Instituto de Computação
Universidade Estadual de Campinas

Indexação Multimídia Escalável e Busca por Similaridade Em Alta Dimensionalidade

Fernando Cesar Akune

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Fernando Cesar Akune e aprovada pela Banca Examinadora.

Campinas, 22 de setembro de 2011.

Ricardo Torres
Prof. Dr. Ricardo da Silva Torres (Orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA POR ANA REGINA MACHADO – CRB8/5467
BIBLIOTECA DO INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E
COMPUTAÇÃO CIENTÍFICA – UNICAMP

Akune, Fernando Cesar, 1976-
Ak84i Indexação multimídia escalável e busca por similaridade em alta dimensionalidade / Fernando Cesar Akune. – Campinas, SP : [s.n.], 2011.

Orientador: Ricardo da Silva Torres.
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Indexação. 2. Estrutura de dados (Computação). 3. Banco de dados. 4. Sistemas multimídia. I. Torres, Ricardo da Silva, 1977-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em inglês: Scalable multimedia indexing and similarity search in high dimensionality
Palavras-chave em inglês:
Indexing
Data structures (Computing)
Database
Multimedia systems
Área de concentração: Ciência da Computação
Titulação: Mestre em Ciência da Computação
Banca examinadora:
Ricardo da Silva Torres [Orientador]
Edson Borin
Caetano Traina Junior
Data da defesa: 01-08-2011
Programa de Pós Graduação: Ciência da Computação

TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 01 de agosto de 2011, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. Caetano Traina Junior
ICMC / USP

Prof. Dr. Edson Borin
IC / UNICAMP

Prof. Dr. Ricardo da Silva Torres
IC / UNICAMP

Instituto de Computação
Universidade Estadual de Campinas

Indexação Multimídia Escalável e Busca por Similaridade Em Alta Dimensionalidade

Fernando Cesar Akune

Agosto de 2011

Banca Examinadora:

- Prof. Dr. Ricardo da Silva Torres (Orientador)
  Instituto de Computação – UNICAMP
- Prof. Dr. Edson Borin
  Instituto de Computação – UNICAMP
- Prof. Dr. Eduardo Alves do Valle Junior (Suplente)
  Faculdade de Engenharia Elétrica e de Computação – UNICAMP
- Prof. Dr. Caetano Traina Junior
  Instituto de Ciências Matemáticas e de Computação – USP
- Profa. Dra. Agma Juci Machado Traina (Suplente)
  Instituto de Ciências Matemáticas e de Computação – USP

Resumo

A disseminação de grandes coleções de arquivos de imagens, músicas e vídeos tem aumentado a demanda por métodos de indexação e sistemas de recuperação de informações multimídia. No caso de imagens, os sistemas de busca mais promissores são os sistemas baseados no conteúdo, que ao invés de usarem descrições textuais, utilizam vetores de características, que são representações de propriedades visuais, como cor, textura e forma. O emparelhamento dos vetores de características da imagem de consulta e das imagens de uma base de dados é implementado através da busca por similaridade. A sua forma mais comum é a busca pelos k vizinhos mais próximos, ou seja, encontrar os k vetores mais próximos ao vetor da consulta. Em grandes bases de imagens, um índice é indispensável para acelerar essas consultas. O problema é que os vetores de características podem ter muitas dimensões, o que afeta gravemente o desempenho dos métodos de indexação. Acima de 10 dimensões, geralmente é preciso recorrer aos métodos aproximados, sacrificando a eficácia em troca da rapidez. Dentre as diversas soluções propostas, existe uma abordagem baseada em curvas fractais chamadas curvas de preenchimento do espaço. Essas curvas permitem mapear pontos de um espaço multidimensional em uma única dimensão, de maneira que os pontos próximos na curva correspondam a pontos próximos no espaço. O grande problema dessa alternativa é a existência de regiões de descontinuidade nas curvas, pontos próximos dessas regiões não são mapeados próximos na curva.

A principal contribuição deste trabalho é um método de indexação de vetores de características de alta dimensionalidade, que utiliza uma curva de preenchimento do espaço e múltiplos representantes para os dados. Esse método, chamado MONORAIL, gera os representantes explorando as propriedades geométricas da curva. Isso resulta em um ganho na eficácia da busca por similaridade, quando comparado com o método de referência. Outra contribuição não trivial deste trabalho é o rigor experimental usado nas comparações: os experimentos foram cuidadosamente projetados para garantir resultados estatisticamente significativos. A escalabilidade do MONORAIL é testada com três bases de dados de tamanhos diferentes, a maior delas com mais de 130 milhões de vetores.

Abstract

The spread of large collections of images, videos and music has increased the demand for indexing methods and multimedia information retrieval systems. For images, the most promising search engines are content-based, which instead of using textual annotations, use feature vectors to represent visual properties such as color, texture, and shape. The matching of feature vectors of query image and database images is implemented by similarity search. Its most common form is the k nearest neighbors search, which aims to find the k closest vectors to the query vector. In large image databases, an index structure is essential to speed up those queries. The problem is that the feature vectors may have many dimensions, which seriously affects the performance of indexing methods. For more than 10 dimensions, it is often necessary to use approximate methods to trade-off effectiveness for speed. Among the several solutions proposed, there is an approach based on fractal curves known as space-filling curves. Those curves allow the mapping of a multidimensional space onto a single dimension, so that points near on the curve correspond to points near on the space. The great problem with that alternative is the existence of discontinuity regions on the curves, where points near on those regions are not mapped near on the curve.

The main contribution of this dissertation is an indexing method for high-dimensional feature vectors, using a single space-filling curve and multiple surrogates for each data point. That method, called MONORAIL, generates surrogates by exploiting the geometric properties of the curve. The result is a gain in terms of effectiveness of similarity search, when compared to the baseline method. Another non-trivial contribution of this work is the rigorous experimental design used for the comparisons. The experiments were carefully designed to ensure statistically sound results. The scalability of the MONORAIL is tested with three databases of different sizes, the largest one with more than 130 million vectors.

Agradecimentos

Eu gostaria de agradecer a Deus, por tudo.

Agradeço à minha família que sempre me incentivou a estudar.

Agradeço ao Prof. Dr. Ricardo da Silva Torres e ao Prof. Dr. Eduardo Alves do Valle Junior pela orientação na realização deste trabalho, fundamentais para o meu crescimento intelectual.

Agradeço também ao Dr. George Teodoro, da UFMG, pela ajuda na parte experimental.

Minha viagem à Istambul para apresentar o artigo do MONORAIL no ICPR'10 foi possível graças ao auxílio financeiro da FAPESP (processo 2009/18438-7) e da FAEPEX/UNICAMP.

Sou grato a todos os colegas do RECOD, do trabalho e funcionários da UNICAMP que me ajudaram durante esse período do mestrado.

Agradeço aos membros da banca examinadora por terem aceitado o convite, pelas correções, críticas e sugestões ao trabalho.

Sumário

Capítulo 1

Introdução

1.1 Contexto

O enorme avanço na tecnologia dos dispositivos de armazenamento e a evolução da Internet têm possibilitado a disseminação de grandes coleções de arquivos de imagens, músicas e vídeos. Com isso, a indexação e a recuperação de informação multimídia emergiram como grandes desafios de pesquisa tanto no Brasil [Medeiros 2008] quanto no mundo.

A abordagem convencional para um sistema de recuperação de imagens é baseada na associação de uma descrição textual às imagens. Dessa forma, os mecanismos tradicionais de consulta em banco de dados podem ser utilizados para encontrar as imagens a partir de palavras-chaves. Contudo, o processo de anotação de imagens (atribuição manual da descrição textual) apresenta diversos problemas: usuários diferentes podem utilizar palavras diferentes para descrever uma mesma característica da imagem, a interpretação do conteúdo visual pode variar de acordo com o conhecimento, o objetivo, a experiência e a percepção de cada usuário [Torres 2006]. Além disso, essa abordagem é inviável em grandes bases de imagens.

Diversas aplicações vêm sendo desenvolvidas utilizando outra abordagem, baseada no conteúdo visual. Essa abordagem, antes reservada ao mundo da pesquisa, atualmente é aplicada nos seguintes exemplos: no site TinEye[1], da empresa Idée Inc., é possível submeter uma imagem e descobrir a origem dela, como ela está sendo usada, se existem versões modificadas, ou encontrar versões com uma resolução mais alta. Segundo a Idée, o TinEye é o primeiro sistema de busca de imagens na Web a usar tecnologia de identificação de imagem ao invés de palavras-chaves; o Goggles[2], da Google, permite a busca de informações sobre pontos de referência (ex.: monumentos), livros, obras de arte, vinhos, logomarcas a partir de fotografias capturadas de dispositivos móveis; a Nokia, por sua vez, propôs um desafio ousado na conferência ACM

[1] Disponível em: <http://www.tineye.com>. Acesso em: 27 abr. 2011.
[2] Disponível em: <http://www.google.com/mobile/goggles>. Acesso em: 27 abr. 2011.

Multimedia de 2009[3], que consiste em determinar onde (localização) e como (orientação) uma dada fotografia foi tirada. Além dessas aplicações, a busca baseada no conteúdo pode ser aplicada em: identificação e autenticação de documentos multimídia; identificação e acompanhamento de reproduções; detecção de violações dos direitos autorais e de cópia; detecção de obras de arte roubadas; reconhecimento de objetos, pessoas e cenas; classificação baseada no conteúdo visual (sem necessidade de palavras-chaves) em coleções de documentos multimídia (fotos, vídeos, áudio, etc.); e muitas outras. Grande parte disso motivado pela disseminação de dispositivos móveis, como telefones celulares com câmeras de alta qualidade e grande capacidade de armazenamento, PDA´s (*Personal Digital Assistants*), *tablets*, etc. O compartilhamento dos documentos nas redes sociais também impulsiona a criação de aplicações inovadoras.

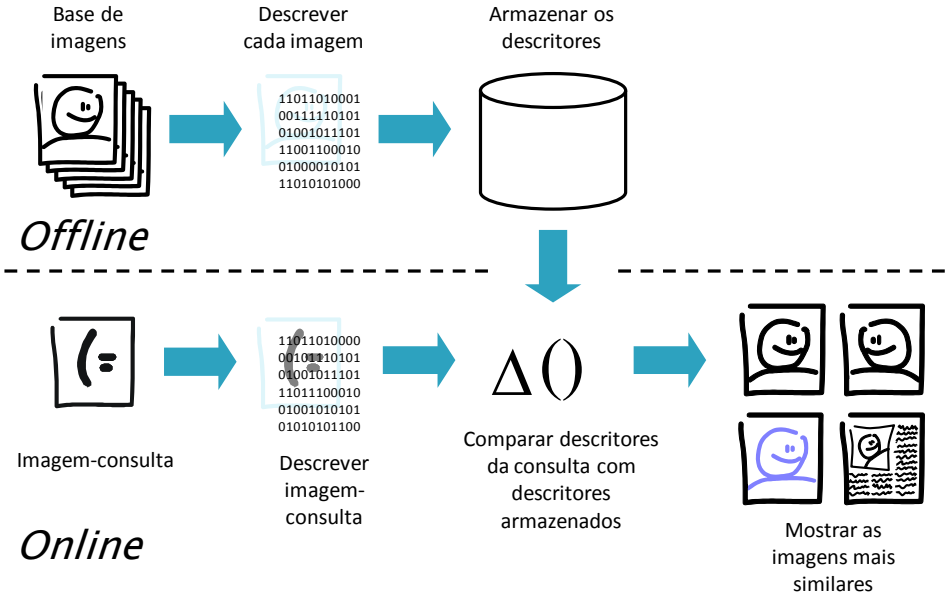

Figura 1.1: Arquitetura típica de um sistema de recuperação de imagens por conteúdo (reproduzida com autorização [Valle 2008]).

1.2 Busca de imagens, descritores e indexação

Os sistemas de busca de imagens baseados no conteúdo são bastante promissores. Esses sistemas, ao invés de usarem as descrições textuais, utilizam propriedades visuais, como cor, textura e forma [Torres 2008]. A Figura 1.1 mostra a arquitetura básica de um sistema de recuperação de imagens por conteúdo (*Content-Based Image Retrieval* — CBIR). Nesses sistemas, a busca é mediada pelo uso de *descritores* que, segundo Torres *et al.* [Torres 2006], são compostos por um

[3] Disponível em: <http://www.sigmm.org/archive/MM/mm09/MMGC.aspx.htm>. Acesso em: 27 abr. 2011.

algoritmo de extração de características, que gera um *vetor de características* para representar uma propriedade da imagem e uma medida de similaridade, que determina a similaridade entre duas imagens a partir de seus vetores de características.

O processo de busca funciona da seguinte forma. Em uma etapa *offline*, os vetores de características das imagens da base são computados e armazenados (Figura 1.1). Quando o usuário faz uma consulta, fornecendo uma imagem de consulta, o seu vetor de características é computado e uma função de distância, como a distância Euclidiana, é usada para avaliar a similaridade entre a imagem de consulta e as imagens da base. Por fim, as imagens mais similares são retornadas para o usuário.

O emparelhamento do vetor de características da consulta com os vetores de características da base é normalmente implementado através da *busca por similaridade*, que consiste em encontrar os vetores mais similares ao vetor da consulta. As formas mais comuns de especificar uma consulta nesse tipo de busca são: a busca por abrangência (*range search* [Samet 2006]) e a busca pelo vizinho mais próximo ou pelos *k* vizinhos mais próximos (busca kVMP, *k nearest neighbors search* [Samet 2006], ou kNN *search*). Esses problemas de busca podem ser resolvidos com um algoritmo sequencial, mas em grandes bases de imagens esse processo precisa ser acelerado com a utilização de estruturas de indexação.

Os vetores de características podem ter muitas dimensões (o *Scale Invariant Feature Transform* — SIFT [Lowe 2004], por exemplo, têm 128). A alta dimensionalidade é um dos principais desafios na indexação dos vetores de características. A tão discutida "maldição da dimensionalidade" [Bellman 1961] está associada a uma série de efeitos que afetam gravemente o desempenho dos métodos, por exemplo: o número de partições do espaço, possíveis em um índice, cresce exponencialmente com o número de dimensões. Acima de 10 dimensões, geralmente é necessário recorrer aos métodos aproximados, sacrificando a eficácia em troca da eficiência.

De acordo com Valle e outros [Valle 2010], os requisitos de um método de indexação multidimensional para grandes bases são:

- Apresentar um bom compromisso entre eficácia e rapidez nos espaços de alta dimensionalidade;
- Ser bem adaptado ao disco, ou seja, usar com parcimônia o acesso aleatório aos dados;
- Permitir a inserção e remoção de dados sem que isso deteriore o desempenho.

Os métodos baseados nas curvas de Peano (curvas de preenchimento do espaço [Sagan 1994]) constituem um caminho promissor, pois atendem bem às duas últimas exigências acima, quando comparados com outras abordagens [Valle 2008]. Basicamente, as curvas de Peano são curvas fractais que permitem mapear pontos de um espaço multidimensional em uma única dimensão, preservando as relações de proximidade dos pontos no espaço. O grande problema desses métodos é a existência de zonas de descontinuidade nas curvas, onde suas propriedades de quase-preservação da vizinhança são violadas. A probabilidade de um ponto cair nessas regiões de descontinuidade aumenta consideravelmente com a dimensionalidade.

1.3 Objetivos

O objetivo deste trabalho é a criação de uma solução eficiente para a indexação de vetores de características de alta dimensionalidade, em grandes bases de dados, a partir da utilização das curvas de preenchimento do espaço. A pesquisa em métodos de indexação escaláveis é indispensável para o desenvolvimento de aplicações que lidam com um enorme volume de dados multimídia. Esse é um dos tópicos incluídos em um grande desafio de pesquisa proposto pela Sociedade Brasileira de Computação (SBC) — [Carvalho 2006]: “Gestão da informação em grandes volumes de dados multimídia distribuídos”.

A avaliação empírica da solução proposta e a comparação do seu desempenho com alguns métodos do estado da arte é outro objetivo. Além disso, objetiva-se testar a escalabilidade da solução, variando o tamanho da base de dados.

1.4 Contribuições

A contribuição principal deste trabalho é um método de indexação, chamado MONORAIL, que utiliza uma curva de Peano e múltiplos representantes para os dados. A criação dos representantes leva em conta as propriedades geométricas da curva. Isso resulta em um ganho significativo na eficácia da busca por similaridade, em comparação à técnica usada como referência.

Uma contribuição não trivial é o rigor experimental usado tanto na exploração do espaço paramétrico dos métodos quanto na comparação entre eles. Os experimentos foram cuidadosamente projetados para garantir resultados estatisticamente significativos. A metodologia é geral o suficiente para ser usada em outras avaliações da busca por similaridade.

Finalmente, apresentamos uma versão do método proposto e de outros usados nas comparações, com uma curva fractal mais simples do que a adotada nas demais técnicas baseadas nesses tipos de curvas. Experimentos mostraram, por exemplo, que a utilização da curva Z-order [Morton 1966] reduz consideravelmente o tempo de criação dos índices, mas com uma degradação mínima na eficácia dos métodos.

1.5 Estrutura do texto

A dissertação está organizada da seguinte forma. No capítulo 2 são definidos os conceitos da busca por similaridade. Um estudo sobre o funcionamento dos métodos que serviram de referência para esta pesquisa também é feito nesse capítulo.

O método MONORAIL é apresentado no capítulo 3. O processo de criação do índice é detalhado, incluindo as características das curvas de Peano que possibilitaram a inovação na maneira de geração dos representantes.

No capítulo 4, toda a metodologia usada nos experimentos é descrita e os resultados obtidos são apresentados.

As conclusões finais e a algumas sugestões para o aperfeiçoamento do método são feitas no capítulo 5.

Capítulo 2

Fundamentos e Estado da Arte

Neste capítulo são definidos os conceitos pertinentes à busca por similaridade. Alguns efeitos da tão discutida "maldição da dimensionalidade", que afeta profundamente o funcionamento desse tipo de busca, são abordados na seção 2.4.

As diversas técnicas de indexação para acelerar a busca por similaridade são relacionadas na seção 2.5, juntamente com as referências para os trabalhos sobre o funcionamento dos métodos de cada uma delas.

A seção 2.6 apresenta uma introdução às curvas de preenchimento do espaço ou curvas de Peano. Na sequência, apresentamos um estudo dos métodos aproximados de busca por similaridade baseados nessas curvas.

Por fim, detalhamos os principais métodos que serviram de base para a realização desta pesquisa.

2.1 Descritores multimídia e seus espaços métricos

Nos sistemas de recuperação de informação multimídia baseados no conteúdo a busca de um documento é feita por meio de um descritor multimídia. Em um sistema CBIR (*Content-Based Image Retrieval*), segundo Torres *et al.* [Torres 2008], um *descritor* é composto por um algoritmo de extração de características, que gera um *vetor de características* para representar uma propriedade da imagem (ex.: cor, textura, forma, etc.) e uma medida de similaridade, que determina a similaridade entre duas imagens a partir de seus vetores de características. Os descritores podem ser globais, quando um único vetor de características é associado à imagem, ou locais, quando vários vetores de características são usados para representá-la. Exemplos típicos são os histogramas de cor [Swain 1991], que são globais e o SIFT [Lowe 2004], que é local. Os descritores locais são mais robustos a transformações (ex.: corte, rotação, etc.),

entretanto, geram um problema de eficiência, pois várias comparações precisam ser feitas para responder uma única consulta [Valle 2008].

A função de similaridade varia de acordo com as propriedades do espaço no qual os dados estão inseridos. Nos *espaços métricos*, a similaridade é dada por funções de distância que respeitam as propriedades de não negatividade, identidade, simetria e desigualdade triangular. Nos *espaços vetoriais*, os dados são representados por vetores de *d* dimensões. Esses espaços podem se tornar métricos com o uso de uma norma, como as métricas L_p (ou Minkowski), onde $p = 1, \ldots, \infty$. O escopo deste trabalho será o *espaço Euclidiano*, definido pela distância Euclidiana ou métrica L_2. Em algumas aplicações, o custo de calcular a distância entre os dados é muito alto (por exemplo, *earth mover's distance* [Rubner 2000]). Nesses casos, uma abordagem denominada *embeddings* (por exemplo, FastMap [Faloutsos 1995]) permite embutir os dados em um espaço vetorial e fazer o cálculo através de uma função mais simples, de forma que a distância entre os dados embutidos seja próxima da distância no espaço original.

2.2 Busca por similaridade

Nos bancos de dados tradicionais, cujos registros são compostos por campos de uma única dimensão (por exemplo, código, nome, data, etc.), as buscas quase sempre são feitas com o objetivo de retornar exatamente os registros especificados pelas consultas. Em uma base de imagens nem sempre existe uma cópia idêntica da imagem de consulta na base. Nesse caso, emprega-se a busca por similaridade que permite encontrar as imagens mais similares à consulta.

Existem diversas formas de especificar uma consulta na busca por similaridade, como a busca por abrangência (*range search* [Samet 2006]), a busca pelos *k* vizinhos mais próximos (*k nearest neighbors search* [Samet 2006], ou *kNN search*), entre outras. O escopo deste trabalho será somente a busca pelo vizinho mais próximo e sua versão mais geral, a busca pelos *k vizinhos mais próximos* (busca kVMP).

A definição formal da busca pelo vizinho mais próximo é encontrada a seguir:

- Dado um espaço vetorial domínio *D* com *d* dimensões;
- uma base de dados *B*, com os elementos $b_1, b_2, \dots, b_n \in D$;
- um vetor-consulta $q \in D$;
- uma função de distância Δ: $D \times D \rightarrow \mathbb{R}$, que por padrão será a distância Euclidiana;
- a busca pelo vizinho mais próximo de q consiste em encontrar o elemento $s \in B \mid \forall b \in B - \{s\},\ \Delta(q, s) \leq \Delta(q, b)$.

E para a busca pelos *k* vizinhos mais próximos de *q*:

- Dados *D*, *B*, *q* e Δ definidos acima;
- a busca kVMP consiste em encontrar o conjunto $S = \{s_1, s_2, \dots, s_k\} \mid \forall b \in B - S$, $\Delta(q, s_i) \leq \Delta(q, b)$, para $1 \leq i \leq k$ e $\Delta(q, s_i) \leq \Delta(q, s_{i+1})$, para $1 \leq i < k$.

A solução óbvia para ambos os casos é a busca sequencial. Nela, o vetor-consulta é comparado com todos os vetores da base de dados. Infelizmente, isso é aceitável somente para bases muito pequenas. Em grandes bases é preciso utilizar algum tipo de índice, que permita responder a consulta sem a necessidade de percorrer todos os elementos.

Nos espaços de alta dimensionalidade, o desenvolvimento de um método de indexação que resolva o problema, mais rápido que a busca sequencial, é uma tarefa bem difícil (a ser discutida na seção 2.4). Uma alternativa é utilizar métodos aproximados, que priorizam o tempo de busca em detrimento da eficácia.

A definição da busca aproximada pelo vizinho mais próximo que será adotada neste texto é:

- Dadas as definições anteriores;
- e um fator de aproximação $(1 + \varepsilon)$, onde $\varepsilon > 0$;
- a busca aproximada pelo vizinho mais próximo consiste em encontrar o elemento $s' \in B \mid \Delta(q, s') \leq (1 + \varepsilon)\Delta(q, s)$;
- e a busca aproximada kVMP consiste em encontrar o conjunto $S' = \{s'_1, s'_2, \dots, s'_k\} \mid \Delta(q, s'_i) \leq (1 + \varepsilon)\Delta(q, s_i)$, para $1 \leq i \leq k$ e $\Delta(q, s'_i) \leq \Delta(q, s'_{i+1})$, para $1 \leq i < k$.

O lado esquerdo da Figura 2.1 mostra a busca pelo vizinho mais próximo. O ponto escuro é o vizinho mais próximo da consulta (rotulada com q) e δ indica a distância entre eles. No lado direito temos a busca aproximada pelo vizinho mais próximo, levando em conta o fator de aproximação $(1 + \varepsilon)$.

