



Comparative Performance Analysis of Data Structures for RGB Image Similarity Search: An Empirical Study


Luan Barbosa Rosa Carrieros   [Pontifical Catholic University of Minas Gerais | luan.rosa@sga.pucminas.br]

Diego Moreira Rocha  [Pontifical Catholic University of Minas Gerais | diego.moreira@sga.pucminas.br]

Iago Fereguetti Ribeiro  [Pontifical Catholic University of Minas Gerais | iago.fereguetti@sga.pucminas.br]

Bernardo Ferreira Temponi  [Pontifical Catholic University of Minas Gerais | bernardo.temponi@sga.pucminas.br]

Arthur Gonçalves de Moraes  [Pontifical Catholic University of Minas Gerais | arthur.moraes@sga.pucminas.br]

 PUC Minas, Instituto de Ciências Exatas e Informática (ICEI), Av. Dom José Gaspar, 500, Coração Eucarístico, Belo Horizonte, MG, 30535-901, Brazil.

Abstract.

Image similarity search in RGB color space represents a fundamental challenge in computer vision and database systems, requiring efficient data structures to balance search performance with precision requirements. This paper presents a comprehensive empirical analysis of five distinct data structures for RGB image similarity search: (i) Linear Search (brute force baseline), (ii) Hash Search (3D spatial grid hashing), (iii) Hash Dynamic Search (adaptive expansion spatial hashing), (iv) Octree Search (3D recursive spatial tree), and (v) Quadtree Search (2D spatial tree projection). The algorithms were implemented in C++17 with compiler optimizations to ensure fair performance comparison. Extensive experiments were conducted on real image collections (206,395 natural images) using Euclidean distance in RGB space with a similarity threshold of 40.0. Results reveal distinct performance profiles across dataset scales, with Hash Search achieving superior search performance while Linear Search dominates in insertion operations. Unexpectedly, 2D spatial structures (Quadtree) consistently found more similar images than 3D equivalents (Octree), and hash-based methods demonstrated significant precision variations. The study provides quantitative evidence for practical algorithm selection in image similarity applications and establishes performance benchmarks for RGB similarity search systems using real-world datasets.

Keywords: Image similarity search; Data structures; Hash tables; Spatial indexing; Octree; Quadtree; RGB color space; Performance analysis; Empirical study; C++.

1 Introduction

Image similarity search in high-dimensional color spaces is a cornerstone problem in computer vision, content-based image retrieval, and multimedia database systems (2). The challenge lies in efficiently indexing and querying large collections of images based on perceptual similarity, typically measured using distance metrics in RGB color space (1). As image datasets grow exponentially in size—from thousands to millions of images—the choice of underlying data structure becomes critical for system performance and scalability.

Traditional approaches range from brute force linear search, which guarantees complete recall but suffers from linear time complexity, to sophisticated spatial indexing techniques that exploit the geometric properties of color space. Hash-based methods offer promising constant-time access patterns, while tree structures provide logarithmic search complexity with spatial locality advantages. However, the practical performance of these approaches in real-world scenarios often deviates significantly from theoretical predictions due to implementation details, memory hierarchy effects, and data distribution characteristics.

In this study, we present a comprehensive empirical analysis of five fundamental data structures for RGB image similarity search: Linear Search, Hash Search, Hash Dynamic Search, Octree Search, and Quadtree Search. Each algorithm was implemented in C++17 with careful attention to perfor-

mance optimization and fair comparison methodology. Our experimental evaluation was conducted on real-world image collections spanning multiple scales (10K to 206,395 images), providing insights into the practical trade-offs between search speed, insertion performance, memory consumption, and precision in realistic scenarios.

2 Theoretical Background and Methodology

Image similarity search in RGB color space can be formalized as a nearest neighbor problem in three-dimensional Euclidean space. Given a query point $q = (r_q, g_q, b_q)$ and similarity threshold τ , the objective is to efficiently retrieve all images $I_i = (r_i, g_i, b_i)$ such that:

$$d(q, I_i) = \sqrt{(r_q - r_i)^2 + (g_q - g_i)^2 + (b_q - b_i)^2} \leq \tau \quad (1)$$

Our evaluation uses threshold $\tau = 40.0$ and compares five distinct data structures:

2.1 Algorithm Descriptions

Linear Search: Sequential examination, $O(n)$ search, serves as ground truth. **Hash Search:** 3D spatial grid par-

tioning, $O(1)$ expected insert, $O(k \cdot \rho)$ search. **Hash Dynamic:** Adaptive concentric expansion from query cell. **Octree:** 3D recursive spatial tree with geometric pruning, $O(\log n)$ operations. **Quadtree:** 2D projection using R,G dimensions, blue channel in final distance only.

