Universidade Estadual de Campinas
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE TELEMÁTICA

Modelo para o Gerenciamento de Bancos de Dados de Imagens

Autor: Sahudy Montenegro González

Orientador: Prof. Dr. Akebo Yamakami

Banca:

Prof. Dr. Cláudia M. Bauzer Medeiros (IC/UNICAMP)
Prof. Dr. Clésio Luiz Tozzi (FEEC/UNICAMP)
Prof. Dr. Yuzo Iano (FEEC/UNICAMP)
Prof. Dr. Anésio dos Santos Júnior (FEEC/UNICAMP)

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos necessários para a obtenção do título de Doutor em Engenharia Elétrica.

Campinas, Dezembro de 2003

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

M764m

Montenegro González, Sahudy
Modelo para o gerenciamento de bancos de dados de imagens / Sahudy Montenegro González.--Campinas, SP: [s.n.], 2003.

Orientador: Akebo Yamakami
Tese (Doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Banco de dados. 2. Processamento de imagens. 3. Modelagem de dados. 4. Linguagem de consulta (Banco de dados). 5. Lógica difusa. I. Yamakami, Akebo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Projeto de Pesquisa referente ao Processo No. 99/04150-8
FAPESP
FUNDAÇÃO DE AMPARO À PESQUISA DO ESTADO DE SÃO PAULO

Resumo

Muitos dos problemas na recuperação de imagens são originados pelas necessidades de domínio específico e por isto as soluções encontradas são limitadas somente a um conjunto de domínios de aplicação. A falta de modelos genéricos para dados e para recuperação de imagens traz, freqüentemente, grandes dificuldades aos pesquisadores. Por outro lado, o termo banco de dados de imagens é usado indistintamente nas pesquisas que envolvem imagens tanto na área de bancos de dados quanto na de visão computacional. Contudo, seria mais eficiente a integração das vantagens que oferecem os bancos de dados com a manipulação, interpretação e tratamento próprio das imagens em um sistema de gerenciamento de imagens sobre bancos de dados.

Nosso trabalho visa oferecer um sistema de propósito geral, baseado em um modelo e uma arquitetura flexíveis, orientadas a facilitar o gerenciamento das imagens, com as quais uma aplicação pode ser desenvolvida de maneira simples. O nosso modelo de gerenciamento de imagens está composto por um modelo para a recuperação de imagens e uma linguagem de consulta fuzzy, que caracteriza a similaridade entre imagens. O modelo preocupa-se em suportar um grande número de consultas sobre imagens, integrando informação semântica, não semântica e abstração de informação em vários níveis. Propomos um mecanismo de retroalimentação das consultas que leva em consideração a subjetividade dos usuários. O sistema fornece uma maneira bem flexível para as aplicações definir suas características particulares. Oferece um repositório de algoritmos para integrar as operações sobre imagens, de maneira que elas estejam disponíveis para serem consultadas e compartilhadas pelos usuários.

Abstract

Most of the solutions proposed in image database applications are limited to a specific application domain. The lack of generic models frequently introduces troubles to researchers. On the other side, the advantages of the database management systems and the image processing manipulation will be integrated in a single image database management system.

This work develops a general purpose image management system, based on an extensible and flexible model and architecture, oriented to ease the work of the image application developers. Our image management model proposes an image retrieval and a fuzzy query language models that reflects the image similarity. The model supports a large stock of image query integrating semantic and no semantic features. A relevance feedback mechanism is defined to consider the user subjectivity in the image retrieval process. The system provides a simple way to applications define their domain-dependent features. The system offers a repository to integrate image retrieval algorithms. This repository acts as a support to developers and to the development of image retrieval applications.