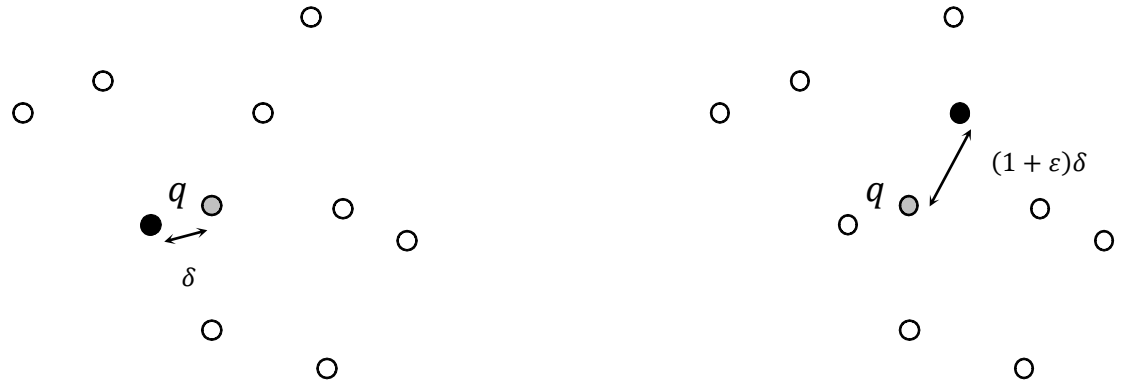

Figura 2.1: O lado esquerdo mostra a busca exata pelo vizinho mais próximo e o direito a interpretação geométrica da busca aproximada.

2.3 Índices

Os índices trabalham particionando o conjunto de elementos da base de dados em páginas, de forma que o algoritmo de busca tenha que acessar um número limitado dessas páginas.

A medida mais importante do desempenho de um índice é a *seletividade*, quanto maior a seletividade, menor será a quantidade de páginas acessadas para responder uma consulta. Portanto, essa medida deve ser alta, para minimizar o número de acessos aleatórios (que tem um custo alto quando feitos em discos).

De maneira inversa, podemos utilizar um parâmetro para medir o custo de responder uma consulta: o *PD* (*Probe Depth* [Valle 2008]), que é o número de elementos da base de dados examinados para se obter a resposta.

2.4 A "maldição da dimensionalidade"

A expressão "maldição da dimensionalidade" foi criada por Bellman [Bellman 1961] para representar a dificuldade de resolver problemas de otimização em alta dimensionalidade, particionando o espaço de soluções. Uma série de efeitos matemáticos [Böhm 2001] que ocorrem quando a dimensionalidade dos dados cresce e que afetam a indexação foram associados a essa expressão, entre esses estão:

- O número de partições do espaço, possíveis em um índice, cresce exponencialmente com o número de dimensões. Se cada dimensão fosse dividida ao meio, por exemplo, o número de partições resultante seria uma potência de 2. Durante a busca, a quantidade de acessos a essas partições poderia fazer com que o tempo fosse superior ao da busca sequencial;
- A chance de um ponto cair perto de uma fronteira entre duas partições de um índice aumenta consideravelmente com a dimensionalidade. Dessa forma, o número de partições que precisam ser visitadas, para responder uma consulta, também é aumentado;
- A diferença entre a distância de um ponto de consulta para o vizinho mais próximo e para o mais distante torna-se cada vez menor à medida que a dimensionalidade aumenta (Figura 2.2). Isso prejudica a busca pelo vizinho mais próximo, pois uma perturbação poderia fazer com que o mais próximo fosse trocado pelo mais distante. Na versão aproximada da busca, o fator de aproximação precisaria ser cada vez menor para evitar essa troca.

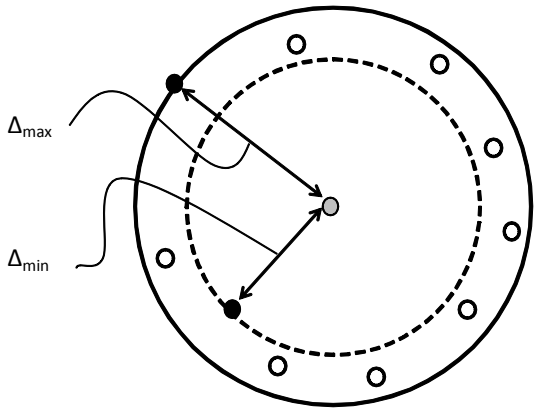

Figura 2.2: A diferença entre Δ_{max} e Δ_{min} torna-se cada vez menor quando a dimensionalidade cresce [Moënne-Loccoz 2005].

Para um número baixo de dimensões, até 10, a maioria dos métodos não é muito afetada pelos efeitos anteriores, mas para dimensionalidades maiores é necessário recorrer aos métodos aproximados, sacrificando a eficácia em troca da eficiência.

2.5 Classificação dos métodos de indexação para a busca kVMP

Um estudo completo dos métodos de indexação, usados para acelerar a busca pelos *k* vizinhos mais próximos, pode ser encontrado em [Gaede 1998] até o final dos anos 1990, e mais recentemente em [Castelli 2002], [Samet 2006] e [Valle 2008]. Em [Markov 2008] pode ser visualizada uma árvore genealógica dos métodos atualizada até o ano de 2006.

Seguindo a classificação de Valle [Valle 2008], os métodos de indexação para a busca kVMP podem ser agrupados nas seguintes famílias:

- **Particionamento do espaço**: a KD-Tree [Bentley 1975, Friedman 1976] é o método clássico, que é uma árvore binária, na qual em cada nível uma dimensão diferente é usada para separar os nós de cada sub-árvore. A 3-Way Trees [Valle 2007], uma variação mais recente, é uma árvore ternária, com duas sub-árvores equivalentes à KD-Tree e uma terceira contendo uma parte dos nós das anteriores. Essa redundância evita que duas sub-árvores tenham que ser visitadas, quando a consulta cai em uma região próxima da fronteira entre elas;
- **Particionamento dos dados**: a R-Tree [Guttman 1984] é uma estrutura similar à árvore B [Bayer 1972]. Os dados na R-Tree são representados por hiper-retângulos. Os ponteiros dos elementos apontam para sub-árvores, cujos hiper-retângulos dos elementos filhos estão todos contidos no hiper-retângulo do elemento pai. A SR-Tree [Katayama 1997] utiliza a interseção entre hiper-retângulos e hiper-esferas para determinar as regiões representadas pelos nós da árvore;
- **Clusterização**: no CLINDEX [Li 2002], os elementos similares são agrupados progressivamente, utilizando um algoritmo chamado *cluster-forming* (ilustrado no próprio artigo). A DAHC-tree [Almeida 2010] é um método mais recente;

- **Métodos métricos**: são capazes de manipular dados não vetoriais, uma vez que utilizam somente a distância entre os dados. O método pioneiro é a M-Tree [Ciaccia 1997], uma árvore balanceada, na qual a desigualdade triangular é usada para descartar os nós que não precisam ser visitados durante a busca. A Slim-Tree [Traina 2000] é uma variação, com um menor número de acessos em disco do que a M-Tree, e a Omni-family [Traina 2007] é uma referência mais recente. Um estudo específico dessa família foi feito por Zezula *et al.* [Zezula 2005];
- **Aproximação dos dados**: esses métodos criam aproximações geométricas (usando poucos *bits* em cada dimensão) para os pontos, que são percorridas sequencialmente para filtrar os candidatos. Em seguida, os candidatos são usados para obter os vizinhos mais próximos. Exemplos dessa família de métodos incluem o VA-file [Weber 1997] e o LPC-file [Cha 2002];
- **Projeção e *hashing*:** projetam os dados em linhas retas como o MEDRANK [Fagin 2003], ou em estruturas piramidais [Berchtold 1998, Zhang 2004], ou linhas retas com particionamento como a NV-Tree [Lejsek 2009], ou fractais (ver seção 2.7). No caso de *hashing* o LSH [Indyk 1998, Gionis 1999, Datar 2004] é o método clássico. Nesse método, a ideia é usar funções de *hashing* que associam os dados, indicando a posição de armazenamento desses dados, ao invés de indexar as chaves. Os dados próximos no espaço tendem a ser associados às mesmas chaves, enquanto dados distantes tendem a ser associados a chaves distintas. Uma revisão desse método foi feita por Shakhnarovich *et al.* [Shakhnarovich 2006].

Apesar da vasta literatura sobre indexação multidimensional, poucos métodos (a NV-Tree [Lejsek 2009], o Multicurves [Valle 2008b], o método de Mainar-Ruiz *et al.* [Mainar-Ruiz 2006], o método de Liao *et al.* [Liao 2001]) satisfazem as características exigidas em um contexto multimídia em grande escala: ter bom desempenho para dados de alta dimensionalidade, ser bem adaptado à memória secundária e ser dinâmico. O foco deste trabalho são os métodos de indexação para a busca aproximada kVMP baseados nas curvas de preenchimento do espaço, que são dinâmicos e funcionam bem em disco.

2.6 Curvas de preenchimento do espaço

Curvas de preenchimento do espaço [Sagan 1994] ou curvas de Peano[4] [Peano 1890] são curvas fractais que preenchem todo o espaço, permitindo mapear pontos de um espaço multidimensional em uma única dimensão.

A Figura 2.3 mostra três exemplos bidimensionais de curvas de preenchimento do espaço, a de Hilbert [Hilbert 1891], a de Lebesgue ou "Z-order" e a de Sierpiński. Três iterações de cada uma delas são apresentadas. O espaço contínuo, no caso o plano, é preenchido integralmente quando o número dessas iterações tende ao infinito.

Essas curvas têm a capacidade de preservar as relações de ordem entre a proximidade dos pontos no espaço, ou seja, pontos que são mapeados próximos na curva tendem a corresponder a pontos próximos no espaço. Essa propriedade é aproveitada na indexação dos vetores de características.

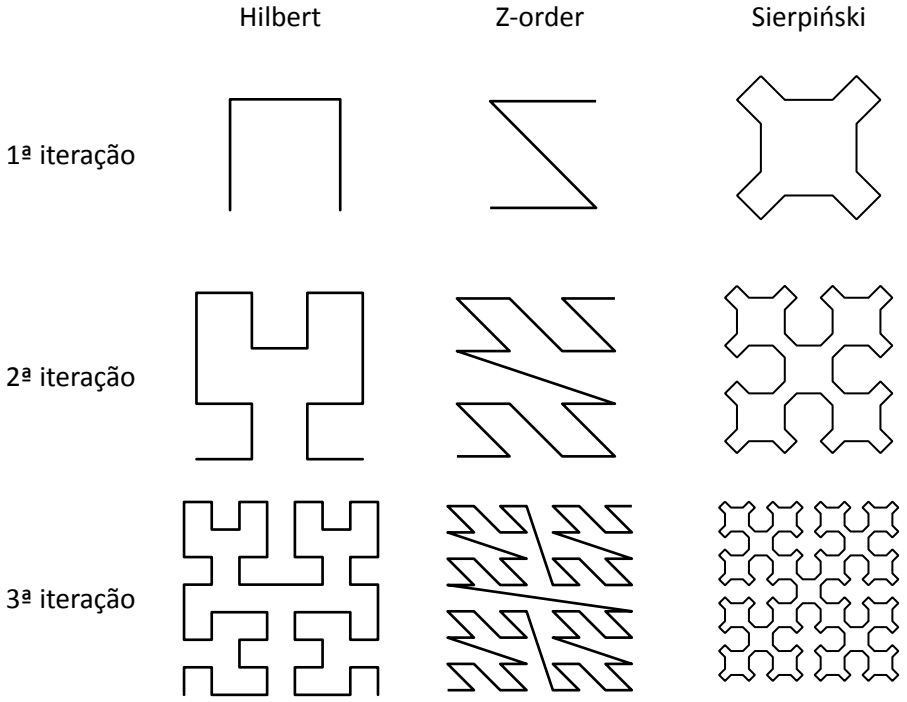

Figura 2.3: Três exemplos bidimensionais de curvas de Peano: a de Hilbert, a de Lebesgue ou "Z-order" e a de Sierpiński. Três iterações de cada curva estão presentes: o plano é preenchido integralmente quando o número de iterações tende ao infinito.

[4] A nomenclatura "curvas de Peano" refere-se, em português, tanto à família das curvas de preenchimento do espaço, quanto a uma formulação específica destas curvas, proposta por Peano.

2.7 O estado da arte da busca aproximada kVMP baseada nas curvas de Peano

A primeira referência explícita do uso de curva de Peano na busca por similaridade foi feita por Faloutsos e Roseman [Faloutsos 1988, Faloutsos 1989], em um método que mapeia pontos do espaço multidimensional para pontos na curva, obtendo uma coordenada de uma dimensão que representa suas posições relativas dentro da curva, nomeada aqui de *chave-estendida*. A chave-estendida é usada então, para realizar a busca por similaridade de forma monodimensional. A Figura 2.4 mostra um exemplo do mapeamento de 7 pontos do espaço bidimensional (coordenadas à esquerda) em uma única dimensão (coordenadas à direita), utilizando a curva Z-order.

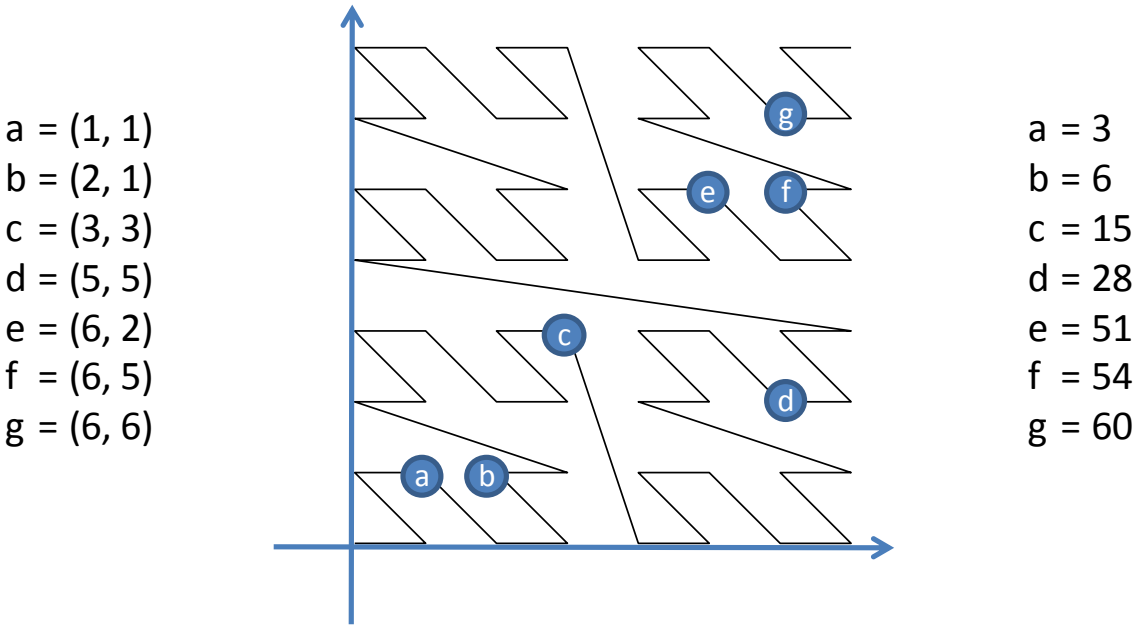

Figura 2.4: Exemplo do mapeamento de pontos de um espaço bidimensional em uma única dimensão, utilizando a curva Z-order.

Os autores fizeram experimentos para comparar três curvas: a de Hilbert, a curva Z-order, que eles chamaram de Peano, e a *Reflected Binary Gray-code* (RBG) [Faloutsos 1986]. O desempenho das curvas na busca por abrangência foi medido pela quantidade média de agrupamentos acessados para responder às consultas. A busca pelo vizinho mais próximo foi avaliada pela distância média do vizinho mais distante dentro de um intervalo na curva (ambas as medidas definidas pelos autores [Faloutsos 1989]). Os resultados mostraram que nas duas formas de busca o desempenho da curva de Hilbert foi superior ao das demais.

O problema desse método é a existência de “zonas de descontinuidade” nas curvas, onde suas qualidades de quase-preservação da vizinhança são perdidas. Isso faz com que pontos próximos dessas regiões não sejam necessariamente mapeados para pontos próximos na curva. Na Figura 2.5 os pontos no centro do espaço estão mais afastados na curva do que os pontos no quadrante inferior-direito, devido às regiões de descontinuidade (indicadas pelas linhas tracejadas).

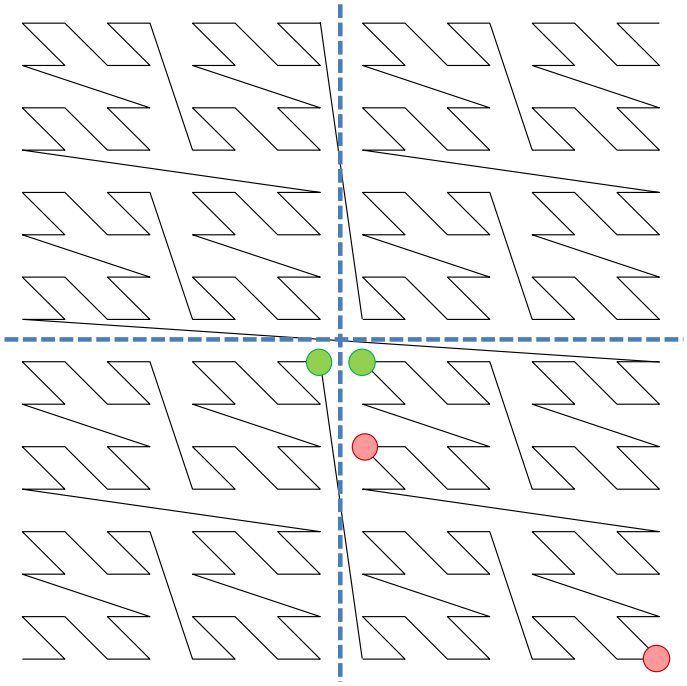

Figura 2.5: O problema das regiões de descontinuidade na curva Z-order. Os pontos no centro do espaço estão mais afastados na curva do que os pontos no quadrante inferior-direito.

Para tentar resolver esse problema das regiões de descontinuidade, o método *CurveIx,* proposto por Shepherd *et al.* [Shepherd 1999], utiliza múltiplas curvas de Hilbert mapeadas após a aplicação de um número aleatório de rotações e translações nos pontos do espaço. O objetivo é que em pelo menos uma delas o problema não aconteça. Nos experimentos, os autores utilizaram 40 curvas (examinando 15, 23 e 31 candidatos por curva) e compararam a eficácia e a eficiência do método com a busca sequencial, em uma base com 10.000 vetores de características de 256 dimensões. A proporção do número de vizinhos corretos entre os 10 retornados, para cada uma das 25 consultas, foi usada como medida de eficácia, que nos melhores resultados atingiu 90% em média. A eficiência do método, medida pelo número de distâncias calculadas, foi superior à da busca sequencial a partir de uma base de dados com 10.000 ou mais imagens. O número de

curvas necessário para manter a eficácia, em diferentes bases, e o número de elementos examinados em cada uma delas não foram estudados pelos autores.

Em uma abordagem apresentada por Liao e outros em 2001 [Liao 2001], foram demonstrados tanto o número quanto o tipo de transformações nos pontos, necessários para obter um limite superior do erro de aproximação na busca pelo vizinho mais próximo. Para um espaço de d dimensões, a ideia consiste em usar $d + 1$ representações dos pontos deslocadas de uma unidade na direção de cada dimensão. Os autores provaram que, com essas condições, o fator de aproximação é $O(d^{1+1/t})$, onde $t = 1, \dots, \infty$ indica o tipo da distância métrica L_1, L_2..., e assim sucessivamente. Isso pode ser conseguido com um número máximo de $(d + 1)\log_p n$ páginas acessadas, onde p é a ordem da árvore B, utilizada para armazenar as listas ordenadas induzidas pelas curvas. Os experimentos foram feitos em duas bases: uma com 67.040 vetores de características de 32 dimensões e outra com 270.000 vetores de características de 60 dimensões. No primeiro conjunto deles, 1.000 consultas pelo vizinho mais próximo foram realizadas em cada base e a eficiência, medida pelo número de páginas acessadas, foi comparada com o método SR-Tree [Katayama 1997]. O método proposto foi consideravelmente mais rápido que a SR-Tree. Em um segundo conjunto de experimentos, os autores mostraram que o erro de aproximação médio foi bem menor que o limite teórico. Na comparação com o desempenho do LSH [Gionis 1999] os autores especulam sobre a superioridade do método, mas sem resultados experimentais conclusivos.

Outra alternativa de utilização de múltiplas curvas a partir de versões transformadas dos pontos foi proposta por Pérez e Vidal [Pérez 1998]. No método baseado em *Extended General Spacefilling Heuristic* (EGSH), uma quantidade de mapeamentos para curvas de Sierpiński são feitos após aplicar rotações, normalizações e translações nos pontos originais. Semelhante ao trabalho do Liao *et al.* [Liao 2001], os autores também sugeriram a utilização de um número de curvas igual ou proporcional ao número de dimensões. Os coeficientes usados nas transformações foram determinados empiricamente a partir de um conjunto de vetores aleatórios. Os autores fizeram diversos experimentos para avaliar a eficácia do método variando o tamanho da base de dados (sintética) e o número de dimensões.

O Multicurves, proposto por Valle *et al.* [Valle 2008b], também usa múltiplas curvas de Peano. A diferença em relação aos anteriores é que cada curva no Multicurves é responsável

somente por uma fração das dimensões do espaço. Uma descrição mais detalhada deste método será feita na seção 2.8.2.

Todos esses métodos utilizam múltiplas curvas, apresentando, portanto, a necessidade de um acesso aleatório por curva.

Asano *et al.* [Asano 1997] desenvolveram uma curva de preenchimento do espaço com o objetivo de minimizar o número de acessos aleatórios nas consultas por abrangência. O relacionamento entre a curva Z-order, a curva RBG e a curva de Hilbert foi estudado por Chen e Chang [Chen 2005]. Em uma avaliação da busca pelo vizinho mais próximo, a curva Z-order apresentou o menor tempo de computação entre as três e a curva de Hilbert obteve o menor tempo de acesso ao disco e o menor tempo total (computação mais acesso ao disco), devido às suas propriedades de agrupamento. Esses resultados confirmaram a superioridade da curva de Hilbert, que já havia sido constatada por Faloutsos e Roseman [Faloutsos 1989].

Um método bem diferente foi proposto por Mainar-Ruiz e Pérez-Cortés [Mainar-Ruiz 2006]: a utilização de múltiplos *representantes*[5] para os pontos em uma única curva ao invés de utilizar múltiplas curvas. Este método será discutido com detalhes na seção 2.8.1.

A Tabela 2.1 sintetiza as referências dos métodos de indexação multidimensional baseados nas curvas de Peano.

[5] Representante foi usado neste texto como tradução de *representative,* embora a palavra dublê explique melhor o seu papel.

Referência	Múltiplas curvas / única curva	Curvas	Solução para o problema das regiões de descontinuidade
[Faloutsos 1989]	Única curva	Z-order, RBG e Hilbert	Não fornece.
[Pérez 1998]	Múltiplas curvas	Sierpiński	Uma quantidade de mapeamentos, proporcional a d (dimensionalidade dos pontos), são feitos após aplicar rotações, normalizações e translações nos pontos originais
[Shepherd 1999]	Múltiplas curvas	Hilbert	Aplicação de uma quantidade aleatória de rotações e translações nos pontos do espaço
[Liao 2001]	Múltiplas curvas	Hilbert	$d + 1$ mapeamentos feitos após deslocar os pontos em uma unidade (sucessivamente) na direção de cada dimensão
[Mainar-Ruiz 2006]	Única curva	Sierpiński	Múltiplos representantes gerados por uma perturbação dos pontos originais, através de pequenas variações aleatórias nos valores em cada eixo — método descrito na § 2.8.1
[Valle 2008b]	Múltiplas curvas	Hilbert	Cada curva é responsável somente por uma fração das dimensões do espaço — método descrito na § 2.8.2

Tabela 2.1: Resumo dos métodos de indexação multidimensional baseados nas curvas de Peano.

2.8 Métodos de referência

Nesta seção são abordados os métodos de Mainar-Ruiz e Pérez-Cortés e o Multicurves.

