Efficient Recall-Guaranteed Spectral Similarity Search on Massive Datasets: 2 Billion Stars from Gaia

Final Report — A Four-Bound Cell Stage for Recall-Guaranteed Threshold Queries

Yasir Alharbi, **Sultan Aloqayli**, **Naif Alharbi**,
Faris Aljohani,

Mentor: Reem Alghamdi

yasir.alharbi.1@kaust.edu.sa, sultan.aloqayli@kaust.edu.sa, naif.alharbi.2@kaust.edu.sa,
faris.aljohani@kaust.edu.sa, reem.alghamdi@kaust.edu.sa

Abstract

Vector search has become a core operation in modern retrieval systems, but many scientific and interactive workloads do not fit the top-k model that most systems are built around. Given a query object, a similarity radius, and optional filters over structured attributes, these workloads require every matching record to be returned, a threshold query, whose answer size varies from one query to the next. In this setting recall cannot be traded away: extra candidates can be discarded by an exact check, but a silently missing result can invalidate the analysis that depends on it. This project develops a similarity search system that guarantees recall by construction and makes that guarantee affordable at massive scale. Compressed vector indexes and provable geometric bounds discard only those records that cannot possibly satisfy the query, so no valid result is ever lost, and the research problem is efficiency: a guarantee is useful only if the set of surviving candidates is small enough to verify quickly. On spectral measurements from the Gaia catalogue the system cuts the fraction of the database that is scanned from 88% to 40%, and the vectors reaching exact-distance verification from 179 per true answer to 1.11, while running roughly twice as fast as the recall-guaranteed baseline and holding recall at exactly 1.0 on every query. Four composable cell bounds, a covering sphere, a bounding box, a Voronoi bisector test and per-cell sub-group boxes, achieve this at zero storage cost per vector, leaving memory spent on the quantizer as the one remaining trade-off, which the work characterises explicitly.

Date: August 16, 2026

Contents

1 Introduction **2**

2 Related Work **2**

3 The Data **3**

3.1 What a vector is here . . . 3

3.2 The shape of the space . . . 4

3.3 Two similarity measures, two questions . . . 4

4 Method **5**

4.1 The baseline, and what was wrong with it . . . 5

4.2 The things that did not work . . . 5

4.3 The four-bound cell stage . . . 6

4.4 Product quantization against scalar quantization . . . 7

4.5 Protocol . . . 7

4.6 A note on the upper bound . . . 8

5 Results **8**

5.1 Which bound does the work . . . 8

5.2 The funnel, end to end . . . 9

5.3 What the guarantee costs, and how the bits buy it back . . . 10

5.4 Memory, which is the one cost that did not go down . . . 10

6 Discussion **12**

6.1 Is the system efficient . . . 12

6.2 Why the bounds worked when the partition did not . . . 12

6.3 The two metrics behave differently, and that is geometry, not physics . . . 13

6.4 The trade-off, and how much of it is under control . . . 13

7 Limitations and Future Work **14**

8 Conclusion **14**

References **15**

1 Introduction

Most vector search systems are built to answer a top-k question: given a query, return the k closest records. A large class of scientific workloads asks something different. An astronomer looking for stars whose spectra resemble a given one does not want the ten most similar; they want *every* star within a similarity radius τ, because the number of genuine matches is itself the finding. That is a threshold query, and its answer size varies from one query to the next, from a handful of objects to hundreds of thousands.

Threshold queries change what an index is allowed to do. In a top-k system a missed neighbour is a small quality loss that a recall metric absorbs. In a threshold query a missed neighbour is a wrong answer, and it is wrong silently, because nothing in the output signals that something is absent. Extra candidates are cheap by comparison, since an exact distance check discards them. So the design target for this project has been recall of exactly 1.0, achieved by construction rather than by tuning, with the entire research effort spent on making that guarantee affordable.

The mechanism is a three-stage funnel. An inverted-file index partitions the database into cells; a *cell bound* decides which cells could possibly contain an answer; a compressed representation of each vector in the opened cells produces an approximate distance $a(x)$ together with a stored per-vector error $\varepsilon(x)$, and any vector with $a(x) - \varepsilon(x) > \tau$ is provably outside the query ball and is discarded; whatever survives is checked exactly. Every test in that chain is a sound lower bound on the true distance, so nothing that belongs in the answer can ever be dropped. What varies is how much work reaches each stage, and that is what this project has spent four weeks reducing.

The starting point was a recall-guaranteed IVF-PQ baseline that worked but was extremely wasteful: at one million stars it sent 88% of the database into the scan stage and 179 vectors into exact verification for every one that was a genuine answer. Reducing those two numbers turned out to require testing, and mostly discarding, a long list of plausible ideas. Better clustering algorithms did not help. More cells did not help enough to justify their build cost. Optimized product quantization recovered almost nothing. A hierarchical coarse index made the search slower. What did work was replacing the cell test itself, and this report is mainly about that: an inverted file whose cells are certified by four independent geometric bounds, paired with a scalar quantizer whose bit depth is an explicit dial between memory and work.

The rest of the report is deliberately narrow. Section 2 places the work in context. Section 3 describes the Gaia spectra and the two similarity measures used on them. Section 4 explains how the search reached its final form, including the ideas that failed, and then describes the four bounds and the quantizer sweep in full. Section 5 reports what the final system does at both scales and on both metrics. The remaining sections discuss the one cost that was not eliminated, which is memory, and what can be done about it.

2 Related Work

Compressed inverted-file indexes.. The baseline this project builds on is IVF-PQ, the combination of an inverted file over k-means cells with product quantization [1], which splits a vector into sub-blocks and replaces each with a codebook index. Its usual purpose is approximate top-k search, and the standard implementation, including the precomputed per-cell distance tables used here, follows FAISS [2]. Optimized product quantization [3] improves the codebook by learning a rotation first. Scalar quantization takes the opposite route, quantizing each dimension independently and spending more bytes to obtain a much smaller reconstruction error, and the compressed-index line of work [4] shows that at scale the extra bytes can pay for themselves in reduced downstream work. The present project uses PQ as its reference point and scalar

quantization as its final choice, but the decisive component is neither of those, it is the cell test.

Geometric bounds for exact similarity search.. The idea of certifying that a region cannot contain an answer predates vector databases. Branch-and-bound nearest-neighbour search [5] and metric trees such as the M-tree [6] prune with a covering radius, which is exactly the sphere test the mentor's baseline uses. Spatial indexes prune with the minimum distance to a bounding rectangle [7, 8]. Generalised hyperplane partitioning [9] and GNAT [10] prune with the perpendicular bisector between two reference points, which is precisely the certificate an IVF partition already carries, because a k-means partition *is* a Voronoi diagram of its centroids [11]. This report's contribution is not a new bound but the observation that these classical tests compose, that their intersection is still sound, and that on anisotropic data they are worth far more together than the covering radius is alone.

Partitioning.. The cells themselves come from mini-batch k-means [12] with k-means++ seeding [13]. Alternatives were tested and rejected, including bisecting k-means [14], BIRCH [15], a Gaussian mixture, a capacity-constrained balanced assignment, and density methods, DBSCAN [16] and HDBSCAN [17], which were also used to ask whether the data has any intrinsic cluster structure at all.

