

δ -EMG: A Monotonic Graph Index for Approximate Nearest Neighbor Search

Liming Xiang
Beijing Institute of Technology
Beijing, China
3120240967@bit.edu.cn

Jing Feng
Beijing Institute of Technology
Beijing, China
scererymc@gmail.com

Ziqi Yin
Nanyang Technological University
Singapore
ziqu003@e.ntu.edu.sg

Zijian Li
Huawei Technologies Ltd
China
lizijian8@huawei.com

Daihao Xue
Huawei Technologies Ltd
China
xuedaihao@huawei.com

Hongchao Qin
Beijing Institute of Technology
Beijing, China
qhc.neu@gmail.com

Ronghua Li
Beijing Institute of Technology
Beijing, China
lironghuabit@126.com

Guoren Wang
Beijing Institute of Technology
Beijing, China
wanggrbit@gmail.com

Abstract

Approximate nearest neighbor (ANN) search in high-dimensional spaces is a foundational component of many modern retrieval and recommendation systems. Currently, almost all algorithms follow an ϵ -Recall-Bounded principle when comparing performance: they require the ANN search results to achieve a recall of more than $1 - \epsilon$ and then compare query-per-second (QPS) performance. However, this approach only accounts for the recall of true positive results and does not provide guarantees on the deviation of incorrect results. To address this limitation, we focus on an Error-Bounded ANN method, which ensures that the returned results are a $(1/\delta)$ -approximation of the true values. Our approach adopts a graph-based framework. To enable Error-Bounded ANN search, we propose a δ -EMG (Error-bounded Monotonic Graph), which, for the first time, provides a provable approximation for arbitrary queries. By enforcing a δ -monotonic geometric constraint during graph construction, δ -EMG ensures that any greedy search converges to a $(1/\delta)$ -approximate neighbor without backtracking. Building on this foundation, we design an error-bounded top- k ANN search algorithm that adaptively controls approximation accuracy during query time. To make the framework practical at scale, we introduce δ -EMQG (Error-bounded Monotonic Quantized Graph), a localized and degree-balanced variant with near-linear construction complexity. We further integrate vector quantization to accelerate distance computation while preserving theoretical guarantees. Extensive experiments on the ANN-Benchmarks dataset demonstrate the effectiveness of our approach. Under a recall requirement of 0.99, our algorithm achieves 19,000 QPS on the SIFT1M dataset, outperforming other methods by more than 40%. Additional results show that, under the same precision requirements, our δ -Error-bounded approach achieves higher QPS than all existing SOTA methods.

Keywords

ANN Search; Error-Bounded; Graph-based; Vector Database

1 Introduction

Nearest neighbor search on high-dimensional vector data is a fundamental problem and has many real-world applications, including image retrieval [34, 42], large language models [10, 36] and recommendation system [40]. In simple terms, given a dataset $V \subset \mathbb{R}^d$ and a query point $q \in \mathbb{R}^d$, the goal of top- k NN search is to find the k points in V closest to q based on a distance metric, typically the Euclidean distance. To improve search efficiency, much of the current research focuses on Approximate Nearest Neighbor (ANN) search, which trades a small loss in accuracy for significant gains in speed. Nowadays, most ANN search methods rely on the ϵ -Recall-Bounded principle, requiring a recall of more than $1 - \epsilon$ and comparing query-per-second (QPS) performance. However, this approach does not ensure error bounds for the returned results.

To enable error-bounded ANN Search, researchers have conducted various studies. From a theoretical perspective, Har-Peled's work showed that an approximate Voronoi diagram can achieve a $(1 + \epsilon)$ -approximation for top-1 NN queries in logarithmic time, though it requires space on the order of $O(n/\epsilon^d)$ [26]. Arya et al. later improved this by reducing the ANN problem to an approximate polytope membership query, achieving near-optimal space complexity $O(n/\epsilon^{d/2})$ while maintaining logarithmic query time [3, 4]. In the database field, methods like NSG [18] can guarantee finding the exact top-1 NN if the query point is part of the dataset ($q \in V$). Similarly, FANNG [27] and τ -MG [41] offer exact top-1 NN guarantees only when the distance to the true nearest neighbor is within a predefined threshold τ . While these approaches contribute to error-bounded ANN search, they are limited to top-1 NN problems, which is insufficient for practical applications. **In this paper, we propose a graph-based method that achieves a $(1/\delta)$ -approximation for top- k nearest neighbor search with a space complexity of $O(n \ln n)$.** To the best of our knowledge, this is the first method to provide an error-bounded guarantee for top- k ($k > 1$) ANN search.

Besides, to improve search efficiency, current mainstream methods involve vector quantization to accelerate distance calculations. Among these, the state-of-the-art method, SymphonyQG[24], has

integrated RaBitQ[20] into a proximity graph. We also incorporate our proposed error-bounded proximity graph into quantization and introduce an exploration strategy that adaptively selects between approximate and exact distance computations. This significantly reduces the errors that can arise from using only approximate distances. **Experimental results show that our error-bounded approach achieves higher QPS than all existing state-of-the-art methods under the same precision requirements.**

To achieve efficient error-bounded ANN search, we first introduce the δ -Error-Bounded Monotonic Graph (δ -EMG). This is a proximity graph where a monotonic search ensures that the final node’s distance from any query q is no more than $1/\delta$ times the true distance. We define the Occlusion Region on δ -EMG, using it to select neighbors and construct the graph. During ANN search on this graph, finding a local optimum node (a node with no closer neighbors to query q) ensures the search is Error-Bounded. Furthermore, to enhance performance, we propose a quantized version of the proximity graph, δ -EMQG, together with an exploration strategy that balances approximate and exact distance computations.

Our contributions are summarized as follows:

Models. We introduce the δ -Error-Bounded Monotonic Graph (δ -EMG), a novel proximity graph model designed to provide formal, rank-aware search guarantees. The core promise of this model is that for any query q , the i -th returned neighbor is a provable $(1/\delta)$ -approximation of the true i -th nearest neighbor, under the practical condition that a local optimum node is found during the search. This guarantee is upheld by the graph’s fundamental navigational property: for any starting point, a simple monotonic search is guaranteed to lead into a target region around the query, which we term the δ -neighborhood. This prevents the search from getting trapped in poor local optimum node far from the query, thus ensuring the error bound is met.

Algorithms. We propose a suite of algorithms for constructing and searching the δ -EMG. To realize the aforementioned error guarantees, we design a graph construction method based on a unique geometric occlusion rule. Our exact implementation of this rule yields a graph with an expected out-degree of $O(\log n)$ and $O(n \log n)$ space complexity. However, its $O(n^2 \log n)$ time complexity makes it impractical for large datasets.

To overcome this, we develop a highly efficient approximate construction algorithm that reduces the time complexity to near-linear ($O(Ln^{(d+1)/d} \ln(n^{1/d})/\Delta)$) and the space complexity to $O(n)$. This method produces a graph that serves as a localized and degree-balanced approximation of a true δ -EMG. Furthermore, we present a new search method for the δ -EMG that dynamically determines the candidate set size to ensure the desired error bounds.

Finally, we integrate vector quantization to create the δ -EMQG, a quantized variant that utilizes RaBitQ [20] for highly efficient distance estimation. This is paired with a novel probing search strategy that intelligently balances fast, approximate distance calculations with precise, exact ones to maximize throughput while maintaining high accuracy.

Experiments. We perform experiments on six real-world datasets to show the effectiveness and efficiency of our proposed methods. The results are compared with the existing methods such as NSG,

HNSW, NGT-QG and SymphonyQG. Our algorithm achieves a single-thread QPS of 19,000 on the SIFT1M dataset. At 99% recall with $k = 1$, it outperforms the best baseline by 1.2x to 3.2x. The error analysis matches our theoretical proof, confirming the accuracy of our method. Additionally, we conduct scalability and ablation experiments to further validate the algorithm.

2 Related Work

The ANN search problem has been widely explored, resulting in various indexing and search strategies [5, 6, 43, 48]. Below, we categorize the existing work into Non-Graph-Based and Graph-Based approaches and introduce them separately.

2.1 Non-Graph-Based ANN Search

Traditional ANN methods fall into several major categories. Tree-based methods [9, 12, 44] recursively divide the data space to support hierarchical search. While effective in low-dimensional settings, their performance deteriorates sharply in high dimensions due to the curse of dimensionality. Hashing-based methods [13, 19, 23] map nearby points to the same hash bucket with high probability, enabling sublinear search in expectation. Inverted file (IVF) based methods[8] cluster the dataset and build an inverted index from cluster centroids to data points, drastically reducing the search space to a few promising clusters. Quantization-based methods, such as Product Quantization (PQ) [1, 22, 31], compress high-dimensional vectors by partitioning them into subspaces and quantizing each subspace independently.

2.2 Graph-Based ANN Search

Graph-based methods represent the state-of-the-art in ANN search. Given a set of vectors V , they construct a graph $G = (V, E)$ where nodes correspond to vectors and edges define neighborhood relationships among them. Such graphs are collectively referred to as proximity graphs [29]. Queries are answered by traversing the graph, typically using a greedy search, to find the approximate nearest neighbors. Existing graph-based approaches can be broadly grouped into three main families.

k -NN Graph Variants. k -Nearest Neighbor (k -NN) graph connects each node to its k nearest neighbors. It serves as a practical approximation of the theoretically ideal but computationally expensive Delaunay Graph (DG)[7, 35]. Unlike the DG, which becomes nearly dense in high dimensions, the k -NN graph remains sparse. It can be efficiently built using methods like NN-Descent, which exhibits an empirical complexity of about $O(N^{1.14})$ [15]. The k -NN graph serves as the foundational structure for many well-known ANN search algorithms, including GNNS[25], IEH[32] and DPG[37].

