*Journal of the Brazilian Computer Society, 202X, XX:1*doi: XX.XXXX/jbcs.202X.XXXXXX

Busca por Similaridade em Alta Dimensionalidade: Implementação e Análise Comparativa de LSH e M-Tree Aplicadas a Imagens Coloridas

Guilherme Henrique Vieira Nascimento [1382418@sga.pucminas.br]
Leonardo de Oliveira Carvalho [1306910@sga.pucminas.br]
Luca Lourenço Araújo Dal Bianco Gonzaga [1320691@sga.pucminas.br]
Matheus Fagundes Araujo [1446152@sga.pucminas.br]
Pedro Rodrigues Alves [1446100@sga.pucminas.br]
Rafael Vilela Padilha Clark [1317127@sga.pucminas.br]
Victor Lopes Azevedo Araujo [1348805@sga.pucminas.br]

Instituto de Ciências Exatas e Informática, Pontifícia Universidade Católica de Minas Gerais, Av. Dom José Gaspar, 500 – Coração Eucarístico, Belo Horizonte, MG, 30535-901 – Brasil.

Este trabalho apresenta a segunda etapa de um projeto de análise de estruturas de dados aplicadas ao problema de busca por similaridade em documentos representados por imagens coloridas. A primeira etapa avaliou Lista, Tabela Hash e QuadTree, evidenciando limitações dessas estruturas em cenários de alta dimensionalidade. Nesta segunda fase, investigamos duas estruturas mais adequadas a esse contexto: Locality Sensitive Hashing (LSH) e M-Tree. Ambas foram implementadas em C++ e avaliadas sobre vetores de 64 dimensões extraídos a partir de histogramas RGB normalizados.

Os experimentos foram conduzidos sobre subconjuntos de diferentes tamanhos do conjunto CIFAR-10, bem como sobre a sua base completa. Os resultados mostram que LSH e M-Tree reduzem significativamente o custo de busca em relação à varredura sequencial, especialmente em bases maiores, ao custo de uma pequena perda de precisão (no caso do LSH). A análise comparativa evidencia os benefícios e os trade-offs de estruturas projetadas para alta dimensionalidade frente às abordagens tradicionais.

Keywords: Busca por Similaridade, Estruturas de Dados, Alta Dimensionalidade, Locality Sensitive Hashing, M-Tree, Recuperação de Imagens.

1 Introdução

A busca por similaridade é um problema central em recuperação de informação, visão computacional e mineração de dados. Em tarefas onde documentos são representados por vetores de características, o objetivo é, dado um vetor de consulta, encontrar os itens mais próximos sob uma métrica definida. No contexto deste trabalho, as entidades analisadas são imagens coloridas representadas por histogramas normalizados no espaço RGB.

Na primeira etapa deste projeto foram implementadas e analisadas estruturas tradicionais de organização de dados — Lista, HashTable e QuadTree — revelando que tais abordagens sofrem degradação significativa em espaços de alta dimensionalidade, fenômeno conhecido como *curse of dimensionality* Beyer *et al*. (1999). Ainda que úteis como *baseline*, essas estruturas não escalam bem para bases maiores.

Para contornar essas limitações, a literatura propõe estruturas métricas e probabilísticas capazes de lidar eficientemente com dados de alta dimensão. Entre elas, destacam-se o **Locality Sensitive Hashing (LSH)** Charikar (2002), amplamente utilizado em busca aproximada, e a **M-Tree** Ciaccia *et al*. (1997), uma árvore métrica balanceada projetada para suportar buscas exatas com poda eficiente.

O objetivo deste artigo é implementar essas duas estruturas com base no código desenvolvido pelo grupo, aplicá-las a subconjuntos de diferentes tamanhos do CIFAR-10 e compará-las sistematicamente às estruturas da Etapa 1. Essa comparação permite uma análise evolutiva de eficiência e precisão, observando como estruturas projetadas para alta dimensionalidade se comportam em relação às abordagens tradicionais.

2 Fundamentos Teóricos

2.1 Representação das Imagens

Todas as imagens utilizadas foram descritas por vetores de características construídos a partir de histogramas de cores RGB normalizados. Cada canal (R, G, B) foi quantizado em 4 *bins*, resultando em um vetor final de $4^3 = 64$ dimensões. Essa representação é comum na literatura de recuperação de imagens Swain and Ballard (1991) e oferece um compromisso entre simplicidade e poder discriminativo.