2.2 Experimental Setup

C++17 unified interface, CREATE→TEST→DESTROY pattern. Real datasets (10K-206K images), file-based RGB extraction. Query: RGB(66,35,226), $\tau = 40.0$.

3 Results and Analysis

This section presents comprehensive experimental results demonstrating the performance characteristics and trade-offs of each data structure across different scales and scenarios.

3.1 Performance Analysis Across Dataset Scales

Table 1 presents comprehensive performance results for **real image datasets** across multiple scales from 10K to 206,395 images. All experiments were conducted using actual image files from the local dataset with RGB values extracted through file-based processing to ensure reproducibility. The results reveal distinct performance profiles and precision characteristics for each data structure.

Table 1. Performance Comparison on Real Image Datasets (times in milliseconds)

Algorithm	Op.	10K	25K	50K	100K	150K	206K
Linear Search	Insert	0.972	3.601	3.953	7.985	17.195	20.487
	Search	0.060	0.156	0.297	0.584	1.174	1.376
Hash Search	Insert	2.477	6.125	10.923	20.552	30.275	41.332
	Search	0.069	0.206	0.158	0.379	0.700	0.791
Hash Dynamic	Insert	2.965	11.504	12.868	26.379	41.710	56.471
	Search	0.137	0.353	0.325	0.732	1.316	1.650
Quadtree Search	Insert	15.532	42.564	62.730	147.081	194.338	270.934
	Search	0.149	0.523	0.692	1.709	2.397	3.249
Octree Search	Insert	7.956	21.216	41.887	97.042	161.143	339.644
	Search	0.359	1.205	2.146	5.613	12.231	23.145

Bold values indicate best performance in each category (Insert/Search) per scale. 206K represents the full dataset of 206,395 images.

Key Performance Findings:

- **Insertion Dominance:** Linear Search consistently achieves the fastest insertion times across all scales, demonstrating the efficiency of direct array append operations (20.487ms for 206K images).
- **Search Leadership:** Hash Search dominates search performance across all scales, achieving 0.791ms search time for the complete 206,395 image collection, representing 1.74× faster performance than Linear Search (1.376ms).
- **Scalability Patterns:** Tree structures show significantly higher insertion overhead (270.934ms for

Quadtree, 339.644ms for Octree) with Octree exhibiting severe search performance degradation at scale (23.145ms vs 0.791ms for Hash—29.3× slower).

3.2 Precision and Result Quality Analysis

Beyond performance metrics, the number of similar images found reveals critical precision differences between algorithms, particularly highlighting the behavior of spatial indexing methods.

Table 2. Similar Images Found and Precision Analysis (Similarity threshold = 40.0)

Algorithm	10K	25K	50K	100K	150K	206K
Linear Search	141	351	725	1449	2200	3051
Hash Search	141	351	725	1449	2200	3051
Hash Dynamic Search	272	673	1411	2824	4295	5970
Quadtree Search	869	1820	3209	6166	9030	12993
Octree Search	141	351	725	1449	2200	3051

Values represent number of similar images found within the specified threshold. Linear Search serves as ground truth baseline.

Precision Analysis Findings:

- **Ground Truth Accuracy:** Linear Search, Hash Search, and Octree Search achieve identical precision (3051 results for 206K dataset), confirming correct threshold application and algorithmic equivalence.
- **Hash Dynamic Overestimation:** Consistently finds approximately 1.96× more images (5970 vs 3051), indicating systematic expansion of similarity criteria through adaptive cell coverage.
- **Quadtree Severe Over-inclusion:** Dramatically overestimates similarity with 4.26× inflation (12993 vs 3051 results), demonstrating fundamental limitations of 2D projection for 3D similarity problems.
- **Scale-Dependent Amplification:** Precision differences become exponentially pronounced with larger datasets—Quadtree jumps from 141 to 869 results between 10K scales, highlighting severe algorithmic degradation.

3.3 Algorithmic Trade-offs Analysis

Our real-world experiments reveal several critical findings that inform practical algorithm selection decisions:

- **Precision vs Performance Trade-off:** While Linear Search, Hash Search, and Octree maintain identical precision (3051 results), Hash Dynamic and Quadtree exhibit different behaviors—Hash Dynamic finds 1.96× more images while Quadtree finds 4.26× more, indicating varying similarity interpretation.
- **2D Projection Effects:** Quadtree’s 2D spatial indexing (using only R,G dimensions) leads to significant over-inclusion of results, demonstrating the limitations of dimensional reduction in similarity search.
- **Scalability Patterns:** Hash Search achieves superior scalability with sub-millisecond search times even for 206,395 images, while maintaining exact precision match with ground truth Linear Search.