Conteúdo

1 Introdução **1**

1.1 Definição do Problema 1

1.2 Motivação e Objetivos 2

2 Gerenciamento de Bancos de Dados de Imagens **4**

2.1 Esforços Correntes para Desenvolvimento de Sistemas Gerenciadores 5

2.2 Trabalhos Relacionados 6

2.2.1 Sistemas de Bancos de Dados Estendidos e Extensíveis 7

2.2.2 Sistemas de Bancos de Dados Orientados a Objetos 8

2.2.3 Sistemas Orientados a Imagens 9

2.3 Impacto das Imagens sobre o Projeto de um Sistema de Gerenciamento de Bancos de Dados 10

2.4 Tópicos Relacionados com o Gerenciamento de Imagens 11

2.4.1 Recuperação de Imagens 11

2.4.1.1 Mecanismo de Retroalimentação 12

2.4.1.2 Avaliação de Sistemas de Recuperação de Imagens 13

2.4.2 Modelagem das Imagens 13

2.4.2.1 Características das Imagens 13

2.4.2.2 Padronização dos Metadados das Imagens 15

2.4.3 Critério de Similaridade e Normalização 16

2.4.4 Estruturas de Indexação e Métodos de Acesso 17

2.4.4.1 Indexação baseada em Conceito (*Concept-based Indexing*) 17

2.4.4.2 Indexação baseada em Conteúdo (*Content-based Indexing*) . . . 18

2.4.4.3 Combinação dessas duas abordagens 18

2.4.5 Linguagem e Interface de Consulta 19

2.4.5.1 Primitivas de Consultas sobre Imagens 19

2.5 Sistemas de Recuperação de Imagens 21

2.5.1 Quadro Resumo 32

2.5.2 Algoritmos de Recuperação de Imagens 33

2.5.3 Arquitetura de um Sistema de Recuperação de Imagens 33

2.6 Usuários das Aplicações 33

2.7 Sumário 35

3 Modelo para o Gerenciamento de Imagens sobre Bancos de Dados 36
3.1 Introdução . . . 36
3.2 Esquema para o Gerenciamento de Bancos de Dados de Imagens . . . 37
3.3 Modelo das Imagens . . . 38
3.3.1 Atributos . . . 38
3.3.2 Descritores . . . 39
3.3.3 Imagem . . . 40
3.4 Modelo de Recuperação de Imagens . . . 42
3.4.1 Tuplas de Operações . . . 42
3.4.2 Modelo Fuzzy para as Consultas . . . 43
3.4.2.1 Modificadores Fuzzy . . . 45
3.4.2.2 Resolução dos Operadores Lógicos . . . 53
3.4.3 Mecanismo de Retroalimentação das Consultas . . . 54
3.4.3.1 Exemplo . . . 56
3.5 Modelo de Manipulação de Imagens sobre Banco de Dados . . . 57
3.6 Modelo baseado no Padrão ODMG . . . 58
3.6.1 Exemplo . . . 58
3.6.2 Modelo de Objetos para Imagens . . . 60
3.6.3 Declaração dos Bancos de Dados de Imagens . . . 61
3.6.3.1 ODL-I . . . 61
3.6.3.2 Linguagem de Especificação das Aplicações . . . 63
3.6.4 Linguagem de Consulta: OQL-I . . . 67
3.6.4.1 Operadores de Similaridade . . . 67
3.6.4.2 Relacionamentos Espaciais dos Objetos . . . 67
3.6.4.3 Outros Operadores . . . 68
3.6.4.4 Definição de Conceitos . . . 68
3.6.5 Resolução de Primitivas de Consultas . . . 69
3.7 Sumário . . . 71

4 AMID: Sistema para o Gerenciamento de Aplicações de Bancos de Dados de Imagens 74
4.1 Introdução . . . 74
4.2 Arquitetura Geral do Sistema AMID . . . 74
4.3 Interface de AMID com as aplicações . . . 75
4.3.1 Serviços de Interface: Conexão ao Banco de Dados . . . 76
4.3.2 Aplicações . . . 78
4.4 Processador das Solicitações . . . 78
4.5 Tradutor . . . 80
4.6 Catálogo do Sistema . . . 80
4.6.1 Catálogo de Conceitos . . . 81
4.7 Usuários do Sistema Amid . . . 81
4.8 Detalhes de Implementação . . . 82
4.9 Sumário . . . 84

