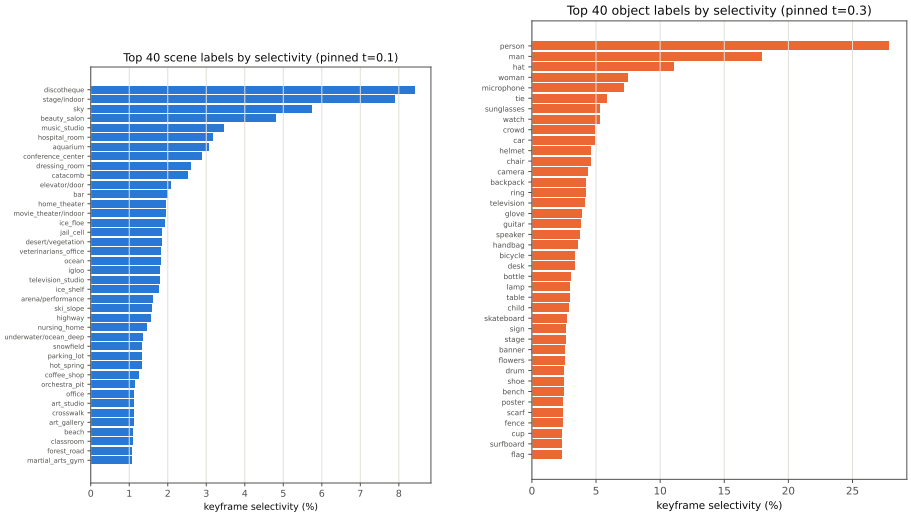


Figure 4: Extraction confidence distributions against the thresholds pinned at ingest. The pinned value determines which keyframes satisfy a predicate, and therefore fixes predicate selectivity identically for every system under test.

The derived metadata is substantially denser and more skewed than the uniform synthetic label sets used by the reviewed benchmarks, which is why realistic conjunctions behave differently from swept-selectivity workloads. Object detection covers every keyframe, so no selectivity figure is confounded by partial coverage. Per keyframe, the extraction passes produce up to 5 scene categories, up to 37 distinct object labels, and up to 314 individual detections (Figure 6). Label frequency is strongly skewed (Figure 5): the two broadest object labels, both person-denoting, select 27.8% and 17.9% of keyframes, while approximately 62 of the 142 object labels each select under 1% of the corpus. Over a grid of the scene and object labels that appear in the query set together with the most frequent labels overall, 2,695 of 2,703 pairs co-occur at least once (Figure 7), so conjunctions are rarely empty, though the near-universal person-denoting labels contribute almost no selectivity and are excluded from filter construction.

The logical schema (Figure 3) is normalised. Scene labels and object detections are stored as separate relations with one row per (keyframe, label) pair, and the 768-dimensional embedding is stored once, on the keyframe. The metadata is set-valued and many-per-keyframe, so a normalised layout is the natural relational representation. No claim is made that this schema is optimal for the workload. Whether a normalised or denormalised physical layout better serves filtered nearest-neighbour search is a question the experiments measure.



(a) Scene categories. (b) Object labels.

Figure 5: Label frequency across the corpus.

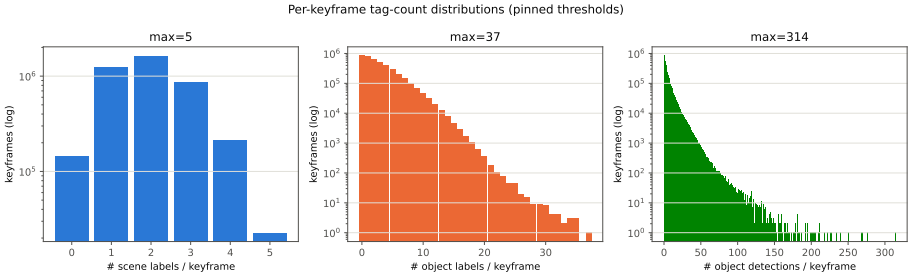


Figure 6: Labels per keyframe at the pinned thresholds.

The consequence of the normalised schema, however, is what separates FRAME from every single-table benchmark in Table 1. A filtered query is not a scan of one column stored beside the vector, but a nearest-neighbour search correlated with a predicate over separate relations. In SQL this takes the form of an `EXISTS` semi-join against the label tables combined with an `ORDER BY` on vector distance. Systems that cannot express a cross-relation predicate must denormalise the schema during ingestion before they can answer such a query at all, a divergence taken up in the discussion of the system interface.

The corpus is prepared into a backend-neutral canonical form before any database sees it. Preparation runs once per shard and is independent of any system under test. Raw video and keyframes enter, and columnar metadata files together with an identifier-keyed embedding archive emerge.

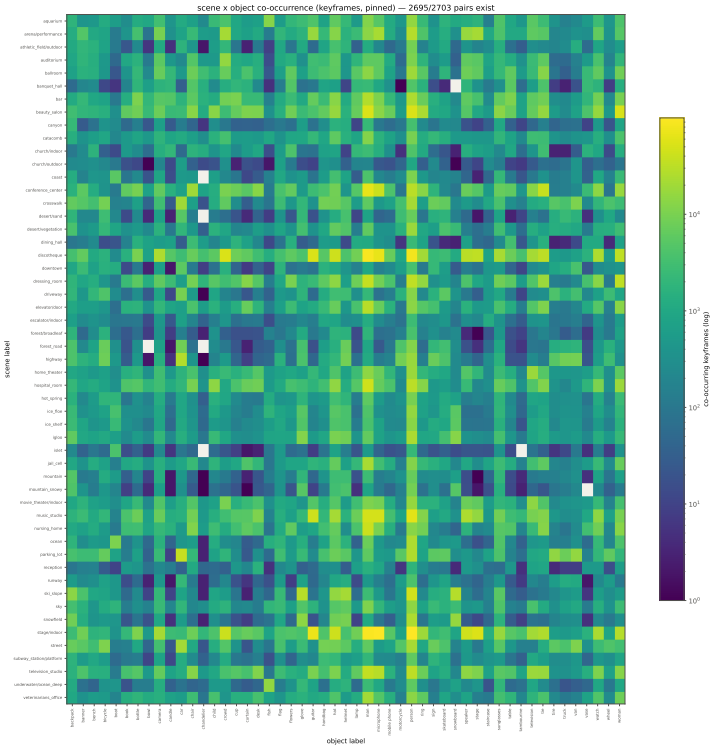


Figure 7: Scene-by-object co-occurrence over the labels the query set draws on together with the most frequent labels overall.

5.3 Query set

The origin of this workload is established in the workload characterisation section. Each item derives from the interaction logs of the Video Browser Showdown, so every phrasing is a genuine retrieval attempt by a motivated user with a known correct answer. The current query set comprises 41 items: 35 derived from the 2021 competition logs [41] and 6 from the 2026 edition.

Each item is a decomposition which separates what a user expressed semantically from what they expressed as a constraint. As illustrated in Figure 8, every item carries four components: the full user phrasing, the semantic part alone, an AND-ed structured predicate over the schema of the previous subsection, and the known item.

A predicate is a judgement about what a user's phrasing was asking for. A scene predicate must name one of the 365 Places365 categories and an object predicate one of the 143 detector terms. The mapping from user language to the fixed vocabularies is inherently lossy. Places365 is a scene taxonomy built for classification, not for expressing what a user is looking for, so query terms routinely map loosely or not at all.