2.8.1 O método de Mainar-Ruiz e Pérez-Cortés

Neste método, proposto por Mainar-Ruiz e Pérez-Cortés [Mainar-Ruiz 2006], o problema das regiões de descontinuidade é atacado com a utilização de múltiplos representantes para os pontos em uma única curva ao invés de utilizar múltiplas curvas. Representantes são criados ao redor dos pontos originais, para que esses tenham mais chances de serem alcançados a partir da consulta. Na Figura 2.6 são mostrados dois pontos da base de dados, os círculos, e o ponto-consulta, a estrela. O mapeamento desses pontos é mostrado na curva de Hilbert, que possui zonas de descontinuidade semelhantes à curva de Sierpiński, usada originalmente pelos autores. O ponto central (verde-escuro) é o mais próximo da consulta, porém sua posição na curva é mais distante quando comparada à do ponto à esquerda (vermelho-escuro).

Os representantes são gerados a partir de uma pequena perturbação nos valores dos pontos originais em cada eixo. A Figura 2.7 ilustra um possível resultado do método, após a criação de 7 representantes para cada ponto da base. Os pontos verdes-claros são os representantes do ponto original verde-escuro (central) e os pontos vermelhos-claros representam o ponto original vermelho-escuro (mais à esquerda). No artigo, valores aleatórios no intervalo [-0,14; 0,14] foram somados aos valores em cada eixo (com os dados normalizados em [0; 1]). Depois do mapeamento, uma checagem é feita para eliminar os representantes repetidos dentro de uma faixa (50 vizinhos de cada lado da curva foi o parâmetro usado pelos autores).

O objetivo dos autores foi obter um resultado similar ao da utilização de múltiplas curvas, mas sem ter que fazer um mapeamento para cada curva, economizando tempo na construção do índice e, principalmente, reduzindo o número de acessos aleatórios em disco durante a busca.

Os experimentos feitos pelos autores mostraram um desempenho superior ao método com múltiplas curvas [Pérez 1998], utilizando-se como parâmetros 10 e 20 representantes no método proposto contra 3 e 5 curvas no método de Pérez e Vidal [Pérez 1998].

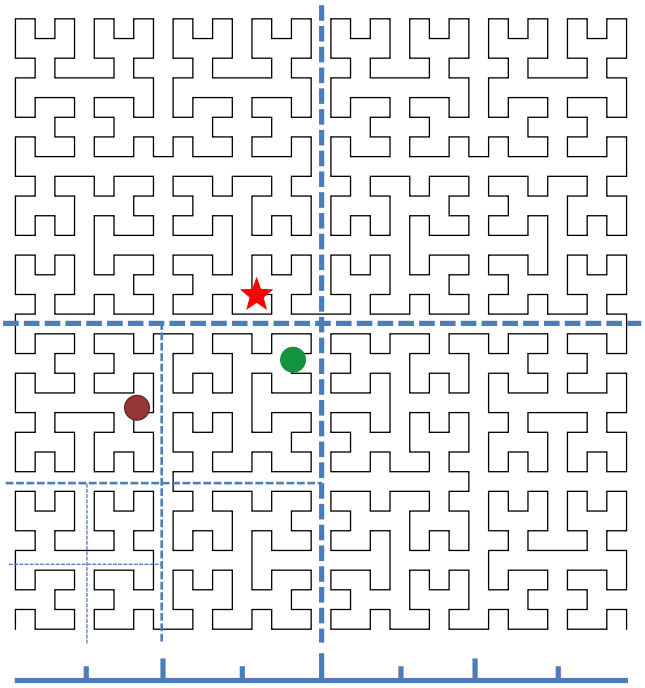

Figura 2.6: Exemplo de dois pontos da base de dados, os círculos, e o ponto-consulta, a estrela. O ponto central é o mais próximo da consulta, porém sua posição na curva é mais distante quando comparada à do ponto à esquerda.

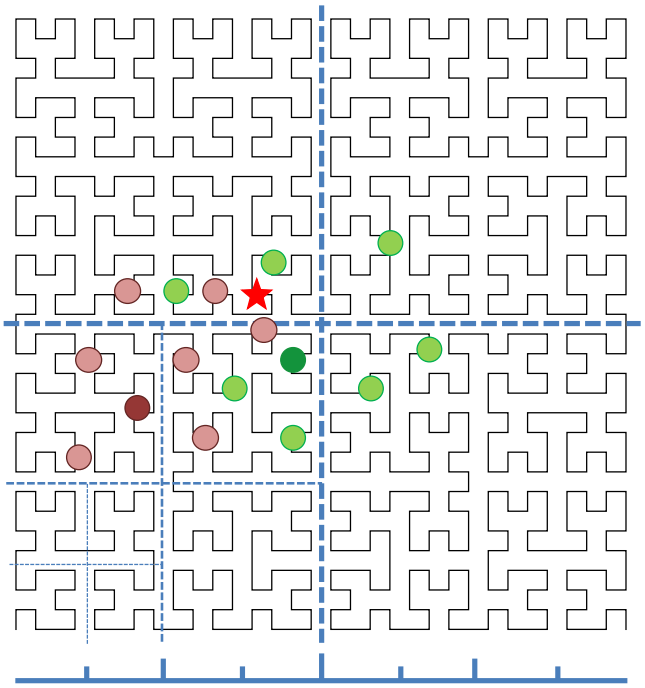

Figura 2.7: Possível resultado da criação dos representantes pelo método de Mainar-Ruiz *et al.* [Mainar-Ruiz 2006]. Esses novos pontos permitem que o verdadeiro vizinho mais próximo da consulta (estrela) seja encontrado na curva.

2.8.2 O método Multicurves

O método Multicurves [Valle 2008b] é baseado na utilização de múltiplas curvas de Peano de dimensionalidade reduzida. A ideia é projetar os dados em diversos sub-espaços, fazendo com que cada curva seja responsável somente por uma fração das dimensões do espaço. A redução de dimensionalidade ameniza o problema das regiões de descontinuidade.

Na implementação original do método, as dimensões dos dados são divididas uniformemente entre um certo número de curvas. Cada curva produz seu próprio mapeamento, gerando o sub-índice correspondente. A Figura 2.8 mostra um exemplo da divisão das dimensões do espaço da base de dados entre as dimensões dos sub-índices e o mapeamento desses sub-espaços, utilizando a curva Z-order.

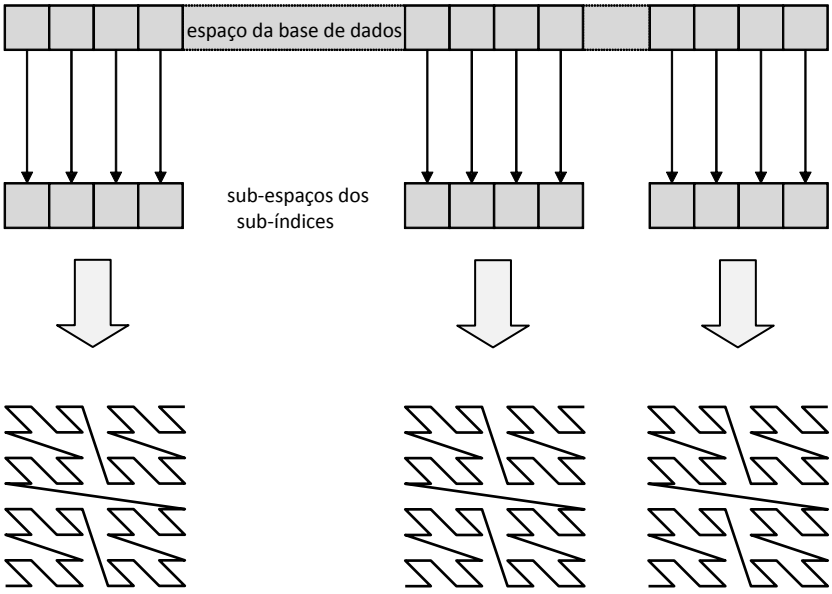

Figura 2.8: Um exemplo dos sub-índices do Multicurves com as dimensões correspondentes.

A busca é realizada em cada sub-índice e os resultados parciais são agregados para produzir a resposta final.

No artigo [Valle 2010], o Multicurves foi comparado ao LSH [Datar 2004], além dos métodos [Mainar-Ruiz 2006] e [Liao 2001]. O Multicurves e o LSH apresentaram um melhor compromisso entre a eficiência e a eficácia na busca aproximada kVMP que os demais e o Multicurves levou vantagem sobre o LSH, pelo menor número de acessos aleatórios realizados.

A curva de Hilbert foi escolhida pelos autores por apresentar características importantes para a busca kVMP [Mokbel 2003] e pela existência de um algoritmo rápido para computar as chaves-estendidas [Butz 1971]. Nós implementamos uma versão do Multicurves usando a curva Z-order, e a comparamos com a original (experimentos da seção 4.5).

Além disso, desenvolvemos uma extensão do Multicurves que permite a escolha arbitrária das dimensões, exemplificada pela Figura 2.9.

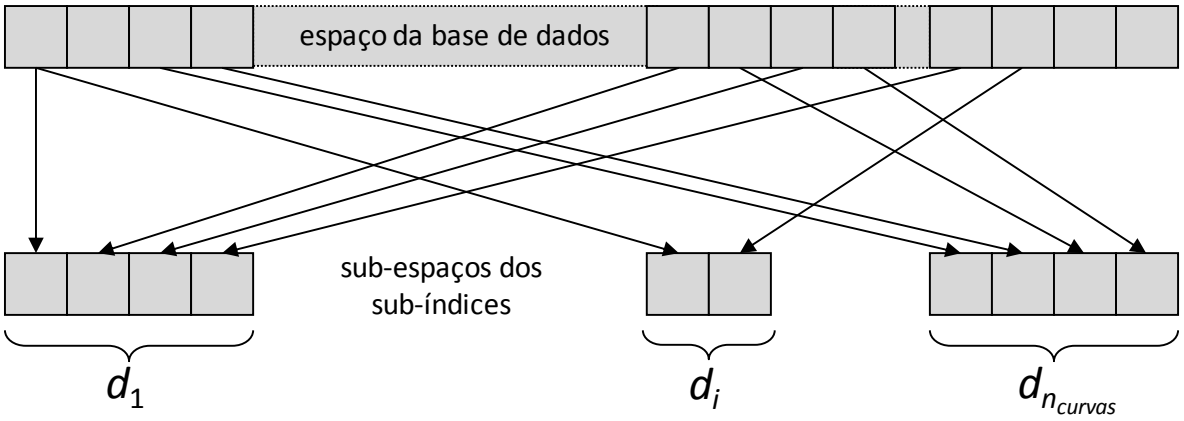

Figura 2.9: Um exemplo dos sub-índices do Multicurves utilizando a extensão que permite a escolha arbitrária das dimensões.

2.9 Conclusão

Em um sistema CBIR a busca de imagens é feita por meio de vetores de características, que são computados a partir de propriedades como cor, textura e forma. A busca por similaridade é realizada para encontrar os vetores mais similares a um vetor-consulta. A forma mais comum de especificar a consulta, nesse tipo de busca, é através da busca pelos *k* vizinhos mais próximos. A função usada para avaliar a similaridade varia de acordo com as propriedades do espaço em que os dados se encontram: espaço métrico, espaço vetorial ou espaço Euclidiano. O interesse neste trabalho é por esse último, que é definido pela distância L_2. Os *embeddings* podem ser utilizados para embutir dados complexos (distância *earth mover's distance* [Rubner 2000]) em espaços mais simples (distância Euclidiana).

As estruturas de indexação multidimensional, que são usadas para acelerar a busca kVMP, são prejudicadas por uma série de fenômenos, associados à expressão "maldição da dimensionalidade". Isso faz com que versões aproximadas sejam adotadas para permitir um bom compromisso entre eficácia e eficiência. Dentre as diversas técnicas de busca kVMP, os métodos

baseados nas curvas de Peano constituem um caminho promissor, pois são dinâmicos e bem adaptados à memória secundária.

As curvas de Peano possuem propriedades que permitem mapear pontos em um espaço multidimensional para uma única dimensão, preservando as relações de proximidade. Os métodos utilizam essa característica para reduzir a busca kVMP em alta dimensionalidade para uma busca monodimensional. O problema é que nem sempre os pontos próximos no espaço são mapeados para pontos próximos na curva, devido à existência de zonas de descontinuidade.

O grande desafio é resolver o problema das regiões de descontinuidade sem deteriorar a eficiência e a eficácia da busca kVMP. No método de Mainar-Ruiz e Pérez-Cortés [Mainar-Ruiz 2006] múltiplos representantes para os pontos são gerados aleatoriamente, em uma única curva. O Multicurves [Valle 2008b] procura fazer isso dividindo as dimensões entre diversas curvas. Esses dois métodos são as principais referências para este trabalho.

Capítulo 3

Método MONORAIL

Neste capítulo apresentamos a principal contribuição deste trabalho, o método de indexação multidimensional que chamamos de MONORAIL.

O método MONORAIL, que utiliza uma curva de Peano e múltiplos representantes para os dados, é descrito na seção 3.2. O processo de criação dos representantes leva em conta as propriedades geométricas da curva, otimizando a quantidade e o posicionamento desses representantes.

Na última seção são apresentados os detalhes de implementação do índice MONORAIL e do algoritmo utilizado para computar a chave-estendida com a curva Z-order.

3.1 O método MONORAIL

Nós desenvolvemos um novo método de criação dos representantes, chamado MONORAIL, levando em conta a desvantagem do método de Mainar-Ruiz e Pérez-Cortés [Mainar-Ruiz 2006] e as propriedades geométricas das curvas de Peano, detalhadas adiante na seção 3.2.1.

O método de indexação MONORAIL tem uma estrutura simples. O índice é construído como uma lista ordenada. Qualquer estrutura adequada à manipulação eficaz das listas pode ser empregada. Os pares <chave-estendida, dado> são inseridos nessa lista ordenada, onde a chave-estendida é a posição linear na curva, computada a partir das coordenadas do ponto no espaço. Não é necessário representar a curva explicitamente, basta computar as chaves-estendidas a partir das coordenadas do espaço. Existem algoritmos rápidos para fazer isso em diversas curvas (por exemplo, para a curva Z-order, basta intercalar os *bits* das coordenadas; para a curva de Hilbert, ver o artigo [Butz 1971]).

3.2 Detalhamento

A criação do índice exige um parâmetro, a multiplicidade, que corresponde ao número máximo de representantes que podem ser criados para cada ponto da base de dados. O processo de criação é simples e compreende a criação de diversos representantes para cada elemento da base de dados e sua inserção na lista ordenada.

3.2.1 Criação do índice

O MONORAIL é baseado na ideia de usar uma única curva de Peano com múltiplos representantes para cada ponto, mas de forma a evitar a poluição, maior desvantagem do método de Mainar-Ruiz e Pérez-Cortés [Mainar-Ruiz 2006], que os gera aleatoriamente. Na realidade, a geração dos representantes pode ser feita de forma muito mais ordenada observando onde a propriedade de preservação da vizinhança do mapeamento da curva de Peano é violada. Devido à sua construção recursiva, a maioria das curvas de Peano (incluindo a curva de Hilbert, a Z-order, etc.) é bem previsível nesse aspecto: a violação mais grave ocorre no plano que divide cada dimensão pela metade. Uma violação secundária e menos grave, ocorre nos dois planos que dividem a dimensão em quatro partes e assim sucessivamente. Na Figura 3.1 as descontinuidades principais da curva Z-order são mostradas pelas linhas tracejadas e barra inferior. O ponto-consulta (a estrela) poderia estar longe do ponto original (círculo sem rótulo) na curva, mesmo eles estando próximos no espaço. As descontinuidades principais da curva de Hilbert são mostradas, de maneira análoga, na Figura 3.2.

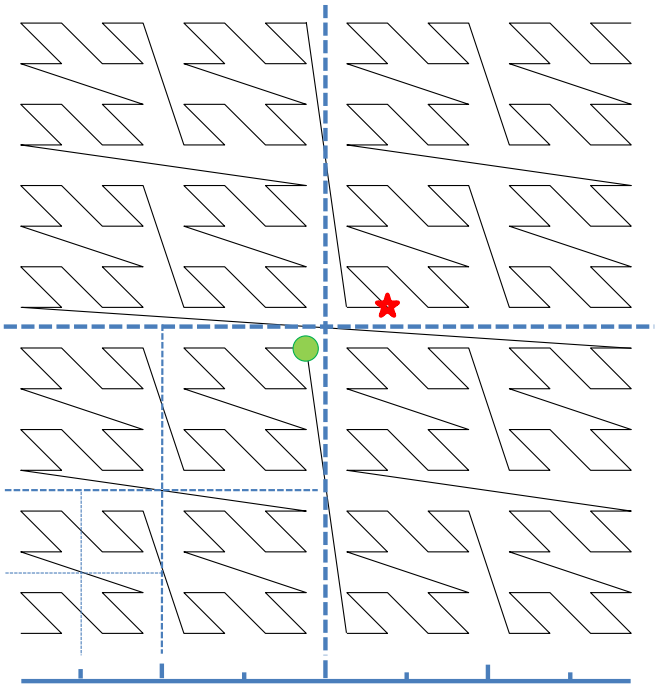

Figura 3.1: As descontinuidades principais da curva Z-order ocorrem nas “costuras” regulares junto com cada dimensão (mostradas pelas linhas tracejadas e barra inferior). O ponto-consulta (a estrela) poderia estar longe do ponto original (círculo sem rótulo) na curva, mesmo eles estando próximos no espaço.

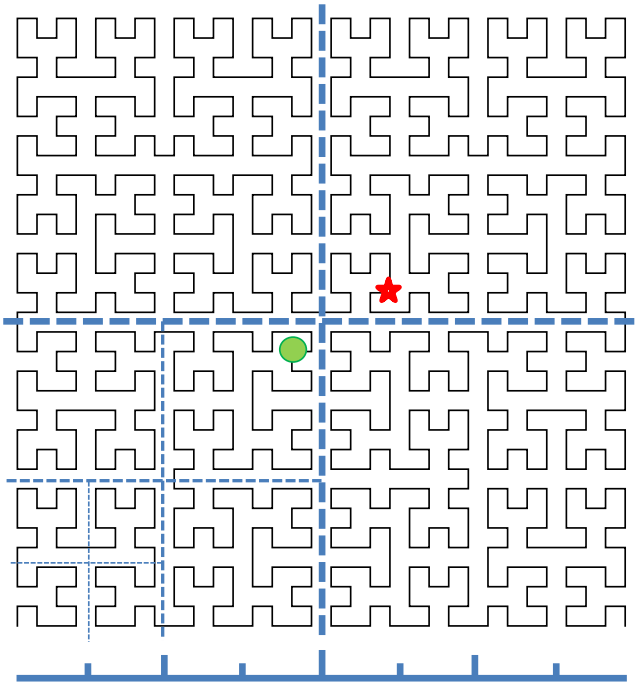

Figura 3.2: As descontinuidades principais da curva de Hilbert também ocorrem nas “costuras” regulares junto com cada dimensão.

O MONORAIL usa a característica anterior para criar os representantes onde eles são mais necessários, como é explicado a seguir:

1. Para cada elemento da base de dados pode ser gerado até um número máximo (parametrizável) de representantes. Cada elemento da base é composto por um dado, por exemplo, um identificador para um objeto de interesse, e um vetor de características que representa esse objeto num espaço multidimensional, no qual a noção de distância (Euclidiana) é utilizada como medida de dissimilaridade entre os objetos. Um exemplo desse vetor é mostrado na Figura 3.3a;

2. As descontinuidades são tratadas sucessivamente, começando pela mais grave. A zona de descontinuidade mais grave é vizinha ao ponto médio de uma determinada dimensão; as descontinuidades seguintes são, recursivamente, vizinhas ao ponto médio de cada meio-intervalo naquela dimensão. Caso não haja mais nenhuma descontinuidade a ser tratada, o processamento do elemento é finalizado e passa-se para o próximo elemento na base de dados (etapa 1);

3. Uma dimensão do vetor do elemento corrente (etapa 1), que ainda não foi verificada, é escolhida aleatoriamente. Caso não haja mais nenhuma dimensão, o tratamento da descontinuidade é finalizado e passa-se para a próxima zona de descontinuidade (etapa 2). Na Figura 3.3b a primeira dimensão escolhida, por exemplo, é a que corresponde ao eixo vertical do plano. A segunda dimensão verificada, correspondente ao eixo horizontal, é mostrada na Figura 3.3d;

4. É verificado para esse elemento, na dimensão escolhida na etapa 3, se ele encontra-se próximo da região de descontinuidade que está sendo tratada (etapa 2). Isso ocorre quando o valor absoluto da diferença entre a posição do ponto e a posição da descontinuidade é menor que um parâmetro, o tamanho da vizinhança a considerar no entorno dos pontos médios (t). Esse parâmetro é indicado pelas setas vertical da Figura 3.3b e horizontal da Figura 3.3d. Caso outra descontinuidade mais grave já tenha sido tratada, para esse elemento e nessa dimensão, o processo retorna à etapa 3, para que uma nova dimensão seja escolhida;

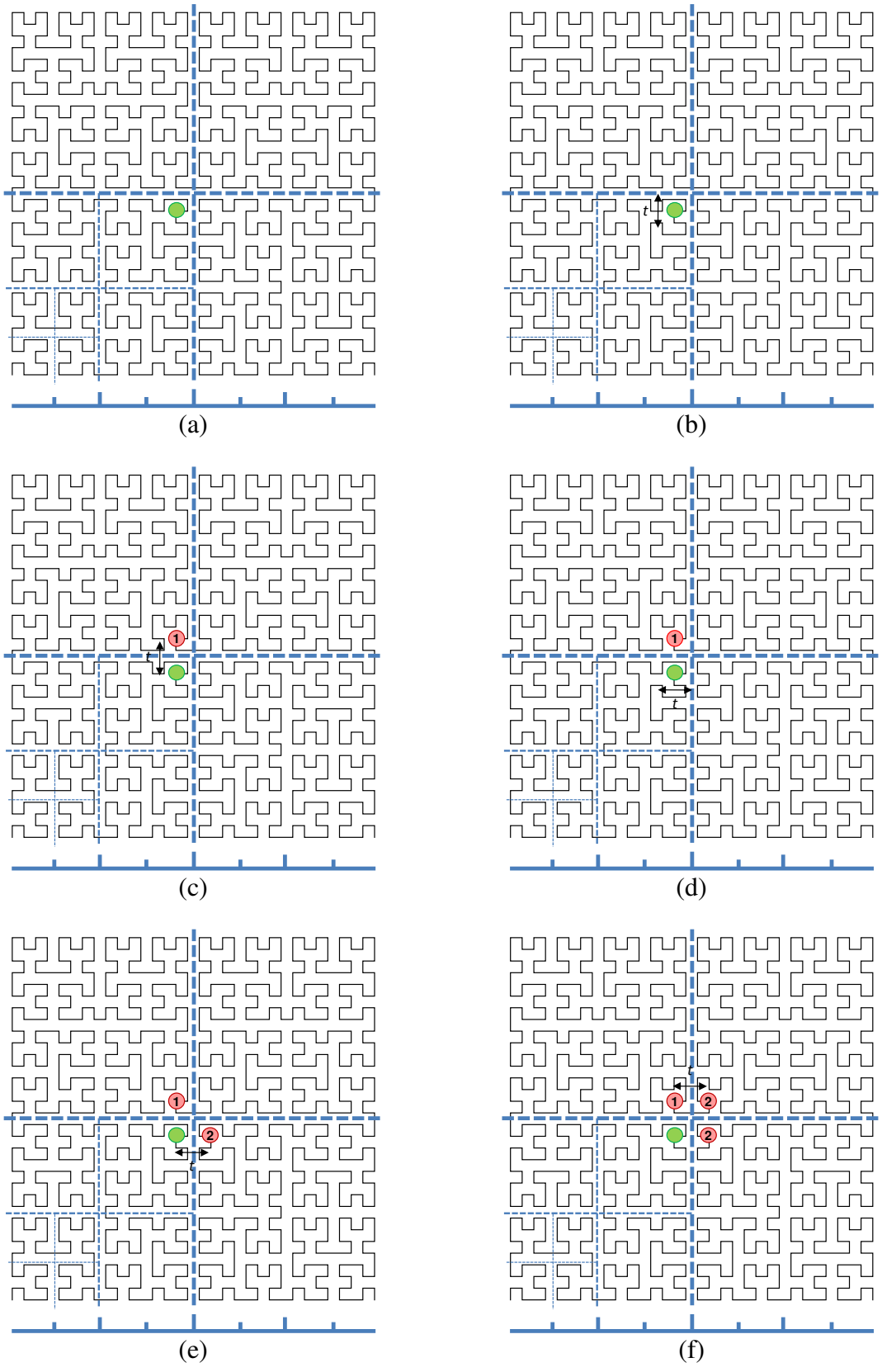

Figura 3.3: Sequência de criação dos representantes.