5 Repositório de Algoritmos de Recuperação de Imagens **85**
5.1 Introdução . . . 85
5.2 Objetivos do Repositório de Algoritmos . . . 86
5.3 Definição de Metadados para os Algoritmos . . . 86
5.3.1 Metadados da Interface . . . 87
5.3.1.1 Definição de Esquema . . . 88
5.3.2 Metadados Básicos dos Algoritmos . . . 88
5.3.2.1 Definição de Esquema . . . 89
5.3.2.2 Exemplo . . . 90
5.3.3 Metadados sobre a Criação de Algoritmos . . . 90
5.3.3.1 Histórico de Uso . . . 90
5.3.3.2 Metadados dos Desenvolvedores . . . 91
5.3.3.3 Definição de Esquema . . . 91
5.3.3.4 Exemplo . . . 93
5.3.4 Metadados para a Descrição dos Algoritmos . . . 93
5.3.4.1 Algoritmos de Extração de Descritores . . . 94
5.3.4.2 Algoritmos de Comparação por Similaridade . . . 94
5.3.4.3 Algoritmos de Normalização . . . 94
5.3.4.4 Algoritmos de Indexação . . . 94
5.3.4.5 Definição de Esquema . . . 95
5.3.5 Exemplos de Representação de Algoritmos . . . 97
5.4 Definição do Esquema para Consultas . . . 100
5.4.1 Definição de Esquema . . . 100
5.4.2 Exemplo . . . 100
5.5 Arquitetura do Repositório de Algoritmos . . . 100
5.6 Implementação do Repositório de Algoritmos . . . 102
5.6.1 Armazenamento dos Algoritmos . . . 102
5.6.2 Geração de Identificafores para os Algoritmos . . . 102
5.6.3 Servidor . . . 104
5.6.4 Interface Visual . . . 104
5.6.4.1 Importar Algoritmos . . . 106
5.6.4.2 Pesquisar sobre os Algoritmos . . . 106
5.7 Sumário . . . 108

6 Conclusões e Trabalhos Futuros **110**

A Glossário de Termos **112**

B Tópicos Relevantes sobre a Lógica Fuzzy **113**

C Especificação da Linguagem de Consulta **117**

D Especificação do XML-Schema para os Algoritmos **120**

Lista de Figuras

2.1 Arquitetura que descreve os Sistemas de Recuperação de Imagens. 34

3.1 Esquema em três camadas que define o gerenciamento de bancos de dados de imagens. . 38
3.2 Gráfico da variável lingüística *modifiers* quando a dimensão do T(u) é 1. 49
3.3 Gráfico da variável lingüística *modifiers* quando a dimensão do T(u) é 2. 50
3.4 Gráfico da variável lingüística *modifiers* para um exemplo de três termos. 52
3.5 Modelo em camadas baseado no padrão ODMG. 59
3.6 Exemplo: Banco de dados de Cadastro de Pessoas e Assinaturas para Bancos. 60
3.7 Diferentes classes de consultas sobre as imagens. 72

4.1 Arquitetura do Sistema AMID. 75
4.2 Serviços de criação dos bancos de dados. 76
4.3 Serviços de AMID para as aplicações. 78
4.4 Processamento das solicitações. 79
4.5 Esquema do Catálogo de Sistema de AMID. 80

5.1 Arquitetura dos Metadados dos Algoritmos. 87
5.2 Arquitetura do Repositório de Algoritmos. 101
5.3 Página Web inicial da Interface do Repositório. 105
5.4 Página Web para importar um documento XML. 105
5.5 Interface para importar uma operação no Repositório de Operações. 106
5.6 Interface para pesquisar uma operação no Repositório de Operações. 107
5.7 Interface para apresentar os resultados de uma pesquisa no Repositório de Operações. . 108

B.1 Definição dos subconjuntos fuzzy sobre a grandeza *altura*, U = [0,2.50]. 115
B.2 Operadores fuzzy. (a) α-cut. (b) Operador de Dilatação. (c) Operador de Concentração. 116

Lista de Tabelas

C.1 Predicados Espaciais 118
C.2 Funções Espaciais 119

Capítulo 1

Introdução

Muitas pesquisas estão sendo encaminhadas no sentido de minimizar os problemas que o uso das imagens introduz em aplicações computacionais. Como resultado da necessidade de projetar aplicações que usem imagens nas mais diversas áreas, os bancos de dados de imagens são cada vez mais comuns. Dentre as áreas de aplicação onde as imagens são utilizadas podem-se citar: os sistemas de informação geográficos [6, 58], os sistemas de informações médicas [24, 48] e os sistemas de bibliotecas digitais históricas para o gerenciamento de museus e galerias de arte [4].