Seja uma imagem representada por um vetor

$$v = (v_1, v_2, \ldots, v_{64}),$$

a similaridade entre duas imagens a e b é calculada pela

distância euclidiana:

$$d(a,b) = \sqrt{\sum_{i=1}^{64}(a_i - b_i)^2}.$$

Essa métrica foi utilizada em todas as estruturas avaliadas.

2.2 Locality Sensitive Hashing (LSH)

O LSH permite realizar buscas aproximadas em alta dimensionalidade utilizando funções de hash projetadas para preservar proximidade. Neste trabalho, adotamos o método dos **hiperplanos aleatórios**, originalmente apresentado por Charikar (2002).

Para cada tabela hash, gera-se um conjunto de vetores normais aleatórios

$$r_1, r_2, \ldots, r_k \in \mathbb{R}^{64}.$$

O hash de um vetor v é definido como o vetor de bits:

$$h(v) = \big(\text{sign}(v \cdot r_1), \ldots, \text{sign}(v \cdot r_k)\big),$$

em que $\text{sign}(x)$ retorna 1 se $x \geq 0$ e 0 caso contrário. Vetores próximos têm alta probabilidade de produzir o mesmo hash e, portanto, caem nos mesmos *buckets*.

Neste trabalho, utilizamos:

- 5 tabelas de hash independentes;
- 12 hiperplanos por tabela (hash de 12 bits);
- buckets implementados como mapas que associam cada hash a uma lista de índices de imagens.

A busca consiste em:

1. calcular os hashes do vetor de consulta;
2. acessar os buckets correspondentes;
3. comparar apenas os candidatos recuperados.

Esse paradigma reduz substancialmente o número de comparações em relação à varredura completa, ao custo de uma busca aproximada, isto é, nem sempre retornando exatamente o mesmo vizinho encontrado pela busca exata.

2.3 M-Tree

A M-Tree, introduzida por Ciaccia, Patella e Zezula (1997) Ciaccia *et al*. (1997), é uma estrutura métrica balanceada projetada para busca por similaridade em grandes bases de dados. Ela se baseia em:

- **routing objects**: representantes dos nós internos;
- **raios de cobertura**: definem as regiões métricas abrangidas pelos nós;
- **poda métrica**: elimina subárvores impossíveis de conter o vizinho mais próximo;
- **promoção**: escolha dos representantes durante divisão de nós.

A implementação segue o algoritmo clássico:

- nós armazenam até 10 entradas;
- quando cheios, são divididos via promoção do par mais distante;
- a busca por vizinho mais próximo utiliza desigualdade triangular para poda;
- entradas são visitadas em ordem crescente de distância aos *routing objects*.

3 Implementação

Todas as estruturas foram implementadas em `C++17`. Nesta seção, descrevemos resumidamente a lógica dos métodos centrais, em especial LSH e M-Tree, adicionados sobre a base apresentada na Etapa 1 do projeto.

3.1 Implementação do LSH

A geração dos hiperplanos é realizada com distribuição normal, garantindo aleatoriedade adequada. Em alto nível, o cálculo do hash para um vetor de características v procede como:

1. para cada hiperplano r_i, calcular o produto escalar $v \cdot r_i$;
2. atribuir o bit i como 1 se $v \cdot r_i \geq 0$ ou 0 caso contrário;
3. concatenar todos os bits para formar um inteiro que indexa o bucket.

No código, as tabelas são estruturas do tipo `std::unordered_map<size_t, std::vector<int>>`, mapeando hashes para índices de imagens. A inserção de uma imagem consiste em:

- armazenar o `ImageData` em um `data_store`;
- para cada tabela, computar o hash e inserir o índice da imagem no vetor correspondente ao bucket.

Durante a busca, apenas os elementos presentes em buckets compatíveis com a consulta são considerados candidatos, reduzindo o custo de comparação em relação à varredura completa. O número de candidatos inspecionados é reportado como “Candidatos” na saída do sistema.

3.2 Implementação da M-Tree

A M-Tree utiliza três componentes principais:

1. **Escolha da subárvore**: seleciona-se a subárvore cujo raio de cobertura precisa de menor aumento para incluir o novo objeto.
2. **Promoção**: quando um nó excede a capacidade, o par de entradas mais distante é promovido como representantes de dois novos nós.
3. **Poda métrica**: durante a busca, subárvores são descartadas se a distância do representante ao ponto de consulta, menos o raio de cobertura, for maior que a melhor distância atual.