Navigable Small-World (NSW) Models. Inspired by the small-world phenomenon[30], this family of methods constructs graphs with both short- and long-range links to facilitate efficient greedy routing[33]. The Navigable Small World (NSW) algorithm [38] pioneered a practical, data-agnostic approach to building these graphs. Hierarchical NSW (HNSW) algorithm [39] extended NSW with a hierarchical multi-layer structure that supports coarse-to-fine traversal—rapidly covering large distances in sparse upper layers and refining locally in the dense base layer. To date, HNSW remains one

Table 1: Notations used in this paper

Notation	Description
$\mathbb{R}^d$	The d -dimensional real vector space.
$V = \{v_1, v_2, \dots, v_n\}$	A dataset of n vectors in $\mathbb{R}^d$.
$d(x, y), \ x - y\ $	Euclidean distance between vectors x, y .
$\tilde{d}(x, y)$	Approximate Euclidean distance between x, y .
$G = (V, E)$	A Proximity Graph with vertices V and edges E .
$\mathcal{N}(v)$	The set of neighbors of node v in the graph G .
q	A query vector in $\mathbb{R}^d$.
$N_k(q)$	The exact top- k nearest neighbors of q .
$R_k(q)$	The results of top- k ANN search of q .
$C[i]$	The i -th nearest vector to q in the set C .
$C[1:l]$	The subset of C containing the l nearest vectors to q (or all of C if $ C < l$).

of the most effective and widely adopted ANN algorithms, though it does not provide formal accuracy guarantees.

RNG-Based Methods. RNG[47] is constructed by eliminating the longest edge in every triangle of points. This principle yields highly sparse graphs with a constant average degree[29]. While the pure RNG is not sufficient for effective navigation[14], it inspired the Monotonic RNG (MRNG)[18], a variant that guarantees finding the exact nearest neighbor for in-dataset queries via greedy search. However, this guarantee fails for out-of-dataset queries. To address this, algorithms like FANNG[27] and τ -Monotonic Graph (τ -MG) [41] were developed, which provide exact search guarantees for queries that are very close (within a radius τ) to the dataset, but reverting to heuristic behavior otherwise. LMG [49] extends the guarantee of τ -MG by preserving edges for a range of τ values. Other works such as Vamana[46] and SSG[17] have proposed heuristic modifications to the RNG rules to improve search performance.

Additionally, significant effort has been invested in optimizing ANN search for modern hardware, leading to specialized systems for SSDs [11, 45, 46] and GPUs [21, 50]. Unlike previous work, ours is the first to offer an error-bounded guarantee for top- k ($k > 1$) ANN search and achieves high QPS during searches.

3 Preliminaries

In this section, we formally define the nearest neighbor search problem and establish the notation used throughout this paper. Default notations are listed in Table 1.

3.1 Nearest Neighbor (NN) Search

Let $V = \{v_1, v_2, \dots, v_n\}$ be a dataset of n vectors in $\mathbb{R}^d$. The distance between any two vectors $x, y \in \mathbb{R}^d$ is measured by the Euclidean distance, denoted as $d(x, y) = \|x - y\|$.

DEFINITION 1 (TOP- k NN SEARCH). *Given a dataset V and a query vector $q \in \mathbb{R}^d$, the goal of top- k Nearest Neighbor (NN) search is to find a set $N_k(q) \subset V$ of size k such that for any vector $v \in N_k(q)$ and any vector $v' \in V \setminus N_k(q)$, the inequality $d(q, v) \leq d(q, v')$ holds. In cases of distance ties, the selection of $N_k(q)$ is arbitrary.*

For subsequent definitions, we treat $N_k(q)$ as a sequence $(v_{(1)}, v_{(2)}, \dots, v_{(k)})$ ordered by non-decreasing distance to q . Solving the exact k -NN problem via a linear scan has a computational cost of $O(nd)$, which is prohibitive for large-scale and high-dimensional

datasets. This has motivated the development of approximate solutions that offer a trade-off between search accuracy and efficiency.

3.2 Approximate Nearest Neighbor (ANN) Search

Due to the high computational cost of exact Nearest Neighbor search and the fact that real-world applications often do not require highly accurate k -nearest neighbors, most current research focuses on Approximate Nearest Neighbor (ANN) search. To facilitate performance comparisons, the field has largely standardized on an ϵ -Recall-Bounded evaluation paradigm, which can be defined as follows:

DEFINITION 2 (TOP- k ϵ -RECALL-BOUNDED ANN SEARCH). *Given a dataset V , a query vector $q \in \mathbb{R}^d$ and a parameter $\epsilon \in (0, 1)$, the goal of ϵ -Recall-Bounded ANN Search is to return a set of k vectors $R_k(q)$, that satisfies $\text{recall}(R_k(q)) = \frac{|R_k(q) \cap N_k(q)|}{|N_k(q)|} = \frac{|R_k(q) \cap N_k(q)|}{k}$, is at least $1 - \epsilon$.*

While the ϵ -Recall-Bounded metric is widely adopted in the literature [18, 24, 39], its fundamental drawback is its exclusive focus on retrieving true positives. It does not guarantee the error margin in cases of mismatches. In this paper, we focus on a novel approach called δ -Error-Bounded ANN search, which ensures that every returned result has an error bound relative to the ground truth neighbor.

DEFINITION 3 (TOP- k δ -ERROR-BOUNDED ANN SEARCH). *Given a dataset V , a query vector q , and an approximation factor $\delta \in (0, 1)$, the goal is to return a sequence of k vectors, $R_k(q) = (r_{(1)}, r_{(2)}, \dots, r_{(k)})$, ordered by non-decreasing distance to q , that satisfies the following condition for all $i \in \{1, \dots, k\}$: $d(q, r_{(i)}) \leq (1/\delta) \cdot d(q, v_{(i)})$.*

Our rank-aware definition is more practical than the stricter alternative of requiring $d(q, r_{(i)}) \leq (1/\delta) \cdot d(q, v_{(1)})$ for all i [17, 18], a condition that rapidly becomes infeasible as k increases.

Hardness of achieving δ -Error-bounded ANN Search. Achieving the guarantee in Definition 3 for arbitrary queries in high-dimensional space is a notoriously difficult problem. The core difficulty lies in the fact that most existing index structures are constructed based solely on the dataset V . Their geometric and navigational properties are therefore defined relative to the data points themselves, not for the entire continuous space $\mathbb{R}^d$.

Consequently, even the most advanced provable methods offer only limited guarantees, as their guarantees typically hinge on restrictive assumptions about the query’s location relative to the dataset [18, 27, 41]. They fall short of providing a universal error bound for an arbitrary 1-ANN query, let alone the rank-aware guarantee for k -ANN. Our work is primarily motivated by addressing this limitation.

4 How to Achieve δ -Error-bounded ANN Search

4.1 ANN Search on Proximity Graph

Graph-based ANN search methods construct a proximity graph over the dataset V , where the search is performed via graph traversal. Algorithm 1 outlines a common greedy strategy for this purpose.

The algorithm maintains a candidate set C and a set T of visited nodes. Vectors in C are sorted by their ascending distance to the query q . We use the notation $C[1 : l]$ to denote the top- l candidates

Algorithm 1 ANN search on a (Proximity) Graph

Require: (Proximity) Graph $G = (V, E)$, query vector q , result size k , start node $v_s \in V$, candidate set size l

Ensure: $R_k(q)$: a set of k approximate nearest neighbors of q

```

1: candidate set  $C \leftarrow \{v_s\}$ ; visited set  $T \leftarrow \emptyset$ ;
2: while  $\exists u \in C[1:l], u \notin T$  do
3:    $u \leftarrow \arg \min_{u \in C[1:l] \setminus T} d(q, u)$ ;
4:    $T \leftarrow T \cup \{u\}$ ;
5:   for all  $v \in N(u) \setminus T$  do
6:      $C \leftarrow C \cup \{v\}$ ;
7:   end for
8:   Prune  $C$  to retain the top  $l$  candidates closest to  $q$ ;
9: end while
10: return  $C[1:k]$ ;
```

in C . Correspondingly, $C[i]$ denotes the i -th candidate in this sorted list. The algorithm iteratively selects the closest unvisited node from candidate set C , adds its neighbors to the set, and prunes the set to maintain the top- l best candidates. This process terminates when all nodes within $C[1:l]$ have been visited.

This search algorithm is fundamentally a heuristic. Without a specially designed graph structure, there is no formal assurance that the search will not terminate in a poor local optimum, failing to find high-quality neighbors.

4.2 The Proximity Graph to be Monotonic

To prevent searches from terminating in such local optima, the graph itself must possess stronger navigational properties. A key property is monotonicity, which guarantees that for any destination node, a path of progressively closer nodes always exists. This concept was originally introduced as the Monotonic Search Network (MSNET) in [14]. To formalize this, we first define a monotonic path with respect to an arbitrary query.

DEFINITION 4 (MONOTONIC PATH). Given a directed graph $G = (V, E)$ defined on vector set $V \subset \mathbb{R}^d$, and a query $q \in \mathbb{R}^d$, a path $(v_1, v_2, \dots, v_k)$ in G is monotonic with respect to q if the distance to q strictly decreases at every step. That is, for all $i \in \{1, \dots, k-1\}$, the condition $d(q, v_i) > d(q, v_{i+1})$ holds.

Using this definition, we can now define a monotonic graph.

DEFINITION 5 (MONOTONIC GRAPH). A directed graph $G = (V, E)$ is a monotonic graph if for any two distinct nodes $u, v \in V$, there exists a monotonic path from u to v in G with respect to v .

DEFINITION 6 (MONOTONIC TOP-1 SEARCH). Given a directed graph $G = (V, E)$, query vector q and start node $v_s \in V$, a monotonic top-1 search iteratively moves from the current node v_t to its neighbor closest to q :

$$v_{t+1} = \arg \min_{u \in N(v_t)} d(q, u), \quad v_1 = v_s$$

where $N(v_t)$ denotes the neighbors of v_t . The process terminates when it reaches a node v_f that is closer to q than all of its neighbors. Such a node v_f is called a local optimum node with respect to q .

THEOREM 1. For any query $q \in V$, the monotonic top-1 search on a monotonic graph $G = (V, E)$ is guaranteed to terminate at q .

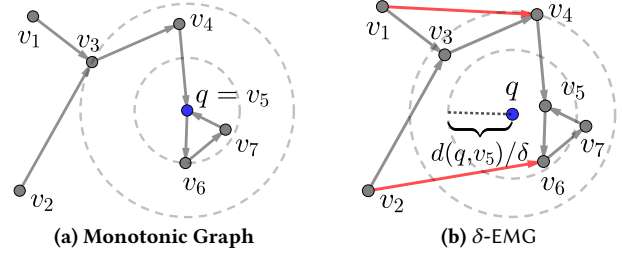


Figure 1: A comparison of Monotonic Graph and δ -EMG. $v_1, \dots, v_7$ are nodes in the graph and q is the query. The red edges indicate additional links introduced in δ -EMG compared to the Monotonic Graph.