As aplicações precisam desenvolver ferramentas para trabalhar sobre os bancos de dados de imagens. Entende-se como o gerenciamento de imagens o conjunto de funções que manipulam as imagens e suas informações. O gerenciamento de bancos de dados de imagens inclui o processamento de imagens, a inserção, exclusão, atualização e recuperação de imagens e seus dados associados no banco de dados.

A recuperação de imagens é a chave para o desenvolvimento das aplicações. Ela é voltada a recuperar as imagens mais relevantes às consultas dos usuários em um banco de dados de imagens. A relevância é determinada pela natureza da aplicação. Por exemplo, uma aplicação pode estar interessada em desenvolver mecanismos para localizar determinada região em um conjunto de imagens de sensoriamento remoto, ou em executar algoritmos para o detecção de faces, ou aplicar o processamento da linguagem natural para reconhecer objetos em fotografias.

Um Sistema de Gerenciamento de Bancos de Dados de Imagens (SGBDI) deve suportar o conjunto de funcionalidades que faz das imagens mais um tipo de dado nativo no sistema. Assim, as operações sobre todos os tipos de dados devem ser aplicadas de maneira uniforme e integrada. O propósito de tal sistema de gerenciamento é facilitar o desenvolvimento das aplicações de recuperação de imagens. Ele deve ser capaz de satisfazer requisitos das aplicações que trabalham com imagens.

1.1 Definição do Problema

O termo *banco de dados de imagens* é usado indistintamente nas pesquisas que envolvem imagens tanto na área de bancos de dados quanto na de visão computacional (processamento de imagens). No entanto, muitos trabalhos na área de bancos de dados somente utilizam textos, palavras chave e ícones para representar as imagens; na área de processamento de imagens, os sistemas

utilizam arquivos de imagens que são manipulados pelos sistemas operacionais convencionais. Um consenso deve ser atingido com vistas a desenvolver melhores e mais eficientes aplicações. Isto é, as aplicações precisam integrar as vantagens que oferecem os bancos de dados com o suporte necessário para armazenar, processar, manipular, consultar e apresentar as imagens.

A grande variedade de aplicações de imagens, faz com que cada uma delas desenvolva mecanismos de domínio específico para satisfazer suas próprias necessidades na busca e recuperação de imagens. No entanto, os desenvolvedores de aplicações buscam mecanismos de propósito geral para o gerenciamento de imagens sobre bancos de dados.

Resumidamente, do nosso ponto de vista, quatro problemas que atingem o gerenciamento de imagens são os seguintes:

- Falta de integração das vantagens que oferecem os bancos de dados com a manipulação, interpretação e tratamento próprio das imagens.
- As aplicações de imagens possuem diferentes necessidades de recuperação e estratégias de processamento das imagens.
- Falta de modelos de propósito geral, para o gerenciamento de imagens sobre bancos de dados.
- Falta de arquiteturas que integrem mecanismos de propósito geral, orientados ao gerenciamento de imagens sobre bancos de dados.

1.2 Motivação e Objetivos

Muitas das aplicações não convencionais impõem requisitos aos sistemas de bancos de dados que não estavam presentes nas aplicações tradicionais, tais como lidar com maior volume de dados, a necessidade de manipulação de dados em formatos não alfanuméricos (incluindo informação multimídia, dados complexos, semânticas) e o alto desempenho imposto pelos sistemas interativos. As necessidades dessas novas categorias de aplicações constituem a maior motivação para o surgimento de novas abordagens para o gerenciamento de bancos de dados.