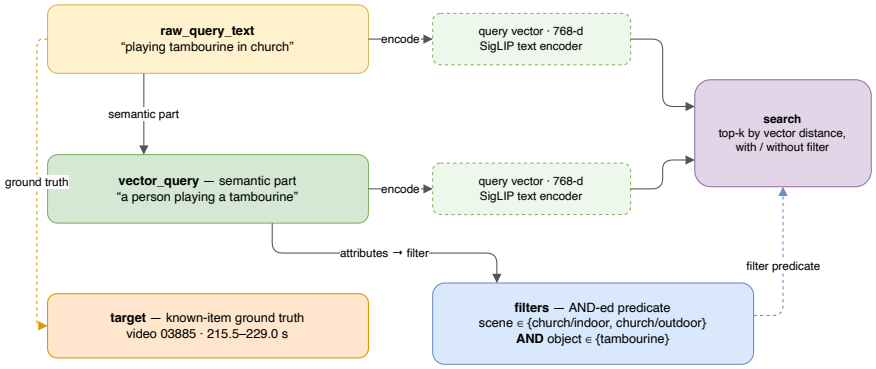


Figure 8: One item. A phrasing typed by a competitor is split into a semantic part and a structured predicate. Both the full phrasing and the semantic part alone are encoded by the same shared text model, and each query vector is searched with and without the filter.

The query set is small and this bounds what may be concluded from it. At 41 items the set is small relative to the reviewed benchmarks, which query over sets of 10^3 to 10^4. The per-item pool of logged phrasings partly offsets this on the semantic axis, but the number of distinct targets, and therefore of distinct predicates, remains small.

Predicate selectivity across the filtered items spans roughly 0.002% to 14% of the corpus (Figure 9), a range that arises from what users happened to describe rather than from a swept parameter. Three items conjoin a scene and an object attribute, and each lands one to two orders of magnitude below its tighter conjunct: $0.36\% \wedge 0.40\% \rightarrow 0.002\%$ (214×), $1.86\% \wedge 0.66\% \rightarrow 0.005\%$ (143×), and $11.06\% \wedge 1.00\% \rightarrow 0.069\%$ (14×), as shown in Figure 10. The mechanism is that a describer names attributes that co-occur rarely.

Predicates that exclude their own answer are kept deliberately. Repairing them would model a user who never makes mistakes. A per-item flag controls whether such a predicate is applied, making the effect measurable. Under exact filtering, an excluded answer is unreachable at any retrieval depth (the rank is undefined, not merely poor), while a scoring or re-ranking strategy that treats the predicate as a preference could still surface it. That contrast is the reason the flag exists. Four items are of this kind (two of the three conjunctions, one object item, and the single OCR pattern-match item). They are reported individually as harm exemplars and excluded from the filtered aggregates. A distinct failure mode is illustrated by one further item, in which the filter keeps its target but the residual semantic phrasing is too weak to localise it, so none of its logged phrasings places the target within reach even without a filter.

The query set is versioned so that an invalid comparison is refused. A single identifier covers both the query set and its ground truth. Version semantics are defined by comparability so that adding items is a minor bump that leaves prior results valid on the shared subset, while changes to a query’s meaning or

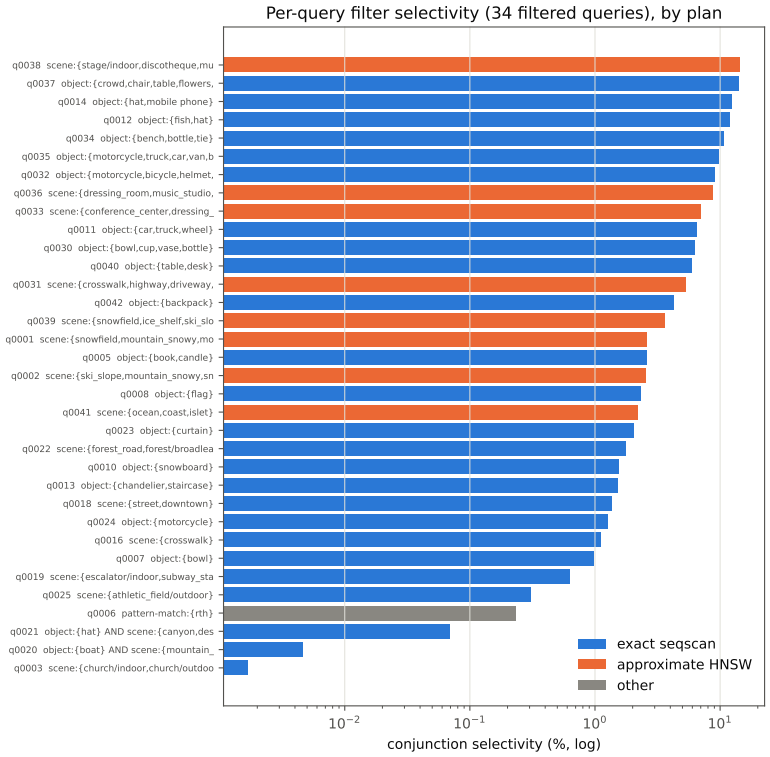


Figure 9: Predicate selectivity per filtered query

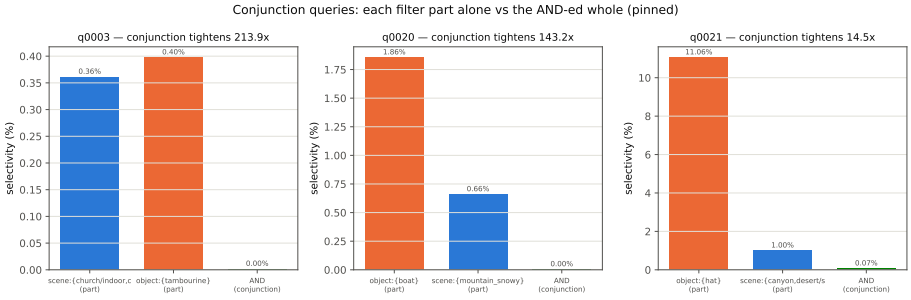


Figure 10: Each two-attribute predicate against its own conjuncts. The conjunction lands one to two orders of magnitude below the tighter of its parts.

its ground-truth parameters are a major bump voiding prior results. A separate identifier records which conditions and metric definitions a results file was produced under, so that a change to the measurement procedure correctly marks older results incomparable even when the query set itself is untouched. This scheme lets the competition's yearly cadence remain non-destructive. Extracting queries from a later year's logs and adding them to the set is a minor bump against a corpus partition that must already exist in canonical form, so the benchmark can grow without resetting its own history.

5.4 Ground truth

FRAME carries two distinct notions of a correct answer. The first is the *geometric* ground truth familiar from the reviewed benchmarks. For each query and each condition, an oracle computes the exact k nearest neighbours by exhaustive search over the subset of keyframes satisfying the predicate. The second is the *task-level* ground truth. This ground truth exists independently of the embedding model, the index structure, and the predicate. It is reported as the rank at which the known item appears in the result list, and aggregated as mean reciprocal rank over the query set.