PROOF. Assume the search terminates at node $v \neq q$. Since $q \in V$ and G is monotonic, there exists at least one monotonic path from v to q . Let u be the immediate successor of v on this path. By Definition 4, $d(q, u) < d(q, v)$, contradicting the termination condition that v is a local optimum node. Hence, the search must terminate at $v = q$. $\square$

EXAMPLE 1. Figure 1a illustrates the navigational property of a Monotonic Graph. The directed edges depict a subgraph that guarantees monotonic paths toward the destination node $q = v_5$. The definition of a monotonic graph guarantees that such a navigable structure exists between any pair of nodes. However, consider a case where $q \neq v_5$, as shown in Figure 1b. In this case, using only the edges in the Monotonic Graph (gray edges in the figure), monotonic searches starting from v_1, v_2 , or v_3 fail to reach v_5 because v_3 acts as a local optimum node.

Theorem 1 provides a strong guarantee of finding the exact nearest neighbor, but it holds only when the query is a dataset point and only for the 1-NN case. Real-world scenarios, however, involve arbitrary queries that are not in the dataset and often require finding the top- k neighbors.

This motivates the development of our framework, the δ -Error-Bounded Monotonic Graph (δ -EMG), which guarantees a top- k δ -error-bounded ANN search for arbitrary queries.

4.3 Error-Bounded Monotonic Graph (δ -EMG)

This chapter introduces our primary contribution: a novel theoretical framework and a corresponding graph structure that provide provable error bounds for any arbitrary query $q \in \mathbb{R}^d$. We first define the δ -Neighborhood for a query q below.

DEFINITION 7 (δ -NEIGHBORHOOD). Given a dataset V and a query q , let $v_{(1)} = \arg \min_{v \in V} d(q, v)$ be the true nearest neighbor of q . For a parameter $\delta \in (0, 1)$, the δ -neighborhood of q is defined as the closed ball centered at q with radius $d(q, v_{(1)})/\delta$:

$$\{x \in \mathbb{R}^d \mid d(q, x) \leq \frac{1}{\delta} \cdot d(q, v_{(1)})\}$$

Based on this definition, we generalize the concept of δ -error-bounded monotonic graph as follows.

DEFINITION 8 (δ -ERROR-BOUNDED MONOTONIC GRAPH). Given a dataset V and $\delta \in (0, 1)$. A proximity graph $G = (V, E)$ is a δ -error-bounded monotonic graph (δ -EMG) if for any query $q \in \mathbb{R}^d$ and any

starting node $v_s \in V$, there exists a monotonic path $(v_1, \dots, v_k)$ in G such that $v_1 = v_s$ and v_k lies within the δ -neighborhood of q .

EXAMPLE 2. Figure 1b illustrates this concept. The inner circle represents the δ -neighborhood of q . While most nodes (e.g., v_1, v_3, v_4, v_7) have monotonic paths directly to the true nearest neighbor v_5 , node v_2 has a monotonic path to v_6 . Since v_6 lies within the δ -neighborhood, the condition is satisfied for all nodes.

The navigational properties of a δ -EMG provide approximation guarantees for both top-1 ANN and top- k ANN search.

Error Bound for 1-NN: δ -EMG guarantees to find a $(1/\delta)$ -approximate nearest neighbor for any query.

THEOREM 2. For any query $q \in \mathbb{R}^d$, the monotonic top-1 search on a δ -EMG is guaranteed to return a node $r \in V$ that is a $(1/\delta)$ -approximation of the true nearest neighbor $v_{(1)}$:

$$d(q, r) \leq \frac{1}{\delta} \cdot d(q, v_{(1)})$$

PROOF. Let $r \in V$ be the node where the monotonic search terminates. By the Definition 6, r is a local optimum node, meaning it has no neighbor in G that is closer to r . Assume that the proposition is false, i.e., $d(q, r) > \frac{1}{\delta} \cdot d(q, v_{(1)})$. This would imply that r lies outside the δ -neighborhood of q . However, by the definition of a δ -EMG, for any node outside the δ -neighborhood, there must exist a monotonic path starting from it. The existence of such a path requires r to have a neighbor r' such that $d(q, r') < d(q, r)$. This directly contradicts the fact that r is a local optimum node. Therefore, the assumption must be false, and we must have $d(q, r) \leq \frac{1}{\delta} \cdot d(q, v_{(1)})$. $\square$

Error Bound for k -NN: For any query and our proposed search strategy in Section 5.2, the δ -EMG ensures a top- k δ -error-bounded ANN search when there is a locally optimum node in the candidate set. This condition is frequently met in our experiments in Sec.7.3. The detailed proof can be seen at Theorem 4 in Sec.5.2.

Based on the above properties, **the key challenge of achieving δ -error-bounded ANN search has now shifted to the problem of constructing a δ -EMG**. Below, we present the algorithms to construct this graph and to perform efficient ANN search upon it.

5 Algorithms for δ -EMG

The core challenge lies in constructing a graph that satisfies the δ -EMG property. A naive approach would require verifying an infinite set of queries in $\mathbb{R}^d$ to ensure a monotonic path exists into each query's δ -neighborhood, which is computationally intractable. The solution, therefore, is to translate this global guarantee into a practical, local rule that operates only on the finite data points in V . The following construction achieves this by introducing a carefully designed geometric occlusion condition.

5.1 Construction of δ -EMG

We begin with the central element of our construction: a unique edge occlusion rule that determines the graph's topology.

DEFINITION 9 (OCCLUSION REGION OF δ -EMG). For vectors $u, v \in \mathbb{R}^d$ and $\delta \in (0, 1)$, the Occlusion Region of δ -EMG is defined as:

$$\text{Occlusion}_\delta(u, v) = \{x \in \mathbb{R}^d \mid d(x, u) < d(u, v) \text{ and } d^2(x, v) + 2\delta \cdot d(u, v) \cdot d(x, u) < d^2(u, v)\}$$

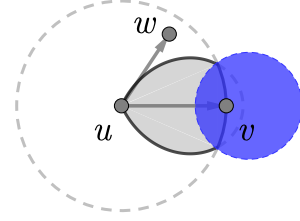


Figure 2: The construction rule for δ -EMG. The gray shaded area represents the Occlusion Region for (u, v) . For any query q inside blue shaded area, u is guaranteed to have a neighbor closer to q .

As illustrated in Figure 2, the occlusion region is the intersection of an open ball centered at u and a teardrop-shaped volume with its cusp at v . The parameter δ modulates its geometry: as $\delta \rightarrow 0$, the region expands towards the lune of an MRNG. Conversely, as $\delta \rightarrow 1$, the region contracts significantly.

This specific geometry induces a "navigable region" around v (the blue area in Figure 2). For any query q within this region, any vector w in $\text{Occlusion}_\delta(u, v)$ is guaranteed to be closer to q than u is. The following lemma formalizes this.

LEMMA 1. For any vectors u, v and any $w \in \text{Occlusion}_\delta(u, v)$, any query $q \in \mathbb{R}^d$ that satisfies the condition $d(q, v) < \delta \cdot d(q, u)$, the inequality $d(q, w) < d(q, u)$ holds.

PROOF. Without loss of generality, we can translate the coordinate system such that $u = \mathbf{0}$, where $\mathbf{0}$ is the zero vector. The inequality to be proven, $d(q, w) < d(q, u)$ is equivalent to $\|q - w\|^2 < \|q\|^2$, which reduces to showing:

$$2q \cdot w > \|w\|^2$$

First, we characterize the region containing the query q . The condition $d(q, v) < \delta \cdot d(q, u)$ becomes $\|q - v\| < \delta \|q\|$. Squaring both sides and expanding the expression yields:

$$(1 - \delta^2)\|q\|^2 - 2q \cdot v + \|v\|^2 < 0$$

For $\delta \in (0, 1)$, completing the square with respect to q gives:

$$\left\|q - \frac{v}{1 - \delta^2}\right\|^2 < \left(\frac{\delta \|v\|}{1 - \delta^2}\right)^2$$

This demonstrates that q must lie within an open hypersphere $B(c, R)$ with center $c = \frac{v}{1 - \delta^2}$ and radius $R = \frac{\delta \|v\|}{1 - \delta^2}$.

To prove that $2q \cdot w > \|w\|^2$ for all $q \in B(c, R)$, it is sufficient to show that the inequality holds for the minimum value of the linear function $2q \cdot w$ over the ball. This minimum is attained on the boundary of the ball in the direction opposite to w , resulting in:

$$\min_{q \in B(c, R)} (2q \cdot w) = 2(c \cdot w - R\|w\|) = \frac{2}{1 - \delta^2} (v \cdot w - \delta \|v\| \|w\|)$$

Next, we use the condition on w . Since $w \in \text{Occlusion}_\delta(\mathbf{0}, v)$, it must satisfy the second inequality from Definition 9:

$$\|w - v\|^2 + 2\delta \|v\| \|w\| < \|v\|^2$$

Expanding $\|w - v\|^2$ and simplifying this expression yields:

$$2(v \cdot w - \delta \|v\| \|w\|) > \|w\|^2$$

Finally, we substitute this result into our expression for the minimum of $2q \cdot w$:

$$\min_{q \in B(c, R)} (2q \cdot w) > \frac{\|w\|^2}{1 - \delta^2} \geq \|w\|^2$$

This confirms that $2q \cdot w > \|w\|^2$ holds for any valid query q in the navigable region, thus concluding the proof. $\square$

We now give the construction of δ -EMG using the occlusion region.

THEOREM 3. *Given a dataset $V \subset \mathbb{R}^d$ and $\delta \in (0, 1)$, a directed graph $G = (V, E)$ is δ -EMG if for any pair of nodes $(u, v) \notin E$, there exists an edge $(u, w) \in E$ such that $w \in \text{Occlusion}_\delta(u, v)$.*

PROOF. Let $v_{(1)} = \arg \min_{v \in V} d(q, v)$ be the true nearest neighbor of a given query q . Consider any node $u \in V$ that lies outside the δ -neighborhood of q , i.e., $d(q, u) > \frac{1}{\delta} \cdot d(q, v_{(1)})$.