5. Se o ponto estiver na zona de descontinuidade, conforme avaliado na etapa 4, um novo valor para a dimensão problemática é calculado do seguinte modo: se o valor do ponto original for menor que o ponto médio (essa é a situação mostrada pelas Figuras 3.3b e 3.3d), da descontinuidade que está sendo tratada (etapa 2), o valor será igual ao valor do ponto original mais o parâmetro do tamanho da vizinhança; caso contrário, o valor será igual ao valor do ponto original menos o tamanho da vizinhança. O limite superior ou inferior (escala da dimensão) é respeitado ao somar ou subtrair o tamanho da vizinhança respectivamente;

6. Enquanto o número máximo de representantes não for atingido, ocorre o processo de criação de representantes para "cruzar" a zona de descontinuidade. Isso consiste em criar cópias dos representantes existentes até então (inclusive o elemento original) efetuando uma translação dessas cópias de forma que elas sejam posicionadas no outro lado do plano da descontinuidade: para a dimensão problemática, determinada na etapa 4, o valor dessa cópia será igual ao valor calculado na etapa 5 e para as demais dimensões, será o mesmo valor do ponto origem (representante pré-existente ou elemento original). O primeiro representante criado é mostrado na Figura 3.3c, pelo círculo rotulado com 1. A Figura 3.3e ilustra o segundo representante e a cópia do primeiro representante transladada, devido ao tratamento da segunda dimensão, aparece na Figura 3.3f (círculo superior rotulado com 2);

7. Todos os representantes (inclusive o elemento original) são incluídos em uma lista, formada pelos pares <chave-estendida, dado> e ordenada pelo campo chave-estendida. Essa chave é computada, a partir das dimensões do vetor de cada representante, de forma a refletir as posições diversas dos mesmos: isso é feito para curva de Hilbert utilizando o algoritmo descrito em [Butz 1971] ou para a curva Z-order intercalando os *bits* das dimensões. Todos os representantes são inseridos com o mesmo valor de dado (referente ao elemento original). Depois disso, o índice passa por uma etapa de limpeza para eliminar os representantes repetidos que são encontrados dentro de uma faixa de elementos. O tamanho dessa faixa é igual ao número de elementos candidatos (parâmetro *PD*) que serão examinados durante o processo de busca.

Apesar de, por motivos didáticos, o algoritmo ter sido explicado como um procedimento sequencial, isso não é essencial para o MONORAIL. O índice pode ser criado de forma incremental e até mesmo concorrentemente, desde que a estrutura de dados usada para a lista ordenada suporte essas operações. Isso também permite que as novas inserções sejam facilmente

manipuladas. Essas outras formas de criação do índice serão objeto de estudo em trabalhos futuros.

Exclusões também podem ser feitas, se algumas precauções forem tomadas. O algoritmo não é estritamente determinístico, porque as dimensões são processadas em uma ordem aleatória diferente a cada inserção. Contudo, isso é feito apenas para evitar vieses sistemáticos e um simples gerador pseudo-aleatório linear congruente [Lehmer 1951] pode ser empregado. Alimentando a semente desse gerador com um valor *hash* do dado, o procedimento torna-se completamente determinístico, e todos os representantes podem ser localizados a partir do ponto original.

3.2.2 Busca

O processo de busca consiste em localizar na lista ordenada um certo número de elementos candidatos (parâmetro *PD*), cujas chaves-estendidas estejam mais próximas da chave-estendida da consulta, e é ilustrado pela Figura 3.4. Os representantes permitem que o ponto original (círculo sem rótulo) seja encontrado pela consulta (estrela). A distância real entre os candidatos e a consulta é calculada e os elementos mais próximos são retornados.

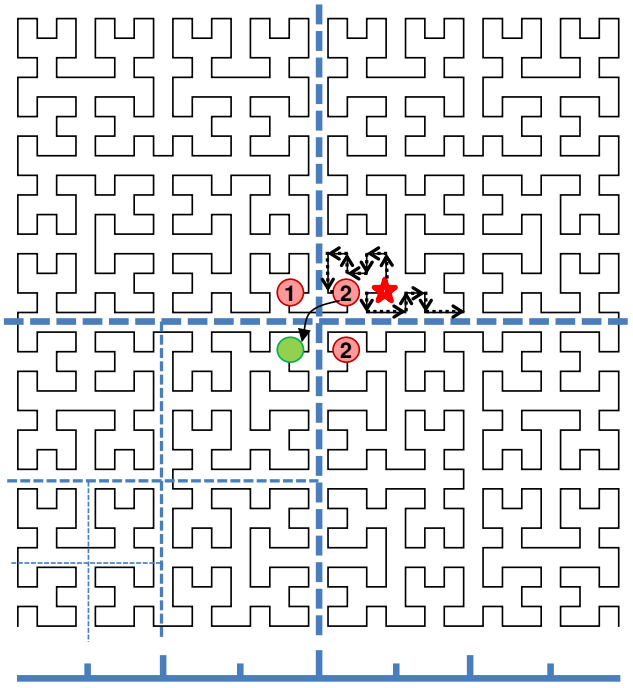

Figura 3.4: Exemplo de como um ponto (círculo sem rótulo) é encontrado pela consulta (a estrela), com a ajuda dos representantes criados.

3.3 Implementação

Utilizamos as implementações dos métodos de Mainar-Ruiz e Pérez-Cortés [Mainar-Ruiz 2006] e Multicurves [Valle 2008b] escritas em Java[6] e com a curva de Hilbert. Escrevemos duas versões, também em Java, para o MONORAIL: uma utilizando a curva de Hilbert e outra com a curva Z-order. Além disso, implementamos uma versão dos dois métodos de referência usando a curva Z-order.

3.3.1 Criação do índice

Para criar o índice MONORAIL são necessários os seguintes parâmetros:

- *d*: é o número de dimensões dos vetores de características (pontos);
- *f*: é o número de *bits* necessário para representar o valor de cada dimensão dos pontos;
- *n*: é a quantidade de pontos na base de dados;
- *M*: é o número máximo de representantes que podem ser criados para cada ponto da base de dados;
- *t*: é o tamanho da vizinhança a considerar no entorno dos pontos médios;
- *PD*: é o número de elementos candidatos que serão examinados no processo de busca.

O processo de criação do índice MONORAIL, explicado na seção 3.2.1, é mostrado pela Figura 3.5.

[6] Foram gentilmente cedidas pelo Prof. Dr. Eduardo Alves do Valle Junior.

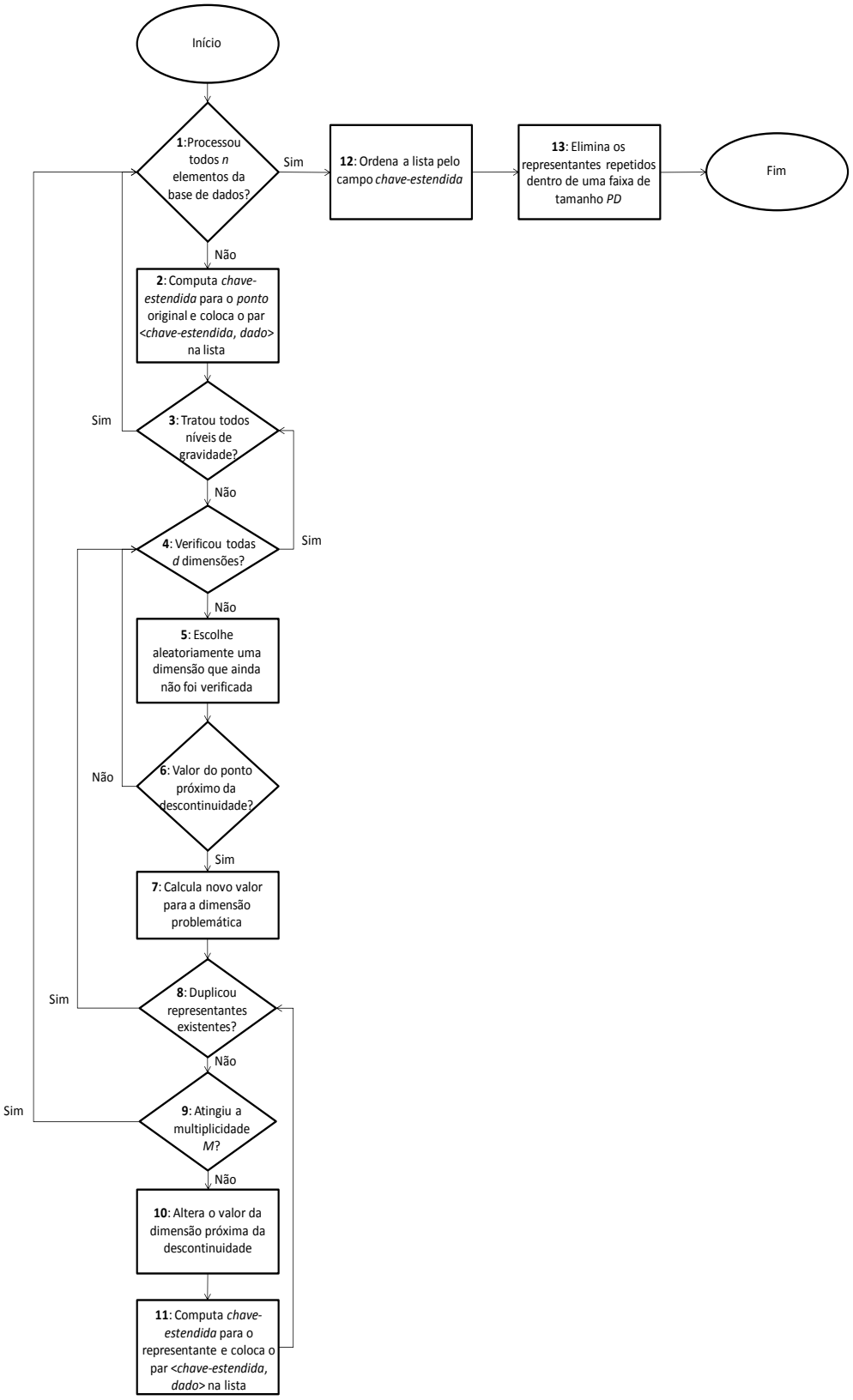

Figura 3.5: Processo de criação do índice MONORAIL.

Assumindo uma estrutura de acesso aleatório e eficiente para a lista ordenada, o custo amortizado de construção do índice é $O(mn \log mn)$, onde m é a multiplicidade efetiva e n o número de pontos na base de dados. A multiplicidade efetiva (o número esperado de representantes por ponto) depende da base de dados, mas a experiência mostra que para descritores de alta dimensionalidade, m é muito próximo de M, para qualquer valor desse último menor que 100. Para valores práticos, o custo da construção do índice é log-linear com o tamanho da base de dados (o efeito do m dentro do logaritmo é desprezível).

Inserções ou exclusões de um único ponto no índice com n pontos custariam $O(m \log mn)$.

O algoritmo detalhado para computar a chave-estendida, utilizando a curva de Hilbert, também pode ser encontrado na seção 3.11.5 em [Valle 2008]. O processo para computar a chave-estendida usando a curva Z-order é mostrado pela Figura 3.6. Esse cálculo consiste em intercalar os *bits* das dimensões de um ponto.

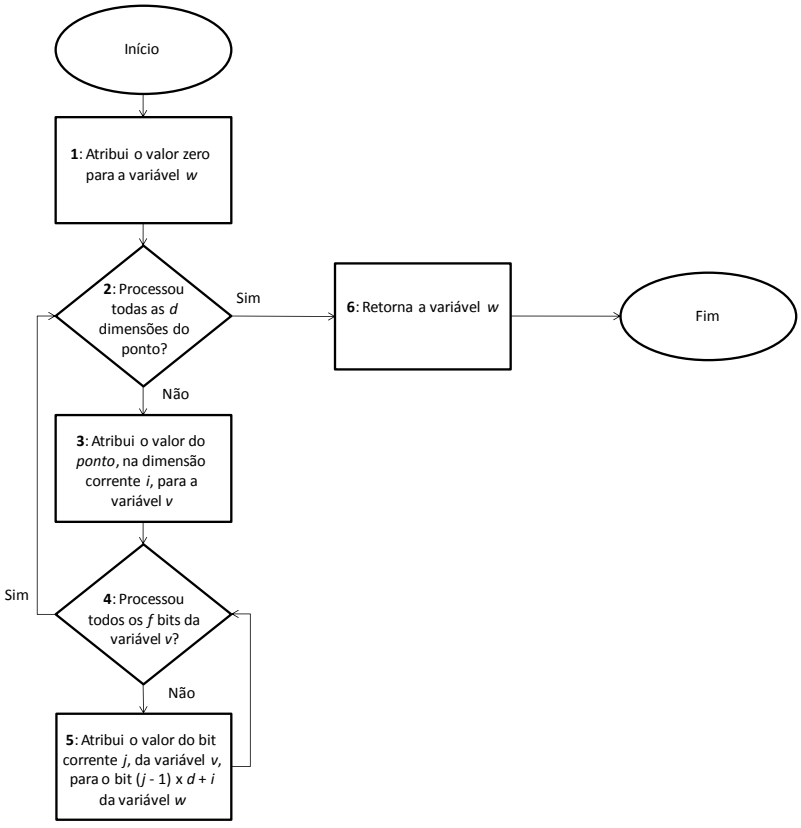

Figura 3.6: Processo de cálculo da chave-estendida usando a curva Z-order.

A complexidade do algoritmo anterior é $O(df)$, de acordo com os laços das caixas 2 e 4 da figura. Na versão com a curva de Hilbert [Butz 1971] a complexidade também é $O(df)$.

Contudo, a constante escondida na ordem da complexidade assintótica dessa versão é bem superior à da curva Z-order.

3.3.2 Busca

O processo genérico de busca é mostrado pela Figura 3.7. Após computar a chave-estendida do vetor-consulta (caixa 1 da figura), uma busca é realizada no índice (caixa 2) para retornar os elementos candidatos. Em seguida, a distância Euclidiana entre esses candidatos e a consulta é calculada (laço formado pelas caixas 3, 4 e 5) e no final as referências para os k elementos mais próximos são retornadas (caixa 6).

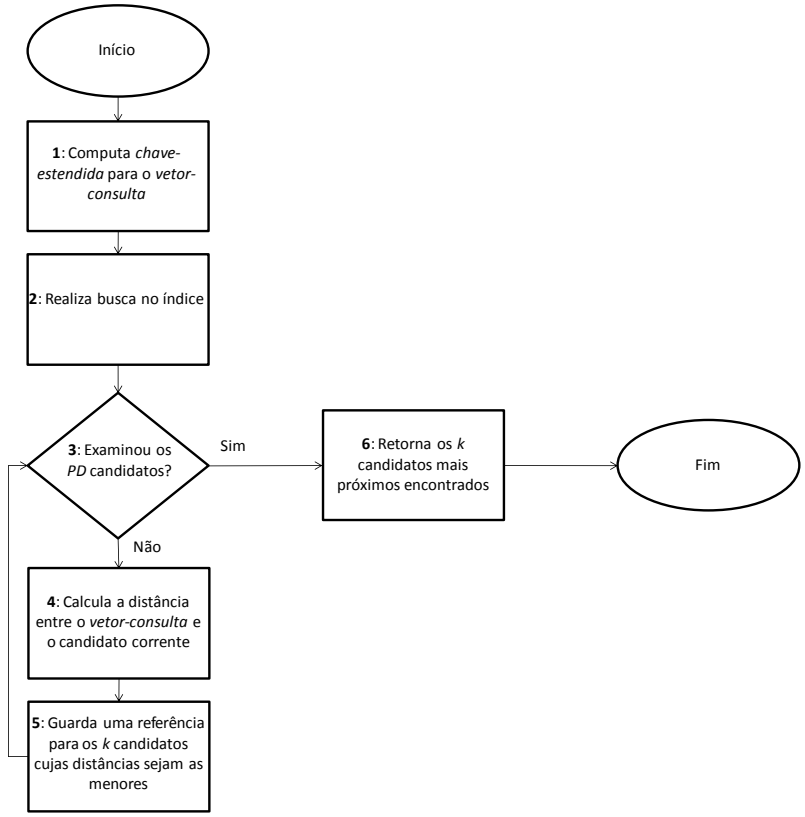

Figura 3.7: Processo genérico de busca.

Esse algoritmo de busca tipicamente irá fazer $O(\log mn + PD)$ operações para retornar os candidatos — caixa 2 — e $O(PD\ d)$ operações para selecionar os melhores, correspondente ao laço das caixas 3 a 5 (com d sendo a dimensionalidade dos pontos).

No entanto, deve ser notado que a hipótese de acesso aleatório uniforme não se aplica à memória secundária, onde o custo desse tipo de acesso é normalmente muito maior que um acesso sequencial. Se uma estrutura de dados eficiente para o disco, como uma árvore B, for

usada para implementar a lista ordenada, o custo dos acessos aleatórios para inclusões e exclusões deverá ser $O(m \log mn)$, uma vez que todos os representantes devem ser atualizados. Esse custo, entretanto, é compensado na busca, já que uma única estrutura deve ser consultada, com custo $O(\log mn)$. Para fins de comparação, considere um índice trabalhando com *c* curvas diferentes para resolver o problema das descontinuidades. Esse índice não teria o custo de *m* representantes para cada ponto, mas por outro lado, ele teria que manipular cada mapeamento independente em uma lista ordenada separada. Portanto, implicaria um custo de acesso aleatório de $O(c \log n)$ para a busca e atualização. Para $c = m$, o MONORAIL incorre em uma atualização um pouco mais cara (o *m* dentro do logaritmo) para alcançar buscas muito mais rápidas (a eliminação do fator linear *c*, fora do logaritmo).

3.4 Conclusão

Nesse capítulo, foi apresentado o método de indexação multidimensional MONORAIL, que utiliza uma curva de Peano e múltiplos representantes para os dados, estrategicamente posicionados para resolver o problema das regiões de descontinuidade.

Em muitas das curvas de Peano, como a de Hilbert, a Z-order, etc., as posições onde a propriedade de preservação da vizinhança é violada ocorrem ao redor do ponto que divide cada dimensão pela metade, depois nos pontos que as dividem em quatro, oito e assim sucessivamente. No MONORAIL os representantes são gerados para "cruzar" as zonas de descontinuidade, aproveitando essa característica.

O processo de criação do índice é simples e sua complexidade é log-linear com o tamanho da base de dados. O mapeamento dos pontos do espaço de alta dimensionalidade para as coordenadas de uma dimensão pode ser feito utilizando a curva Z-order, que apesar de não ter a mesma capacidade de agrupamento da curva de Hilbert, é mais fácil de ser calculado.

A avaliação empírica do MONORAIL e da curva Z-order, incluindo comparações com outro método que gera os representantes aleatoriamente [Mainar-Ruiz 2006] e um que trabalha com múltiplas curvas [Valle 2008b], é descrita no próximo capítulo.

Capítulo 4

Experimentos

Neste capítulo são apresentados a metodologia adotada nos experimentos e os resultados obtidos. Na seção 4.1, iniciamos com a descrição das bases de dados utilizadas.

Uma contribuição não trivial deste trabalho é o rigor experimental usado tanto na exploração do espaço paramétrico dos métodos quanto na comparação entre eles. Os experimentos foram cuidadosamente projetados para garantir resultados estatisticamente significativos. A metodologia, detalhada na seção 4.2, é geral o suficiente para ser usada em outras avaliações da busca por similaridade.

O experimento da seção 4.4 compara a eficácia do MONORAIL com o método de Mainar-Ruiz e Pérez-Cortés [Mainar-Ruiz 2006], ambos implementados com a curva de Hilbert.

No segundo conjunto de experimentos, descrito na seção 4.5, o objetivo é avaliar o impacto da troca da curva de Hilbert pela curva Z-order no tempo de criação dos índices, tempo de busca e eficácia do MONORAIL e do Multicurves [Valle 2008b].

O terceiro experimento (seção 4.6) testa a escalabilidade do MONORAIL e do Multicurves com três bases de dados de tamanhos diferentes.

O experimento da seção 4.7 compara a eficácia e o tempo de busca do MONORAIL com o Multicurves.

No experimento da seção 4.8 o objetivo é comparar a eficácia do MONORAIL com o método de Mainar-Ruiz e Pérez-Cortés, ambos utilizando a curva Z-order.

O último experimento (seção 4.9) compara a eficácia da versão original do Multicurves com a versão estendida, que permite a escolha arbitrária das dimensões.

4.1 Bases de dados

Quatro bases de dados foram utilizadas nos experimentos.

Para gerar a primeira delas, foram coletadas 233.852 imagens da Web [Penatti 2010], cujos vetores de características foram usados como pontos de confusão, e adicionadas 225 imagens, pertencentes à coleção pessoal de um dos membros do laboratório, e que não estão na Web. Os descritores locais SIFT (*Scale Invariant Feature Transform* [Lowe 2004]) foram computados para todas as imagens, resultando em 130.463.526 vetores de características (média de 557 descritores por imagem) de 128 dimensões.

Na segunda base, o número de imagens de confusão foi reduzido para 97.291 e foram mantidas as 225 imagens pessoais. Dessa forma, a quantidade total de vetores de características (registros) caiu para 55.473.496.

A terceira base tem 57.162 imagens de confusão e as mesmas 225 imagens pessoais, totalizando 32.973.335 vetores de características.

Para obter consultas que fazem sentido para as bases de dados acima, as 225 imagens pessoais foram submetidas a grandes transformações (corte, rotação, alteração de escala, cisalhamento, correção de gama e *dithering*). Depois disso, os descritores SIFT foram computados para as imagens obtidas, resultando em 187.839 vetores-consultas.

A quarta base de dados, chamada APM, foi gerada a partir de transformações em 100 imagens de fotografias antigas (do século XIX e da primeira metade do século XX) da coleção do Arquivo Público Mineiro, do estado de Minas Gerais. Cada imagem passou por 15 transformações (rotações, alterações de escala, etc.) e em seguida os descritores SIFT foram computados, resultando em 2.871.300 vetores de características. As consultas dessa base são descritores SIFT computados a partir das imagens originais, totalizando 263.968 vetores-consultas.

4.2 Metodologia

A *verdade terrestre*[7] é o conjunto das respostas da busca kVMP exata e serve para avaliar os resultados da busca aproximada kVMP. Para as três primeiras bases, esse conjunto é composto pelos cinco vizinhos mais próximos de cada consulta. Na base APM o arquivo contém os 30 vizinhos mais próximos. Esses arquivos foram gerados por uma versão da busca sequencial com um alto grau de paralelismo [Teodoro 2008]. Mesmo tendo utilizado um *cluster* de máquinas, a tarefa demorou mais de uma semana para ser concluída.

Infelizmente, essa verdade terrestre não pode ser usada diretamente nesta avaliação empírica, porque algumas das consultas são muito mais fáceis de responder do que outras. O grau de dificuldade para responder uma consulta se torna um fator de perturbação, que pode arruinar a significância estatística do experimento. A forma tradicional de lidar com essa situação em um projeto experimental é dividir o experimento em blocos, realizando medições diferentes para níveis diferentes de dificuldade.