The two notions operate at different granularities, so an explicit judging rule bridges them. Systems return keyframes; the competition designates a video and a time interval within it. A retrieved keyframe is judged correct when it belongs to that video and its timestamp falls inside the designated interval. Because the interval typically spans several shots, more than one keyframe can satisfy the rule, and the rank of the known item is taken to be the rank of the *first* such keyframe, since it determines how far down the result list they must look before seeing what they wanted. Two consequences follow from this rule. First, a query whose target interval covers many shots is easier than one covering a single shot, because more keyframes can satisfy the win condition. The number of qualifying keyframes per item is therefore reported alongside the ranks. Second, a keyframe temporally adjacent to but outside the interval is judged incorrect even when visually near-identical, which is a deliberately strict reading of the competition's own win condition.

5.5 Conditions for a fair comparison

Comparing two approaches on the same workload is only licensed if a specific list of things is identical between them.

The first invariant is the embedding model. Query encoding is a shared harness step, never delegated to an adapter. Every system and every condition receives identical query vectors from the same SigLIP encoder. The second invariant is what may be called the *passing universe*. Since confidence thresholds are pinned at ingest time rather than applied at query time, the same keyframe satisfies a given predicate in every system, and predicate selectivity is therefore a property of the benchmark rather than of the system under test. The third is the execution protocol. Retrieval depth, query order, and the set

of conditions are fixed by the shared runner, so no adapter can inadvertently run a different k, reorder queries, or skip a condition. The fourth is the index. Both systems build an HNSW graph with the same construction parameters ($m = 16$, ef_construction $= 64$), and the query-time search parameter ef_search is set identically per run, so a difference in retrieval quality reflects the systems' filtering and layout rather than a difference in index configuration.

5.6 The system interface

Three methods constitute the entire surface an adapter must implement: ingest the corpus into the system's physical layout, prepare a run, and answer one query returning ranked identifiers for a given query vector, predicate, and depth. Everything else (encoding, timing, scoring, ground-truth comparison) is inherited from the shared harness. The ingest method has no default implementation. A system integrator must specify how their store represents set-valued metadata distributed across multiple relations.

Under FRAME's contract, a relational system loads the canonical files into separate relations and answers a predicate by semi-join at query time. A single-collection vector store cannot express a predicate across relations at all, so its ingest must flatten the set-valued metadata into per-vector fields before any query can be answered. Both start from identical canonical input, so what differs downstream is the physical representation.

Adapters return ranked identifiers only, with no scores or distances. A consequence of this contract is that FRAME measures hard filtering only. A predicate can exclude a candidate from the result set, but it cannot down-weight one. Soft filtering (treating a predicate as a preference rather than a gate) is the strategy that would recover a correct answer excluded by its own predicate. Both filtered and unfiltered correct answers are already stored per item, so the data representations need not change. Only the return type would need to extend, and the change is confined to the adapter interface. Hard filtering is thus a deliberate scope boundary of the design.

System-specific facilities (collecting table statistics, inspecting query plans, setting index parameters) live behind the adapter interface. The shared harness has no notion of any of them, so adding a system that lacks a particular facility requires no change to shared code, and a system that offers one not anticipated by the harness can expose it without extending a common abstraction.

5.7 Measurement

Each filtered query is run in four configurations, defined by the crossing of two binary axes. The first axis is the predicate: whether the structured filter is applied or withheld. This axis measures what pushing an attribute out of the embedding and into a database predicate does to retrieval quality. The second axis is the phrasing: whether the text given to the encoder is the user's full phrasing, which already contains the attribute in natural language, or only the semantic part from which that attribute has been removed. This is measured

because a filter may be adding nothing the embedding did not already encode. Whether a strong text encoder makes explicit filtering redundant is a question this design can answer. Each of the four resulting configurations is scored against its own exact ground truth.

Every configuration requires its own ground truth, but only the two filtered configurations additionally require the known item to survive the predicate. An item whose predicate excludes its own answer is unscorable under the filtered configurations, reported as such and never as zero recall. The same item still has a valid unfiltered measurement. Cross-configuration aggregates therefore use the subset of items scorable in every configuration.

Geometric correctness is measured as Recall@k against the oracle's exact filtered or unfiltered neighbours, at $k \in \{5, 25, 50, 100, 1000\}$. Task success is measured as mean reciprocal rank over the rank of the known item, computed at capped depths $k \in \{1000, 100, 50, 10\}$. Beyond the cap an item counts as missed. A cap at or above the retrieval depth cannot change the value, so the deepest cap on a given run reproduces the uncapped MRR. Only caps below the retrieval depth are independent numbers, and rows are labelled accordingly. The reported operating point is $k = 50$, a depth consistent with interactive known-item search in which a user scans the top of a result list. The deepest slice, $k = 1000$, is retained only as a recall ceiling and does not represent a realistic retrieval depth. Latency is reported as median and 95th percentile across queries. Rank distributions are shown with every query visible as an individual observation, annotated with the count each distribution contains.

Mean reciprocal rank is reported in two versions. The first is taken over all scorable items. The second is restricted to the succeeding subset, defined as the items whose target appears in the unfiltered top-100. The restriction separates a filtering or system effect from the case in which the semantic content of a query is too weak to place the target within reach at all.

Because each target carries many logged phrasings, the phrasing axis is additionally measured over the full distribution of user wordings rather than a single representative phrasing per item. In this variant grading each logged phrasing is scored as its own query under the no-filter and shared-filter conditions, yielding a naturalistic sample of task success (824 phrasings under no-filter and 634 under the shared filter, across 38 items).

5.8 Workloads

In this thesis a *workload* is a predicate-type slice of the query set. The four run configurations and any parameter sweeps are applied *to* a workload.

The roster follows the predicate taxonomy defined in the query-set subsection, so that a workload name always answers which predicate type it exercises. **Scene** comprises the 13 items whose sole predicate is scene-category set-containment; their selectivities span the broad-to-moderate range where the approximate and exact execution plan cutover lives. **Object** comprises the 16 items whose sole predicate is object-presence set-containment, occupying the same selectivity regime as Scene. **Unfiltered** comprises the 7 items that carry

no predicate. Unfiltered is also a configuration within each filtered workload (the no-filter cells of the condition matrix), but a standalone unfiltered workload exists so that an external filterless query set could be run against the same harness without modification.

5.9 Experiments

Three experiments consume the workloads and configurations defined above, each isolating one question. They share the corpus, the query set, the ground truth, and the fairness invariants. They differ in which workload they draw on, which of the four configurations they run, and which parameters they vary.

Experiment A establishes the operating point and the systems comparison. It runs the Scene and Object workloads under the reduced configuration pair (semantic phrasing, with and without the filter) and the Unfiltered reference, sweeping the retrieval depth $k \in \{10, 25, 50, 100\}$ and the search parameter ef_search $\in \{10, 25, 50, 100, 200, 500, 1000\}$. It reports Recall@$k$ as a function of ef_search together with median and 95th-percentile latency, tracing the recall–latency frontier for each system and locating the smallest ef_search that reaches a filtered Recall of 0.9 at $k = 50$. This experiment fixes the operating point used by the other two and situates the two systems on the throughput-versus-recall axes that the reviewed benchmarks use.

Experiment B measures what a user gains or loses when an attribute moves from the embedding into a structured predicate. It runs the full four-configuration matrix on the Scene and Object workloads, reported separately, at retrieval depth $k = 100$ with ef_search $= 1000$. Depth 100 makes the succeeding subset visible, while mean reciprocal rank is unaffected by the deeper retrieval. From the mean reciprocal rank in each cell it derives the change across the predicate axis and the change across the phrasing axis, with Recall@k reported alongside. Because the two systems apply the same predicates to the same passing universe, the deltas are expected to agree across systems where the effect is a property of the encoder and workload rather than the database.