Now, consider the pair of nodes $(u, v_{(1)})$. According to the construction rule, there are two mutually exclusive possibilities for the edge $(u, v_{(1)})$:

- (1) The edge $(u, v_{(1)})$ exists in E . In this case, $v_{(1)}$ is a neighbor of u . From the initial condition, we have $d(q, u) > \frac{1}{\delta} \cdot d(q, v_{(1)}) > d(q, v_{(1)})$. Thus, $v_{(1)}$ is strictly closer to q than u is, and $(u, v_{(1)})$ is a monotonic step.
- (2) The edge $(u, v_{(1)})$ does not exist in E . By theorem 3, its absence implies that it must be occluded by some other node w . This means there exists an edge $(u, w) \in E$ where $w \in \text{Occlusion}_\delta(u, v_{(1)})$. According to Lemma 1, we have $d(q, w) < d(q, u)$. Thus, u has a neighbor w that is strictly closer to q .

In both cases, any node u outside the δ -neighborhood is guaranteed to have a neighbor in G that is strictly closer to q . Therefore, any greedy search starting from u must proceed along a monotonic path and cannot stop at u . Since the dataset V is finite and the distance to q strictly decreases at each step, this path must terminate. As it cannot terminate outside the δ -neighborhood, it must eventually enter it. This satisfies the definition of a δ -EMG. $\square$

We analyze the expected out-degree of δ -EMG, which is a key measure of the graph's structural complexity.

LEMMA 2. *For a set V of n vectors sampled uniformly from a bounded region in $\mathbb{R}^d$, the expected out-degree of any vertex in the corresponding δ -EMG is $O(\ln n)$.*

PROOF. We assume that the dataset V consists of n points uniformly distributed in a bounded region of $\mathbb{R}^d$, with a constant point density ρ . Let $P(r)$ denote the probability that an edge (u, v) exists between two points $u, v \in V$ at distance $r = d(u, v)$. Since the distribution is uniform and isotropic, $P(r)$ depends only on the distance r .

By Theorem 3, an edge (u, v) exists only if no other vertex w creates an edge (u, w) that occludes it, i.e. $w \in \text{Occlusion}_\delta(u, v)$. For convenience, denote the occlusion region by

$$\Omega_\delta(u, v) = \text{Occlusion}_\delta(u, v)$$

The probability that (u, v) survives occlusion can be written as

$$P(r) = \mathbb{E} \left[\prod_{w \in V \cap \Omega_\delta(u, v)} (1 - P(d(u, w))) \right]$$

Since the occlusion region expands as r increases, it becomes increasingly likely to contain occluding vertices. Therefore, $P(r)$ is a monotonically decreasing function of r .

For any $w \in \Omega_\delta(u, v)$, by the definition of the occlusion region we have

$$d^2(w, v) + 2\delta d(u, v) d(w, u) < d^2(u, v).$$

Combining this with the triangle inequality $d(w, v) \geq |d(u, v) - d(w, u)|$, and noting that $d(w, u) > 0$ since there are no duplicate points, we obtain

$$d(u, w) < 2(1 - \delta) d(u, v)$$

Let $c = 2(1 - \delta)$. Since $P(\cdot)$ is decreasing, for any such $w \in \Omega_\delta(u, v)$,

$$P(d(u, w)) > P(c d(u, v)) = P(cr)$$

Hence,

$$P(r) < \mathbb{E} \left[\prod_{w \in V \cap \Omega_\delta(u, v)} (1 - P(cr)) \right]$$

Let the volume of the occlusion region be $\text{Vol}(\Omega_\delta(u, v)) = C_{\delta, d} r^d$ for some constant $C_{\delta, d} > 0$ depending only on δ and d . Under a Poisson point process approximation, the number of vertices inside the occlusion region follows a Poisson distribution with rate parameter

$$\lambda = \rho \cdot \text{Vol}(\Omega_\delta(u, v)) \cdot P(cr) = \rho C_{\delta, d} r^d P(cr)$$

The probability that this region is empty of occluding vertices is approximately $e^{-\lambda}$, giving

$$P(r) < e^{-K r^d P(cr)},$$

where $K = \rho C_{\delta, d} > 0$

We now show that the recursive inequality above implies that $P(r)$ decays at least as fast as r^{-d} . Suppose, for contradiction, that there exist constants $\epsilon, \beta > 0$ such that

$$P(r) > \beta r^{\epsilon - d}$$

for all sufficiently large r . Substituting this into the inequality gives

$$\beta r^{\epsilon - d} < P(r) < e^{-K r^d P(cr)} < e^{-K r^d \beta (cr)^{\epsilon - d}} = e^{-K' r^\epsilon},$$

where $K' = K \beta c^{\epsilon - d} > 0$. However, as $r \rightarrow \infty$, the polynomial term $\beta r^{\epsilon - d}$ decays much more slowly than the super-exponential term $e^{-K' r^\epsilon}$, which yields a contradiction. Therefore, $P(r)$ must decay at least as fast as $O(r^{-d})$.

The expected out-degree of a vertex u is

$$\mathbb{E}[\deg^+(u)] \approx \int \rho P(r) dV = \int_{r_{\min}}^{r_{\max}} \rho P(r) S_{d-1} r^{d-1} dr,$$

where S_{d-1} denotes the surface area of the unit $(d - 1)$ -sphere. Using $P(r) = O(r^{-d})$, we obtain

$$\mathbb{E}[\deg^+(u)] \leq O \left(\int_{r_{\min}}^{r_{\max}} r^{-d} r^{d-1} dr \right) = O \left(\ln r \Big|_{r_{\min}}^{r_{\max}} \right)$$

For a dataset of n uniformly distributed points with fixed density ρ , the dataset volume scales as $O(n)$, so the maximum inter-point

Algorithm 2 Construct δ -EMG

Require: Vector set V , parameter $\delta \in (0, 1)$
Ensure: A δ -EMG $G = (V, E)$

- 1: Initialize edge set $E \leftarrow \emptyset$
- 2: **for all** node $u \in V$ **do**
- 3: $\mathcal{N}(u) \leftarrow \text{SELECTNEIGHBORS}(u, V, \delta)$
- 4: $E \leftarrow E \cup \{(u, v) \mid v \in \mathcal{N}(u)\}$
- 5: **end for**
- 6: **return** $G = (V, E)$

- 7: **function** $\text{SELECTNEIGHBORS}(u, V, \delta)$
- 8: $\mathcal{N}(u) \leftarrow \emptyset$
- 9: **for all** $v \in V$ sorted by increasing distance from u **do**
- 10: **if** $\nexists w \in \mathcal{N}(u)$ and $w \in \text{Occlusion}_\delta(u, v)$ **then**
- 11: $\mathcal{N}(u) \leftarrow \mathcal{N}(u) \cup \{v\}$
- 12: **end if**
- 13: **end for**
- 14: **return** $\mathcal{N}(u)$
- 15: **end function**

distance satisfies $r_{\max} = O(n^{1/d})$, while the minimum distance $r_{\min} = O(1)$. Substituting these bounds yields

$$\mathbb{E}[\deg^+(u)] = O(\ln r_{\max} - \ln r_{\min}) = O(\ln n).$$

Thus, the expected out-degree of any vertex in a δ -EMG constructed from uniformly sampled points in $\mathbb{R}^d$ is $O(\ln n)$. $\square$

Algorithm 2 provides a direct, exact implementation of the construction principle from Theorem 3. For each node u , it considers all other nodes as potential neighbors, sorted by distance. It then iteratively adds an edge (u, v) only if v is not occluded by any of the shorter, already-accepted neighbors of u .

Complexity of Algorithm 2. Recall that Lemma 2 guarantees that δ -EMG has an expected out-degree of $O(\ln n)$. Algorithm 2 requires space of $O(n \ln n)$. Algorithm 2 requires examining all pairs of points to test occlusion relationships. Consequently, its overall time complexity remains $O(n^2 \ln n)$.

5.2 ANN Search on δ -EMG

While the δ -EMG provides a foundational top-1 ANN guarantee, practical applications demand both the retrieval of k nearest neighbors and the ability to specify a more stringent error bound at query time. The standard approach is to employ greedy search with a heuristically chosen candidate set size $l > k$, offering no formal connection between l and the resulting error.

To address this, we introduce an error-bounded k -ANN search algorithm that operates on a δ -EMG, presented as Algorithm 3. Rather than using a fixed candidate set size, our algorithm adaptively expands the candidate set size l , starting from $l = k$. The search terminates when the distance to the k -th candidate is sufficiently close to the query relative to the current search frontier. This termination is governed by a user-specified parameter $\alpha \geq 1$.

The formal guarantee of this algorithm depends on the discovery of a local minimum during the search (defined as a node with no neighbors closer to q). Let $R_k(q) = (r_{(1)}, \dots, r_{(k)})$ be the result

Algorithm 3 Error-Bounded top- k ANN Search on δ -EMG

Require: δ -EMG Graph $G = (V, E)$, query q , start node $v_s \in V$, result size k , accuracy parameter $\alpha \geq 1$
Ensure: $R_k(q)$: k approximate nearest neighbors of q

- 1: candidate set $C \leftarrow \{v_s\}$, visited set $T \leftarrow \emptyset$
- 2: **for** candidate set size $l = k, k+1, k+2, \dots$ **do**
- 3: **while** $\exists u \in C[1:l], u \notin T$ **do**
- 4: $u \leftarrow \arg \min_{u \in C[1:l] \setminus T} d(q, u)$
- 5: $T \leftarrow T \cup u$
- 6: **for all** $v \in \mathcal{N}(u) \setminus T$ **do**
- 7: $C \leftarrow C \cup \{v\}$
- 8: **end for**
- 9: keep top $l+1$ candidates in C in ascending distance
- 10: **end while**
- 11: **if** $d(q, C[l]) \geq \alpha \cdot d(q, C[k])$ **then break**
- 12: **end for**
- 13: **return** $C[1:k]$

returned by the algorithm, and let $N_k(q) = (v_{(1)}, \dots, v_{(k)})$ be the true k nearest neighbors, both ordered by distance to q .