4.2.1 Blocagem das consultas

Em qualquer experimento existem fatores que afetam os resultados, mas não fazem parte do conjunto de fatores que o experimentador está interessado em avaliar. Esses fatores são chamados de fatores de perturbação e podem, quando negligenciados, arruinar a significância estatística do experimento. Um exemplo seria comparar experimentos do tempo de execução de diferentes algoritmos em máquinas com diferentes configurações. A diferença no poder de computação das máquinas gera um efeito, chamado de confusão (em modelagem estatística), ao atribuir as diferenças observadas para o resíduo (efeitos não considerados pelo modelo) ou mesmo para os fatores de interesse, quando na verdade são causadas pelo fator de perturbação. A forma tradicional de lidar com essa questão, em um projeto experimental, é dividir o experimento em blocos dentro dos quais a influência do fator de perturbação seja reduzida. Na comparação de dois métodos de busca aproximada kVMP, com diversas parametrizações, o nível de dificuldade das consultas certamente contribui para a variação na eficácia de ambos os métodos. Para isolar essa interferência indesejada e identificar a real contribuição dos parâmetros de interesse, as medidas precisam ser comparadas considerando as consultas com um mesmo grau de dificuldade.

[7] Verdade terrestre foi adotada como tradução do termo *ground truth*, utilizado em sensoriamento remoto.

Para quantificar a dificuldade de responder uma determinada consulta nós usamos um critério de contraste [Lowe 2004, Lejsek 2009]. A ideia é observar a seguinte razão:

$$C(q) = \Delta(q, \text{1º VMP de } q) / \Delta(q, \text{Xº VMP de } q),$$

onde q é o vetor-consulta, Δ é a função de distância (Euclidiana para o SIFT) e X é a posição de um vizinho próximo de q mais distante que o primeiro (nós utilizamos o 5º para as três primeiras bases e o 30º para a base APM). Se o valor de $C(q)$ é próximo de zero, isso significa que o primeiro vizinho de q é muito mais próximo que os demais — indicando um bom contraste e uma provável facilidade para responder a consulta. Já se o valor de $C(q)$ é próximo de 1, todas as respostas são mais ou menos equidistantes, indicando uma consulta muito difícil, que pode até não ter sentido [Beyer 1999].

Usando esse critério, as consultas foram divididas em 5 blocos, cujos valores do contraste encontram-se dentro dos intervalos [0; 0,2), [0,2; 0,4), [0,4; 0,6), [0,6; 0,8) e [0,8; 1]. Nós consideramos as consultas do primeiro bloco *fáceis* e as consultas do segundo bloco *difíceis*. Os últimos três blocos apresentam um contraste tão ruim que eles não têm sentido do ponto de vista do funcionamento do descritor SIFT [Lowe 2004] e nem da teoria do vizinho mais próximo [Beyer 1999]. Nós chamamos esses três blocos de *ruído*.

A Figura 4.1 mostra o que aconteceria com os resultados do primeiro experimento, realizado na terceira base (32.973.335 registros) e detalhado na seção 4.4, se as consultas não fossem divididas em blocos. O desvio padrão é muito grande e consequentemente o intervalo de confiança (α=0,05), representado pela barra vertical, também. Os quatro valores do eixo x são iguais para ambos os métodos, mas os pontos foram deslocados um pouco, apenas para permitir a visualização separada dessas barras. Nessa análise, o efeito do fator de perturbação é alocado no resíduo do modelo e os intervalos de confiança acabam se cruzando: um experimentador desatento consideraria a comparação das médias como inconclusiva. A Figura 4.2 mostra os resultados separados pelos blocos: claramente as médias são diferentes.

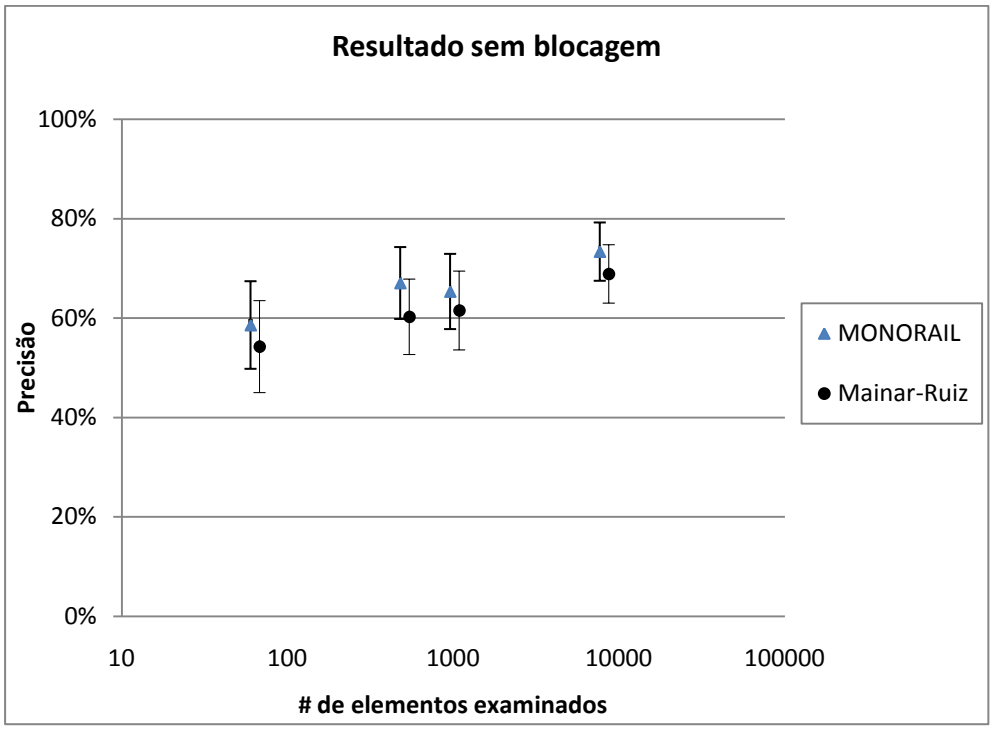

Figura 4.1: Resultado do primeiro experimento sem considerar os blocos de consultas. A comparação das médias é inconclusiva.

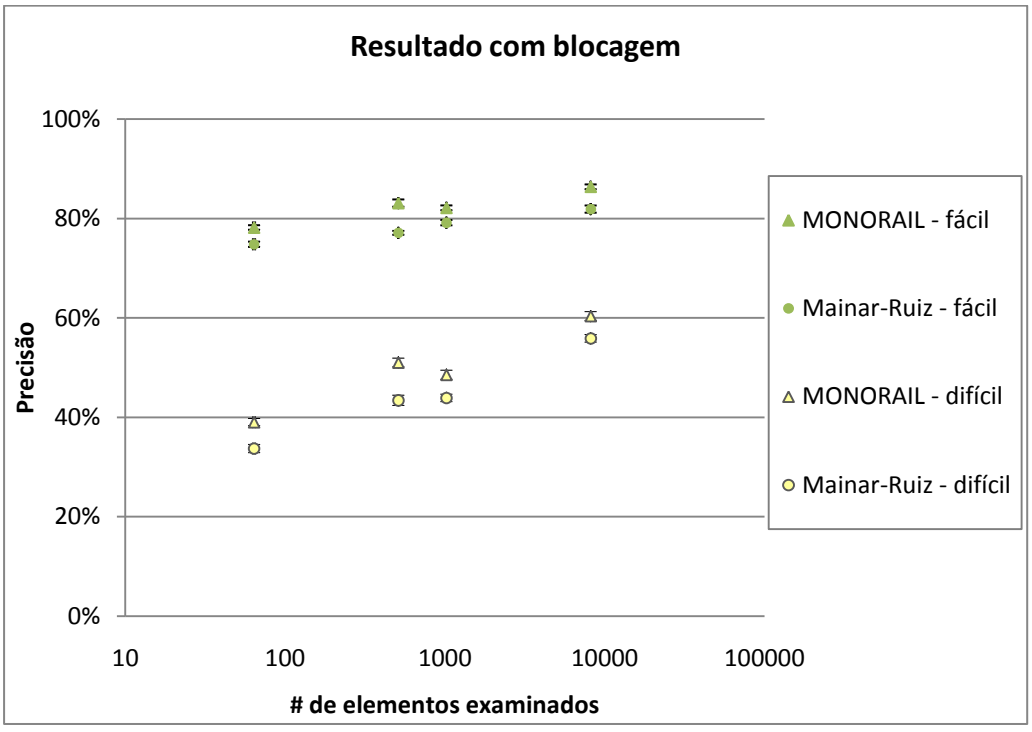

Figura 4.2: Resultado do primeiro experimento considerando os blocos de consultas. As médias são diferentes.

4.2.2 Métricas utilizadas

As medidas de eficácia e eficiência obtidas nos experimentos são mostradas na Tabela 4.1.

Medida	Descrição	Tipo
Precisão	Proporção entre o número de respostas corretas e o número de respostas retornadas	Eficácia
MPM	Média das precisões médias: $$MPM = Média\left(\frac{1}{\lvert\mathcal{V}\rvert}\sum_{j=1}^{k}\begin{cases}F(j), & se\ s'_j \in \mathcal{V} \\ 0, & caso\ contrário\end{cases}\right)$$ onde $\mathcal{V}$ é o conjunto verdade terrestre e $$F(j) = \frac{\#\ de\ respostas\ 1^{a} \ldots j^{ésima} \in \mathcal{V}}{j}$$	Eficácia
EAR	Erro de aproximação relativo: $$EAR = Média\left(\frac{1}{k}\sum_{i=1}^{k}\left\lvert\frac{\Delta(q,s_i) - \Delta(q,s'_i)}{\Delta(q,s_i)}\right\rvert\right)$$	Eficácia
Tempo de busca	Tempo, em milissegundos, gasto para responder cada consulta	Eficiência
Tempo de criação	Tempo, em horas, gasto na construção do índice, sem contar a etapa de ordenação	Eficiência
Tamanho do índice	Tamanho ocupado pelo índice no disco, em *gigabytes* (GB)	Eficiência

Tabela 4.1: Medidas obtidas nos experimentos.

As medidas MPM e EAR foram usadas somente nos experimentos com k>1. Essas medidas são importantes, pois atribuem um peso maior para os vizinhos encontrados corretamente nas posições mais próximas da consulta do que os vizinhos mais afastados. Todos os tempos foram obtidos em um computador com 2 processadores Intel Xeon X5670 de 2,93 GHz, 12 GB de memória, 4 discos de 1.500 GB ligados em RAID-5 por *software* e sistema operacional Linux.

4.3 Implementação da lista ordenada

A busca foi implementada usando um índice de dois níveis, ilustrado pela Figura 4.3. O primeiro nível, chamado *índice agrupado*, é a própria lista ordenada gerada pelo processo de criação do índice. O segundo nível, chamado *índice esparso*, é gerado a partir do primeiro, contendo uma chave-estendida e sua posição no índice agrupado a cada *S* entradas. O parâmetro *S,* chamado de passo, deve ser suficientemente grande para que esse nível caiba inteiramente na memória principal.

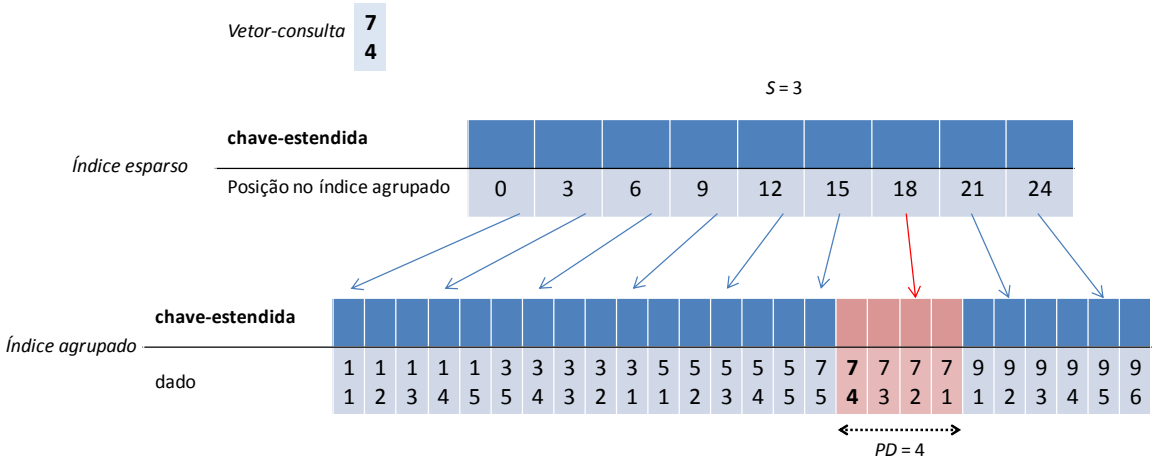

Figura 4.3: Índice de dois níveis.

Os índices agrupados do MONORAIL e do método de Mainar-Ruiz e Pérez-Cortés [Mainar-Ruiz 2006] passam adicionalmente por uma etapa de limpeza antes da geração do índice esparso. Esse processo é feito para eliminar os representantes repetidos que são encontrados dentro de uma faixa de elementos, cujo tamanho é igual ao valor do parâmetro *PD*.

O processo de busca, com o índice de dois níveis, é mostrado na Figura 4.4. A busca é dividida em duas etapas: na primeira, após computar a chave-estendida do vetor-consulta (caixa 1 da figura), uma busca binária é realizada no índice esparso (caixa 2); na segunda, o índice agrupado é percorrido para trás e para frente a partir da posição determinada na fase anterior (laço formado pelas caixas 3, 4 e 5). O número de elementos examinados é indicado pelo parâmetro do número de elementos candidatos (*PD*). A distância Euclidiana entre esses candidatos e a consulta é calculada (caixa 4) e no final as referências para os *k* elementos mais próximos são retornadas (caixa 6).

Essa estrutura em dois níveis foi adotada por ser de simples implementação e por minimizar o número de acessos aleatórios em disco. Uma árvore B [Bayer 1972] também poderia ser empregada, vista como uma generalização disso em múltiplos níveis.

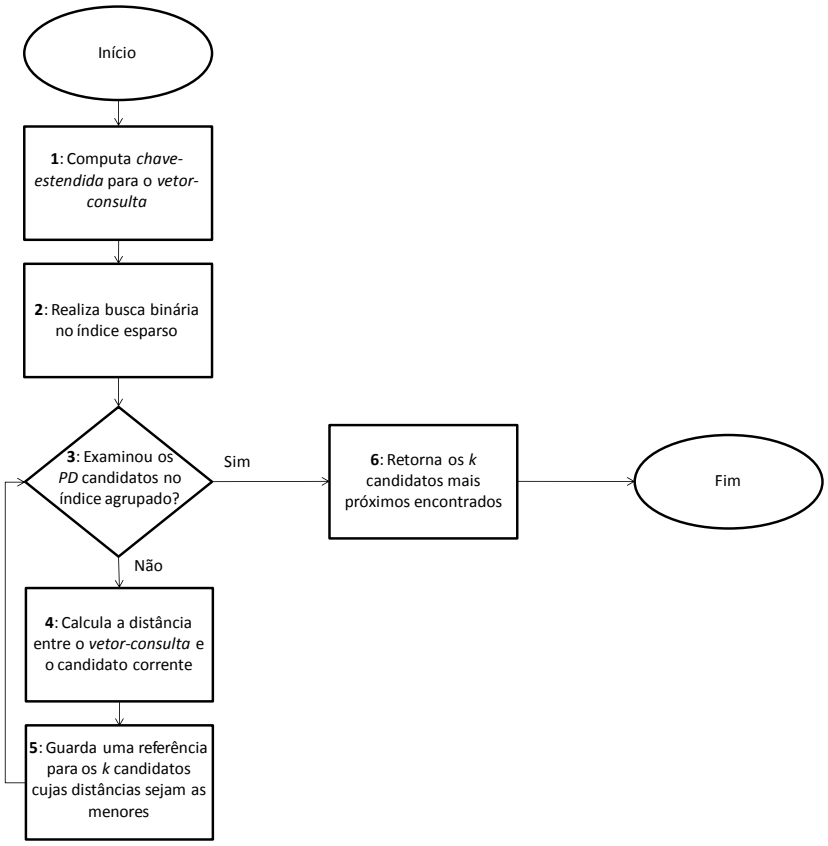

Figura 4.4: Processo de busca usando um índice de dois níveis.

A complexidade desse processo de busca é dada por $O(\log(\frac{mn}{s})) + O(PD\ d)$. A parcela logarítmica corresponde à busca binária no índice esparso (caixa 2). O último termo é referente ao cálculo das distâncias (Euclidiana) entre a consulta e os PD candidatos (laço das caixas 3 a 5).

4.4 Comparação do método de Mainar-Ruiz e Pérez-Cortés com o MONORAIL

Os objetivos deste experimento são:

- Comparar a precisão do MONORAIL com o método de Mainar-Ruiz e Pérez-Cortés [Mainar-Ruiz 2006];
- Testar a blocagem das consultas, utilizando o critério de contraste;
- Verificar as contribuições dos fatores, método, multiplicidade e número de elementos candidatos para a variação na precisão em cada bloco de consultas;
- Comparar os tamanhos dos índices, tempos de criação e tempos de busca.

Montamos um projeto experimental fatorial completo com o objetivo de revelar a importância relativa de cada fator, incluindo os efeitos cruzados. Consideramos o método (ambos implementados com a curva de Hilbert) como um dos fatores, além da multiplicidade (*M*) e do número de elementos candidatos (*PD*). Os valores são mostrados na Tabela 4.2. As consultas foram divididas em três blocos: *Fácil*, *Difícil* e *Ruído* de acordo com o contraste, explicado na seção 4.2.1. Utilizamos o valor 8 para o parâmetro do tamanho da vizinhança (*t*) do método MONORAIL, correspondente a três níveis (2^3) de recursão da curva (usando números inteiros para representar as coordenadas).

Fatores			**Contribuições**		
Fator	**Níveis**		***Fácil***	***Difícil***	***Ruído***
Método	Mainar-Ruiz *et al*.	MONORAIL	34,6%	11,0%	0,6%
PD	$16 \times M$	$256 \times M$	32,7%	39,0%	44,9%
M	4	32	24,5%	46,8%	47,6%
	Valores-P dos modelos:		< 0,01	< 0,01	< 0,01

Tabela 4.2: Resultados da ANOVA para cada tipo de consulta.

Dez repetições foram executadas para cada bloco, cada uma com 1.000 consultas escolhidas aleatoriamente (sem reposição). A medida obtida em cada execução foi a precisão, ou seja, a proporção entre o número de respostas corretas e o número de respostas retornadas. Somente o primeiro vizinho mais próximo foi procurado (k=1). Apenas a terceira base (32.973.335 registros) foi considerada nesse experimento.

A Tabela 4.2 resume os resultados da análise fatorial de variação (*analysis of variance* — ANOVA). Analisamos separadamente os resultados para as consultas fáceis, difíceis e ruído, obtendo três modelos diferentes. Os valores-P dos modelos indicam uma alta significância estatística. Poucos efeitos cruzados foram encontrados como estatisticamente significativos, mas suas contribuições individuais sempre foram abaixo de 3%.

É importante notar que os percentuais na Tabela 4.2 não indicam as alterações na precisão, mas ao invés disso eles mostram a contribuição relativa de cada fator para a variação observada na precisão. Para as consultas fáceis, por exemplo, o valor de 34,6% no método não significa que o nosso método seja 34,6% melhor do que o método de Mainar-Ruiz e Pérez-Cortés [Mainar-Ruiz 2006], mas que para as consultas fáceis a alteração no método é responsável sozinha por 34,6% da variação observada.

A Figura 4.5 ilustra os resultados de uma forma mais intuitiva — aqui os percentuais indicam as medidas de precisão. A multiplicidade foi fixada em 32 representantes e a média da precisão foi calculada para as 10 repetições, nas diferentes configurações do método, número de elementos candidatos e dificuldade da consulta. Com exceção do ruído, o MONORAIL sempre apresenta um ganho de eficácia. Na Figura 4.6 a multiplicidade foi fixada em 4 representantes, nesse caso a precisão do MONORAIL é superior à do método de Mainar-Ruiz e Pérez-Cortés em todos os blocos de consultas.

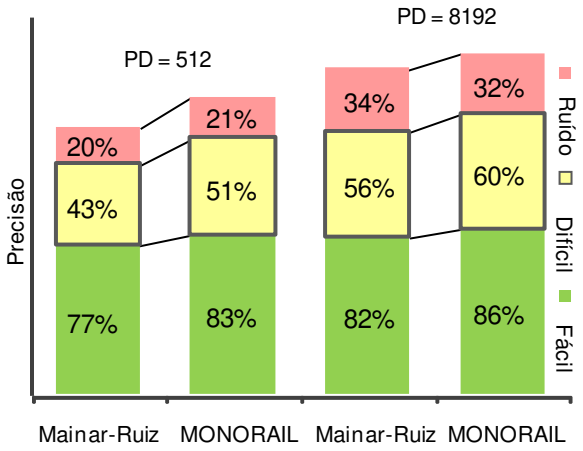

Figura 4.5: Resultados do experimento para comparar o método de Mainar-Ruiz *et al.* [Mainar-Ruiz 2006] com o MONORAIL, para M=32. O percentual dentro de cada bloco indica a precisão alcançada em cada configuração.

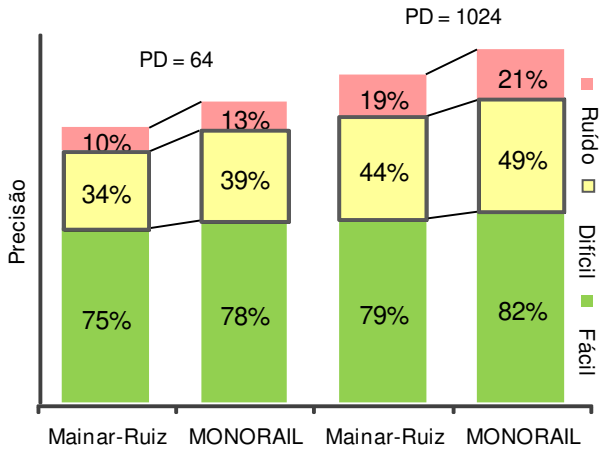

Figura 4.6: Resultados do experimento para comparar o método de Mainar-Ruiz *et al.*[Mainar-Ruiz 2006] com o MONORAIL, para *M*=4. O percentual dentro de cada bloco indica a precisão alcançada em cada configuração.

As medidas de eficiência (tempo de criação do índice e tempo de busca) foram praticamente as mesmas para ambos os métodos. A Figura 4.7 mostra o tamanho dos índices para cada um dos métodos, com o fator *PD* igual a 16 vezes a multiplicidade. O tamanho do índice também varia com o número de elementos candidatos (*PD*), devido ao processo de limpeza do índice, explicado na seção 4.3. Os índices MONORAIL ficaram pelo menos 30% menores em termos de tamanho ocupado em disco, isso confirma a geração de uma quantidade maior de representantes desnecessários pelo método de Mainar-Ruiz e Pérez-Cortés [Mainar-Ruiz 2006].

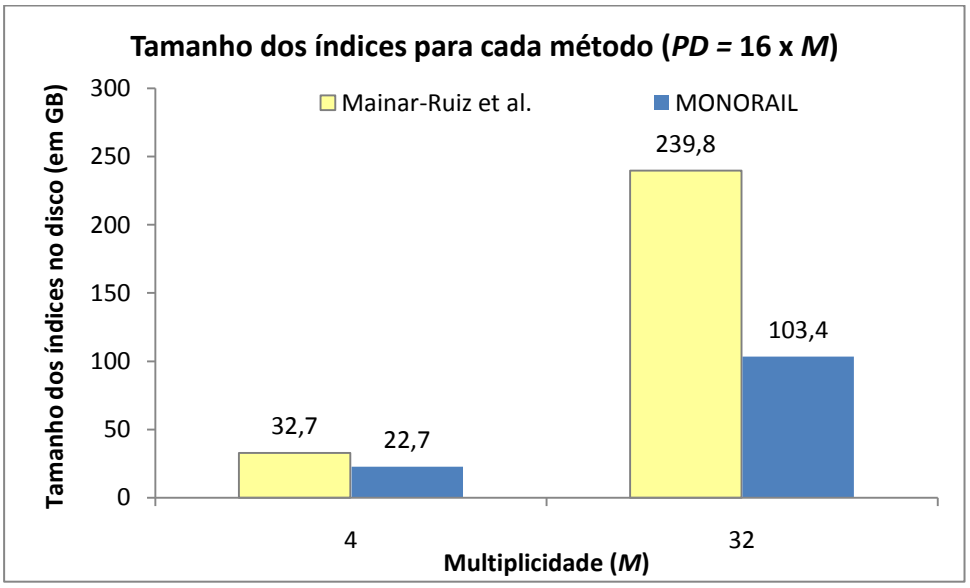

Figura 4.7: Tamanho dos índices para cada método, com o número de elementos candidatos (*PD*) igual a 16 vezes a multiplicidade (*M*).