Experiment C measures task success across the natural distribution of user wordings. Using variant grading, it scores every logged phrasing of each target under the no-filter and shared-filter conditions at $k = 100$, and reports mean reciprocal rank in both versions (all phrasings and the succeeding subset) together with per-item rank distributions. This experiment quantifies how often a target is findable given how differently users phrase the same need, and it is where geometric correctness and task success are placed side by side, so that a high Recall@k can be read against the rank at which the known item is actually returned.

6 Results

6.1 Experiment A: The recall–latency frontier

Table 4 and Figures 11–12 report the recall–latency frontier from Experiment A, sweeping ef_search at $k = 50$.

On pgvector, ef_search controls recall directly. Sweeping it from 50 to 1000 moves filtered Recall@50 from 0.800 to 0.936, with median latency rising from 3.5 to 17 ms and 95th-percentile latency remaining at or below 322 ms (Figure 12). On Chroma, filtered Recall@50 is flat at approximately 0.963 regardless of ef_search; median latency is approximately 2.1 s and 95th-percentile latency approximately 4.7 s.

At matched recall (filtered Recall@50 ≈ 0.9, $k = 50$), the two systems sit approximately two orders of magnitude apart in latency: pgvector at a median of 17 ms (ef_search = 1000) and Chroma at 2137 ms (ef_search = 50), a factor of approximately 125. Task success at this operating point is comparable: MRR@10 is 0.173 on pgvector and 0.169 on Chroma. Recall differs slightly between systems by direction. pgvector achieves higher recall on the raw-phrasing cells, while Chroma achieves higher recall under the filter (Section 6.2.1).

System	ef_search	R@50 (filt)	MRR@10	p50	p95
pgvector	1000	0.936	0.173	17.0 ms	321.7 ms
Chroma	50	0.963	0.169	2137 ms	4740 ms

Table 4: Head-to-head at the iso-recall operating point ($k = 50$, filtered Recall@50 ≈ 0.9).

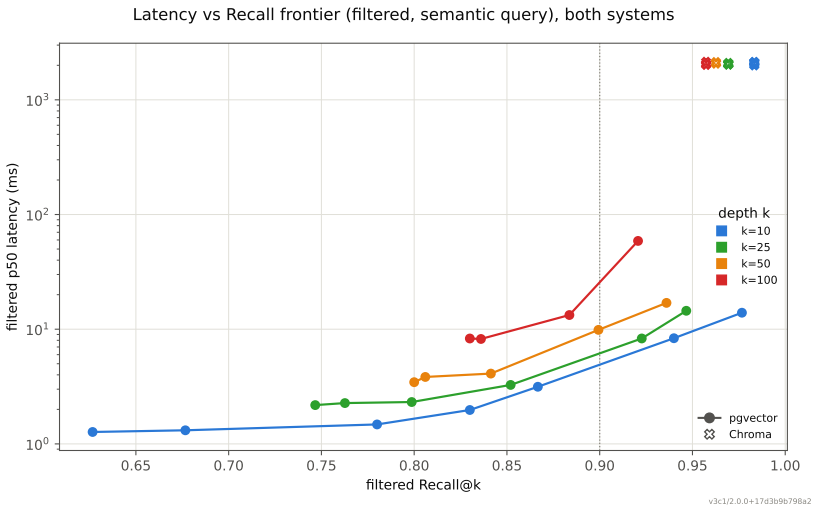


Figure 11: The recall-latency frontier for both systems over the $k \times$ ef_search sweep. Colour encodes depth k and marker encodes system.

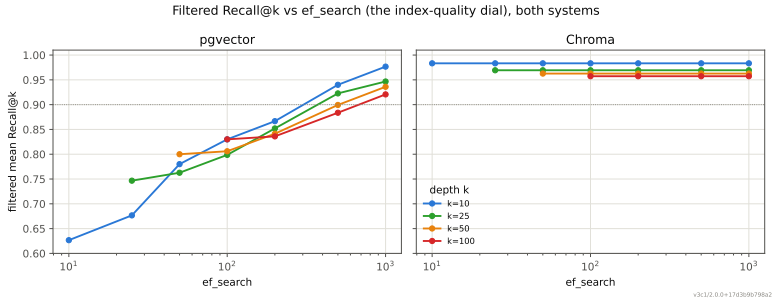


Figure 12: Filtered mean Recall@k against ef_search, one line per retrieval depth k (colour).

6.2 Experiment B: Moving an attribute into the filter

Experiment B runs the full four-configuration matrix on the Scene and Object workloads over the 30 items scorable in every cell, at $k = 100$. It is read through two lenses: geometric recall against the oracle's exact neighbours (Section 6.2.1), and the mean-reciprocal-rank deltas across the predicate and phrasing axes (Section 6.2.2).

6.2.1 Geometric recall is high across all configurations

Table 5 and Figure 13 report Recall@k against the oracle's exact neighbours for both systems across all four configurations, over the 30 items scorable in every

cell. Recall@50 ranges from 0.896 to 0.963 across all cells and both systems. Recall@100 from 0.887 to 0.957. On pgvector, recall is nearly flat across configurations: Recall@50 spans from 0.936 under semantic+filter to 0.951 under raw+nofilter, so applying the predicate shifts recall by at most approximately 0.015. On Chroma the pattern differs: the filter raises recall, with Recall@50 rising from 0.896 without the filter to 0.953 with it under the raw phrasing, and from 0.923 to 0.963 under the semantic phrasing. The predicate does not lower geometric recall on either system. The oracle ground truth is capped at 100 neighbours per item, so Recall@k is reported up to $k = 100$; retrieval beyond that depth recovers no additional oracle neighbours.

	pgvector		Chroma	
Configuration	R@50	R@100	R@50	R@100
raw + nofilter	0.951	0.939	0.896	0.887
raw + filter	0.942	0.925	0.953	0.945
semantic + nofilter	0.951	0.938	0.923	0.898
semantic + filter	0.936	0.921	0.963	0.957

Table 5: Recall@k against the oracle's exact neighbours, per configuration and system, over the 30 scorable items.

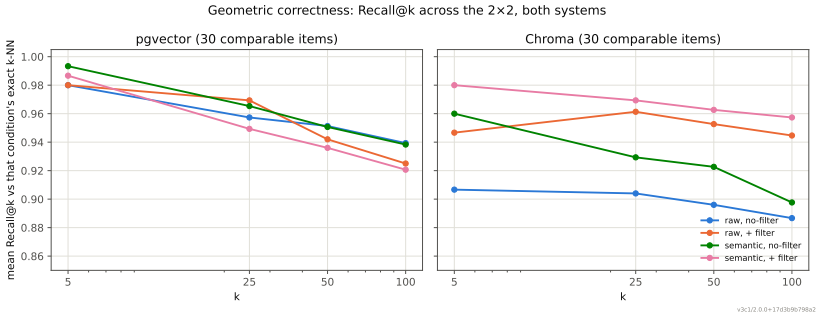


Figure 13: Recall@k for both systems across the four configurations, shown up to the oracle's ground-truth depth of 100.

6.2.2 The 2×2 delta: moving an attribute into the filter

Table 6 and Figure 14 report the four-cell configuration matrix for both systems over the 30 scorable items, at one canonical phrasing per item and $k = 100$.