THEOREM 4. Let C be the final candidate set upon termination of Algorithm 3, and let $R_k(q)$ be the returned result. If $\exists u \in C \setminus R_k(q)$ and u is a local optimum node, then $R_k(q)$ satisfies the following error bound for all $i \in \{1, \dots, k\}$:

$$d(q, r_{(i)}) \leq \frac{1}{\delta'} \cdot d(q, v_{(i)}) \quad \text{where} \quad \delta' = \delta \cdot \frac{d(q, u)}{d(q, r_{(k)})}$$

PROOF. Since u is a local optimum node, it has no neighbor closer to q . By the property of the δ -EMG in Theorem 2, it must satisfy $\delta \cdot d(q, u) \leq d(q, v_{(1)})$. From the definition of δ' in the theorem statement, we have $\delta \cdot d(q, u) = \delta' \cdot d(q, r_{(k)})$. Substituting this into the previous inequality gives:

$$\delta' \cdot d(q, r_{(k)}) \leq d(q, v_{(1)})$$

We know that for any $i \in \{1, \dots, k\}$, $d(q, r_{(i)}) \leq d(q, r_{(k)})$ and $d(q, v_{(1)}) \leq d(q, v_{(i)})$. Combining these inequalities yields the full chain:

$$\delta' \cdot d(q, r_{(i)}) \leq \delta' \cdot d(q, r_{(k)}) \leq d(q, v_{(1)}) \leq d(q, v_{(i)})$$

This proves the theorem. $\square$

The effective error bound δ' , as derived from Theorem 4, is determined after the search process. Its value depends on whether there exists a local optimum u in the final candidate set C . This condition is reasonable because the sparsity of high-dimensional spaces makes local optima statistically likely to exist.

Note that, the parameter α provided by the user directly influences the search accuracy. A larger α enforces a stricter stopping criterion, which requires the algorithm to expand its candidate set size. This wider search increases the chances of finding a local optimum u farther from the query, leading to a stronger approximation guarantee. Thus, the quality of the approximation is closely tied to the choice of α .

Exp-6 and 7 in Sec. 7.3 demonstrate how α impacts the achieved error bound δ' . The results show that if $\alpha > 2$, we can find such a

local optimum node with a probability of over 95%, and in this case, the algorithm is theoretically guaranteed to be error-bounded.

6 Algorithms for Quantized δ -EMG (δ -EMQG)

While the exact δ -EMG provides a robust theoretical foundation, its direct construction is computationally intractable and likely to cause a highly non-uniform degree distribution. Nodes in dense regions accumulate excessive edges, leading to high search overhead, which is a common challenge for many proximity graph methods.

To overcome these limitations, we introduce an approximate construction of δ -EMG. The core insight is to differentiate the roles of edges based on their length: long-range edges for coarse navigation can have relaxed guarantees, while short-range edges for fine-grained convergence must be robust. This motivates a principled relaxation where the parameter δ becomes an adaptive function of the edge length for any $u, v \in V$:

$$\delta_t(u, v) = 1 - \frac{d(u, v)}{d(u, v_{(t)})}$$

where $v_{(t)}$ is the t -th nearest neighbor of u in V .

This adaptive definition creates a multi-scale graph structure. For long-range edges where $d(u, v) > d(u, v_{(t)})$, $\delta(u, v)$ becomes negative, the deterministic guarantee is relaxed in favor of probabilistic progress. As the search enters a query's neighborhood, the value of $\delta(u, v)$ increases towards 1. This approximation thus locally recovers the strong navigable properties of a δ -EMG with high- δ .

Practically, it provides two major benefits:

- (1) **Locality.** Long-range edges are pruned early, so each node only explores a small local subset during construction, reducing complexity to near-linear.
- (2) **Degree Balancing.** Dense regions prune more aggressively, preventing degree explosion; sparse regions retain enough edges for connectivity.

The practical approximate construction of a δ -EMG, detailed in Algorithm 4, is an iterative process that refines an initial bootstrap graph (e.g., an approximate k-NN graph). Each iteration rebuilds the graph by first using beam search to generate local candidates for every node, and then applying our adaptive occlusion rule to prune these candidates into the final neighbor sets. This process typically converges to a high-quality graph within 3-4 iterations.

Complexity of Algorithm 4. The dominant computational cost of Algorithm 4 lies in lines 6 and 7. Line 6 performs a greedy search on the current graph to obtain a local candidate set. According to the analysis in [18], the complexity of such a search is approximately $O(L n^{1/d} \ln(n^{1/d})/\Delta)$, where Δ is the smallest distance between any two distinct points in V . Line 7 processes at most L candidates and checks occlusion against at most M accepted neighbors, yielding a per-node complexity of $O(LM)$. Since both M and L are small constants relative to n , the overall time complexity per iteration is $O(Ln^{(d+1)/d} \ln(n^{1/d})/\Delta)$. Considering space complexity, since the out-degree of each node is hard-capped at a constant M , the space complexity of Algorithm 4 is $O(Mn) = O(n)$.

Algorithm 4 Approximate Construction of δ -EMG

Require: Vector set V , max out-degree M , candidate set size L , neighborhood-scale parameter $t \leq L$, iterations I

Ensure: An Approximate δ -EMG $G = (V, E)$

```

1: Let  $v_s$  be the approximate medoid of  $V$ .
2: Initialize  $G = (V, E)$  from a top- $M$  approximate NN graph.
3: for iter from 1 to  $I$  do
4:   New edge set  $E_{new} \leftarrow \emptyset$ 
5:   for all  $u \in V$  do
6:      $R_u \leftarrow \text{GREEDYSEARCH}(G, v_s, u, L, L)$ 
7:      $\mathcal{N}(u) \leftarrow \text{LOCALLYSELECTNEIGHBORS}(u, R_u, t)$ 
8:     if  $|\mathcal{N}(u)| > M$  then
9:        $\mathcal{N}(u) \leftarrow$  the  $M$  closest nodes in  $\mathcal{N}(u)$ 
10:    end if
11:     $E_{new} \leftarrow E_{new} \cup \{(u, v) \mid v \in \mathcal{N}(u)\}$ 
12:  end for
13:   $G \leftarrow (V, E_{new})$ 
14:  Add reverse edges to  $G$  within the degree  $M$ .
15:  Connect any nodes unreachable from  $v_s$  to their nearest
    reachable neighbors, subject to the degree limit  $M$ .
16: end for

17: function  $\text{LOCALLYSELECTNEIGHBORS}(u, R_u, t)$ 
18:    $\mathcal{N}(u) \leftarrow \emptyset$ 
19:   Let  $r_{(t)}$  be the  $t$ -th closest node to  $u$  in  $R_u$ 
20:   for all  $r \in R_u$  sorted by increasing distance from  $u$  do
21:      $\delta \leftarrow 1 - \frac{d(u, r)}{d(u, r_{(t)})}$ 
22:     if  $\nexists w \in \mathcal{N}(u)$  and  $w \in \text{Occlusion}_\delta(u, r)$  then
23:        $\mathcal{N}(u) \leftarrow \mathcal{N}(u) \cup \{r\}$ 
24:     end if
25:   end for
26:   return  $\mathcal{N}(u)$ 
27: end function
28: return  $G = (V, E)$ 

```

6.1 Construction of δ -EMQG

Building upon the proposed approximate construction framework, we further incorporate vector quantization to reduce the cost of distance computations during ANN search. Specifically, we adopt the RaBitQ quantization scheme [20], as it provides an unbiased distance estimator with a rigorous theoretical error bound. Moreover, RaBitQ achieves an exceptionally high compression ratio, reducing vectors to as little as one bit per dimension. While prior work [24] has successfully integrated RaBitQ into a graph-based index, their approach relies on a heuristic NSG-like structure, lacking the geometric guarantees offered by our δ -EMG formulation.

The estimation of approximate distances from RaBitQ codes is performed using FastScan [1, 2] that leverages SIMD instructions. FastScan processes vectors in fixed-size batches, typically a multiple of the SIMD width (e.g., 32). Consequently, if the number of a node's neighbors is not a multiple of this batch size, the final batch incurs wasted computational cycles.

To ensure perfect alignment with FastScan's operational model, we first set the maximum out-degree M to a multiple of the SIMD batch size. Recall that the parameter t in Algorithm 4 monotonically

Algorithm 5 Probing top- k ANN Search on δ -EMQG

Require: δ -EMQG $G = (V, E)$, query vector q , start node $v_s \in V$, result size k , accuracy parameter $\alpha \geq 1$

Ensure: $R_k(q)$: k approximate nearest neighbors of q

```

1: candidate set  $C_e \leftarrow \{v_s\}$ ,  $C_a \leftarrow \emptyset$ , visited set  $T_e \leftarrow \emptyset$ ,  $T_a \leftarrow \emptyset$ 
2:  $d_{last} \leftarrow d(q, v_s)$ 
3: for candidate set size  $l = k, k + 1, k + 2, \dots$  do
4:   while true do
5:      $u \leftarrow \arg \min_{u \in C_e[1:l] \setminus T_e} d(q, u)$ 
6:      $w \leftarrow \arg \min_{w \in C_a[1:l] \setminus T_a} \tilde{d}(q, w)$ 
7:     if  $u$  is null and  $w$  is null then break
8:     else if  $\text{NEEDPROBING}(u, w, d_{last})$  then
9:       Compute exact distance  $d(q, w)$ 
10:       $C_e \leftarrow C_e \cup \{w\}$ ,  $T_a \leftarrow T_a \cup \{w\}$ 
11:      maintain  $C_e$  in ascending exact distance
12:    else
13:       $d_{last} \leftarrow d(q, u)$ 
14:      Compute approx. distances  $\tilde{d}(q, v)$  for all  $v \in \mathcal{N}(u)$ 
15:       $C_a \leftarrow C_a \cup \mathcal{N}(u) \setminus T_a$ ,  $T_e \leftarrow T_e \cup \{u\}$ 
16:      maintain  $C_a$  in ascending approximate distance
17:    end if
18:  end while
19:  if  $d(q, C_e[l]) \geq \alpha \cdot d(q, C_e[k])$  then break
20: end for
21: return  $C_e[1:k]$ 

```

```

22: function  $\text{NEEDPROBING}(u, w, d_{last})$ 
23:   if  $u$  is null then return true
24:   else if  $d(q, u) > d_{last}$  and  $w$  is not null and  $\tilde{d}(q, w) <$ 
      $d(q, u)$  then
25:     return true
26:   else
27:     return false
28:   end if
29: end function

```

influences the resulting neighborhood size: a larger t expands the local distance scale $d(u, v_{(t)})$, thereby relaxing the pruning condition and yielding more neighbors. During construction, for any node whose initial neighbor set (pruned with a default t) is smaller than M , we perform a binary search on the parameter t within the candidate range $[1, L]$ to find the smallest value that produces a neighborhood of exactly size M .