4.5 Avaliação do impacto da curva Z-order no MONORAIL e no Multicurves

Os objetivos deste experimento são:

- Avaliar o impacto da troca da curva de Hilbert pela curva Z-order nos tempos de criação dos índices do MONORAIL e do Multicurves;
- Avaliar o impacto da utilização da curva Z-order nos tempos de busca desses métodos;
- Verificar a alteração na precisão dos métodos em cada bloco de consultas, com a utilização da curva Z-order.

Neste experimento, utilizamos as versões para os métodos MONORAIL e Multicurves [Valle 2008b] implementadas com a curva Z-order. Nós montamos um projeto experimental com os fatores e valores para cada método apresentados nas Tabelas 4.3 e 4.4 respectivamente. Como o número de elementos candidatos (*PD*) no método Multicurves é especificado por sub-índice, no MONORAIL esse parâmetro é multiplicado pelo número de representantes, para que o número total de elementos examinados seja o mesmo nos dois métodos. O valor 8 foi usado para o parâmetro do tamanho da vizinhança (*t*) do método MONORAIL.

Fator	**Níveis**		
Curva	Hilbert		Z-order
Multiplicidade (*M*)	4	8	32
# de elementos candidatos (*PD*)	$16 \times M$	$64 \times M$	$128 \times M$

Tabela 4.3: Parâmetros usados na avaliação do MONORAIL, curva de Hilbert x curva Z-order.

Fator	**Níveis**		
Curva	Hilbert		Z-order
# de sub-índices (*N*)	4	8	32
# de elementos candidatos (*PD*)	16	64	128

Tabela 4.4: Parâmetros usados na avaliação do Multicurves, curva de Hilbert x curva Z-order.

As consultas foram divididas em três blocos, de acordo com o contraste explicado na seção 4.2.1. Dez repetições foram executadas para cada bloco, cada uma com 4.000 consultas

escolhidas aleatoriamente (com reposição). Somente a terceira base (32.973.335 registros) foi usada nesse experimento.

A Figura 4.8 mostra o tempo, em horas, gasto na construção dos índices MONORAIL com 4, 8 e 32 representantes, com as curvas de Hilbert e Z-order. O tempo de criação dos representantes, com suas respectivas chaves-estendidas, cresce linearmente (demonstrado pela regressão) com o número total de pontos, que é o número de vetores de características (32.973.335) da base de dados mais o número total de representantes. O tempo de cálculo da chave-estendida com a curva Z-order foi aproximadamente cinco vezes menor do que com a curva de Hilbert.

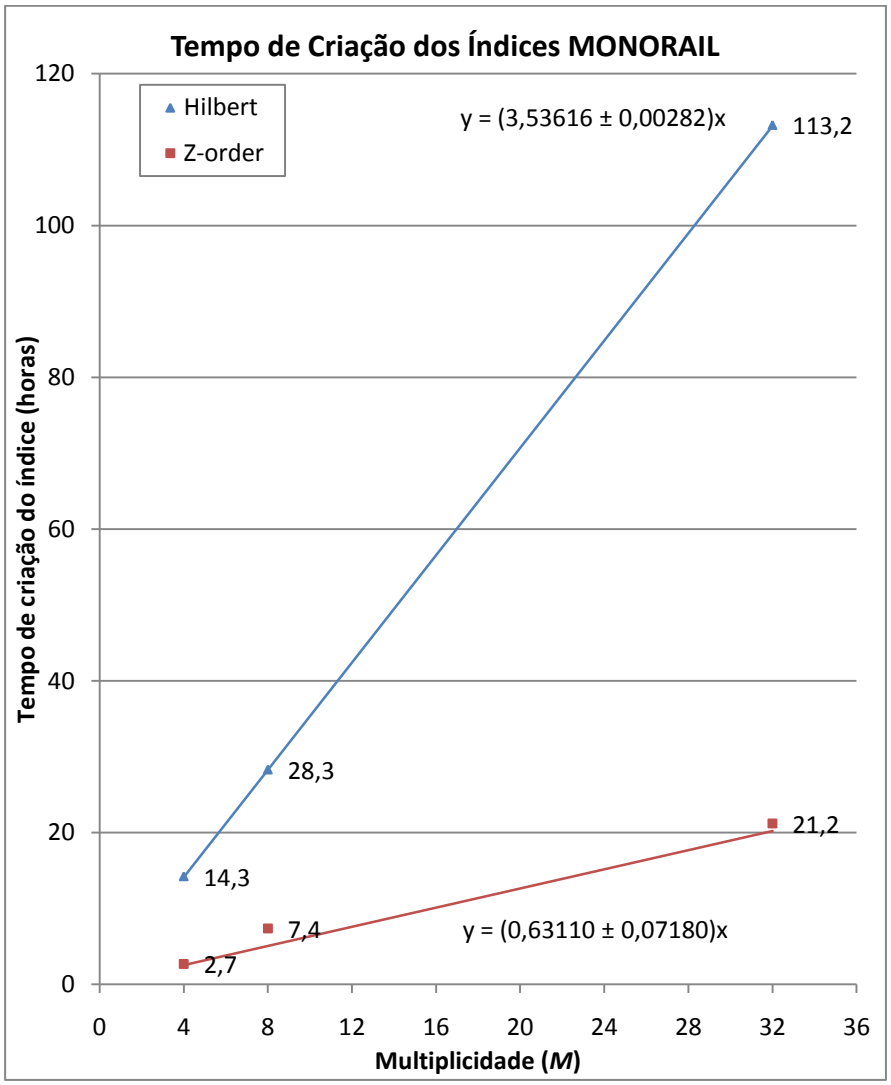

Figura 4.8: Tempo de criação dos índices MONORAIL. O coeficiente de correlação da regressão linear para a versão com a curva de Hilbert foi 0,9999997 e 0,9935906 para a curva Z-order. O *speed-up* da curva Z-order foi aproximadamente 5.

Os tempos de criação das chaves-estendidas dos sub-índices do Multicurves são apresentados na Figura 4.9. Aqui a regressão linear foi feita considerando o tamanho da chave como a variável independente. Novamente o tempo gasto com a curva Z-order foi significativamente menor do que com a curva de Hilbert. Contudo, o *speed-up* (aproximadamente 1,2 para a configuração com 32 sub-índices, 1,5 com 8 sub-índices e 2,0 com 4 sub-índices) não foi tão alto quanto o observado no caso anterior, porque o Multicurves computa a chave-estendida para os sub-espaços (tamanho da chave igual a 4, 16 e 32 dimensões respectivamente), enquanto o MONORAIL faz isso para o espaço completo (128 dimensões).

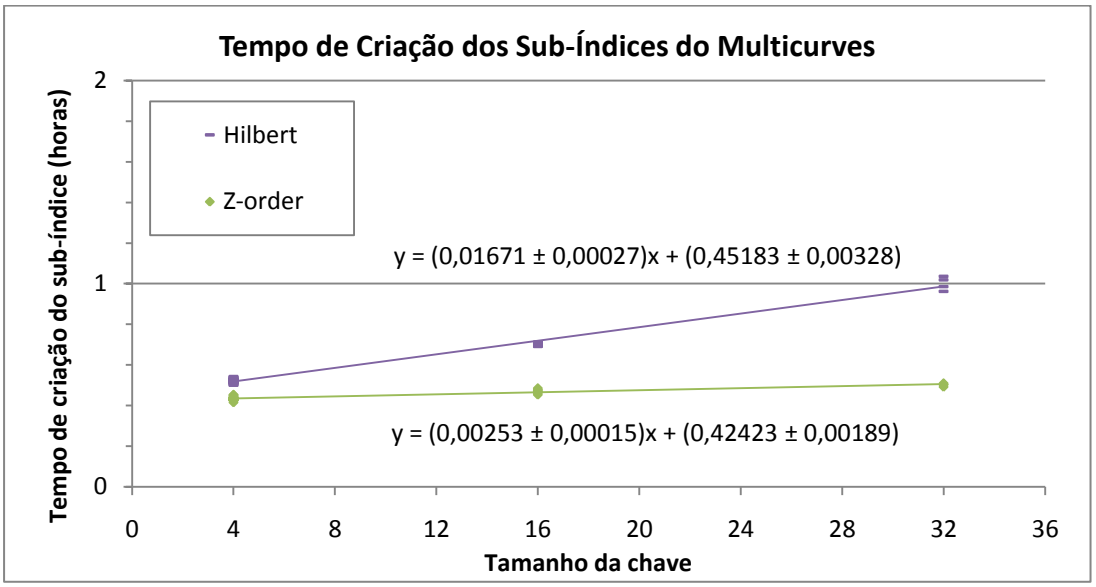

Figura 4.9: Tempo de criação dos sub-índices do Multicurves. O coeficiente de correlação da regressão linear para a versão com a curva de Hilbert foi 0,99470 e 0,93100 para a curva Z-order.

Para a busca, colocamos em um gráfico os tempos de consulta obtidos para todos os blocos, totalizando 50 execuções com 4.000 consultas em cada configuração. A Figura 4.10 mostra os tempos de busca do MONORAIL para as seguintes configurações: M=4 × PD=16, M=4 × PD=64 e M=4 × PD=128. Na Figura 4.11 são mostrados os tempos para as configurações: M=8 × PD=16, M=8 × PD=64 e M=8 × PD=128.

As Figuras 4.12 e 4.13 mostram os resultados do Multicurves para as configurações (PD=16, PD=64 e PD=128) com 4 e 8 sub-índices respectivamente. Nesses gráficos do tempo de busca, o ajuste linear foi realizado em função do número total de elementos examinados. Como podemos observar, tanto para o MONORAIL quanto para o Multicurves, a alteração da curva trouxe muito pouca contribuição para a variação no tempo de busca.

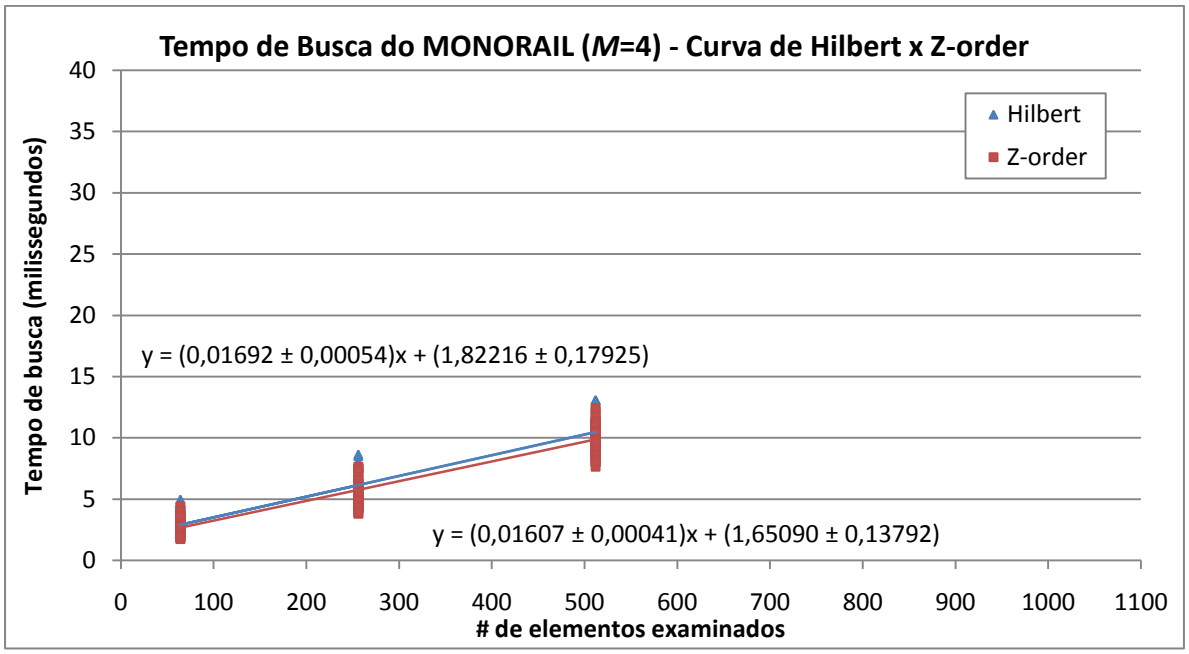

Figura 4.10: Tempo de busca do MONORAIL com a curva de Hilbert e a curva Z-order, considerando o número total de elementos examinados nas diferentes configurações com 4 representantes. O coeficiente de correlação da regressão linear para a versão com a curva de Hilbert foi 0,93243 e 0,95406 com a Z-order.

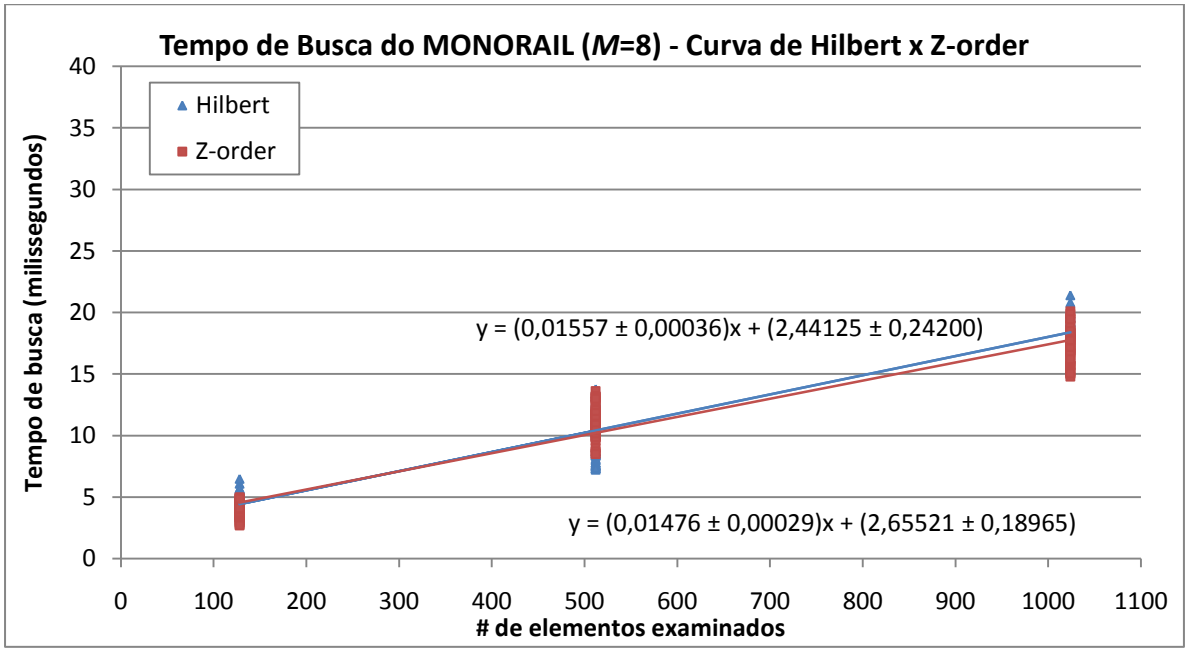

Figura 4.11: Tempo de busca do MONORAIL com a curva de Hilbert e a curva Z-order, considerando o número total de elementos examinados nas diferentes configurações com 8 representantes. O coeficiente de correlação da regressão linear para a versão com a curva de Hilbert foi 0,96188 e 0,97348 com a Z-order.

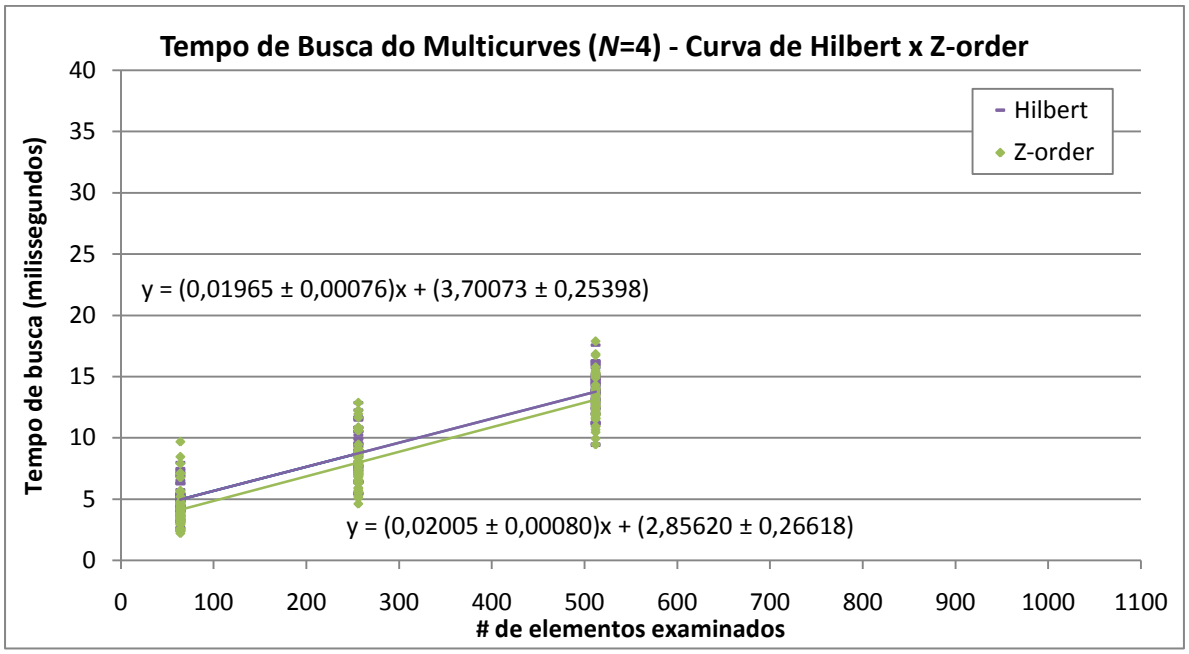

Figura 4.12: Tempo de busca do Multicurves com a curva de Hilbert e a curva Z-order, considerando o número total de elementos examinados para as configurações com 4 sub-índices. O coeficiente de correlação da regressão linear para a versão com a curva de Hilbert foi 0,90404 e 0,89957 com a Z-order.

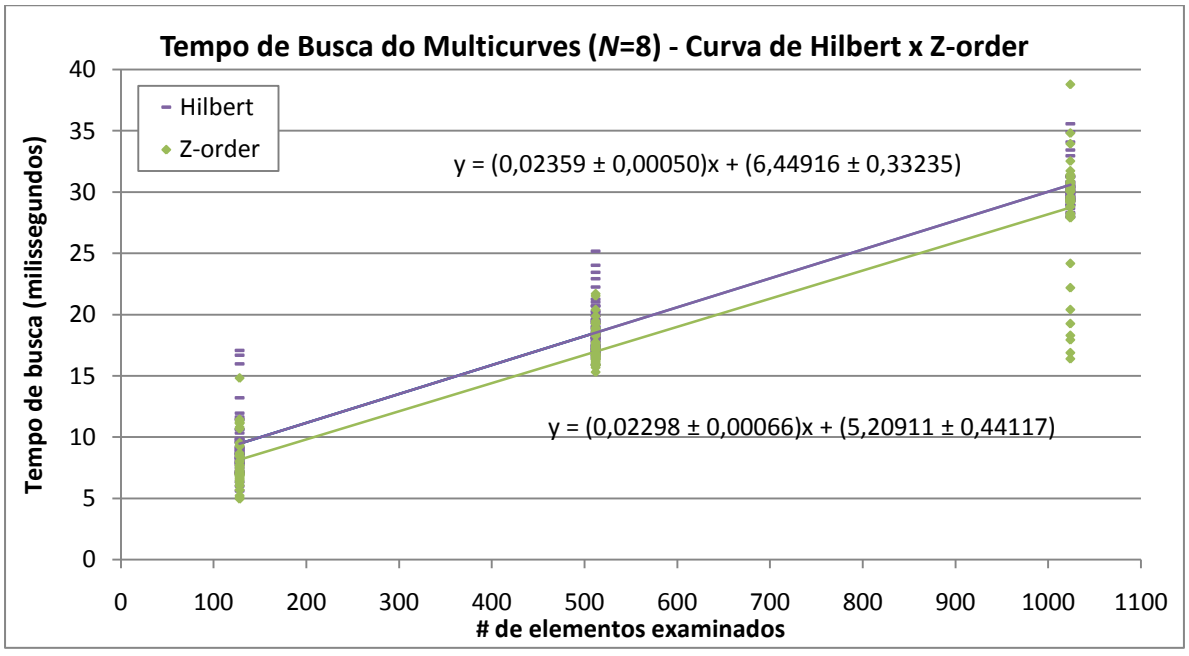

Figura 4.13: Tempo de busca do Multicurves com a curva de Hilbert e a curva Z-order, considerando o número total de elementos examinados para as configurações com 8 sub-índices. O coeficiente de correlação da regressão linear para a versão com a curva de Hilbert foi 0,96836 e 0,94354 com a Z-order.

A Tabela 4.5 mostra a precisão média obtida para as 10 execuções nas diferentes configurações do MONORAIL, com as curvas de Hilbert e Z-order, nos blocos das consultas fáceis e difíceis. Na Tabela 4.6 nós temos os valores para o Multicurves. Como é possível ver em ambos os blocos, a eficácia dos métodos ao longo das configurações não foi muito afetada pelo uso da curva Z-order.

# de elementos examinados	**Fáceis**		**Difíceis**	
	Hilbert	**Z-order**	**Hilbert**	**Z-order**
***M*=4 × *PD*=16**	77,4 ± 0,4	76,8 ± 0,3	39,7 ± 0,5	38,9 ± 0,5
***M*=8 × *PD*=16**	79,3 ± 0,3	78,5 ± 0,3	44,0 ± 0,5	42,8 ± 0,6
***M*=4 × *PD*=128**	80,3 ± 0,3	79,4 ± 0,3	47,0 ± 0,5	46,1 ± 0,6
***M*=8 × *PD*=128**	82,2 ± 0,3	80,8 ± 0,2	51,2 ± 0,7	49,9 ± 0,7

Tabela 4.5: Precisão média do MONORAIL (confiança maior que 0,95) com as curvas de Hilbert e Z-order, para as consultas *fáceis* e *difíceis*, considerando diferentes configurações.

# de elementos examinados	**Fáceis**		**Difíceis**	
	Hilbert	**Z-order**	**Hilbert**	**Z-order**
***N*=4 × *PD*=16**	94,3 ± 0,1	94,2 ± 0,1	60,1 ± 0,5	58,7 ± 0,6
***N*=8 × *PD*=16**	96,7 ± 0,2	96,6 ± 0,1	58,9 ± 0,3	58,8 ± 0,4
***N*=4 × *PD*=128**	96,6 ± 0,1	96,7 ± 0,1	73,1 ± 0,5	71,5 ± 0,4
***N*=8 × *PD*=128**	98,9 ± 0,1	98,8 ± 0,1	77,6 ± 0,5	76,7 ± 0,3

Tabela 4.6: Precisão média do Multicurves (confiança maior que 0,95) com as curvas de Hilbert e Z-order, para as consultas *fáceis* e *difíceis*, considerando diferentes configurações.

Os resultados desse experimento nos levam a concluir que a principal vantagem da utilização (além da facilidade de implementação) de uma curva fractal mais simples, a curva Z-order, nos métodos MONORAIL e Multicurves, está no ganho de tempo na geração dos índices. A desvantagem fica por conta da degradação mínima na precisão dos dois métodos.

4.6 Avaliação da escalabilidade do MONORAIL e do Multicurves

Os objetivos deste experimento são:

- Avaliar os tempos de busca do MONORAIL e do Multicurves com três bases de tamanhos diferentes;
- Comparar a precisão do MONORAIL e do Multicurves, em cada bloco de consultas, nas três bases.