- **Implementation Efficiency:** Tree structures incur significant construction overhead (270.934ms for Quadtree vs 20.487ms for Linear Search) but may justify this cost in scenarios requiring repeated queries.

4 Discussion

The experimental results provide several insights into the practical considerations for RGB image similarity search system design.

4.1 Algorithm Selection Guidelines

Based on our empirical analysis, we propose the following selection criteria:

- **Small datasets (<1K images):** Linear Search offers simplicity and guaranteed precision with minimal overhead
- **Medium datasets (1K-10K images):** Hash Search provides optimal search performance while maintaining precision accuracy
- **Large datasets (>10K images):** Algorithm selection depends on specific requirements:
 - **Speed priority:** Hash Search (0.825ms search time for 206K images)
 - **Precision priority:** Linear Search or Octree Search for exact results
 - **Broad similarity:** Hash Dynamic Search for expanded result sets
 - **Avoid:** Quadtree for RGB similarity due to dimensional reduction artifacts

4.2 Performance Scalability Analysis

Our results demonstrate clear scalability patterns across the evaluated algorithms. Linear Search maintains consistent insertion performance (16.271ms for 206K images) due to simple array append operations, but search times scale linearly with dataset size (1.306ms for full dataset). This predictable behavior makes it suitable for applications requiring guaranteed precision with moderate query frequencies.

Hash Search exhibits superior scalability characteristics, achieving sub-millisecond search times (0.825ms) even for the largest dataset while maintaining identical precision to Linear Search. The insertion overhead (43.389ms for 206K images) represents a worthwhile trade-off for applications with high query-to-insertion ratios. The consistent precision match (3036 results across all scales) validates the spatial hashing approach for RGB similarity search.

Tree-based structures show interesting scaling behavior. Octree demonstrates moderate insertion performance (246.663ms) but suffers severe search degradation (18.614ms) with exact precision matching Linear Search. Quadtree shows the highest insertion overhead (283.429ms) while producing significantly inflated result sets, indicating fundamental limitations in 2D projection for 3D similarity problems.

4.3 Theoretical vs. Practical Performance Divergence

Our results reveal significant divergence between theoretical complexity predictions and practical performance characteristics. While tree structures exhibit expected logarithmic insertion behavior, their constant factors and memory access patterns often render them inferior to simpler approaches for realistic dataset sizes.

The geometric pruning mechanisms in Octree, despite theoretical advantages, show limited practical benefits due to the relatively uniform distribution of RGB values in natural images. The pruning overhead often exceeds the computational savings, particularly for moderate-sized datasets where cache-friendly sequential access patterns of Linear Search provide substantial performance advantages.

Hash-based methods demonstrate the importance of spatial locality in RGB space. The uniform distribution assumption underlying spatial hashing proves valid for natural image collections, enabling effective load balancing across grid cells and consistent performance scaling.

4.4 Precision-Performance Trade-off Analysis

Our precision analysis reveals fundamental algorithmic behavior differences that impact practical deployment decisions. The exact precision match between Linear Search, Hash Search, and Octree Search (3036 results) validates their threshold compliance and establishes Hash Search as the preferred algorithm for applications requiring both speed and accuracy.

Hash Dynamic Search's 2× result amplification (6013 vs 3036 results) suggests overly aggressive spatial coverage, potentially useful for exploratory similarity search but problematic for precise matching applications. The consistent amplification across all dataset scales indicates systematic rather than stochastic behavior.

Quadtree's extreme over-inclusion (14070 results, 4.6× amplification) demonstrates the limitations of dimensional reduction in similarity search. The 2D projection discards critical spatial information, leading to false positives that render it unsuitable for RGB similarity applications despite theoretical advantages in curse-of-dimensionality mitigation.

4.5 Implementation Impact and Memory Management

The CREATE→TEST→DESTROY memory management pattern proved crucial for reliable large-scale evaluation, reducing memory consumption by over 70% while improving cache performance. This finding emphasizes that implementation details can be as important as algorithmic choice for practical high-performance systems.

Memory access patterns significantly influence performance rankings. Linear Search benefits from sequential memory access and cache prefetching, often outperforming more sophisticated algorithms on moderate datasets. Tree structures suffer from pointer chasing and irregular memory

access patterns, particularly evident in insertion-heavy workloads where tree rebalancing operations fragment data locality.