Filtered vector search.. The wider setting for this work is filtered approximate nearest-neighbour search, where a similarity query is combined with predicates over structured attributes [18–21]. The index built here carries an attribute-filter hook through both query paths, and Section 7 is explicit that it was never exercised, so this report covers the unfiltered half of that problem.

The data.. The vectors are BP/RP spectral coefficients from Gaia Data Release 3 [22]. Visualisation and multi-resolution access to the same catalogue is the subject of related work by one of the project's mentors [23].

3 The Data

3.1 What a vector is here

Gaia observes each star with two low-resolution prism spectrographs, a blue photometer (BP) and a red photometer (RP), and the pipeline stores each spectrum not as a sampled curve but as 55 coefficients of a basis expansion, one set per photometer. A record therefore carries two blocks of 55 numbers that describe different halves of the same object, the blue end and the red end of its spectral energy distribution, and concatenating them gives the $d = 110$ dimensional vector this project searches. Records also carry scalar attributes such as position, parallax, magnitudes and effective temperature. Those are the predicates that a filtered query would restrict on, and they are deliberately *not* part of the search vector: the similarity being measured is spectral, and mixing physical quantities with different units into the same distance would make that distance meaningless.

Gaia DR3 published these spectra for roughly 220 million sources, and a future release is expected to extend them toward the full catalogue of about two billion. Working at either of those sizes was not possible inside a summer school, so every experiment in this report runs on two subsets, 10,000 stars and 1,000,000 stars, drawn from the same file with the same pipeline. The two scales are not redundant. Several effects in this project reverse sign between them, and a method that is only ever measured at one scale can easily be a measurement of the harness rather than of the method.

3.2 The shape of the space

Figure 1 shows why the geometry of this data matters for an index. Both panels are drawn on a random subsample of 20,000 stars rather than the whole file, so they describe the distribution rather than enumerate it, but the effect is unmistakable and it appears at both scales. In the standardised space the index actually uses, a two-dimensional histogram of the first two principal components puts 99.0% of the 10K subsample and 97.9% of the 1M subsample into a single bin. Clipping the axes to the central 99% of the range recovers the structure, so part of that collapse is a viewing-range artefact rather than a property of the data. What is not an artefact is the third panel: the raw vector norms span 5.4 decades at 10K and 6.5 decades at 1M. A handful of extremely bright stars are hundreds of thousands of times longer than a typical one, all 110 coefficients are strongly skewed, and the first principal component alone carries most of the variance.

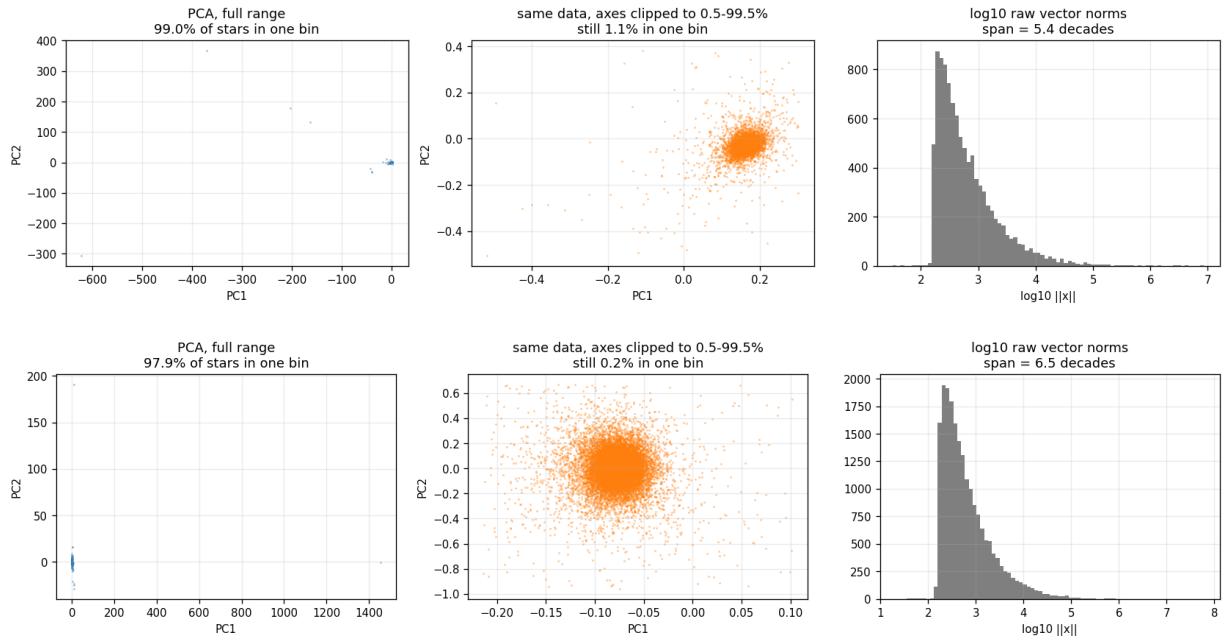


Figure 1 The standardised search space, 10K above and 1M below, on a random 20,000 star subsample. Left, the first two principal components over their full range, where almost the whole subsample falls into one bin of a 200×200 histogram. Centre, the same points with the axes clipped to the 0.5–99.5 percentile range, which recovers the structure and shows that much of the collapse in the left panel is a viewing-range effect. Right, the distribution of $\log_{10}\|x\|$, spanning 5.4 and 6.5 decades. That last panel is the one that matters for the index: it is why a cell built around a centroid can have one member hundreds of times further away than all the others.

This is the single most consequential fact about the dataset for everything that follows. A k-means cell in such a space is long, thin and occasionally has one extreme member far outside the rest. The classical way to certify that a cell is irrelevant to a query is a covering sphere around the centroid, sized by the furthest member, and in this data that sphere is set by the outlier and therefore certifies almost nothing. Measured on the 1M index, the median covering radius is 0.2492 while the largest is 5,107.63, against query radii in the range $[0.0655, 0.1066]$. One star makes one cell permanently unprunable.

3.3 Two similarity measures, two questions

Every experiment is run twice, once under Euclidean distance on the standardised coefficients and once under cosine distance, which is Euclidean distance on the same vectors after normalising

each to unit length. This is not redundancy, the two measures ask different questions about a star.

Euclidean distance keeps the amplitude of the spectrum, so two stars are close only if their spectra agree in both shape and overall scale. It is therefore sensitive to how much light the star delivers as well as to how that light is distributed. Cosine distance divides that amplitude out and compares only the *shape* of the spectrum, which is the part most directly tied to the star's effective temperature and spectral type: a hot blue star and a cool red one have differently shaped spectra regardless of how bright either appears. A scientist looking for objects of the same physical kind wants the cosine index; one looking for objects that also present similarly wants the Euclidean one.

The two behave very differently under every method in this report, and the reason is geometric rather than astrophysical. Normalising to unit length puts every vector on a sphere, which makes the cells nearly round and their centroids closely spaced. Geometric bounds exploit elongation, so they have much less to work with on cosine than on L2. This asymmetry recurs in every table below, and it is stated once here so that it does not need restating each time.