Nós testamos a escalabilidade do MONORAIL e do Multicurves [Valle 2008b] com as três primeiras bases de dados. Os valores dos parâmetros usados são mostrados nas Tabelas 4.7 e 4.8 respectivamente. A multiplicidade igual 32 no MONORAIL e o número de curvas igual a 8 no Multicurves correspondem a melhor configuração em cada um dos métodos. O valor 8 foi utilizado para o parâmetro do tamanho da vizinhança (t) do MONORAIL.

Fator	**Níveis**		
# de vetores de características	32.973.335	55.473.496	130.463.526
Curva	Z-order		
Multiplicidade (*M*)	32		
# de elementos candidatos (*PD*)	$128 \times M$		

Tabela 4.7: Tamanhos das bases de dados e parâmetros usados na avaliação da escalabilidade do MONORAIL.

Fator	**Níveis**		
# de vetores de características	32.973.335	55.473.496	130.463.526
Curva	Z-order		
# de sub-índices (*N*)	8		
# de elementos candidatos (*PD*)	512		

Tabela 4.8: Tamanhos das bases de dados e parâmetros usados na avaliação da escalabilidade do Multicurves.

Dez repetições foram executadas para cada bloco de consultas, cada uma com 4.000 consultas escolhidas aleatoriamente (com reposição). O tempo médio de busca do MONORAIL para cada base de dados é mostrado na Figura 4.14. Os tempos médios de busca do Multicurves para as bases são mostrados na Figura 4.15. Mesmo sendo poucos os pontos, a tendência parece seguir o esperado, ou seja, em ambos os casos, o crescimento é logarítmico com o número de

registros da base de dados. O MONORAIL apresenta um tempo de busca inferior ao do Multicurves.

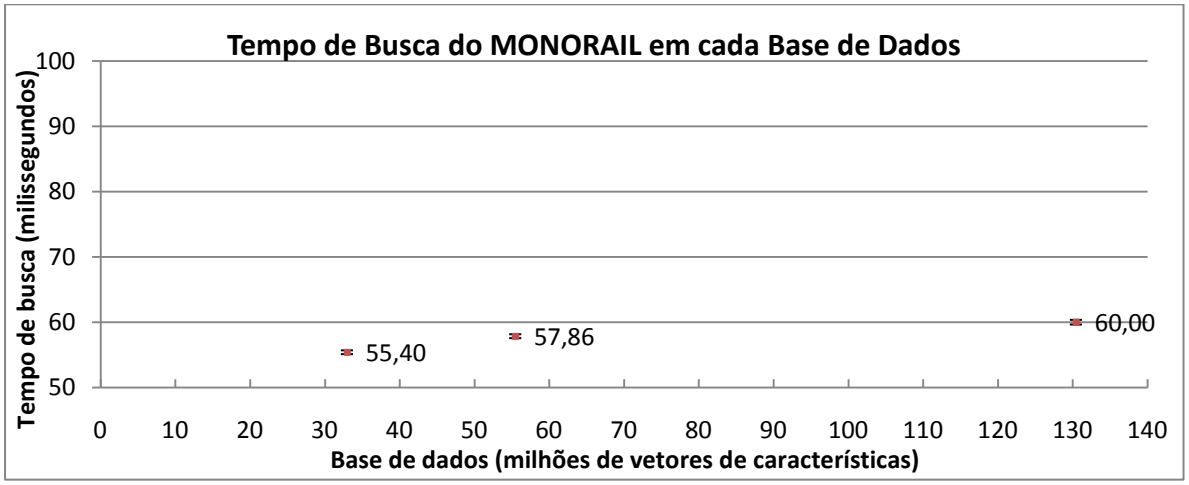

Figura 4.14: Tempo de busca do MONORAIL nos diferentes tamanhos de bases de dados. Os intervalos de confiança (α=0,05) são representados pelas barras de erros verticais.

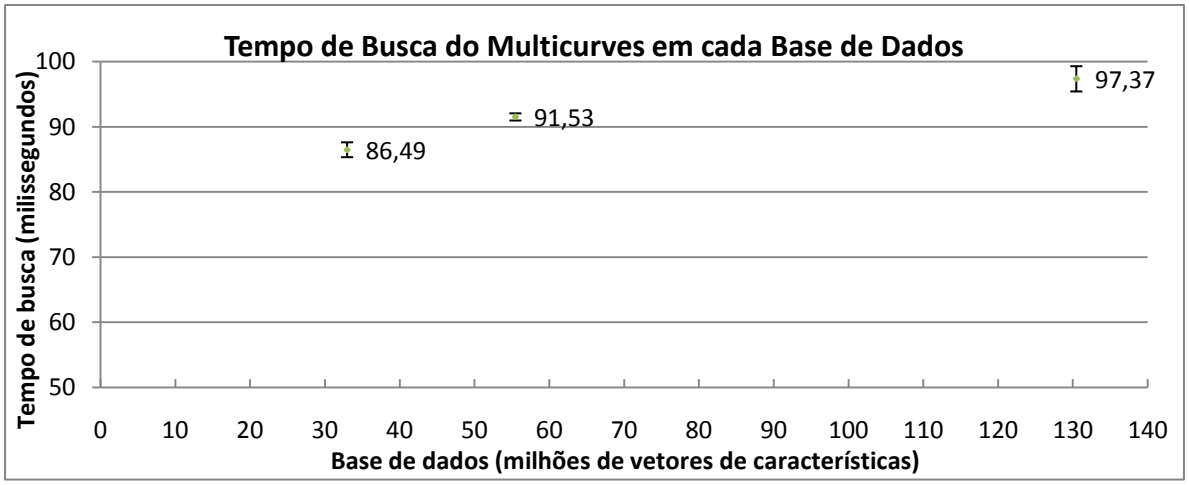

Figura 4.15: Tempo de busca do Multicurves nos diferentes tamanhos de bases de dados. Os intervalos de confiança (α=0,05) são representados pelas barras de erros verticais.

A Figura 4.16 mostra que a precisão do MONORAIL, em cada bloco de consultas, caiu muito pouco na medida em que foram utilizadas bases de dados cada vez maiores. A precisão do Multicurves para os diferentes tamanhos de bases de dados, mostrada na Figura 4.17, apresentou uma queda mais acentuada, principalmente nos blocos das consultas mais difíceis. Mesmo assim, sua precisão foi superior à do MONORAIL.

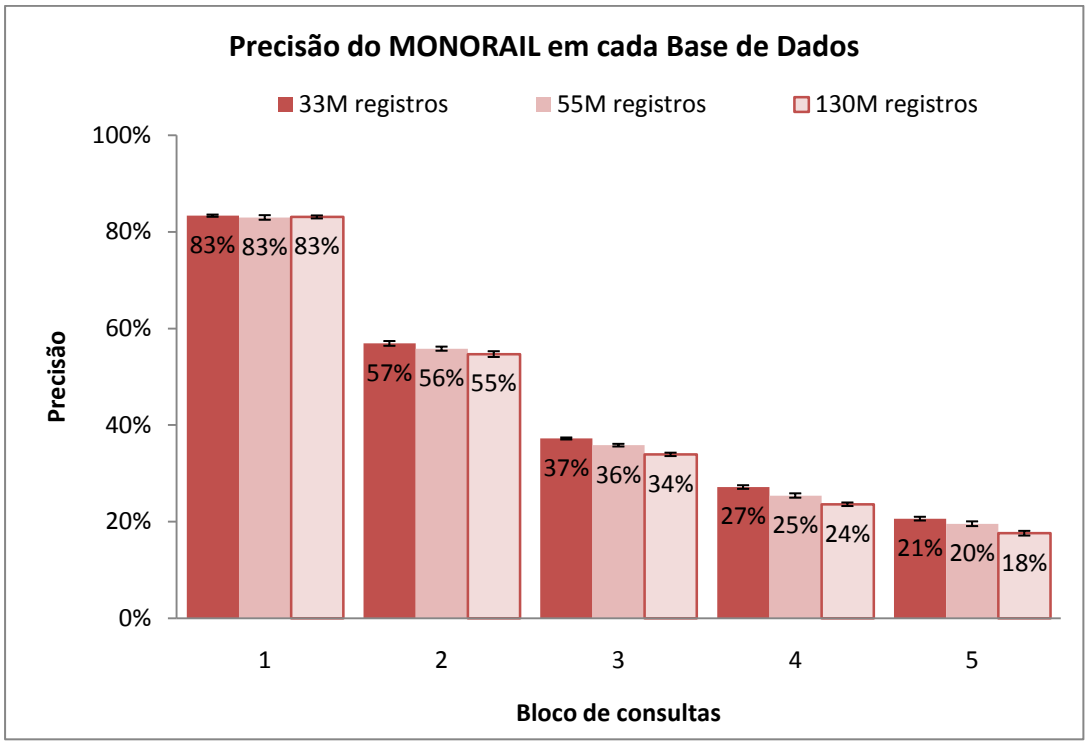

Figura 4.16: Precisão média do MONORAIL em cada base de dados. Os intervalos de confiança (α=0,05) são representados pelas barras de erros verticais.

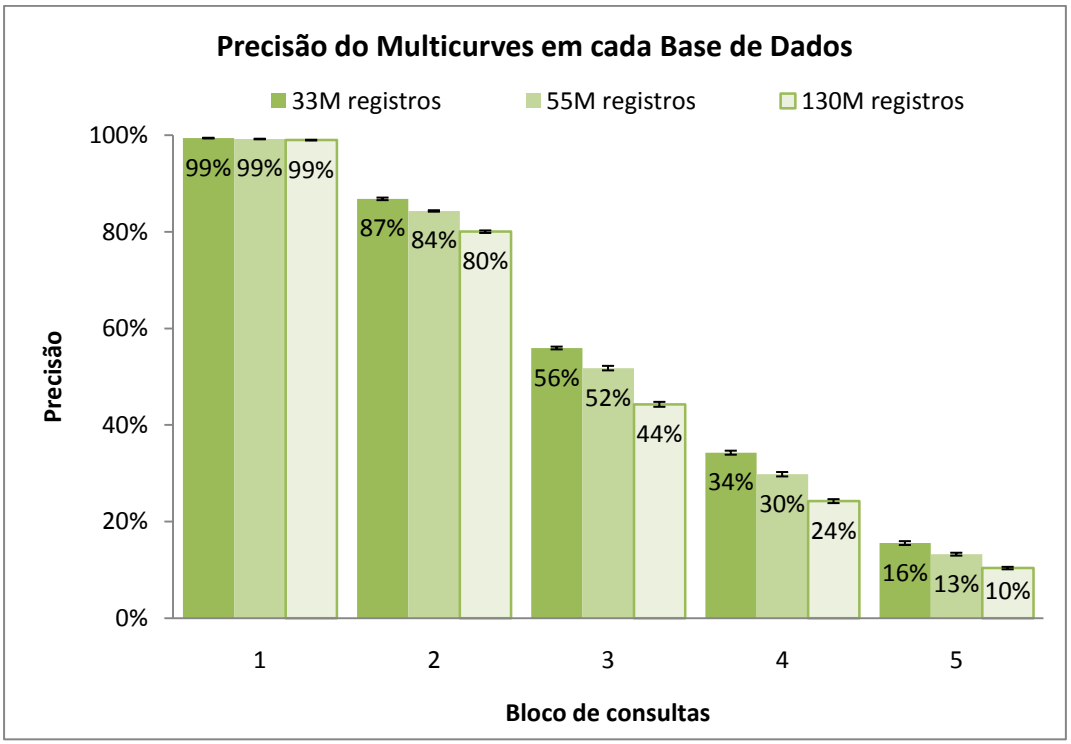

Figura 4.17: Precisão média do Multicurves em cada base de dados. Os intervalos de confiança (α=0,05) são representados pelas barras de erros verticais.

A Tabela 4.9 mostra o tamanho do índice MONORAIL e a soma dos tamanhos dos sub-índices do Multicurves para cada base de dados.

Método	32.973.335	55.473.496	130.463.526
MONORAIL	94,5 GB	168,4 GB	428,3 GB
Multicurves	39,3 GB	66,1 GB	155,5 GB

Tabela 4.9: Tamanho dos índices para cada base de dados.

4.7 Comparação do MONORAIL com o Multicurves

Os objetivos deste experimento são:

- Comparar a média das precisões médias (MPM) e o erro de aproximação relativo (EAR) do MONORAIL com o Multicurves, em cada bloco de consultas;
- Comparar os tempos de busca desses métodos.

Neste experimento, o desempenho do MONORAIL é comparado ao do Multicurves, ambos implementados com a curva Z-order. Os valores dos parâmetros para cada método são mostrados nas Tabelas 4.10 e 4.11 respectivamente.

Cinco repetições foram executadas para cada bloco, cada uma com 1.000 consultas escolhidas aleatoriamente (com reposição). As medidas obtidas em cada execução foram a média das precisões médias (MPM), o erro de aproximação relativo (EAR) e o tempo de busca. A busca foi feita pelos 9 vizinhos mais próximos (k=9) de cada consulta. Somente a base de dados APM (2.871.300 registros) foi usada nesse experimento.

Fator	**Níveis**	
Curva	Z-order	
Multiplicidade (*M*)	32	
# de elementos candidatos (*PD*)	$32 \times M$	$128 \times M$

Tabela 4.10: Parâmetros do MONORAIL usados na comparação.

Fator	**Níveis**	
Curva	Z-order	
# de sub-índices (*N*)	8	
# de elementos candidatos (*PD*)	128	512

Tabela 4.11: Parâmetros do Multicurves usados na comparação.

Os resultados das medidas de eficácia, para as configurações com o menor número de candidatos examinados (1.024), são mostrados pelas Figuras 4.18 e 4.19. Os tempos de busca são apresentados na Tabela 4.12.

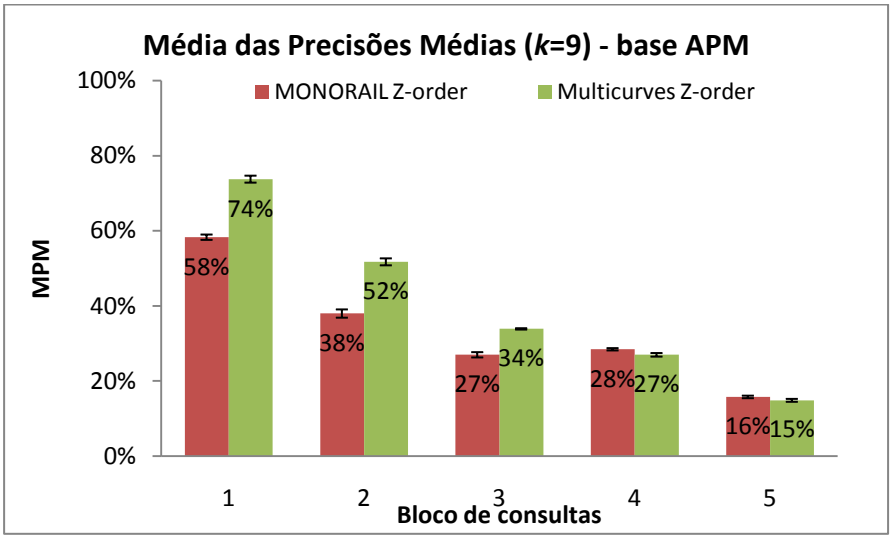

Figura 4.18: Média das precisões médias (busca com *k*=9), MONORAIL x Multicurves na base APM.

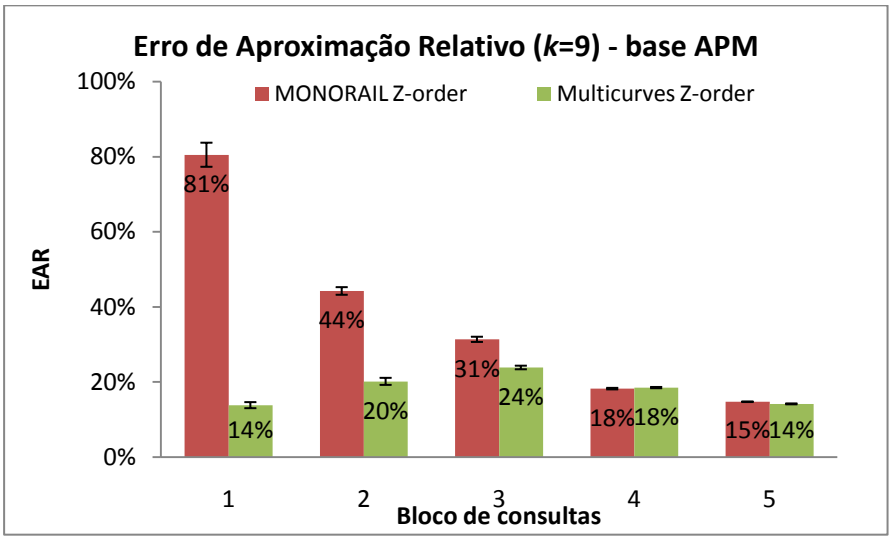

Figura 4.19: Erro de aproximação relativo (busca com *k*=9) dos métodos na base APM.

Método	Tempo de busca (milissegundos)
MONORAIL	17,82 ± 0,29
Multicurves	19,54 ± 0,86

Tabela 4.12: Tempo médio de busca (α=0,05) para as 25 execuções de cada método.

Os resultados indicam que a precisão do MONORAIL é inferior à do Multicurves, mas o MONORAIL é mais rápido, certamente devido ao menor número de acessos aleatórios.

4.8 Comparação do método de Mainar-Ruiz e Pérez-Cortés com o MONORAIL, utilizando a curva Z-order

O objetivo deste experimento é:

- Comparar a precisão do MONORAIL com o método de Mainar-Ruiz *et al*., ambos utilizando a curva Z-order.

Os valores dos parâmetros nos dois métodos, usados neste experimento, são mostrados na Tabela 4.13. O valor 8 foi utilizado para o parâmetro do tamanho da vizinhança (*t*) do MONORAIL.

Fator	**Níveis**			
Método	Mainar-Ruiz *et al*.		MONORAIL	
Curva	Z-order			
Multiplicidade (*M*)	8	16	32	64
# de elementos candidatos (*PD*)	$128 \times M$			

Tabela 4.13: Parâmetros usados na comparação dos métodos, usando a curva Z-order.

Dez repetições foram executadas para cada bloco, cada uma com 4.000 consultas escolhidas aleatoriamente (com reposição). A medida obtida em cada execução foi a precisão na busca pelo primeiro vizinho mais próximo (k=1). Somente a terceira base de dados (32.973.335 registros) foi usada nesse experimento.

A Figura 4.20 ilustra a precisão média obtida para 10 repetições executadas nas diferentes configurações do método de Mainar-Ruiz e Pérez-Cortés [Mainar-Ruiz 2006] e do MONORAIL, usando a curva Z-order, para responder as consultas fáceis. Os intervalos de confiança (α=0,05) são representados pelas linhas verticais na parte superior das barras. Na Figura 4.21 são ilustrados os valores para as consultas difíceis. Em ambos os casos o MONORAIL sempre apresenta um ganho de eficácia.

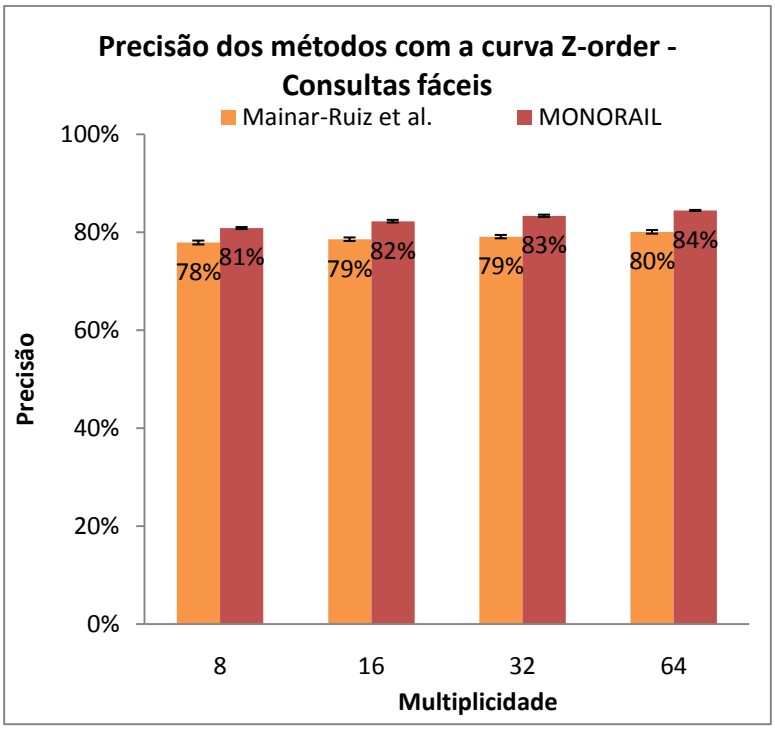

Figura 4.20: Precisão dos métodos com a curva Z-order para responder as consultas do bloco fácil, nas diferentes configurações.

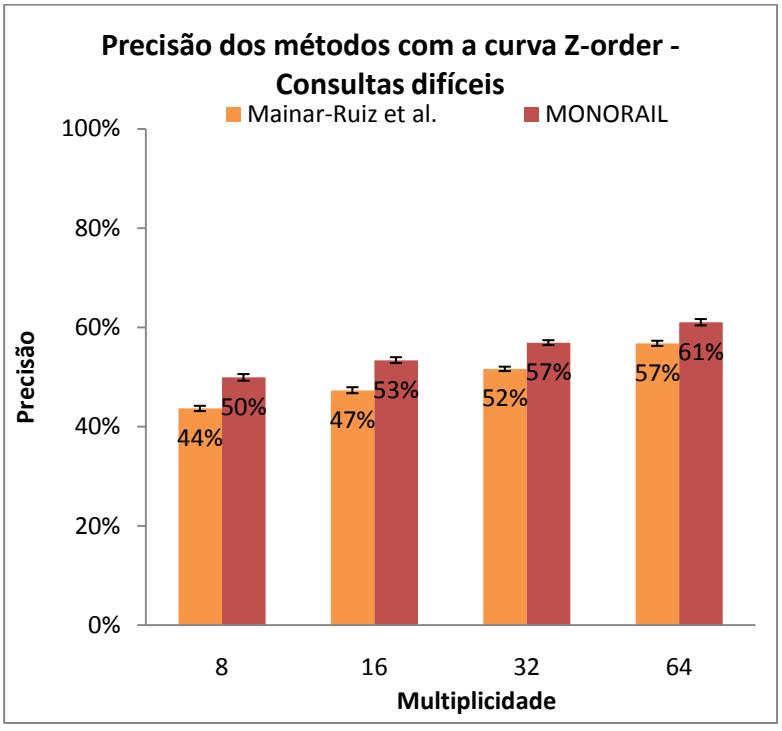

Figura 4.21: Precisão dos métodos com a curva Z-order para responder as consultas do bloco difícil, nas diferentes configurações.

4.9 Exploração de novas parametrizações usando a extensão do Multicurves

O objetivo deste experimento é:

- Comparar a eficácia da versão original do Multicurves com a extensão que permite a escolha arbitrária das dimensões.

A implementação original do Multicurves divide as dimensões uniformemente entre os sub-índices. A versão estendida, descrita na seção 2.8.2, permite que as dimensões sejam distribuídas de forma arbitrária. Os parâmetros usados na versão original, com a curva Z-order, são mostrados na Tabela 4.14.

Fator	**Níveis**	
Curva	Z-order	
# de sub-índices (*N*)	4	8
# de elementos candidatos (*PD*)	128	512

Tabela 4.14: Parâmetros da versão original do Multicurves usados na comparação.

Três tipos de parametrizações foram utilizados na versão estendida. No primeiro deles, foram preparadas cinco configurações com 4 e 8 sub-índices, contendo 32 e 16 dimensões respectivamente, escolhidas aleatoriamente e sem reposição. No segundo, também com cinco configurações de 4 e 8 sub-índices, as dimensões foram escolhidas aleatoriamente com reposição, ou seja, algumas dimensões podem aparecer mais de uma vez, enquanto que outras podem ser ignoradas. No último, tanto a quantidade de sub-índices quanto a quantidade de dimensões foram escolhidas aleatoriamente dentro das faixas indicadas na Tabela 4.15, que sintetiza todos os tipos de parametrizações utilizados.