A busca por vizinho mais próximo percorre a árvore priorizando subárvores cujo representante é mais próximo da consulta e utiliza poda métrica para evitar visitas desnecessárias. O número de comparações efetivas é contabilizado e reportado pelo programa.

4 Integração com o Código-Fonte

O código utilizado nos experimentos produz automaticamente, para cada imagem de consulta:

- tempo de busca para cada estrutura (em milissegundos);
- número de comparações realizadas (ou candidatos analisados, no caso do LSH);
- nome da imagem mais próxima encontrada;
- distância euclidiana entre a imagem de consulta e o vizinho retornado.

Na versão atual, o `main.cpp` realiza, para cada imagem da pasta `imageReference`, uma consulta simultânea em todas as estruturas (Lista, HashTable, QuadTree, LSH e M-Tree) utilizando a base de imagens presente na pasta `images`. Essa lógica é reaproveitada para três tamanhos de base, diferindo apenas nas pastas de entrada.

5 Experimentos

5.1 Conjunto de Dados e Cenários de Avaliação

Os experimentos foram conduzidos sobre subconjuntos de diferentes tamanhos do CIFAR-10 Krizhevsky (2009), além da base completa. Em todos os casos, cada imagem foi convertida em um vetor de 64 dimensões através do histograma RGB normalizado, mantendo consistência com a metodologia da etapa anterior.

Foram considerados três cenários:

- **Base pequena**: 30 imagens na base (`images`) e 10 imagens de referência (`imageReference`), cobrindo as 10 classes do CIFAR-10.
- **Base média**: 100 imagens na base e 50 imagens de referência.
- **Base grande**: base completa do CIFAR-10, com todas as imagens de treino na base de busca e um conjunto ampliado de imagens de referência.

5.2 Metodologia de Agregação

Para comparar as estruturas, foram extraídas dos logs as seguintes métricas:

- **tempo médio de busca** por estrutura, em milissegundos;
- **número médio de comparações** (ou candidatos, no caso do LSH).

Na base pequena (10 consultas) utilizamos todas as linhas produzidas pelo sistema. Na base média, para manter o texto conciso, utilizamos as dez primeiras consultas de cada estrutura como amostra representativa. Na base grande, os resultados seguem as mesmas tendências observadas, com crescimento mais acentuado do tempo para as estruturas da Etapa 1.

5.3 Resultados – Base Pequena

A Tabela 1 apresenta o tempo médio de busca (em milissegundos) e resume o número médio de comparações ou candidatos analisados para a base pequena (30 imagens, 10 consultas). Os valores são médias das dez consultas presentes no log.

Table 1. Tempo médio de busca (ms) e Número médio de comparações/candidatos – base pequena (30 imagens, 10 consultas).

Estrutura	Tempo médio (ms)	Comparações
Lista	0,0011	30 comparações
HashTable	0,0015	30 comparações
QuadTree	0,0443	7 comparações
LSH	0,0044	$\approx 4{,}3$ candidatos
M-Tree	0,0021	≈ 28 comparações

Nessa configuração pequena, todas as estruturas retornaram, na maior parte das consultas, a mesma imagem mais próxima reportada pela Lista, exceto em poucos casos em que o LSH não recuperou nenhum candidato em bucket compatível ou escolheu um vizinho diferente com distância próxima. A M-Tree foi, em geral, consistente com a Lista nesse cenário reduzido.

5.4 Resultados – Base Média

Para a base média (100 imagens, 50 de referência), analisamos as dez primeiras consultas presentes no log. A Tabela 2 apresenta os tempos médios obtidos e o número médio de comparações/candidatos.

Table 2. Tempo médio de busca (ms) e Número médio de comparações/candidatos – base média (100 imagens, 10 consultas amostradas).

Estrutura	Tempo médio (ms)	Comparações
Lista	0,0039	100 comparações
HashTable	0,0048	100 comparações
QuadTree	0,1336	12 comparações
LSH	0,0089	$\approx 13{,}9$ candidatos
M-Tree	0,0057	$\approx 10{,}7$ comparações

Observa-se que o tempo de QuadTree cresce significativamente com o aumento da base, enquanto Lista e HashTable mantêm tempos pequenos, porém com custo $O(n)$ em comparações. LSH e M-Tree passam a apresentar vantagem mais clara em número de comparações, com tempos ainda competitivos.