4 Method

4.1 The baseline, and what was wrong with it

The starting point was the mentor's recall-guaranteed IVF-PQ index. It partitions the database into $n_{\text{list}} = \sqrt{N}$ cells by k-means, stores every vector as a 10-byte product-quantized code of its residual from the cell centroid, and answers a radius query in three stages. Stage one opens every cell whose covering sphere reaches the query ball, $\|q - c_k\| - R_c \le \tau$, where R_c is the distance from the centroid to its furthest member. Stage two computes the approximate distance $a(x)$ for every vector in the opened cells and keeps those with $a(x) - \varepsilon(x) \le \tau$, where $\varepsilon(x)$ is the stored reconstruction error of that vector. Stage three computes exact distances for the survivors. Both tests are sound lower bounds, so recall is 1.0 by construction, and every table in this project verifies that empirically rather than assuming it.

The index worked and it was extremely wasteful. At one million stars the sphere test opened enough cells to send 88% of the database into stage two, and the ε band was wide enough that 179 vectors reached exact verification for every one that was a genuine answer. Both numbers trace back to the same geometry described in Section 3: elongated cells with extreme members make a covering sphere useless as a certificate, and coarse quantization makes ε large.

4.2 The things that did not work

Four ideas were tested at length before the one that worked, and they are summarised briefly here because knowing which levers are dead is part of the result.

A better partitioner. Six ways of making cells were compared at fixed n_{list} under identical conditions, including k-means, a capacity-constrained balanced assignment, bisecting k-means, BIRCH, a Gaussian mixture, and a random partition as a control. The entire spread between the best of them and a partition that ignores the data completely was 11.8 points of candidates at 1M under L2 and 4.7 under cosine, and the best still scanned 88% of the database. The balanced partitioner deserves a sentence of its own: it won on candidates, and when it was tested head to head against k-means on an identical coarse quantizer it raised the number of vectors reaching exact verification by 53.3%, because a capacity cap pushes points off their nearest centroid, which enlarges their residuals and therefore ε. Density methods were also tried and are not usable at all here, since DBSCAN returns one cluster holding 99% of the sample and HDBSCAN labels 84% of it as noise, while posting lists must contain every star.

More cells. Multiplying n_{list} by 32 does reduce candidates, monotonically, exactly as the covering-radius argument predicts. At 1M it bought 7.6% fewer candidates for 20.7 times the build time, and the return got worse with scale rather than better. n_{list} was left at $\sqrt{N}$.

A finer quantizer alone. Optimized product quantization recovered 1.8–9.2% of survivors at 1.4 times the query time. Low-bit scalar quantization did far better on survivors, by factors of two to six, but at 1M it cost 15% more query time than the baseline, because it raises the per-candidate work by an order of magnitude and there were still 880,000 candidates per query to pay it on.

A hierarchy above the cells. Clustering the centroids into groups and skipping whole groups is the textbook way to make stage one cheaper. It never skipped a single group, at any group count, because a group inherits the worst covering radius among its members and one bad cell keeps the whole group open.

The common thread is that all four operate downstream of, or alongside, a cell test that was admitting almost everything. That is what the final method replaces.

4.3 The four-bound cell stage

A cell is opened only if *every* active test says it might contain a point within τ. Four tests are used, shown in Figure 2.

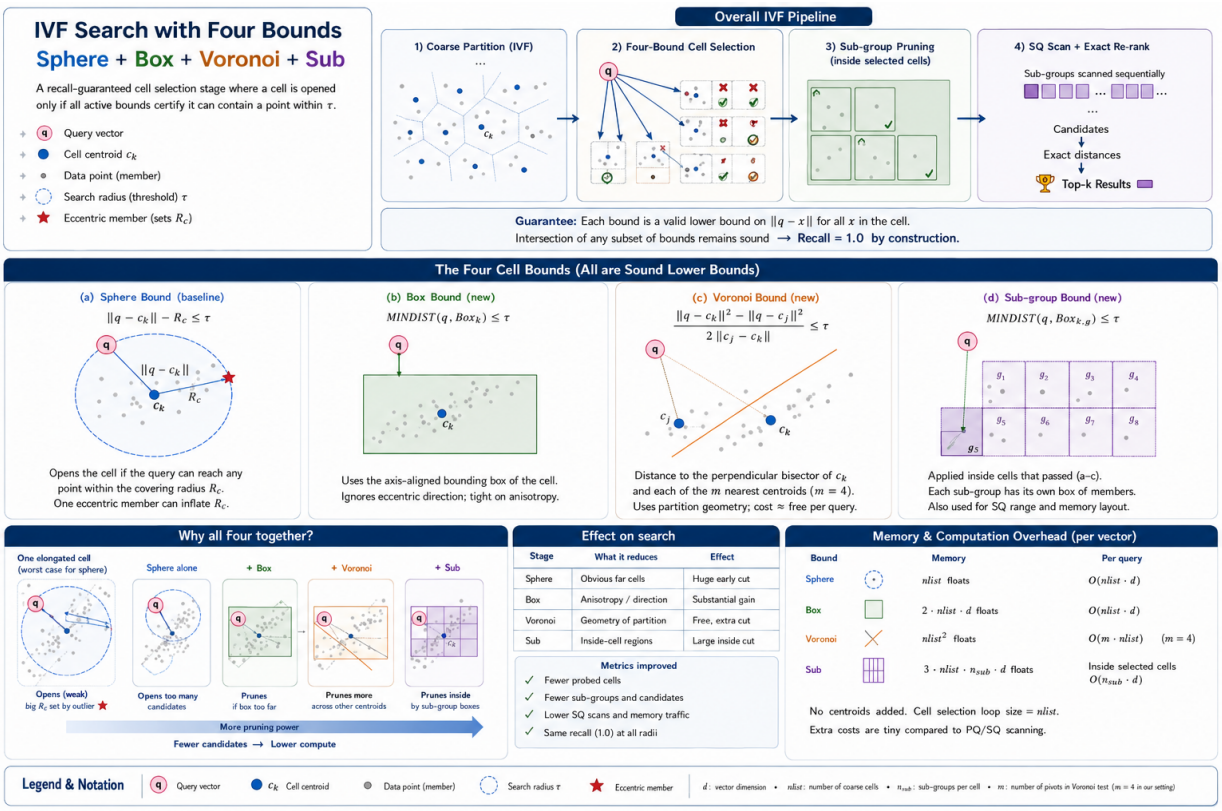


Figure 2 The four-bound cell stage. A cell survives only if all four certificates fail to rule it out. The sphere is the baseline test and is the one an eccentric member can destroy; the box, the Voronoi bisector and the sub-group boxes never read the covering radius at all, which is why they keep working on a partition the sphere test cannot handle.

- **Sphere**, $\|q - c_k\| - R_c \le \tau$. The baseline test. It costs n_{list} floats and nothing per vector, and it is the one that a single eccentric member can render useless.