The overall construction of δ -EMQG follows the iterative framework outlined in Algorithm 4. Specifically, the neighborhood alignment step takes place after global connectivity has been established. Subsequently, for each node, we compute and store the RaBitQ codes for its entire neighborhood.

6.2 Quantized Search on δ -EMQG

Searching on a quantized graph introduces a critical trade-off: relying solely on approximate distances for navigation compromises accuracy, whereas frequent recourse to exact distance computations negates the performance benefits of quantization. To resolve this, we introduce the **Probing Search** algorithm (Algorithm 5). This

method seamlessly integrates the speed of quantized exploration with the accuracy of exact verification, while maintaining the adaptive structure of Algorithm 3 with a dynamic increment candidate set size.

The algorithm maintains two candidate sets:

- (1) Exact Candidate Set (C_e): It stores candidates whose exact distances $d(q, \cdot)$ have been computed. It serves as the basis for the final result and the termination condition.
- (2) Approximate Candidate Set (C_a): It is a set of candidates discovered during graph traversal, ordered by their approximate distances $\tilde{d}(q, \cdot)$.

The search dynamically alternates between two operations: **Expansion** and **Probing**. The decision logic, which depends on the best unvisited candidates from both sets ($u \in C_e[1:l]$ and $w \in C_a[1:l]$) and the distance of the last expanded node (d_{last}), is designed to minimize exact distance computations by invoking them only when the search encounters a potential local optimum.

- (1) Expansion: This is the default operation. The algorithm expands from the most promising exact candidate u by retrieving its neighbors, computing their approximate distances in a batch via FastScan, and inserting them into C_a .
- (2) Probing: This is the operation when the search using exact distances stops improving—where u is farther from the query than d_{last} —and the candidate w from the approximate set looks better. The algorithm then "probes" this candidate w by computing its exact distance and promoting it to C_e .

7 Experiments

In this section, we conduct extensive experiments to evaluate the proposed algorithms. We implement seven different algorithms for comparison:

- (i) NSG[18]. We use an efficient implementation of NSG provided in the open-source *Glass* library.
- (ii) HNSW[39]. We also use the implementation from the *Glass* library for consistency.
- (iii) τ -MNG[41]. Since no official implementation is available, we reimplemented τ -MG based on the NSG source code.
- (iv) NGT-QG[28]. NGT-QG is a quantized graph method developed in the open-source NGT library. It integrates Product Quantization with a proximity graph.
- (v) SymphonyQG[24]. SymphonyQG represents the latest advancement in quantized graph. We use the official open-source implementation released by the authors.
- (vi) δ -EMG. δ -EMG is our proposed error-bounded monotonic graph framework. The index is constructed using Algorithm 4, and queries are performed using the error-bounded k-ANN search (Algorithm 3).
- (vii) δ -EMQG. δ -EMQG extends δ -EMG by integrating RaBitQ-based vector quantization. It employs the Probing Search (Algorithm 5) for queries.

Datasets. Our evaluation is conducted on six widely-used real-world datasets that span diverse modalities, dimensionalities, and data scales: SIFT1M, SIFT50M, GIST, MSong, Crawl, and Deep1M. These datasets are standard benchmarks in ANN research and have been adopted in prior studies [5, 18, 24, 27, 39, 48] to evaluate both accuracy and efficiency. Specifically, SIFT1M and SIFT50M consist

Table 2: Statistics of datasets

Dataset	dimension	# of base	# of queries	LID
SIFT1M	128	1,000,000	10,000	9.3
SIFT50M	128	50,000,000	10,000	9.3
GIST	960	1,000,000	1,000	18.9
MSong	420	992,272	200	9.5
Crawl	300	1,989,995	10,000	15.7
Deep1M	256	1,000,000	1,000	12.1

of 128-dimensional image descriptors; GIST contains holistic image features; MSong includes audio features; Crawl comprises word embeddings; and Deep1M contains deep visual features. Table 2 summarizes their key statistics, including the *Local Intrinsic Dimensionality* (LID), which measures the local geometric complexity of each dataset. Datasets with higher LID values are generally harder for ANN search.

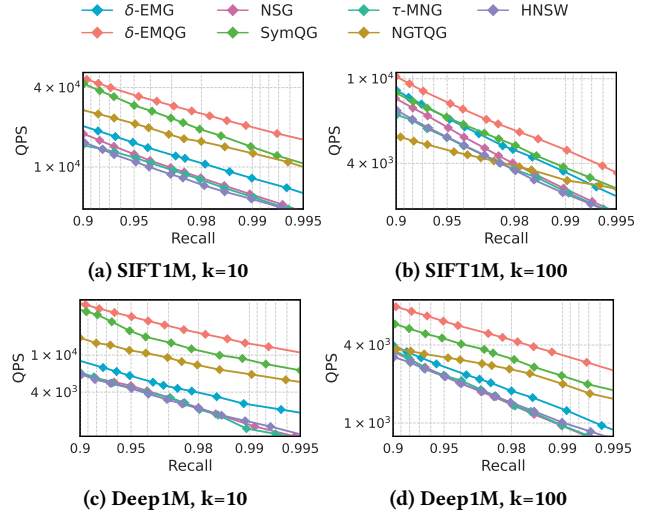
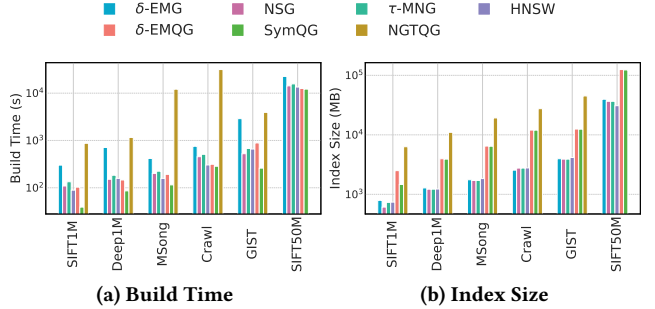
Parameter settings. For the baseline methods, construction parameters were set according to the recommendations in prior studies [18, 24, 39, 41]. For our proposed δ -EMG and δ -EMQG, we fixed the candidate set size at $L = 1000$ and the number of refinement iterations at 3 across all datasets. The maximum out-degree M was generally set to 64. M was reduced to 32 for δ -EMG on MSong and Crawl, and for δ -EMQG on SIFT50M, as this value is sufficient for achieving high performance on these datasets.

At query time, we varied the primary search-time parameters for each algorithm to obtain its performance profile across different accuracy levels. For NSG, HNSW, τ -MG, and SymphonyQG, this corresponds to adjusting the search candidate set size. The search procedure for NGT-QG involves two key parameters, the search radius and reranking size; we varied combinations of these two parameters to generate its accuracy-efficiency curve. For our δ -EMG and δ -EMQG, we varied the accuracy parameter α .

Experimental settings. All experiments were conducted on a server equipped with an AMD Ryzen Threadripper 3990X 64-Core Processor and 320 GB of RAM, running a Linux 4.4 kernel. The processor supports the AVX2 instruction set, which is leveraged by methods employing SIMD acceleration. All algorithms were implemented in C++ and compiled with GCC 13.3 using the ‘-Ofast’ optimization flag. Index construction for all methods was parallelized using 32 threads. All search performance was measured on a single thread. Each reported Queries-Per-Second (QPS) value is the average of five runs.

7.1 Performance V.S. the Baselines

Exp-1. QPS Comparison with the Baselines. We first compare the Queries-Per-Second (QPS) against recall for all methods across the six datasets. For each method, its build-time parameters were fixed according to the parameter settings described earlier, while its search-time parameters were varied to generate the QPS-Recall curves. We tested with $k = 1, 10, 100$ on all datasets, targeting a recall range of 0.9 to 0.995. Figure 3 presents the comprehensive results. The complete results for all datasets are provided in the appendix (Figure A.1). Both axes follow the logarithmic scaling used in ANN-Benchmarks [5], with the x-axis logarithmic in $1 - \text{recall}$ to emphasize the high-recall region. Curves that are higher and to the right are better. We note that NGT-QG results are missing for SIFT50M because it ran out of our memory.

**Figure 3: QPS vs Recall****Figure 4: Construction Cost**

The results clearly show that δ -EMQG outperforms all baseline methods across all datasets, k -values, and recall levels. At 99% recall with $k = 1$, δ -EMQG is 1.2x to 3.2x faster than the best baseline. For $k = 10$ and $k = 100$, this lead is between 1.2x and 2.1x. Our non-quantized method, δ -EMG, also exhibits highly competitive performance. It surpasses all other non-quantized baselines (HNSW, NSG, τ -MG) in every tested configuration. For instance, at 99% recall with $k = 1$, δ -EMG is 2.1x to 2.3x faster than the best non-quantized baseline. Notably, at high recall regimes on the SIFT1M, SIFT50M, and MSong datasets, δ -EMG is even faster than the quantized baselines NGT-QG and SymphonyQG.

Note that our algorithm delivers the best performance across all parameters and all datasets. For the complete results, please refer to the supplementary material.

Exp-2. Index Construction Comparison with the Baselines. We further analyze the index construction time and the final index size in memory. The index size includes the adjacency lists, vector storage, and quantization codes (for quantized methods). Figure 4 shows these results.

Overall, our proposed methods exhibit moderate index building costs in both time and space. The construction time for δ -EMQG is similar to the fastest baselines, HNSW and SymphonyQG. δ -EMG is slightly slower to build but still remains faster than NGT-QG.

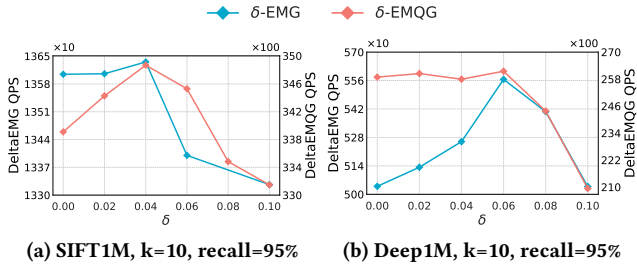


Figure 5: Effect of δ

In terms of index size, δ -EMQG has nearly the same size as SymphonyQG and smaller than NGT-QG. Meanwhile, the index size for δ -EMG is on par with the most space-efficient baselines like NSG and τ -MG. This demonstrates that the superior search performance of our frameworks does not come at the cost of excessive indexing overhead.