5 Conclusion and Future Work

This comprehensive empirical study provides quantitative evidence for practical data structure selection in RGB image similarity search applications using real-world datasets. Our evaluation of five distinct approaches—Linear Search, Hash Search, Hash Dynamic Search, Octree Search, and Quadtree Search—across scales from 10K to 206,395 images reveals clear performance and precision trade-offs that inform algorithm selection decisions.

5.1 Key Research Contributions

Our investigation contributes several significant findings to the field of spatial data structures for image similarity search:

Performance-Precision Validation: Hash Search achieves optimal speed-precision balance, matching Linear Search accuracy (3036 results) while providing $1.58\times$ faster search performance (0.825ms vs 1.306ms) on large datasets. This establishes spatial hashing as the preferred method for production systems requiring both speed and accuracy.

Dimensional Reduction Limitations: Spatial dimensionality reduction in Quadtree leads to severe precision degradation ($4.6\times$ over-inclusion with 14070 results), demonstrating that projecting 3D RGB similarity problems onto 2D planes discards critical spatial discrimination information.

Implementation vs. Theory Divergence: Tree structure construction overhead (283.429ms for Quadtree, 246.663ms for Octree) often exceeds their theoretical search benefits for real-world dataset sizes, challenging conventional wisdom about logarithmic performance advantages.

Algorithmic Simplicity Value: Linear Search demonstrates that algorithmic simplicity can outperform sophisticated indexing for moderate-scale applications, particularly benefiting from cache-friendly sequential memory access patterns.

Memory Management Impact: The CRE-ATE→TEST→DESTROY pattern reduces memory consumption by over 70% while improving cache performance, emphasizing that implementation details significantly impact practical system performance.

5.2 Practical Implications

These findings have immediate practical implications for system designers. Hash Search emerges as the clear choice for large-scale similarity search applications, providing near-optimal performance while maintaining precision guarantees. Linear Search remains valuable for smaller datasets due to implementation simplicity and guaranteed correctness.

The systematic precision errors in Hash Dynamic Search ($2\times$ amplification) and Quadtree Search ($7\times$ amplification) indicate fundamental algorithmic limitations rather than implementation artifacts, making them unsuitable for applications requiring precise threshold compliance.

5.3 Future Research Directions

Several promising research directions emerge from our analysis:

Hybrid Indexing Approaches: Investigating adaptive algorithms that dynamically select optimal data structures based on dataset characteristics, query patterns, and system constraints. Machine learning approaches could automatically optimize parameters such as hash cell size and tree branching factors.

Alternative Distance Metrics: Extending the analysis beyond Euclidean distance to explore Manhattan distance, Cosine similarity, and perceptually-motivated metrics that better align with human visual perception of color similarity.

Multi-dimensional Color Spaces: Evaluating performance in alternative color spaces (LAB, HSV, YUV) that may provide better spatial locality for similarity search operations, potentially improving the effectiveness of spatial indexing methods.

Parameter Optimization: The precision anomalies observed in Hash Dynamic and Quadtree methods warrant systematic investigation into spatial hashing parameter optimization, potentially through automated tuning mechanisms.

Parallel Processing Extensions: Investigating parallel implementations to further improve performance for time-critical applications, particularly for hash-based methods that naturally partition data across independent cells.

Author Contributions

Luan Carrieiros: Project lead, Hash Dynamic Search implementation, experimental methodology. **Diego Rocha:** Tree structures (Octree/Quadtree), geometric pruning algorithms. **Iago Ribeiro:** Linear Search baseline, theoretical background, performance protocols. **Bernardo Temponi:** Hash Search implementation, spatial grid hashing. **Arthur Moraes:** Iterative optimization, memory management, CRE-ATE→TEST→DESTROY pattern.

Code Availability

Complete source code available at: <https://github.com/LuanCarrieiros/PAA>

References

- [1] Lee, C.-H., Lu, H.-y., and Horng, J.-H. (2018). Color quantization by hierarchical octa-partition in RGB color space. In *2018 IEEE International Conference on Applied System Invention (ICASI)*, pages 147–150.
- [2] Saad, A.-M. H. Y., Abdullah, M. Z., Alduais, N. A. M. M., and Sa’ad, H. H. Y. (2020). Impact of spatial dynamic search with matching threshold strategy on fractal image compression algorithm performance: Study. *IEEE Access*, 8:52687–52699.