- **Box**, MINDIST$(q, B_k) \le \tau$, where B_k is the axis-aligned bounding box of the cell's members. A box hugs an elongated cell where a sphere encloses it loosely, so this test is tighter than the sphere in 99.3% of cells at 1M. It costs $2n_{\text{list}}d$ floats and nothing per vector.
- **Voronoi**, $(\|q - c_k\|^2 - \|q - c_j\|^2)/(2\|c_j - c_k\|) \le \tau$, evaluated against the four centroids nearest the query. If x was assigned to cell k then $\|x - c_k\| \le \|x - c_j\|$ for every other centroid j, which is simply what assignment means, so x must lie on the c_k side of the perpendicular bisector of c_k and c_j, and the distance from q to that bisector lower-bounds the distance to any member. The partition is already a Voronoi diagram of its centroids, so this certificate was there all along and was never used. All the required $\|q - c_j\|^2$ are computed for the sphere test anyway, so the new arithmetic is one small division per query and the new storage is an $n_{\text{list}} \times n_{\text{list}}$ matrix of centroid distances, again nothing per vector.
- **Sub**, MINDIST$(q, B_{k,g}) \le \tau$, applied *inside* cells that already passed. Each cell's members are sorted along that cell's widest axis and cut into $n_{\text{sub}} = 8$ equal groups, each with its own bounding box. The same structure does three jobs: it prunes within an opened cell, which no cell-level test can do; it gives the quantizer a narrower range per group and therefore a smaller ε at the same bit depth; and it makes each group's records contiguous, so scanning one is a sequential read. It adds no centroids.

Each of the four is a sound lower bound on $\|q - x\|$ for every x in the cell, so the intersection of any subset of them is still sound and recall remains exactly 1.0 whichever subset is active. That is what makes the ablation in Section 5 a clean measurement: turning bounds on and off changes how much is scanned and cannot change what is returned.

4.4 Product quantization against scalar quantization

The cell stage decides how many vectors are *scanned*. The quantizer decides how many of those *survive* to exact verification. The two are cleanly separable, because the cell test reads only centroids, radii, boxes and the centroid matrix and never touches a code, while the quantizer produces a and ε and cannot open or close a cell.

Product quantization splits the 110 dimensions into $M = 10$ blocks of 11 and replaces each block with one of $K_s = 256$ learned centroids, so a vector costs 10 bytes. It is extremely compact and its reconstruction error is correspondingly large. Scalar quantization takes the same idea to its limit, one codebook per dimension, so $M = d = 110$, and stores b bits per dimension for $\lceil db/8 \rceil$ bytes per vector. Bits are the dial. More bits mean a smaller ε, and since the survivors are exactly the vectors with $a - \varepsilon \le \tau$, the surviving set is essentially everything within $\tau + 2\varepsilon$. Shrinking ε shrinks that shell directly. Five depths are swept, 8, 6, 5, 4 and 3 bits, costing 110, 83, 69, 55 and 42 bytes per vector against product quantization's 10.

The important practical point is that this is a knob the operator controls rather than a property of the method. Nothing about the guarantee, and nothing about the cell stage, changes with bit depth. A deployment that is short of memory picks a low depth and accepts more exact-distance work; one that has memory to spare picks 8 bits and gets an index that returns very nearly the answer set and nothing else.

4.5 Protocol

Identical throughout and identical to the project's first report, so every number here is comparable with every earlier one. Both metrics, 100 query vectors drawn from the database itself, six radii at the 0.1, 0.5, 1, 2, 5 and 10th percentiles of the exact distance distribution of those queries, and brute-force ground truth for grading. $n_{\text{list}} = \sqrt{N}$, mini-batch k-means with `batch_size` = 20,000, $n_{\text{sub}} = 8$, four Voronoi pivots, and $M = 10$, $K_s = 256$ for the product-quantized baseline. The

harness verifies on every row that no true neighbour was pruned and that the refined output equals the brute-force answer set exactly. Precision is reported *micro*, as $\sum \mathrm{TP} / \sum$ survivors, because the quantity being managed is the total refinement bill and a macro average would let a query with a hundred survivors count as much as one with a hundred thousand.

4.6 A note on the upper bound

The same stored ε that gives the lower bound also gives an upper one: a vector with $a(x) + \varepsilon(x) \le \tau$ is provably *inside* the ball and could be returned without an exact check. This was measured throughout and it is not worth much. Under product quantization it removed at most 0.85% of the refinement work, and only at the widest radius, because acceptance needs $a \le \tau - \varepsilon$ and with a large ε that window is nearly empty. It becomes more useful as ε shrinks and as τ grows, so it is worth keeping as a cheap final filter, but it was never the lever.

5 Results

Every number below was produced with recall verified rather than assumed. Across the sweep and the ablation, $\mathrm{recall}_{\mathrm{LB}}$ and the exact-refinement check read 1.000 on all 20,400 audited query–radius combinations, at both scales, on both metrics and at every bit depth. Nothing in this section trades accuracy for speed; it only reports how much work each configuration had to do to return the same answer.

5.1 Which bound does the work

Table 1 turns the bounds on one at a time with the quantizer held fixed. Only the candidate count can move, since a cell test cannot change ε, which makes this a clean measurement of the cell stage alone.

Table 1 Fraction of the database scanned at the tightest radius, adding one bound at a time. Lower is better. The quantizer is held at 8 bits throughout and survivors are identical down each column, so every change here is the cell test.

Bounds active	10K L2	10K cosine	1M L2	1M cosine
sphere (baseline)	0.9113	0.9952	0.8775	0.9550
sphere + box	0.7492	0.9624	0.6000	0.8066
sphere + voronoi	0.6089	0.9259	0.4817	0.7711
sphere + box + voronoi	0.6018	0.9127	0.4557	0.7207
sphere + box + voronoi + sub	**0.4611**	**0.7072**	**0.4001**	**0.6500**

Three things follow. The Voronoi test is the strongest single bound everywhere, removing 45.1% of the candidates on top of the sphere at 1M L2 against the box's 31.6%, and it is also the cheapest of the new tests, since the distances it needs were already computed for the sphere. The box and the Voronoi test are complementary rather than redundant, because neither implies the other, so adding the box on top of sphere+voronoi still buys 5.4% at 1M L2. And `sub` contributes a further 12.2% at 1M L2 and 23.4% at 10K L2 by pruning inside cells that have already been opened, which no cell-level test can do.

Taken together the four bounds cut the fraction of the database scanned from 0.8775 to 0.4001 at 1M L2 and from 0.9550 to 0.6500 at 1M cosine. They do this on the same partition as the baseline, with the pathological covering radius of 5,107.63 still present. The covering radius did not need to be repaired; it needed to stop being the only thing consulted.

5.2 The funnel, end to end

Table 2 is the main result of the project. It reports the four quantities that matter, the number of vectors scanned, the number reaching exact verification, the micro precision of that survivor set, and the wall-clock time, for the product-quantized baseline and for every scalar-quantization depth under the four bounds, at both scales and on both metrics, at the tightest and widest radius.