MRR is highest in the raw+nofilter cell on both systems: 0.216 on pgvector and 0.215 on Chroma. Adding the predicate while retaining the full phrasing (raw+filter) raises MRR slightly, to 0.233 on pgvector ($\Delta_{\text{filter}} = +0.016$) and 0.222 on Chroma (+0.006). Under the semantic phrasing the same effect is

smaller: from 0.171 to 0.176 on pgvector (+0.006) and from 0.168 to 0.173 on Chroma (+0.005). All predicate-axis deltas fall within ±0.02.

The phrasing axis shows a larger shift. Removing the attribute text from the phrasing (the raw-to-semantic transition) lowers MRR by 0.046 on pgvector and 0.047 on Chroma under no-filter, and by 0.056 and 0.048 under the filter. At $n = 30$ the phrasing-axis change of approximately -0.05 exceeds the predicate-axis change of approximately ± 0.005 to 0.02. The pattern holds on both systems.

Scene ($n = 13$) and Object ($n = 16$) items are reported separately. Predicate selectivity across the filtered items spans approximately 0.002% to 15% of the corpus (Figure 15), with the selective-versus-relaxed subdivision at the 3% threshold.

Configuration (MRR, $n = 30$, $k = 100$)	pgvector	Chroma
raw + nofilter	0.216	0.215
raw + filter	0.233	0.222
semantic + nofilter	0.171	0.168
semantic + filter	0.176	0.173
Δ_{filter} (raw row)	+0.016	+0.006
Δ_{filter} (semantic row)	+0.006	+0.005
Δ_{semantic} (nofilter col)	−0.046	−0.047
Δ_{semantic} (filter col)	−0.056	−0.048

Table 6: The 2×2 configuration matrix (MRR over the 30 scorable items, one canonical phrasing per item) and its two deltas.

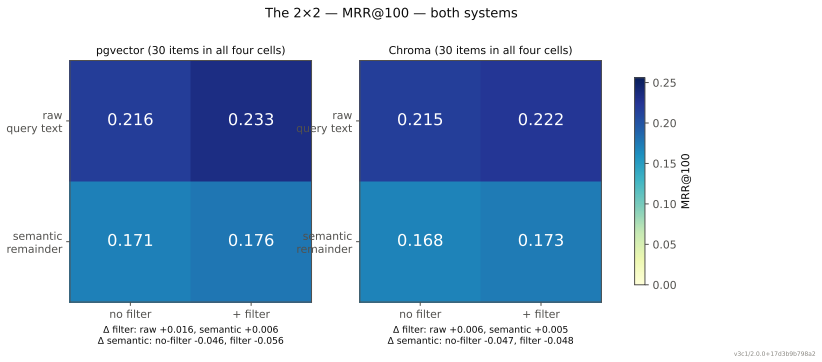


Figure 14: The 2×2 delta for both systems: MRR by predicate axis and phrasing axis.

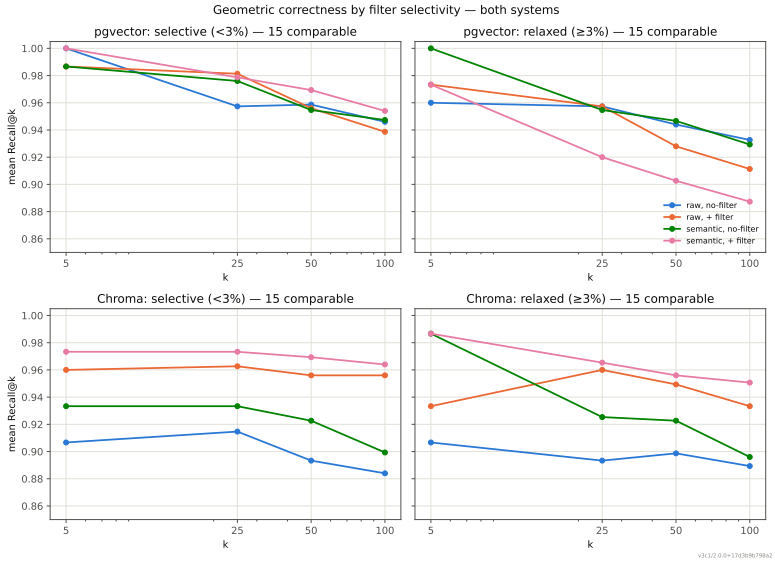


Figure 15: Recall@k split by predicate selectivity (selective <3% vs. relaxed ≥3%), which spans roughly 0.002% to 15% of the corpus across the filtered items.

6.3 Experiment C: Task success over natural phrasings

Over the same items for which Section 6.2.1 reported geometric recall of approximately 0.9, Experiment C measures task success across the natural distribution of logged phrasings at $k = 100$. Mean reciprocal rank is low (Figure 16).

Under no-filter, the pooled MRR over all 824 phrasings across 38 items is 0.071 on pgvector and 0.068 on Chroma. Under the shared filter, where the phrasing count drops to 634, pooled MRR is 0.078 on both systems. Restricted to the succeeding subset, MRR rises to 0.247 (pgvector, $n = 238$) and 0.241 (Chroma, $n = 232$) under no-filter, and to 0.284 ($n = 170$) and 0.261 ($n = 166$) under the filter. The succeeding fractions are 238 of 824 phrasings on pgvector and 232 of 824 on Chroma: the majority of phrasings do not place the target within the top-100.

The per-item outcomes (Figure 17) are highly uneven. A small number of items are found by most of their phrasings, while the majority sit near zero on both success fraction and MRR. One item whose success fraction sits well to the right of its MRR, q0028, is found by roughly three-quarters of its phrasings yet carries an MRR near 0.11. The target is retrieved but ranks deep. Item q0039 illustrates the floor: none of its 49 phrasings places the target in the unfiltered top-100, contributing an MRR of approximately zero.

Placed against Section 6.2.1, geometric recall of approximately 0.9 is observed alongside a pooled MRR of approximately 0.07 over the same items.

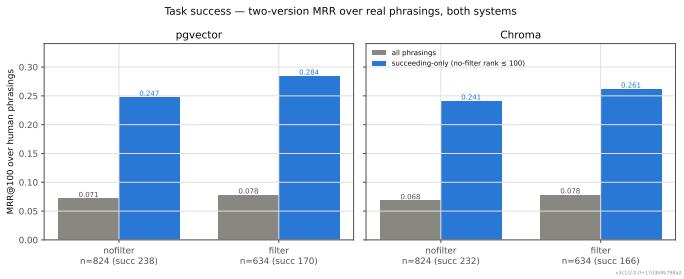


Figure 16: Two-version MRR over human phrasings for both systems: all phrasings versus the succeeding-only subset, under no-filter and the shared filter.