Tipo	# de configurações	# de sub-índices (*N*)	# de dimensões	Escolha aleatória
1	5	4	32	sem reposição
1	5	8	16	sem reposição
2	5	4	32	com reposição
2	5	8	16	com reposição
3	10	4 a 8	16 a 32	com reposição

Tabela 4.15: Parametrizações da versão estendida do Multicurves.

Cinco repetições foram executadas para cada bloco, cada uma com 1.000 consultas escolhidas aleatoriamente (com reposição). A medida obtida em cada execução foi a média das precisões médias (MPM). A busca foi feita pelos 15 vizinhos mais próximos (k=15) de cada consulta. Somente a base de dados APM (2.871.300 registros) foi usada nesse experimento.

Os tipos de parametrizações 1 e 2 foram comparados com a versão original na mesma quantidade de sub-índices. Cinco configurações do tipo 3 foram comparadas com a versão original de 4 sub-índices e cinco com a de 8 sub-índices. Todas as parametrizações do tipo 1 apresentaram um desempenho melhor que a versão original. As parametrizações do tipo 2 com 8 sub-índices também foram superiores e as configurações com 4 sub-índices foram inferiores à versão original. As cinco configurações do tipo 3, comparadas com a versão original de 4 sub-índices apresentaram um desempenho superior. Três das cinco configurações do tipo 3, comparadas com a versão original de 8 sub-índices tiveram um resultado melhor.

A Figura 4.22 mostra os resultados das configurações dos tipos 1 e 2 comparadas com a versão original de 8 sub-índices, considerando o menor número de candidatos examinados (PD=128). As linhas horizontais no final das barras representam os intervalos de confiança (α=0,05). A versão estendida apresenta uma eficácia superior à original em todos os blocos de consultas. Entre a escolha aleatória das dimensões com ou sem reposição, percebemos que ignorar algumas dimensões prejudica a precisão, mas ainda assim é melhor do que a versão original. Há um indício de que a escolha aleatória das dimensões melhora a eficácia do Multicurves.

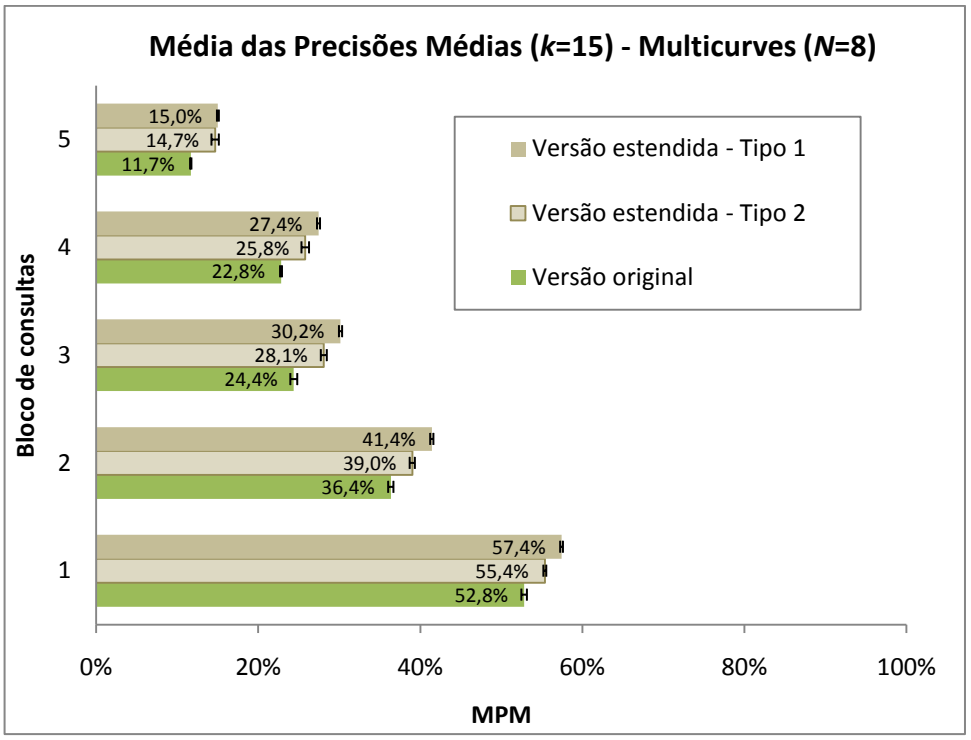

Figura 4.22: Média das precisões médias (busca com *k*=15), versão original do Multicurves (8 sub-índices) x versão estendida (configurações dos tipos 1 e 2).

4.10 Conclusões

Os resultados obtidos nos experimentos nos levam às seguintes conclusões:

- O MONORAIL resolve o problema da poluição do índice, apresentado pelo método de Mainar-Ruiz *et al.* (seção 4.4);
- Os índices MONORAIL ficaram pelo menos 30% menores em termos de tamanho ocupado em disco. Isso confirma a geração de uma quantidade maior de representantes desnecessários pelo método de Mainar-Ruiz *et al.;*
- A superioridade do MONORAIL em comparação com o método de Mainar-Ruiz *et al*. também é verificada com a curva Z-order (seção 4.8);
- A precisão da curva Z-order é ligeiramente inferior à da curva de Hilbert, nas implementações dos métodos MONORAIL e Multicurves (seção 4.5);
- O uso da curva Z-order não contribui muito para a redução no tempo de busca dos métodos (seção 4.5);

- A principal vantagem da curva Z-order em relação à curva de Hilbert é o ganho de tempo obtido na criação dos índices (seção 4.5), aproximadamente 5 vezes mais rápida no caso do MONORAIL;
- A escalabilidade do MONORAIL e do Multicurves foi testada com sucesso, utilizando três bases de dados de tamanhos diferentes (seção 4.6);
- A precisão do MONORAIL é inferior à do Multicurves (seções 4.5, 4.6 e 4.7);
- Há uma indicação de que a escolha aleatória das dimensões no Multicurves aumenta a eficácia desse método (seção 4.9).

Capítulo 5

Conclusões

A indexação multidimensional é fundamental para o bom funcionamento dos sistemas de recuperação de informações multimídia, que utilizam os vetores de características para representar os documentos.

Os índices multimídia sofrem da "maldição da dimensionalidade", associada a um conjunto de fenômenos que afetam profundamente o desempenho da indexação em espaços com muitas dimensões.

A família de técnicas baseadas nas curvas de Peano apresenta vantagens importantes para as grandes bases dados, como o fato dos métodos serem dinâmicos, ou seja, permitem a inserção e a remoção dos dados sem a necessidade de reconstruir o índice. Porém, a existência de zonas de descontinuidade é o maior entrave para a eficácia desses métodos em alta dimensionalidade. Muitos pesquisadores partiram para a adoção de múltiplas curvas, na expectativa de que em pelo menos uma delas o problema não aconteça. Contudo, isso exige um acesso aleatório por curva.

O método de indexação multidimensional, MONORAIL, principal contribuição dessa dissertação, ataca o problema das regiões de descontinuidade utilizando representantes cuidadosamente posicionados (em uma única curva) para preservar a propriedade da curva que permite a indexação multidimensional. Os experimentos comprovaram uma melhoria significativa em relação à técnica pioneira na utilização dos representantes.

A metodologia adotada na parte experimental possibilitou uma análise consistente dos resultados e pode ser empregada em outras avaliações da busca por similaridade.

As implementações do MONORAIL e do Multicurves [Valle 2008b] com a curva Z-order, outra contribuição desse trabalho, mostraram uma precisão dessa curva ligeiramente inferior à curva de Hilbert, mas com uma grande redução no tempo de criação dos índices.

As aplicações dos métodos de indexação multidimensional não se restringem aos dados multimídia, em biometria e segurança essas poderiam ser: classificação e busca baseadas em vetores de características biométricos, como informações de face, impressões digitais, retina, íris, voz, etc.

Outras aplicações variadas: mineração de dados de alta dimensionalidade — por exemplo, aceleração de algoritmos de agrupamento em aplicações OLAP (*On-Line Analytical Processing*) para busca de padrões interessantes de transações, detecção de fraudes, etc. Casamento de sequências por casamento acelerado de *substrings* com diversas aplicações em recuperação de informação textual, bioinformática, etc.

A pesquisa e o desenvolvimento de um método de indexação multidimensional, dinâmico, adaptado à memória secundária (escalável) e com um erro de aproximação previsível (e irrelevante) são indispensáveis para a evolução das aplicações que manipulam o crescente volume de dados de alta dimensionalidade. O estudo teórico do comportamento do MONORAIL poderia culminar em um modelo de precisão do método aumentando dessa forma, por exemplo, a viabilidade de uma implementação em um gerenciador de banco de dados comercial. Esse modelo seria usado no planejamento e otimização de consultas.

Ainda que estejamos um pouco distante de um modelo teórico, uma exploração paramétrica mais abrangente e uma análise mais profunda das consultas perdidas pelo método poderiam induzir um aperfeiçoamento no posicionamento dos representantes.

Algumas sugestões de possíveis melhorias para o MONORAIL:

- Verificar a presença de um representante nas proximidades antes de incluir mais outro;
- Estudar a escalabilidade do método em SSD (*Solid State Drives*);
- Permitir a variação do valor do parâmetro do tamanho da vizinhança a considerar no entorno dos pontos médios (t) de acordo com a gravidade da descontinuidade;
- Avaliar a relação entre os parâmetros PD x M x t;
- Implementar a criação do índice de forma incremental e concorrente.

O método MONORAIL e os resultados dos experimentos da seção 4.4 foram publicados em um artigo da *International Conference on Pattern Recognition* (ICPR) 2010 [Akune 2010]. Além disso, o invento foi depositado em um pedido de patente com o título "Método de Indexação para Dados de Alta Dimensionalidade em Grandes Bases de Dados Usando Uma Curva de Peano (Curva Preenchedora do Espaço) e Múltiplos Representantes para os Dados", número do protocolo: 018100048409, em 20/12/2010, INPI (Instituto Nacional da Propriedade Industrial).

A versão com a curva Z-order e os resultados dos experimentos das seções 4.5, 4.6 e 4.8 serão submetidos em um artigo para a revista *IEEE Transactions on Knowledge and Data Engineering* (TKDE*)*.

Referências Bibliográficas

[Akune 2010] Fernando Akune, Eduardo Valle and Ricardo Torres. MONORAIL: A Disk-Friendly Index for Huge Descriptor Databases. In *Proceedings of the International Conference on Pattern Recognition*, pages 4145–4148, Istanbul, Turkey, August 2010.

[Almeida 2010] Jurandy Almeida, Eduardo Valle, Ricardo da S. Torres, Neucimar J. Leite. DAHC-tree: An Effective Index for Approximate Search in High-Dimensional Metric Spaces. *Journal of Information and Data Management*, vol. 1, n. 3, pages 375–390, October 2010.

[Asano 1997] Tetsuo Asano, Desh Ranjan, Thomas Roos, Emo Welzl and Peter Widmayer. Space-filling curves and their use in the design of geometric data structures. *Theoretical Computer Science*, vol. 181, n. 1, pages 3–15, July 1997.

[Bayer 1972] Rudolf Bayer and E. M. McCreight. Organization and maintenance of large ordered indexes. *Acta Informatica*, vol. 1, n. 3, pages 173–189, September 1972.

[Bellman 1961] Richard Bellman. *Adaptive Control Processes: a guided tour*. Princeton University Press, 1961.

[Bentley 1975] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, vol. 18, n. 9, pages 509–517, September 1975. ACM Press, New York, NY, USA.

[Berchtold 1998] Stefan Berchtold, Christian Böhm and Hans-Peter Kriegal. The pyramid-technique: towards breaking the curse of dimensionality. *ACM SIGMOD Record*, vol. 27, n. 2, pages 142–153, June 1998. ACM, New York, NY, USA.

[Beyer 1999] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, Uri Shaft. When Is "Nearest Neighbor" Meaningful? In *Proceedings of the International Conference on Database Theory*, pages 217–235, Jerusalem, Israel, January 10–12, 1999.

[Böhm 2001] Christian Böhm, Stefan Berchtold and Daniel A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys (CSUR)*, vol. 33, n. 3, pages 322–373, September 2001. ACM, New York, NY, USA.

[Butz 1971] Arthur R. Butz. Alternative Algorithm for Hilbert's Space-Filling Curve. *IEEE Transactions on Computers*, vol. C-20, n. 4, pages 424–426, April 1971. IEEE Computer Society.

[Carvalho 2006] André C. Ponce de Leon F. de Carvalho, *et al.*. *Grandes Desafios da Pesquisa em Computação no Brasil – 2006 - 2016*. Seminário Grandes Desafios da Sociedade Brasileira de Computação, São Paulo, SP, Brasil, May 2006.

[Castelli 2002] Vittorio Castelli and Lawrence D. Bergman. *Image Databases: Search and Retrieval of Digital Imagery*. John Wiley & Sons, 2002.

[Cha 2002] Guang-Ho Cha, Xiaoming Zhu, Dragutin Petkovic and Chin-Wan Chung. An efficient indexing method for nearest neighbor searches in high-dimensional image databases. *IEEE Transactions on Multimedia*, vol. 4, n. 1, pages 76–87, 2002. IEEE, New York, NY, USA.

[Chen 2005] Hue-Ling Chen e Ye-In Chang. Neighbor-finding based on space-filling curves. *Information systems*, vol. 30, n. 3, pages 205–226, 2005.

[Ciaccia 1997] Paolo Ciaccia, Marco Patella and Pavel Zezula. M-Tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Proceedings of the International Conference on Very Large Data Bases*, pages 426–435, August 25–29, 1997. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Datar 2004] Mayur Datar, Nicole Immorlica, Piotr Indyk and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 12th annual symposium on Computational geometry*, pages 253–262, Brooklyn, New York, USA, June 08–11, 2004. ACM, New York, NY, USA.

[Fagin 2003] Ronald Fagin, Ravi Kumar and D. Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 301–312, San Diego, California, USA, June 09–12, 2003. ACM, New York, NY, USA.

[Faloutsos 1986] Christos Faloutsos. Multiattribute hashing using Gray codes. In *Proceedings of the 1986 ACM SIGMOD International Conference on Management of data*, pages 227–238, Washington, DC, USA, May 28–30, 1986. ACM Press, New York, NY, USA.

[Faloutsos 1988] Christos Faloutsos. Gray Codes for Partial Match and Range Queries. *IEEE Transactions on Software Engineering*, vol. 14, n. 10, pages 1381–1393, October 1988.

[Faloutsos 1989] Christos Faloutsos and Shari Roseman. Fractals for secondary key retrieval. In *Proceedings of the 8th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 247–252, Philadelphia, Pennsylvania, USA, March 29–31, 1989. ACM Press, New York, NY, USA.

[Faloutsos 1995] Christos Faloutsos and King-Ip Lin. FastMap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data,* pages 163–174, 1995. ACM, New York, NY, USA.

[Friedman 1976] Jerome Friedman, Jon L. Bentley and Raphael A. Finkel. *An Algorithm for Finding Best Matches in Logarithmic Expected Time*. Technical Report CS-TR-75-482. Stanford University, Stanford, CA, USA, 1976.

[Gaede 1998] Volker Gaede and Oliver Günther. Multidimensional access methods. *ACM Computing Surveys (CSUR)*, vol. 30, n. 2, pages 170–231, June 1998. ACM, New York, NY, USA.

[Gionis 1999] Aristides Gionis, Piotr Indyk and Rajeev Motwani. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529, San Francisco, CA, USA, September 07–10, 1999. Morgan Kaufmann Publishers Inc..

[Guttman 1984] Antonin Guttman. R-Trees: a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, pages 47–57, Boston, MA, USA, 1984. ACM, New York, NY, USA.

[Hilbert 1891] David Hilbert. Über die stetige Abbildung einer Line auf ein Flächenstück. *Mathematische Annalen*, vol. 38, n. 3, pages 459–460, September 1891. Springer, Berlin / Heidelberg.

[Indyk 1998] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the 13th annual ACM symposium on Theory of computing*, pages 604–613, Dallas, Texas, USA, May 24–26, 1998. ACM, New York, NY, USA.

[Katayama 1997] Norio Katayama and Shin'ichi Satoh. The SR-Tree: an index structure for high-dimensional nearest neighbor queries. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 369–380. Tucson, Arizona, United States, May 11–15, 1997. ACM, New York, NY, USA.

[Lehmer 1951] Derrick Henry Lehmer. Mathematical methods in large-scale computing units. In *Proceedings 2nd Symp. on Large-Scale digital calculating machinery*, pages 141–146, 1951. Harvard Univ. Press, Cambridge, MA.

[Lejsek 2009] Herwig Lejsek, Friðrik Heiðar Ásmundsson, Björn Þór Jónsson and Laurent Amsaleg. NV-Tree: An Efficient Disk-Based Index for Approximate Search in Very Large High-Dimensional Collections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, n. 5, pages 869–883, May 2009. IEEE Computer Society, Washington, DC, USA.

[Li 2002] Chen Li, Edward Chang, Hector Garcia-Molina and Gio Wiederhold. Clustering for approximate similarity search in high-dimensional spaces. *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, n. 4, pages 792–808, July 2002. IEEE Educational Activities Department, Piscataway, NJ, USA.

[Liao 2001] Swanwa Liao, Mario A. Lopez and Scott T. Leutenegger. High Dimensional Similarity Search With Space Filling Curves. In *Proceedings of the International Conference on Data Engineering*, pages 615–622, Heidelberg, Germany, April 02–06, 2001. IEEE Computer Society Washington, DC, USA.

[Lowe 2004] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, vol. 60, n. 2, pages 91–110, November 2004. Springer, Netherlands.

[Mainar-Ruiz 2006] Gloria Mainar-Ruiz and Juan-Carlos Pérez-Cortés. Approximate Nearest Neighbor Search using a Single Space-filling Curve and Multiple Representations of the Data Points. In *Proceedings of the 18th International Conference on Pattern Recognition*, vol. 2, pages 502–505, Wan Chai, Hong Kong, August 20–24, 2006. IEEE Computer Society, Washington, DC, USA.

[Markov 2008] Krassimir Markov, Krassimira Ivanova, Ilia Mitov and Stefan Karastanev. Advance of the Access Methods. *International Journal "Information Technologies and Knowledge"*, vol. 2, n. 2, pages 123–135, 2008. Institute of Information Theories and Applications FOI ITHEA.

[Medeiros 2008] Claudia Bauzer Medeiros. Grand Research Challenges in Computer Science in Brazil. *Computer*. IEEE Computer Society, vol. 41, n. 6, pages 59–65, June 2008.

[Moënne-Loccoz 2005] Nicolas Moënne-Loccoz. *High-Dimensional Access Methods for Efficient Similarity Queries*. Technical Report 05.05. University of Geneva, Computer Science Center Computer Vision Group, Geneva, Switzerland, May 2005.

[Mokbel 2003] Mohamed F. Mokbel, Walid G. Aref and Ibrahim Kamel. Analysis of Multi-Dimensional Space-Filling Curves. *Geoinformatica*, vol. 7, n. 3, pages 179–209, September 2003. Kluwer Academic Publishers Hingham, Massachusetts, USA.

[Morton 1966] G. M. Morton. *A computer oriented geodetic data base and a new technique in file sequencing*. Technical Report. IBM Ltd., Ottawa, Canada, 1966.

[Peano 1890] Giuseppe Peano. Sur une courbe, qui remplit toute une aire plane. *Mathematische Annalen*, vol. 26, n. 1, March 1890. Springer, Berlin / Heidelberg, Germany.

[Penatti 2010] Otávio A.B. Penatti and Ricardo da S. Torres. User-Oriented Evaluation of Color Descriptors for Web Image Retrieval. *The European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2010)*. Lecture Notes in Computer Science, vol. 6273, pages 486–489, 2010.

[Pérez 1998] Juan-Carlos Pérez and Enrique Vidal. An approximate nearest neighbours search algorithm based on the Extended General Spacefilling Curves Heuristic. *Advances in Pattern Recognition,* vol. 1451, pages 697–706, 1998. Springer Berlin / Heidelberg.

[Rubner 2000] Yossi Rubner, Carlo Tomasi, Leonidas J. Guibas. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, vol. 40, n. 2, pages 99–121, November 2000.

[Sagan 1994] Hans Sagan. *Space-filling curves*. Springer-Verlag, Berlin / New York. September 2, 1994.

[Samet 2006] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann. San Francisco, CA, USA, 2006.

[Shakhnarovich 2006] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)*. The MIT Press. Cambridge, Massachusetts, 2006.

[Shepherd 1999] John A. Shepherd, Xiaoming Zhu and Nimrod Megiddo. A fast indexing method for multidimensional nearest neighbor search. In *SPIE Conference on Storage and Retrieval for Image and Video Databases VII*, vol. 3656, pages 350–355, San Jose, California, USA, January 26-29, 1999.

[Swain 1991] Michael J. Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, vol. 7, n. 1, pages 11–32, 1991.

[Teodoro 2008] George Teodoro, Daniel Fireman, Dorgival Guedes, Wagner Meira Jr., Renato Ferreira. Achieving Multi-Level Parallelism in the Filter-Labeled Stream Programming Model. *37th International Conference on Parallel Processing*, pages 287–294, 2008.

[Torres 2006] Ricardo da Silva Torres and Alexandre Xavier Falcão. Content-Based Image Retrieval: Theory and Applications. *Revista de Informática Teórica e Aplicada*, vol. 13, n. 2, pages 61–185, 2006.

[Torres 2008] Ricardo da Silva Torres, Javier A. M. Zegarra, Jefersson A. dos Santos, Cristiano D. Ferreira, Otávio A. B. Penatti, Fernanda Andaló, Jurandy Almeida. Recuperação de Imagens: Desafios e Novos Rumos. In *Seminário Integrado de Software e Hardware (SEMISH)*, July 2008.

[Traina 2000] Caetano Traina Jr., Agma Traina, Bernhard Seeger and Christos Faloutsos. Slim-Trees: High Performance Metric Trees Minimizing Overlap between Nodes. In *Advances in Database Technology — EDBT 2000*, pages 51–65. Lecture Notes in Computer Science, vol. 1777, 2000. Springer, Berlin / Heidelberg, Germany.

[Traina 2007] Caetano Traina Jr., Roberto F. Filho, Agma J. Traina, Marcos R. Vieira, and Christos Faloutsos. The Omni-family of all-purpose access methods: a simple and effective way to make similarity search more efficient. *The VLDB Journal*, vol. 16, n. 4, pages 483–505, October 2007.

[Valle 2007] Eduardo Valle, Matthieu Cord and Sylvie Philipp-Foliguet. 3-Way-Trees: A Similarity Search Method for High-Dimensional Descriptor Matching. In *Proceedings of the 2007 IEEE International Conference on Image Processing*, vol. 1, pages I.173–176, San Antonio, TX, USA, September 16–19, 2007. IEEE Computer Society, Washington, DC, USA.

[Valle 2008] Eduardo Valle. *Local-Descriptor Matching for Image Identification Systems*. PhD. Thesis. Université de Cergy Pontoise. France, 2008.

[Valle 2008b] Eduardo Valle, Matthieu Cord and Sylvie Philipp-Foliguet. High-Dimensional Descriptor Indexing for Large Multimedia Databases. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management — CIKM*, pages 739–748, Napa Valley, CA, USA, October 26–30, 2008. ACM, New York, NY, USA.

[Valle 2010] Eduardo Valle, Matthieu Cord, Sylvie Philipp-Foliguet, and David Gorisse. Indexing personal image collections: a flexible, scalable solution. In *IEEE Transactions on Consumer Electronics,* vol. 56, n. 3, pages 1167–1175, August 2010.

[Weber 1997] Roger Weber and Stephen Blott. *An Approximation-Based Data Structure for Similarity Search*. Report TR1997b. ETH Zentrum, Zurich, Switzerland. 1997.

[Zezula 2005] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search: The Metric Space Approach (Advances in Database Systems)*. Springer-Verlag, New York, Inc., Secaucus, NJ, USA, 2005.

[Zhang 2004] Rui Zhang, Beng Chin Ooi and Kian-Lee Tan. Making the Pyramid Technique Robust to Query Types and Workloads. In *Proceedings of the 20th International Conference on Data Engineering*, pages 313–324. Boston, MA, USA, March 30-April 2, 2004. IEEE Computer Society, Washington, DC, USA.