Table 2 The full funnel. `cand` is vectors scanned, `surv` is vectors reaching exact verification, `prec` is micro precision of the survivor set, `ms` is milliseconds per query. The baseline is IVF-PQ at 10 B/vec under the sphere bound; every SQ row uses all four bounds. Candidates are identical across bit depths by construction, because the bounds and not the quantizer decide them. Recall is 1.000 in every row. Timings are single-session and indicative; the counts are deterministic.

	tightest radius				widest radius			
Index	cand	surv	prec	ms	cand	surv	prec	ms
10K — L2 (τ = 0.1350, 10 true answers; τ = 0.2166, 1,000 true answers)								
IVF-PQ + sphere	9,113	1,561	0.0064	16.6	9,301	4,940	0.2024	15.4
SQ 8-bit + 4 bounds	**4,611**	**10.7**	**0.9355**	5.3	**7,014**	**1,022**	**0.9782**	33.8
SQ 6-bit + 4 bounds	4,611	13.4	0.7453	21.1	7,014	1,091	0.9164	9.9
SQ 5-bit + 4 bounds	4,611	17.8	0.5620	5.3	7,014	1,186	0.8433	10.5
SQ 4-bit + 4 bounds	4,611	30.2	0.3319	7.5	7,014	1,391	0.7187	10.2
SQ 3-bit + 4 bounds	4,611	85.1	0.1176	5.6	7,014	1,866	0.5361	10.6
10K — cosine (τ = 0.5880, 10 true answers; τ = 0.8517, 1,000 true answers)								
IVF-PQ + sphere	9,952	2,159	0.0046	14.2	10,000	6,866	0.1456	16.9
SQ 8-bit + 4 bounds	**7,072**	**10.6**	**0.9408**	12.0	**9,596**	**1,023**	**0.9775**	18.2
SQ 6-bit + 4 bounds	7,072	12.7	0.7869	12.1	9,596	1,095	0.9132	18.4
SQ 5-bit + 4 bounds	7,072	16.2	0.6198	11.9	9,596	1,193	0.8382	27.9
SQ 4-bit + 4 bounds	7,072	26.3	0.3806	12.9	9,596	1,413	0.7079	28.9
SQ 3-bit + 4 bounds	7,072	66.8	0.1497	59.5	9,596	1,947	0.5137	25.6
1M — L2 (τ = 0.0655, 1,000 true answers; τ = 0.1066, 100,000 true answers)								
IVF-PQ + sphere	877,517	178,806	0.0056	543.1	894,256	492,445	0.2031	977.7
SQ 8-bit + 4 bounds	**400,140**	**1,111**	**0.9003**	271.5	**645,623**	**102,440**	**0.9762**	698.2
SQ 6-bit + 4 bounds	400,140	1,500	0.6668	**265.9**	645,623	110,021	0.9089	744.8
SQ 5-bit + 4 bounds	400,140	2,200	0.4545	267.3	645,623	120,733	0.8283	707.6
SQ 4-bit + 4 bounds	400,140	4,427	0.2259	266.2	645,623	144,337	0.6928	725.2
SQ 3-bit + 4 bounds	400,140	14,145	0.0707	261.8	645,623	200,199	0.4995	765.8
1M — cosine (τ = 0.5986, 1,000 true answers; τ = 0.9019, 100,000 true answers)								
IVF-PQ + sphere	954,953	217,933	0.0046	592.4	999,936	662,150	0.1510	1,190.4
SQ 8-bit + 4 bounds	**649,965**	**1,092**	**0.9160**	440.8	**932,454**	**103,096**	**0.9700**	997.9
SQ 6-bit + 4 bounds	649,965	1,409	0.7098	444.7	932,454	112,819	0.8864	918.7
SQ 5-bit + 4 bounds	649,965	1,960	0.5102	**438.1**	932,454	126,799	0.7886	**912.1**
SQ 4-bit + 4 bounds	649,965	3,646	0.2743	443.5	932,454	158,321	0.6316	950.9
SQ 3-bit + 4 bounds	649,965	10,959	0.0912	451.6	932,454	236,069	0.4236	1,012.0

The refinement stage collapses. At the tightest 1M L2 radius the baseline sends 178,806 vectors to exact verification for 1,000 true answers; the eight-bit index sends 1,111, an overfetch of 1.11 and a reduction of 161 times. On cosine the same comparison is 217,933 against 1,092, a factor of 200. Micro precision rises from 0.0056 to 0.9003 and from 0.0046 to 0.9160. The advantage narrows as the radius widens, to a factor of 4.8 on L2 and 6.4 on cosine at the widest radius, which is correct behaviour rather than a weakness: when most of what is scanned is a genuine answer, no index can prune what is really there.

Time improves at 1M in every configuration. At the tightest L2 radius the six-bit pipeline runs at 265.9 ms against the baseline's 543.1 ms, a factor of 2.0, and it is still 1.3 times faster at the widest. On cosine the gain is smaller but consistent, the eight-bit pipeline running 1.34 times faster at the tightest radius and 1.19 at the widest. This is worth emphasising because the previous stage of the project had to concede that scalar quantization was 15% *slower* than the baseline at

1M. Nothing about the quantizer changed to reverse that. What changed is that there are now roughly a third fewer candidates to pay the per-candidate cost on.

The 10K timings should not be read the same way. They swing without a pattern that tracks the workload, 21.1 ms for six-bit at the tightest L2 radius against 9.9 ms at the widest on strictly more work, and 59.5 ms for three-bit cosine against 25.6 ms two rows later. At ten thousand vectors the per-query work is small enough that session noise dominates, so the 10K columns of Table 2 should be read for candidates, survivors and precision, which are deterministic, and not for milliseconds. The 1M timings are trustworthy because the work per query dwarfs the noise, and even there the figures are single-session.

5.3 What the guarantee costs, and how the bits buy it back

Figure 3 shows the same comparison as a picture. Each panel contrasts the guaranteed pipeline against simply thresholding the approximate distance without the ε correction, which is what a conventional approximate index would do.

Two readings come out of it. First, the guarantee is not free and its price is visible: on the baseline the guaranteed precision curve sits near zero while the approximate-only curve sits near 0.7, which is the cost of never missing anything. Second, that price is almost entirely removed by spending bytes. At 8 bits the two curves nearly coincide, so the index is delivering the precision of an approximate system while still returning provably every answer.

Figure 4 shows the same effect at 1M as counts rather than ratios. The dashed line is the true answer count and the eight-bit and six-bit curves sit almost on top of it across the whole radius range, while the baseline sits an order of magnitude above.

5.4 Memory, which is the one cost that did not go down

Table 3 is the trade-off stated plainly. Scalar quantization is between four and eleven times larger than product quantization per vector, and at catalogue scale that is the difference between an index that fits in a laptop and one that needs a server.

Table 3 Storage against work. Compression is relative to raw `float32` at 440 B/vec. Index sizes are codes only and are analytic figures for a packed layout; the harness stores one byte per dimension at every depth. The last four columns are measured at 1M L2, ε and overfetch at the tightest radius and MB moved per query at the widest.