7.2 Effect of Construction Parameters

Exp-3. Effect of δ . We investigated the impact of δ on search performance. Since the exact construction of a δ -EMG (Algorithm 2) is computationally prohibitive, we adopt the practical construction framework (Algorithm 4) with a fixed global δ throughout the entire build process, instead of using an adaptive value.

To isolate the effect of δ , we varied its value while keeping all other construction parameters unchanged. For each resulting graph, we measured QPS at 95% recall with $k = 10$. The results are presented in Figure 5. Complete figures across all datasets are in the appendix (Figure A.2).

The results show that QPS initially increases as δ grows, reaches a peak, and then gradually declines. The optimal performance is typically observed when δ lies between 0.04 and 0.06. The best δ for δ -EMG and δ -EMQG is nearly the same. This finding suggests that introducing a small but non-zero δ improves the structural connectivity of the graph and facilitates more efficient navigation during search. However, excessively large δ values lead to overly dense graphs, which in turn increase the traversal cost and reduce query efficiency.

Exp-4. Effect of t . In our practical construction method, the parameter t controls our adaptive δ rule, $\delta_t(u, v) = 1 - d(u, v)/d(u, v_{(t)})$, which is designed to balance graph density and search performance. We tested how different values of t affect the QPS of both δ -EMG and δ -EMQG.

We built indexes with various t values while keeping other parameters constant, and then measured the QPS at 95% recall with $k = 10$. As shown in Figure 6 (see Figure A.3 in appendix for full results), despite minor fluctuations, the QPS for both methods generally rises to a distinct peak and then declines as t increases. We also noted that the optimal t can differ for the quantized and non-quantized methods. This is likely attributable to the degree alignment procedure in the δ -EMQG construction, where t is locally adjusted for nodes to ensure their out-degree is a multiple of the SIMD batch size. This additional constraint naturally shifts the optimal value for the t parameter.

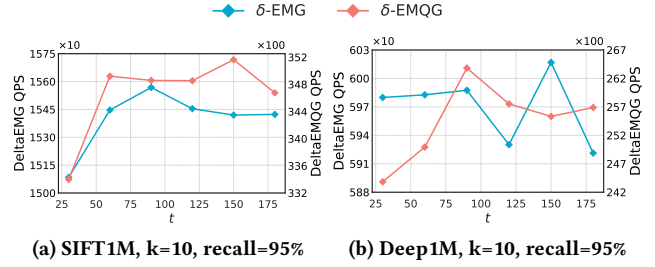


Figure 6: Effect of t

Crucially, the peak QPS achieved with the best adaptive t was higher than the best performance attainable with any fixed δ in the previous experiment. This confirms that our adaptive rule is a more effective approach for building high-performance search graphs.

7.3 Validating the Error-Bounded Framework

Exp-5. Analyzing Practical Search Error. Here, we test the practical impact of our error-bounded design. To quantify the precision of search results beyond recall, we employ the Relative Distance Error, which is defined as the average error $(d(q, r_{(i)}) - d(q, v_{(i)}))/d(q, v_{(i)})$ across all queries and returned vectors. This metric is intrinsically linked to our theoretical foundation; the result from δ -Error-bounded ANN search guarantees a Relative Distance Error of less than $(1/\delta) - 1$. To provide a fair comparison of algorithmic efficiency, independent of implementation-specific factors like memory layout or caching, we use the average number of distance computations per query as the primary performance metric. This metric directly reflects the core workload of an ANN search. Our analysis focuses on a comparison between δ -EMG and other non-quantized baselines. We exclude quantized methods from this specific experiment because their performance is governed by a complex trade-off between approximate distance calculations and exact ones, which would obscure the fundamental graph navigation efficiency we aim to measure.

Figure 7 illustrates the performance curves on different datasets. Full results are in the appendix (Figure A.4). We focus our analysis on the high-precision region (Relative Distance Error < 0.005), which typically corresponds to recall rates exceeding 95%. The results show that δ -EMG consistently requires fewer distance computations to achieve the same Relative Distance Error compared to all baselines. For example, at an error of 0.001, δ -EMG needs only 50% to 80% of the computations of the next best method. This means the geometric guarantees of δ -EMG make the search process inherently more efficient. It finds accurate neighbors by exploring a smaller portion of the graph, which directly explains the higher QPS we observed in earlier experiments.

Exp-6. Probability of Finding a Local Optimum node. Our theoretical guarantee for k -ANN search (Theorem 4) depends on finding a "local optimum node" (a node with no closer neighbors) within the final search candidate $C[k : l]$. In this experiment, we test how often this happens.

To clearly validate the theory and analyze the relationship between δ and its derived δ' , we constructed δ -EMG graphs using the optimal fixed δ for each dataset identified in Exp-3 (e.g., $\delta = 0.04$ for Crawl, GIST, SIFT1M, and MSong; $\delta = 0.06$ for Deep1M). We then

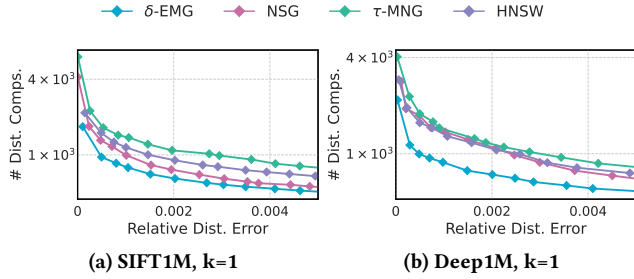


Figure 7: Distance Error

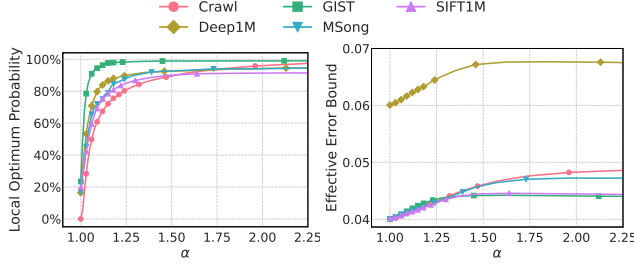


Figure 8: Empirical validation of the ANN error guarantee

performed searches with $k = 10$ and varied the search parameter α . For each value of α , we measured the empirical probability that at least one local optimum exists within the candidate set $C[k : l]$ at the moment of termination. As shown in Figure 8a, this probability quickly rises with α , approaching 95% for α around 2.0. This confirms that the condition for our theoretical guarantee is almost always met in practice.

Exp-7. Analyzing the Theoretical Error Bound δ' . When a local optimum node is found, the achieved bound is given by $\delta' = \delta \cdot d(q, u) / d(q, r_{(k)})$, where u is the discovered local optimum. A larger δ' corresponds to a tighter error guarantee. Using the same experimental setup as above, we measured the average δ' achieved for queries where a local optimum node was found. The results are presented in Figure 8b.

The results show that the average δ' increases with α and then levels off. This is because a larger α leads to a wider search, which tends to find local optimum nodes that are farther away, improving the bound.

Importantly, the achieved δ' is always better than the base δ used to build the graph. For graphs built with $\delta = 0.04$, we achieved a δ' of about 0.045–0.050. For the graph with $\delta = 0.06$, we achieved a δ' of 0.068. This shows our method delivers a practical error guarantee that is even stronger than the one used for construction.

7.4 Scalability and Ablation Experiments

Exp-8. Scalability Studies. In this section, we evaluate how our methods perform as the dataset size increases. We evaluated δ -EMG and δ -EMQG on subsets of the SIFT1B dataset of increasing magnitude, specifically with 1M, 5M, 10M, 20M, 50M, and 100M vectors. We kept the build parameters for δ -EMG and δ -EMQG consistent across all scales. The parameter L was fixed at 1000, and M was set to 64 (except for δ -EMQG on the 50M and 100M datasets where it

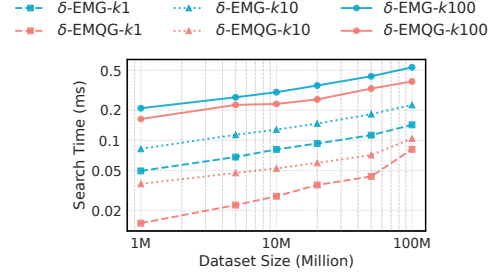


Figure 9: Search Time vs Data Size

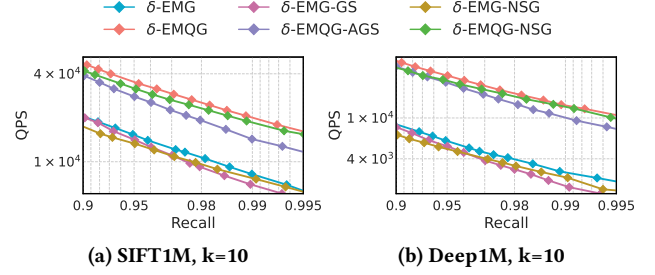


Figure 10: Ablation Study

was reduced to 32 to reduce memory cost without compromising performance). The parameter t was fixed to the best value found on SIFT1M. We measured the search time needed to reach 95% recall for $k \in \{1, 10, 100\}$. As shown in Figure 9, the search time for both of our methods grows nearly linearly with the number of data points. This result confirms that our methods scale efficiently and are suitable for large-scale applications.

Exp-9. Ablation Studies. To deconstruct the performance contributions of our proposed framework, we conducted an ablation study. We tested the importance of both our graph construction and our search algorithms separately.

First, we kept our specialized search algorithms but replaced our graph construction with baseline structures:

- **δ -EMG-NSG:** Employs our Error-Bounded top-k ANN Search (Algorithm 3) on a standard NSG graph.
- **δ -EMQG-NSG:** Employs our Probing Search on a SymphonyQG graph, which is based on NSG.

Next, we used our proposed graph constructions but reverted to simpler, standard search algorithms:

- **δ -EMG-GS:** Employs a standard greedy search (Algorithm 1) on our δ -EMG graph.
- **δ -EMQG-AGS:** Employs an approximate greedy search (AGS) on our δ -EMQG. Originally proposed in SymphonyQG, AGS follows the same logic as a greedy search but uses approximate distances to guide the graph traversal.