5.5 Resultados – Base Grande

Na base grande, composta pela totalidade das imagens do CIFAR-10, o comportamento observado segue as tendências anteriores:

- o tempo de Lista e HashTable cresce linearmente com o número de imagens, tornando-se rapidamente proibitivo para cenários de uso interativo;

- a QuadTree sofre ainda mais com o aumento da dimensionalidade e do tamanho da base, com crescimento acentuado de tempo e comparações;
- LSH e M-Tree mantêm tempos de busca mais estáveis e um número de comparações bem menor do que as estruturas da Etapa 1.

Os logs completos estão disponíveis no código do projeto e podem ser utilizados para análises adicionais.

6 Discussão

Os resultados obtidos permitem observar diferenças fundamentais entre estruturas tradicionais e estruturas projetadas para alta dimensionalidade:

- **Lista** e **HashTable** apresentam custo $O(n)$, produzindo quantidade elevada de comparações à medida que a base cresce. Em bases pequenas, o tempo absoluto continua baixo, mas o custo torna-se rapidamente impraticável em cenários de larga escala.
- **QuadTree** tenta explorar a estrutura espacial dos dados, mas depende de uma projeção dos vetores de 64 dimensões em 2 dimensões, resultando em perda de informação. Na prática, a poda é limitada e o tempo cresce significativamente, especialmente para a base média e grande.
- **LSH** reduz drasticamente o número de comparações graças aos *buckets* probabilísticos. Embora aproximado, obteve vizinhos próximos aos da Lista, com tempos maiores que a HashTable em bases pequenas, mas com melhor relação comparações/tempo à medida que o tamanho da base aumenta.
- **M-Tree** apresenta uma solução métrica estruturada, com poda eficiente. Nos experimentos, reduziu o número de comparações em relação à Lista e à HashTable, mantendo tempos de busca competitivos.

A comparação entre as bases pequena, média e grande mostra que:

- em bases pequenas, a diferença de tempo entre as estruturas é pouco perceptível em termos absolutos, mas LSH e M-Tree já começam a mostrar redução no número de comparações;
- em bases maiores, a vantagem de estruturas projetadas para alta dimensionalidade torna-se evidente, especialmente quando o número de consultas cresce;
- a escolha entre LSH e M-Tree depende do compromisso desejado entre precisão e desempenho: LSH favorece desempenho aproximado, enquanto M-Tree tende a manter maior coerência com a métrica subjacente.

7 Conclusão

Este artigo apresentou a segunda etapa de um estudo sobre busca por similaridade em imagens coloridas representadas por vetores de características. Após analisar estruturas tradicionais na primeira fase, esta etapa implementou e avaliou duas estruturas amplamente utilizadas em cenários de alta dimensionalidade: LSH e M-Tree.

Os resultados demonstram que:

- LSH reduz significativamente o número de comparações por consulta, mostrando-se adequado para buscas aproximadas em grandes bases;
- M-Tree explora a métrica de forma estruturada, diminuindo o número de comparações em relação à Lista e à HashTable, com tempos competitivos e vizinhos, em geral, coerentes com a distância euclidiana;
- em bases pequenas, as diferenças entre as estruturas são pouco visíveis em termos absolutos, mas à medida que o tamanho da base cresce, LSH e M-Tree tornam-se mais vantajosos.

A comparação entre as duas etapas evidencia a importância da escolha da estrutura de dados para busca por similaridade, especialmente em espaços de alta dimensão e com grandes volumes de dados. Como trabalhos futuros, pretendemos:

- avaliar diferentes funções de hash no LSH e ajustar o número de tabelas e hiperplanos;
- investigar estratégias alternativas de promoção e divisão de nós na M-Tree;
- incorporar outros métodos de indexação métrica (VP-Tree, Ball-Tree) e aumentar o conjunto de imagens para explorar cenários de larga escala e consultas em lote.

Responsabilidades dos Autores

- **Implementação das Estrutura LSH :** Matheus Fagundes, Leonardo de Oliveira e Victor Lopes.
- **Implementação da M-Tree:** Rafael Vilela e Guilherme Nascimento.
- **Integração das Estruturas com o Código, Testes, Análises e Documentação:** Luca Gonzaga e Pedro Rodrigues.

References

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When is nearest neighbor meaningful? In *Proceedings of the 7th International Conference on Database Theory (ICDT)*, pages 217–235.

Charikar, M. (2002). Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 380–388.

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

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Swain, M. J. and Ballard, D. H. (1991). Color indexing. *International Journal of Computer Vision*, 7(1):11–32.