Index	B/vec	compr.	1M (MB)	200M (GB)	2B (GB)	$\tilde{\varepsilon}$	$\tilde{\varepsilon}/\tau_{\min}$	overfetch	MB/query
raw `float32`	440	1.0×	419.6	81.9	819.6	0	0	1.00	—
IVF-PQ + sphere	10	44.0×	9.5	1.9	18.6	—	—	178.80	215.17
SQ 8-bit + 4 bounds	110	4.0×	104.9	20.5	204.9	0.00055	0.008	**1.11**	110.71
SQ 6-bit + 4 bounds	83	5.3×	79.2	15.5	154.6	0.00223	0.034	1.50	97.27
SQ 5-bit + 4 bounds	69	6.4×	65.8	12.9	128.5	0.00452	0.069	2.20	**93.15**
SQ 4-bit + 4 bounds	55	8.0×	52.5	10.2	102.4	0.00935	0.143	4.43	94.43
SQ 3-bit + 4 bounds	42	10.5×	40.1	7.8	78.2	0.02005	0.306	14.15	109.87

The column that decides whether a depth is usable is $\tilde{\varepsilon}/\tau_{\min}$, the typical reconstruction error as a fraction of the tightest query radius. Once that ratio approaches one the test $a - \varepsilon \le \tau$ stops firing at all, and the overfetch column tracks it exactly, from 1.11 at 8 bits to 14.15 at 3. Three bits is where the funnel begins to lose its grip.

The last column changes the shape of the argument. Counting bytes *stored*, product quantization wins and always will. Counting bytes *moved per query*, which is the quantity a disk-resident index at two billion sources would actually pay, every bounded depth from 3 to 8 bits moves less than half of what the baseline moves, 93 to 111 MB against 215 MB, because the refinement term

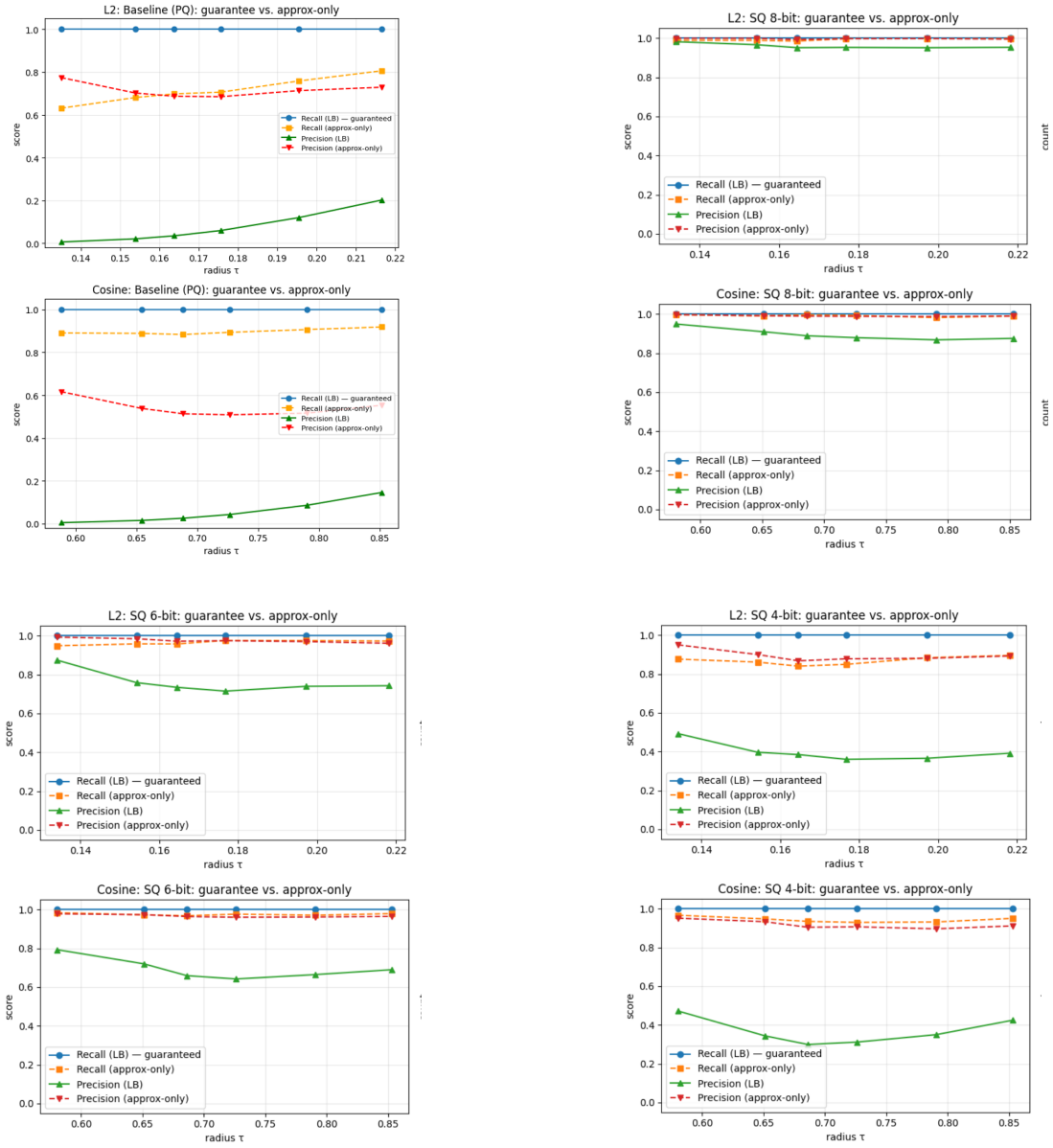


Figure 3 Guaranteed against approximate-only search at 10K, L2 above and cosine below within each panel. Top left, the IVF-PQ baseline; top right, 8-bit; bottom left, 6-bit; bottom right, 4-bit. The solid blue line is recall under the guarantee and it sits on 1.000 in all eight plots, because the depths change only a and ε and never the pruning rule. The dashed orange line is what a conventional approximate index would return, and it drops as low as 0.63 on the baseline: those are answers that would have been silently missing. The vertical gap between the solid green curve and the dashed red one is the price of the guarantee, and watching it close from the baseline to 8 bits is the visual form of the whole method. Precision here is the per-query mean, so it reads slightly higher than the micro precision of Table 2 at the tight radii.