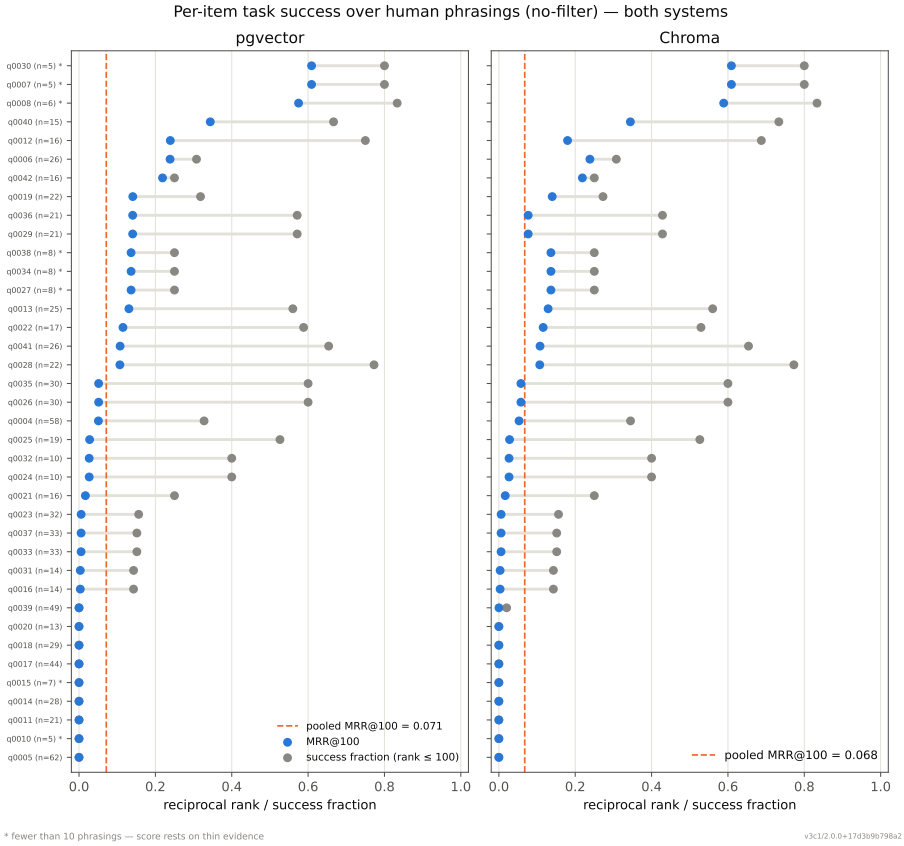


Figure 17: Per-item task success over each item's phrasings, both systems.

7 Discussion

7.1 Geometric correctness does not imply task success

Both systems achieve geometric recall of approximately 0.9 at the reported operating point (Section 6.2.1), indicating that the HNSW index faithfully returns the neighbourhood the brute-force oracle identifies. Yet over the same items, pooled mean reciprocal rank is approximately 0.07 (Section 6.3). The known target usually does not appear among the geometric nearest neighbours. A recall-only evaluation would rate these systems as excellent while the task-success metric contradicts that reading. FRAME makes it directly measurable on a filtered known-item workload with exact target ground truth. This observation addresses objective 3 of the problem statement: geometric and task-oriented evaluation can diverge substantially, and reporting both is necessary to characterise a system's utility for the task it is meant to serve.

The 2×2 configuration matrix (Section 6.2.2, Table 6) allows the decomposition cost to be read directly. On the predicate axis, Δ_{filter} falls within ± 0.02 on both systems and as such, applying the structured filter is approximately task-neutral. The mechanism is that the false positives which outrank the target tend to share the filtered attribute, so the predicate removes distractors the similarity ranking already placed below the target while leaving the competing items in place. The candidate set shrinks, which can help geometric recall, but the items that actually compete with the known target are not separated. On the phrasing axis, $\Delta_{\text{semantic}} \approx -0.05$: removing the attribute text from the query and replacing it with a hard predicate lowers task success. The encoder uses the natural-language attribute phrase as soft conditioning on the query vector. A set-membership test is a lossy substitute because it cannot express the graded, contextual similarity signal the phrase carried. The consequence is that the raw phrasing with no filter is the highest-scoring configuration on both systems, so the common recipe of decomposing a natural-language query into a semantic part plus a structured filter is, on this workload, a net loss for task success.

The two systems under test differ in storage layout (normalised versus denormalised), query-execution strategy (semi-join versus predicate materialisation), and implementation, yet they return near-identical task success. MRR is comparable at the matched-recall operating point (Section 6.1), while geometric recall differs slightly between them. The quantity a user ultimately cares about, i.e. whether the target item appears at a usable rank, is therefore determined upstream of the database, by the embedding model and the query phrasing.

This gap between geometric recall and task success has a concrete implication for systems that consume retrieval results automatically. A retrieval-augmented generation pipeline typically forwards only a small top-k of results to the downstream model, commonly on the order of five to ten. At the observed MRR the known target usually falls outside such a cutoff, so the correct item would never enter the downstream context. A human-in-the-loop interface, by contrast, tolerates a poor rank. A person scanning the top 25 or 50 results can still

locate and select the target. The same retrieval quality is therefore adequate for interactive known-item search and inadequate for automation. The per-item MRR distributions (Figure 17) reinforce this reading. A small number of items are reliably found by most phrasings, while the majority sit near an MRR of zero, so the pooled average understates the failure rate on the harder items.

7.2 What FRAME contributes

The two-lens reading of the preceding subsection is available because FRAME operates in a bounded known-item domain. Mean reciprocal rank over a designated target is computable only when each task has exactly one correct answer defined independently of the retrieval system. A benchmark whose queries are held-out vectors or synthetic text strings has no designated target item and can therefore report geometric correctness alone. The task-success metric is an affordance of the design choice to work with competition-defined known items.

The results also reframe where benchmarking effort is most productively spent. Index-focused benchmarks study how graph or quantisation choices trade geometric recall against query speed. On the present workload that axis is nearly flat for task success. The two engines return near-identical MRR despite differing in schema and execution (Section 7.1), indicating that the embedding model and the query phrasing, not the index, govern whether the user succeeds. A practitioner tuning the index is therefore optimising the axis that matters least for actually finding the target item. This does not deny that the index governs latency. Section 7.4 discusses the substantial latency divergence between the two systems, but it separates the two axes and identifies which one owns task success on this class of workload.

7.3 Workload realism and the limits of the query set

Real known-item search queries produce a filter regime that the synthetic, selectivity-swept benchmarks do not reach. Individual attributes are already highly selective. A single scene predicate such as “church” selects well under one per cent of the corpus and multi-attribute conjunctions fall one to two orders of magnitude below their most selective conjunct. This regime sits outside the envelope that a single synthetic attribute swept by selectivity can produce [6, 31]. That the gap is structural, not incidental, is corroborated by recent evidence that attribute distributions in real corpora are largely independent of the embedding geometry [30].

The conclusions the query set supports are bounded by the proof-of-concept scope of this thesis. The number of distinct targets in the query set, and therefore of distinct predicates, is small relative to the reviewed benchmarks. The query set is therefore presented as version one of an extensible benchmark instrument, not as a settled measurement, and the conclusions drawn from it should be read with that status in mind.

7.4 Systems and architecture

The two systems share the same HNSW index algorithm and operate on the same corpus, so their divergence in latency (Table 4) is attributable to schema and execution strategy rather than to the approximate nearest-neighbour search itself. pgvector stores the filterable metadata in normalised relations and answers a predicate as a semi-join. Chroma denormalises the set-valued attributes into per-vector document fields because it has no join operator. The latency profiles that follow from these choices differ in character. The denormalised layout pays a uniform predicate-materialisation cost on every filtered query, whereas the normalised layout is inexpensive on the typical query but exposes the cost-based query planner to a plan-choice boundary.