We compared the QPS-Recall curves for these variants against our full methods at $k = 10$. For the variants with standard search algorithms (δ -EMG-GS and δ -EMQG-AGS), the curves were generated by varying the candidate set size. As shown in Figure 10, removing either our graph construction or our search algorithm leads to a drop in performance. Full results can be seen in the supplementary material. This proves that both components are essential and work synergistically to achieve the final results.

8 Conclusion

In this paper, we addressed the challenge of achieving efficient and error-bounded top- k Approximate Nearest Neighbor (ANN) search. We proposed a novel graph-based model, the δ -error-bounded monotonic graph (δ -EMG), which ensures that the distances of retrieved top- k approximate nearest neighbors are within a user-specified error tolerance. Our method achieves a $(1/\delta)$ -approximation for top- k NN search with a space complexity of $O(n \ln n)$. We also introduced δ -EMQG, a quantized version of δ -EMG, along with efficient construction and search algorithms. Our experimental results on multiple datasets demonstrated that our approach outperforms state-of-the-art methods in terms of query-per-second (QPS) while maintaining the desired error bounds. Furthermore, we provided extensive experiments, including scalability and ablation studies, to validate the effectiveness and efficiency of our algorithms.

References

- [1] Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2015. Cache locality is not enough: High-Performance Nearest Neighbor Search with Product Quantization Fast Scan. *Proc. VLDB Endow.* 9, 4 (2015), 288–299.
- [2] Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2017. Accelerated Nearest Neighbor Search with Quick ADC. In *ICMR 2017*. 159–166.
- [3] Sunil Arya, Guilherme Dias da Fonseca, and David M. Mount. 2017. Near-Optimal epsilon-Kernel Construction and Related Problems. In *SoCG 2017*, Vol. 77. 10:1–10:15.
- [4] Sunil Arya, Guilherme Dias da Fonseca, and David M. Mount. 2017. Optimal Approximate Polytope Membership. In *SODA 2017*. 270–288.
- [5] Martin Aumüller, Erik Bernhardsson, and Alexander John Faithfull. 2020. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Inf. Syst.* 87 (2020).
- [6] Martin Aumüller and Matteo Ceccarello. 2023. Recent Approaches and Trends in Approximate Nearest Neighbor Search, with Remarks on Benchmarking. *IEEE Data Eng. Bull.* 47, 3 (2023), 89–105.
- [7] Franz Aurenhammer. 1991. Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure. *ACM Comput. Surv.* 23, 3 (1991), 345–405.
- [8] Dmitry Baranchuk, Artem Babenko, and Yuriy Malkov. 2018. Revisiting the Inverted Indices for Billion-Scale Approximate Nearest Neighbors. In *ECCV 2018*. 209–224.
- [9] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1990. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *SIGMOD 1990*. 322–331.
- [10] Wei-Cheng Chang, Felix X. Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. 2020. Pre-training Tasks for Embedding-based Large-scale Retrieval. In *ICLR 2020*.
- [11] Qi Chen, Bing Zhao, Haidong Wang, Mingqin Li, Chuanjie Liu, Zengzhong Li, Mao Yang, and Jingdong Wang. 2021. SPANN: Highly-efficient Billion-scale Approximate Nearest Neighborhood Search. In *NeurIPS 2021*. 5199–5212.
- [12] Paolo Ciaccia, Marco Patella, and Pavel Zezula. 1997. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *VLDB 1997*. 426–435.
- [13] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *SoCG 2004*. 253–262.
- [14] D.W. Dearholt, N. Gonzales, and G. Kurup. 1988. Monotonic Search Networks For Computer Vision Databases. In *Twenty-Second Asilomar Conference on Signals, Systems and Computers*, Vol. 2. 548–553.
- [15] Wei Dong, Moses Charikar, and Kai Li. 2011. Efficient k-nearest neighbor graph construction for generic similarity measures. In *WWW 2011*. 577–586.
- [16] Cong Fu and Deng Cai. 2016. EFANNA : An Extremely Fast Approximate Nearest Neighbor Search Algorithm Based on kNN Graph. *arXiv:1609.07228 [cs.CV]* <https://arxiv.org/abs/1609.07228>
- [17] Cong Fu, Changxu Wang, and Deng Cai. 2022. High Dimensional Similarity Search With Satellite System Graph: Efficiency, Scalability, and Unindexed Query Compatibility. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 8 (2022), 4139–4150.
- [18] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2019. Fast Approximate Nearest Neighbor Search With The Navigating Spreading-out Graph. *Proc. VLDB Endow.* 12, 5 (2019), 461–474.
- [19] Jinyang Gao, H. V. Jagadish, Wei Lu, and Beng Chin Ooi. 2014. DSH: data sensitive hashing for high-dimensional k-nnsearch. In *SIGMOD 2014*. ACM, 1127–1138.
- [20] Jianyang Gao and Cheng Long. 2024. RaBitQ: Quantizing High-Dimensional Vectors with a Theoretical Error Bound for Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 3 (2024), 167.
- [21] Vincent Garcia, Eric Debreuve, and Michel Barlaud. 2008. Fast k nearest neighbor search using GPU. In *CVPR Workshops 2008*. 1–6.
- [22] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2013. Optimized Product Quantization for Approximate Nearest Neighbor Search. In *CVPR 2013*. 2946–2953.
- [23] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity Search in High Dimensions via Hashing. In *VLDB 1999*. 518–529.
- [24] Yutong Gou, Jianyang Gao, Yuxuan Xu, and Cheng Long. 2025. SymphonyQG: Towards Symphonious Integration of Quantization and Graph for Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 3, 1 (2025), 80:1–80:26.
- [25] Kiana Hajebi, Yasin Abbasi-Yadkori, Hossein Shahbazi, and Hong Zhang. 2011. Fast Approximate Nearest-Neighbor Search with k-Nearest Neighbor Graph. In *IJCAI 2011*. 1312–1317.
- [26] Sarel Har-Peled. 2001. A Replacement for Voronoi Diagrams of Near Linear Size. In *FOCS 2001*. 94–103.
- [27] Ben Harwood and Tom Drummond. 2016. FANNG: Fast Approximate Nearest Neighbour Graphs. In *CVPR 2016*. 5713–5722.
- [28] Masajiro Iwasaki and Daisuke Miyazaki. 2018. Optimization of Indexing Based on k-Nearest Neighbor Graph for Proximity Search in High-dimensional Data. *arXiv:1810.07355 [cs.DB]*
- [29] J.W. Jaromczyk and G.T. Toussaint. 1992. Relative neighborhood graphs and their relatives. *Proc. IEEE* 80, 9 (1992), 1502–1517.
- [30] Stanley Milgram Jeffrey Travers. 1977. An Experimental Study of the Small World Problem. In *Social Networks*. 179–197.
- [31] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 1 (2011), 117–128.
- [32] Zhongming Jin, Debing Zhang, Yao Hu, Shiding Lin, Deng Cai, and Xiaofei He. 2014. Fast and Accurate Hashing Via Iterative Nearest Neighbors Expansion. *IEEE Transactions on Cybernetics* 44, 11 (2014), 2167–2177.
- [33] Jon M. Kleinberg. 2000. The small-world phenomenon: an algorithmic perspective. In *STOC 2000*. ACM, 163–170.
- [34] Brian Kulis and Kristen Grauman. 2009. Kernelized locality-sensitive hashing for scalable image search. In *CVPR 2009*. 2130–2137.
- [35] Govinda D. Kurup. 1992. *Database organized on the basis of similarities with applications in computer vision*. Ph. D. Dissertation. USA. UMI Order No. GAX92-01912.
- [36] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *NeurIPS 2020*.
- [37] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2020. Approximate Nearest Neighbor Search on High Dimensional Data - Experiments, Analyses, and Improvement. *IEEE Trans. Knowl. Data Eng.* 32, 8 (2020), 1475–1488.
- [38] Yuriy Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. 2014. Approximate nearest neighbor algorithm based on navigable small world graphs. *Inf. Syst.* 45 (2014), 61–68.
- [39] Yuriy A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836.
- [40] Jason Mohoney, Anil Pacaci, Shihabur Rahman Chowdhury, Ali Mousavi, Ihab F. Ilyas, Umar Farooq Minhas, Jeffrey Pound, and Theodoros Rekatsinas. 2023. High-Throughput Vector Similarity Search in Knowledge Graphs. *Proc. ACM Manag. Data* 1, 2 (2023), 197:1–197:25.
- [41] Yun Peng, Byron Choi, Tsz Nam Chan, Jianye Yang, and Jianliang Xu. 2023. Efficient Approximate Nearest Neighbor Search in Multi-dimensional Databases. *Proc. ACM Manag. Data* 1, 1 (2023), 54:1–54:27.
- [42] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. 2007. Object retrieval with large vocabularies and fast spatial matching. In *CVPR 2007*.
- [43] Larissa Capobianco Shimomura, Rafael Seidi Oyamada, Marcos R. Vieira, and Daniel S. Kaster. 2021. A survey on graph-based methods for similarity searches in metric spaces. *Inf. Syst.* 95 (2021), 101507.
- [44] Chanop Silpa-Anan and Richard I. Hartley. 2008. Optimised KD-trees for fast image descriptor matching. In *CVPR 2008*.
- [45] Aditi Singh, Suhas Jayaram Subramanya, Ravishankar Krishnaswamy, and Harsha Vardhan Simhadri. 2021. FreshDiskANN: A Fast and Accurate Graph-Based ANN Index for Streaming Similarity Search. *CoRR abs/2105.09613* (2021). *arXiv:2105.09613*
- [46] Suhas Jayaram Subramanya, Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnaswamy, and Rohan Kadekodi. 2019. Rand-NSG: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node. In *NeurIPS 2019*. 13748–13758.
- [47] Godfried T. Toussaint. 1980. The relative neighbourhood graph of a finite planar set. *Pattern Recognition* 12, 4 (1980), 261–268.
- [48] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. 2021. A Comprehensive Survey and Experimental Comparison of Graph-Based Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 14, 11 (2021), 1964–1978.
- [49] Jiadong Xie, Jeffrey Xu Yu, and Yingfan Liu. 2025. Graph Based K-Nearest Neighbor Search Revisited. *ACM Trans. Database Syst.* 50, 4 (2025), 14:1–14:30.
- [50] Weijie Zhao, Shulong Tan, and Ping Li. 2020. SONG: Approximate Nearest Neighbor Search on GPU. In *ICDE 2020*. 1033–1044.