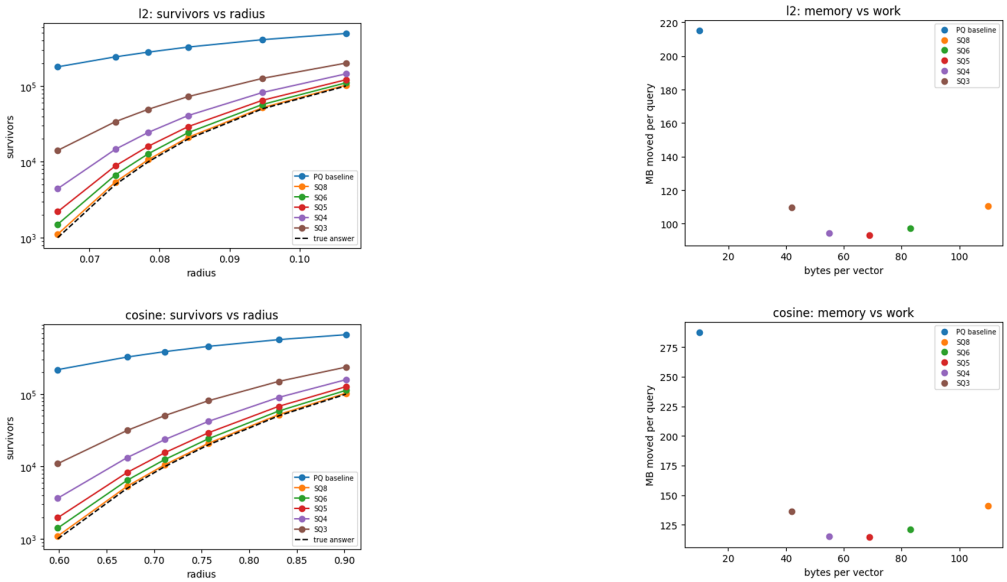


Figure 4 Left, vectors reaching exact verification against radius at 1M, L2 above and cosine below, with the true answer count as the dashed line. The bit depth moves a curve up or down between the baseline and the truth, and 8 bits is nearly indistinguishable from the truth itself. Right, megabytes moved per query against bytes stored per vector at the widest radius. Every bounded scalar-quantization depth moves less than half of what the baseline moves, and the total has a minimum in the middle of the range rather than at either end, because fewer bytes per vector make the scan cheaper but leave more survivors to fetch.

collapsed by two orders of magnitude while the scan term grew by only a factor of four to eleven. The total has a minimum in the middle of the range, at 5 bits, rather than at the compact end.

6 Discussion

6.1 Is the system efficient

The project's name claims efficiency, and it is worth saying exactly what that word is standing on. Three things had to be true at once. The answer must be provably complete, which it is, verified on every query at every radius rather than argued from theory. The work must be substantially less than the baseline's, which it is on both stages of the funnel: 40% of the database scanned instead of 88%, and 1.11 vectors verified per true answer instead of 179. And the wall clock must actually improve, which it does at 1M, by a factor of two at the tight radii and 1.3 at the widest.

That third condition is the one that took the longest, and the way it was finally met is the most transferable lesson here. For three of the four weeks, every gain in precision was paid for in time. Scalar quantization raised M from 10 to 110 and therefore made every candidate eleven times more expensive to score, which at 880,000 candidates per query was a 15% slowdown that no amount of tuning inside the quantizer could fix. The fix was not to make the per-candidate work cheaper. It was to have a third fewer candidates. Once the cell stage stopped admitting almost everything, the same expensive quantizer became a net win, and precision and speed stopped competing.

6.2 Why the bounds worked when the partition did not

A great deal of effort went into making better cells, and almost none of it mattered. Six partitioners spanned 11.8 points of candidates at 1M including a random control; multiplying the number of

cells by 32 bought 7.6% for 20.7 times the build time; fixing a genuine bug in the clustering that cut the median covering radius by 28.6% moved candidates by 0.5%. Three independent attempts at improving the partition, three results near zero.

The reason is that a partition is a set of regions, and what admits a candidate is not where the boundaries fall but how loosely a query can be shown not to intersect a region. In a space where one principal component carries most of the variance, cells are long and thin, and a sphere around a long thin cell is sized by its longest axis, so it certifies almost nothing. The box hugs the cell instead of enclosing it. The Voronoi test does not look at the cell's extent at all, it uses the fact that membership already implies which side of every bisector a point lies on, which is a certificate the index was carrying for free and never reading. The sub-group boxes cut inside a cell that has already been opened, which no cell-level test can do. All three work on the poisoned partition, with the 5,107.63 covering radius left exactly where it was. The covering radius did not have to be fixed; it had to stop being load-bearing.

6.3 The two metrics behave differently, and that is geometry, not physics

Every result in this report is weaker on cosine than on L2. The four bounds take 1M L2 from 0.8775 to 0.4001 of the database and 1M cosine only from 0.9550 to 0.6500. This is not a statement about stellar spectra, it is a statement about the unit sphere. Normalising every vector to unit length makes the cells nearly round and puts the centroids close together, and a box around a round cell is barely tighter than a ball, while a bisector between two nearby centroids has little room to cut. Every geometric bound tried in this project has been worth between two and five times more on L2 than on cosine.

What does move the cosine index is the quantizer, which is indifferent to cell shape. At the tightest 1M cosine radius the eight-bit index reduces survivors by a factor of 200 against the baseline, its largest reduction anywhere in the project. So the two metrics need the same system for different reasons: L2 benefits most from the cell stage, cosine most from the quantizer, and shipping both together serves both.

6.4 The trade-off, and how much of it is under control

The honest summary is that precision and time both improved substantially and memory got worse. A vector costs 110 bytes at 8 bits or 83 at 6, against product quantization's 10, and at two billion sources that is 205 GB or 155 GB against 18.6 GB. There is no way to present that as a win on stored bytes.

Two things make it a manageable cost rather than a disqualifying one. First, it is a dial rather than a property. The cell stage is the part of this system that does the structural work, and it costs zero bytes per vector, no additional centroids and no second clustering pass, so it composes with any quantizer at any depth. An operator short of memory can run 4 bits at 102 GB for two billion sources and still get 0.69 micro precision at the widest radius and a scan of 40% of the database; an operator with memory to spare can run 8 bits and get an index whose survivor set is 1.11 times the answer set. Nothing about the guarantee changes between those two choices. That range, a factor of 2.6 in memory across depths that all still beat the baseline on work, is what makes the memory cost a decision rather than a defect.

Second, the axis that matters at catalogue scale is probably not stored bytes but moved bytes. If exact refinement at two billion sources is an I/O operation, then the quantity being paid per query is the last column of Table 3, and on that column every depth beats the baseline by better than two to one. That reframing is not a proof, it rests on an assumption about the storage hierarchy that this project never measured, and it is named as the first open question in Section 7. But it

moves the argument from one that cannot be won to one that can.

7 Limitations and Future Work

The system does what it claims at the scales it was measured on, and the honest boundaries of that claim are worth stating in one place.

Memory is the one cost that was not reduced, and settling whether it is worth paying needs a number this project never measured: what an exact-distance fetch costs at 200 million or two billion sources. If it is a memory read, stored bytes are the right currency and product quantization keeps its advantage; if it is a disk read, moved bytes are the right currency and every bounded depth here wins comfortably. Everything else in the trade-off is already characterised.

The measurements stop at one million stars. The projection to 200 million in Table 3 holds the measured scan fraction and survival rate constant, and neither has been tested above 1M, although neither showed strong size dependence between the two scales that were tested. Timings are single-session NumPy in Python and are indicative; candidates, survivors, precision and ε are deterministic and are the figures to quote. Low bit depths are simulated rather than packed, since the code array holds one byte per dimension at every depth, so the storage and bandwidth columns are analytic figures for what a packed implementation would move rather than measured ones, and a real packed index should be faster at low depths than these timings suggest.