That boundary is the mechanism behind pgvector's latency variability. For predicates that match a broad fraction of the metadata (common object labels, for instance) the planner can estimate the post-filter candidate set as large enough that an exact sequential scan costs less than the HNSW traversal, and choose it. At union scale this exact path is expensive, and the choice turns on a cardinality estimate (the planner's `n_distinct` statistic) rather than on the true per-query selectivity. Filtered-ANN latency on a relational engine is therefore partly a property of the query planner, an effect a benchmark can expose by recording the plan chosen per query. Chroma has no planner and no join; its cost is the predicate materialisation itself, which is independent of the vector search.

7.5 Future work

FRAME is designed to grow. The query set can expand as new competition logs become available, versioned so that added items leave prior results valid on the shared subset. The corpus and query set can extend to further V3C partitions with multi-shard targets. The unit of retrieval can move from keyframes to video-level embeddings, and further filterable metadata can be folded in as additional predicate dimensions. Task scope can likewise widen beyond known-item search to ad-hoc retrieval (find as many relevant items as possible) and question-answering tasks, which carry different selectivity and ranking profiles and would exercise the metric in ways the current KIS-only set does not.

The one deliberately fixed design choice is the embedding model. Results are comparable across extensions only if every query item is encoded by the same model, so anyone adding items must reuse the pinned encoder. Because the results show that the encoder and query phrasing govern task success (Section 7.1), making the encoder itself a swappable variable is a first-class future experiment. Re-running the full suite under a newer vision language model would test directly whether the recall-to-MRR gap narrows, and if so, how much of the improvement is attributable to the embedding space and how much to the query formulation.

The present benchmark measures single-shot retrieval quality: one query, one ranked list, one rank of the known target. A real user, however, navigates

results and reformulates. An item at a poor rank can still be reached through the interface, and the "best" query strategy is not fixed. As users gain experience with a system's vocabulary and failure modes, how they phrase, filter, and refine changes, so the optimal strategy is a moving target that a one-shot benchmark snapshot cannot capture. An interaction-aware extension of FRAME could address this by grouping phrasings by user experience level or by position within a search session and reporting task success as a function of that progression. This would connect the geometric measurement to how people actually search, bridging the gap between the retrieval metric and interactive task performance.

The failure modes the results expose point to concrete next experiments in filtering and indexing. Hard predicates that exclude the target motivate soft or ranking filters that treat a predicate as a preference rather than a gate, scoring items by how well they satisfy the predicate rather than discarding those that do not. The query set already retains the excluded-target items for exactly this study. Alternative index structures present a separate direction: randomised or compressed indexes trade geometric recall for different cost profiles, and whether a different ranking helps or hinders specific tasks is untested. The results suggest that recall and task success can move independently (Section 7.1), so an index that lowers geometric recall need not lower MRR. Testing this across systems, holding the encoder fixed, is the natural continuation of the present work.

8 Conclusion

This thesis asked whether the workloads exercised by existing filtered-ANN benchmarks reflect those of real multimedia retrieval, and what happens when the patterns observed in a real workload are evaluated on production vector database systems. The investigation was carried out by characterising query logs from the Video Browser Showdown, building a benchmark suite grounded in those queries, and running it on pgvector and ChromaDB over the V3C collection. The three research questions receive the following answers.

On the first question, the existing FANNS benchmarks and the real VBS workload match on one axis and diverge on the rest. On corpus geometry, V3C is indistinguishable from the canonical benchmark corpora on the properties the ANN literature uses to predict search difficulty. The benchmark queries are held out from the corpus while the real queries are text that users typed at a live system. The benchmark filters are single-attribute predicates, typically over synthetic labels in a flat single-table schema while the VBS queries express deep heterogeneous conjunctions across multiple attribute types.

On the second question, both systems handle the workload but through different architectural paths. pgvector stores the filterable metadata in normalised relations and answers predicates as semi-joins, supporting the multi-relation schema directly but exposing a cost-based query planner that can select an expensive execution plan on broad predicates. ChromaDB cannot express a cross-relation predicate and requires the set-valued metadata to be denormalised into per-vector fields at ingest time. At matched geometric recall (filtered Recall@50

of approximately 0.9 on both systems), pgvector returns results at a median latency of 17 ms against ChromaDB's 2137 ms, a difference of approximately two orders of magnitude attributable to schema and execution strategy rather than to the underlying HNSW index. Task success at this operating point is comparable between the two systems.

On the third question, geometric correctness and task success diverge on this workload. Both systems achieve geometric Recall@k of approximately 0.9, indicating that the HNSW index faithfully returns the neighbourhood the brute-force oracle identifies. Over the same items, mean reciprocal rank over the known target is approximately 0.07. The target usually does not appear among the geometric nearest neighbours. The standard Recall@k metric, used by every reviewed benchmark, rates these systems as performing well while the task-level metric does not. The results suggest that the embedding model and query phrasing, rather than the index configuration or the filtering strategy, determine whether the target is found.

The methodological contribution that enables these findings is FRAME, a filtered-ANN benchmark for known-item search. Its design properties (real user queries decomposed into a semantic part and a structured predicate, a normalised multi-relation schema, and dual ground truth reporting geometric recall alongside known-item rank) are what allow the workload-representation gap and the recall-versus-task-success gap to be measured on the same items. The benchmark is extensible by design. Query-set additions from future competition logs are versioned to leave prior results valid, and new systems are integrated by implementing a three-method adapter against the shared harness.

The conclusions are bounded by the proof-of-concept scope of this work. The query set contains 41 items and only two vector database systems are evaluated. The natural extensions are a larger query set drawing on further competition years, additional systems, soft or ranking filters that treat a predicate as a preference rather than a gate, and swapping the embedding model to test directly whether the gap between geometric recall and task success narrows under a stronger encoder.

References

[1] J. Shi, Y. Cai, and W. Zheng, "Filtered Approximate Nearest Neighbor Search: A Unified Benchmark and Systematic Experimental Study [Experiment, Analysis & Benchmark]," Sept. 2025.

[2] L. Ma, R. Zhang, Y. Han, S. Yu, Z. Wang, Z. Ning, J. Zhang, P. Xu, P. Li, W. Ju, C. Chen, D. Wang, K. Liu, P. Wang, P. Wang, Y. Fu, C. Liu, Y. Zhou, and C.-T. Lu, "A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge," 2023.

[3] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing - STOC '98*, (Dallas, Texas, United States), pp. 604–613, ACM Press, 1998.

[4] M. Aumüller, E. Bernhardsson, and A. Faithfull, "ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms," *Information Systems*, vol. 87, p. 101374, Jan. 2020.

[5] H. V. Simhadri, M. Aumüller, A. Ingber, M. Douze, G. Williams, M. D. Manohar, D. Baranchuk, E. Liberty, F. Liu, B. Landrum, M. Karjikar, L. Dhulipala, M. Chen, Y. Chen, R. Ma, K. Zhang, Y. Cai, J. Shi, Y. Chen, W. Zheng, Z. Wan, J. Yin, and B. Huang, "Results of the Big ANN: NeurIPS'23 competition," 2024.

[6] P. Iff, P. Bruegger, M. Chrapek, M. Besta, and T. Hoefler, "Benchmarking Filtered Approximate Nearest Neighbor Search Algorithms on Transformer-based Embedding Vectors," 2025.

[7] M. Li, X. Yan, B. Lu, Y. Zhang, J. Cheng, and C. Ma, "Attribute Filtering in Approximate Nearest Neighbor Search: An In-depth Experimental Study," 2025.