Within the method, three things are unfinished and all three are cheap:

- The Voronoi bound uses four pivots because that was the first value tried, and it was never swept. It is the strongest single bound in every configuration measured, so this is the most likely place a further gain is sitting. Its side table also grows as n_{list}^2, which is 3.81 MB at $n_{\text{list}} = 1{,}000$ but 381 MB at 10,000, so at catalogue scale it needs a pivot subset rather than the full matrix.
- Sub-group pruning keeps improving all the way to 32 groups per cell on candidates, on ε and on bytes moved, but the wall clock turns around after 2 because the scan iterates once per group in Python. Vectorising that loop would unlock roughly another 11% of candidates at 1M L2.
- The attribute filter is wired through both query paths and was never exercised. It is the filtered half of filtered vector search, it is what the project was originally proposed to deliver [18, 19], and it is the one lever that could cut candidates by more than the geometry does.

8 Conclusion

This project set out to make a recall-guaranteed threshold query affordable on a catalogue of two billion stellar spectra, and it ends with a system that returns provably every answer while scanning 40% of the database instead of 88%, verifying 1.11 vectors for every true answer instead of 179, and running twice as fast as the baseline it started from.

The route there was mostly elimination. Better clustering algorithms, more cells, a hierarchy above the cells, a rotated codebook and a capacity-balanced partition were all tested under one protocol and all returned somewhere between nothing and a small fraction of what was needed. What they had in common is that they operated on or around a cell test that was admitting almost the entire database, and the term they were all trying to move was sized by that test rather than by them. Replacing the test was the intervention that mattered. An axis-aligned box, a Voronoi bisector test against the four nearest centroids, and per-cell sub-group boxes are all classical ideas,

none of them new; what this project contributes is the observation that they compose, that their intersection is still a sound lower bound so recall is untouched, and that on spectra with one dominant principal component they are worth together what a covering sphere cannot deliver alone. The Voronoi test in particular was free the whole time, since a k-means partition already is a Voronoi diagram and the distances it needs were already being computed.

Pairing that cell stage with a scalar quantizer turned the project's longest-standing trade-off into a setting. Bit depth moves the survivor set between 1.11 and 14.15 times the true answer count and the index between 78 and 205 GB at two billion sources, and every depth in that range still scans less than half of what the baseline scans and moves less than half the bytes per query. Precision and speed both improved, memory got worse, and the amount by which it gets worse is chosen rather than imposed.

What did not change anywhere in four reports is the guarantee. Recall was 1.000 and the refined output was identical to brute-force truth on every query, at every radius, at both scales, on both metrics and at every bit depth. That was the point of the exercise: not a system that is usually right, but one that is provably complete and, now, fast enough to use.

References

[1] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. "Product quantization for nearest neighbor search". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.1 (2011), pp. 117–128. DOI: 10.1109/TPAMI.2010.57.

[2] Jeff Johnson, Matthijs Douze, and Hervé Jégou. "Billion-scale similarity search with GPUs". In: *IEEE Transactions on Big Data* 7.3 (2021), pp. 535–547. DOI: 10.1109/TBDATA.2019.2921572.

[3] Tiezheng Ge et al. "Optimized product quantization". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.4 (2014), pp. 744–755. DOI: 10.1109/TPAMI.2013.240.

[4] Cecilia Aguerrebere et al. "Similarity search in the blink of an eye with compressed indices". In: *Proceedings of the VLDB Endowment* 16.11 (2023), pp. 3433–3446. DOI: 10.14778/3611479.3611537.

[5] Keinosuke Fukunaga and Patrenahalli M. Narendra. "A branch and bound algorithm for computing k-nearest neighbors". In: *IEEE Transactions on Computers* C-24.7 (1975), pp. 750–753. DOI: 10.1109/T-C.1975.224297.

[6] Paolo Ciaccia, Marco Patella, and Pavel Zezula. "M-tree: An efficient access method for similarity search in metric spaces". In: *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB)*. 1997, pp. 426–435.

[7] Antonin Guttman. "R-trees: A dynamic index structure for spatial searching". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1984, pp. 47–57. DOI: 10.1145/602259.602266.

[8] Nick Roussopoulos, Stephen Kelley, and Frédéric Vincent. "Nearest neighbor queries". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1995, pp. 71–79. DOI: 10.1145/223784.223794.

[9] Jeffrey K. Uhlmann. "Satisfying general proximity/similarity queries with metric trees". In: *Information Processing Letters* 40.4 (1991), pp. 175–179. DOI: 10.1016/0020-0190(91)90074-R.

[10] Sergey Brin. "Near neighbor search in large metric spaces". In: *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB)*. 1995, pp. 574–584.

[11] Franz Aurenhammer. “Voronoi diagrams: A survey of a fundamental geometric data structure”. In: *ACM Computing Surveys* 23.3 (1991), pp. 345–405. DOI: 10.1145/116873.116880.

[12] D. Sculley. “Web-scale k-means clustering”. In: *Proceedings of the 19th International Conference on World Wide Web (WWW)*. 2010, pp. 1177–1178. DOI: 10.1145/1772690.1772862.

[13] David Arthur and Sergei Vassilvitskii. “k-means++: The advantages of careful seeding”. In: *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 2007, pp. 1027–1035.

[14] Michael Steinbach, George Karypis, and Vipin Kumar. “A comparison of document clustering techniques”. In: *KDD Workshop on Text Mining*. 2000.

[15] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. “BIRCH: An efficient data clustering method for very large databases”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 1996, pp. 103–114. DOI: 10.1145/233269.233324.

[16] Martin Ester et al. “A density-based algorithm for discovering clusters in large spatial databases with noise”. In: *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*. 1996, pp. 226–231.

[17] Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. “Density-based clustering based on hierarchical density estimates”. In: *Advances in Knowledge Discovery and Data Mining (PAKDD)*. 2013, pp. 160–172. DOI: 10.1007/978-3-642-37456-2_14.

[18] Yannis Chronis et al. “Filtered vector search: State-of-the-art and research opportunities”. In: *Proceedings of the VLDB Endowment* 18.12 (2025), pp. 5488–5492.

[19] Mocheng Li et al. “Attribute filtering in approximate nearest neighbor search: An in-depth experimental study”. In: *Proceedings of the ACM on Management of Data* 3.6 (2025), pp. 1–26.

[20] Yanjun Lin et al. “Survey of filtered approximate nearest neighbor search over the vector-scalar hybrid data”. In: *arXiv preprint arXiv:2505.06501* (2025).

[21] Jiayang Shi, Yuzheng Cai, and Weiguo Zheng. “Filtered approximate nearest neighbor search: A unified benchmark and systematic experimental study”. In: *arXiv preprint arXiv:2509.07789* (2025).

[22] Gaia Collaboration et al. “Gaia Data Release 3: Summary of the content and survey properties”. In: *Astronomy & Astrophysics* 674 (2023), A1. DOI: 10.1051/0004-6361/202243940.

[23] Reem Alghamdi et al. “Lactea: Web-based spectrum-preserving multi-resolution visualization of the Gaia star catalog”. In: *Computer Graphics Forum* 44.3 (2025), e70117.