[8] L. Rossetto, H. Schuldt, G. Awad, and A. A. Butt, "V3C – A Research Video Collection," in *MultiMedia Modeling* (I. Kompatsiaris, B. Huet, V. Mezaris, C. Gurrin, W.-H. Cheng, and S. Vrochidis, eds.), (Cham), pp. 349–360, Springer International Publishing, 2019.

[9] Sivic and Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proceedings Ninth IEEE International Conference on Computer Vision*, (Nice, France), pp. 1470–1477 vol.2, IEEE, 2003.

[10] J. Pennington, R. Socher, and C. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (A. Moschitti, B. Pang, and W. Daelemans, eds.), (Doha, Qatar), pp. 1532–1543, Association for Computational Linguistics, Oct. 2014.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," 2017.

[12] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), pp. 3980–3990, Association for Computational Linguistics, 2019.

[13] M. Ostendorff, T. Blume, T. Ruas, B. Gipp, and G. Rehm, "Specialized document embeddings for aspect-based similarity of research papers," in *Proceedings of the 22nd ACM/IEEE Joint Conference on Digital Libraries*, JCDL '22, (New York, NY, USA), pp. 1–12, Association for Computing Machinery, June 2022.

[14] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning Transferable Visual Models From Natural Language Supervision," 2021.

[15] C. Schuhmann, R. Vencu, R. Beaumont, R. Kaczmarczyk, C. Mullis, A. Katta, T. Coombes, J. Jitsev, and A. Komatsuzaki, "LAION-400M: Open Dataset of CLIP-Filtered 400 Million Image-Text Pairs," Nov. 2021.

[16] Z. Lu, "A Theory of Multimodal Learning." https://arxiv.org/abs/2309.12458v2, Sept. 2023.

[17] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When Is "Nearest Neighbor" Meaningful?," in *Database Theory — ICDT'99* (G. Goos, J. Hartmanis, J. Van Leeuwen, C. Beeri, and P. Buneman, eds.), vol. 1540, pp. 217–235, Berlin, Heidelberg: Springer Berlin Heidelberg, 1999.

[18] Yu. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs," 2016.

[19] M. Wang, X. Xu, Q. Yue, and Y. Wang, "A Comprehensive Survey and Experimental Comparison of Graph-Based Approximate Nearest Neighbor Search," 2021.

[20] H. Jégou, M. Douze, and C. Schmid, "Product Quantization for Nearest Neighbor Search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 117–128, Jan. 2011.

[21] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with GPUs," 2017.

[22] M. Douze, H. Jégou, and F. Perronnin, "Polysemous Codes," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), vol. 9906, pp. 785–801, Cham: Springer International Publishing, 2016.

[23] W. Li, Y. Zhang, Y. Sun, W. Wang, M. Li, W. Zhang, and X. Lin, "Approximate Nearest Neighbor Search on High Dimensional Data — Experiments, Analyses, and Improvement," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, pp. 1475–1488, Aug. 2020.

[24] E. Jääsaari, "VIBE: Vector Index Benchmark for Embeddings,"

[25] S. Gollapudi, N. Karia, V. Sivashankar, R. Krishnaswamy, N. Begwani, S. Raz, Y. Lin, Y. Zhang, N. Mahapatro, P. Srinivasan, A. Singh, and H. V. Simhadri, "Filtered-DiskANN: Graph Algorithms for Approximate Nearest Neighbor Search with Filters," in *Proceedings of the ACM Web Conference 2023*, (Austin TX USA), pp. 3406–3416, ACM, Apr. 2023.

[26] Y. Lin, K. Zhang, Z. He, Y. Jing, and X. S. Wang, "Survey of Filtered Approximate Nearest Neighbor Search over the Vector-Scalar Hybrid Data," May 2025.

[27] L. Patel, P. Kraft, C. Guestrin, and M. Zaharia, "ACORN: Performant and Predicate-Agnostic Search Over Vector Embeddings and Structured Data," 2024.

[28] C. Zuo, M. Qiao, W. Zhou, F. Li, and D. Deng, "SeRF: Segment Graph for Range-Filtering Approximate Nearest Neighbor Search," *Proceedings of the ACM on Management of Data*, vol. 2, pp. 1–26, Mar. 2024.

[29] A. Liang, P. Zhang, B. Yao, Z. Chen, Y. Song, and G. Cheng, "UNIFY: Unified Index for Range Filtered Approximate Nearest Neighbors Search," 2024.

[30] D. Lu, H. Caminal, M. Chatzakis, Y. Papakonstantinou, Y. Chronis, V. Jain, and F. Özcan, "An In-Depth Study of Filter-Agnostic Vector Search on a PostgreSQL Database System: [Experiments and Analysis]," *Proceedings of the ACM on Management of Data*, vol. 4, pp. 1–26, May 2026.

[31] G. Kang, Z. Ge, J. Hu, X. Zhang, L. Wang, and J. Zhan, "BigVectorBench: Heterogeneous Data Embedding and Compound Queries are Essential in Evaluating Vector Databases," *Proceedings of the VLDB Endowment*, vol. 18, pp. 1536–1550, Jan. 2025.

[32] A. Amanbayev, B. Tsan, T. Dang, and F. Rusu, "Filtered Approximate Nearest Neighbor Search in Vector Databases: System Design and Performance Analysis," Feb. 2026.

[33] M. Chen, K. Zhang, Z. He, Y. Jing, and X. S. Wang, "RoarGraph: A Projected Bipartite Graph for Efficient Cross-Modal Approximate Nearest Neighbor Search," *Proceedings of the VLDB Endowment*, vol. 17, pp. 2735–2749, July 2024.

[34] Q. Yue, M. Wang, X. Xu, C. Long, Y. Wang, and J. Wang, "Efficient and Robust Out-Of-Distribution Vector Similarity Search with Cross-Distribution Monotonic Graph," *Proceedings of the ACM on Management of Data*, vol. 4, pp. 29:1–29:29, Apr. 2026.

[35] M. Aumüller and M. Ceccarello, "The Role of Local Intrinsic Dimensionality in Benchmarking Nearest Neighbor Search," July 2019.

[36] Z. Wang, Q. Zhang, B. Lu, Q. Chen, and C. Tan, "Towards Robustness: A Critique of Current Vector Database Assessments," Apr. 2026.

[37] A. Salemi and H. Zamani, "Evaluating Retrieval Quality in Retrieval-Augmented Generation," Apr. 2024.

[38] Transaction Processing Performance Council, "TPC Benchmark H (TPC-H): Standard Specification," tech. rep., 2022.

[39] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 Million Image Database for Scene Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 1452–1464, June 2018.

[40] M. Minderer, A. Gritsenko, and N. Houlsby, "Scaling Open-Vocabulary Object Detection," May 2024.

[41] S. Heller, V. Gsteiger, W. Bailer, C. Gurrin, B. . Jónsson, J. Lokoč, A. Leibetseder, F. Mejzlík, L. Peška, L. Rossetto, K. Schall, K. Schoeffmann, H. Schuldt, F. Spiess, L.-D. Tran, L. Vadicamo, P. Veselý, S. Vrochidis, and J. Wu, "Interactive video retrieval evaluation at a distance: Comparing sixteen interactive video search systems in a remote setting at the 10th Video Browser Showdown," *International Journal of Multimedia Information Retrieval*, vol. 11, no. 1, pp. 1–18, 2022.