As propostas acima descritas para o gerenciamento de bancos de dados não oferecem ferramentas reais para que um desenvolvedor consiga construir aplicações de imagens com economia de esforço. Os sistemas particularmente dedicados ao gerenciamento de imagens costumam oferecer um conjunto de funções limitadas para processar e recuperar as imagens.

Na nossa opinião, cada sistema aborda apenas alguns dos aspectos relevantes em um sistema de gerenciamento de bancos de imagens. Por exemplo, muitos sistemas não fornecem mecanismos para integrar a recuperação das imagens com os diversos tipos de dados tradicionais. O processamento das imagens é desenvolvido para uma classe de aplicações ou restringido a um conjunto de operações.

Nosso trabalho é diferente porque visa oferecer um sistema de propósito geral, baseado em um modelo e uma arquitetura flexíveis, orientadas a facilitar o gerenciamento das imagens, com as quais uma aplicação pode ser desenvolvida de maneira simples.

O nosso modelo de gerenciamento de imagens está composto por um modelo para a recuperação de imagens e uma linguagem de consulta fuzzy que reflete a similaridade entre imagens.

O modelo preocupa-se em suportar um grande número de consultas sobre imagens, integrando informação semântica, não semântica, e abstração de informação em vários níveis. Propomos um mecanismo de retroalimentação das consultas que leva em consideração a subjetividade dos usuários.

O sistema fornece uma maneira bem flexível para as aplicações definir suas características particulares. Oferece um repositório de algoritmos para integrar as operações sobre imagens, de maneira que elas estejam disponíveis para serem consultadas e compartilhadas pelos usuários.

Os objetivos da nossa proposta são:

- Atender às necessidades das aplicações de bancos de dados de imagens.
- Aproveitar as facilidades e a tecnologia oferecidas pelos SGBDs já existentes.
- Definir um modelo e um sistema de propósito geral para as aplicações que manipulam imagens sobre bancos de dados.
- Oferecer ao desenvolvedor de aplicações meios simples para criar suas aplicações sobre bancos de dados de imagens.

No próximo capítulo descrevemos tópicos relacionados com o gerenciamento de bancos de dados de imagens. Apresentamos um estudo aos sistemas de recuperação de imagens, suas características e necessidades. O capítulo 3 define nosso modelo para o gerenciamento de imagens, incluindo definição de consultas, uma abordagem baseada no padrão de dados ODMG, dentre outros tópicos. No capítulo 4 especificamos o sistema AMID, (*Architecture for Management of Image Databases*), sua arquitetura e sua implementação. O capítulo 5 descreve nossa proposta a um repositório de algoritmos envolvidos na recuperação das imagens. O capítulo 6 expõe as conclusões e trabalhos futuros baseados em nossa proposta. Este trabalho possui quatro anexos. O anexo A apresenta um glossário de termos utilizados ao longo deste trabalho. O anexo B apresenta aspectos gerais da teoria de lógica fuzzy que são utilizados na nossa abordagem. No terceiro C, especificamos a linguagem de consulta dos bancos de dados de imagens. O anexo D apresenta a especificação completa do esquema XML para os algoritmos de recuperação de imagens.

Como parte deste trabalho, os autores desenvolveram três artigos, citados a seguir.

A General Purpose Retrieval Architecture for Image Databases. Proceedings of the IEEE International Conference on DEXA'2000, Greenwich, London, UK.

A Metadata Repository for Image Retrieval. Aceito no 6^{th} International Conference on Enterprise Information Systems, ICEIS'2004, Porto, Portugal.

A Query Model with Relevance Feedback for Image Retrieval in Databases. A ser submetido. 2004.

Capítulo 2

Gerenciamento de Bancos de Dados de Imagens

A inclusão das imagens em um banco de dados causa um forte impacto no projeto, nas propriedades e nas funções dos sistemas de gerenciamento de bancos de dados. Utilizando conhecimento semântico mais profundo sobre as imagens, pode se tentar acrescentar o valor das funcionalidades de um sistema de gerenciamento para um sistema de gerenciamento de bancos de dados de imagens.

Diversas soluções estão sendo propostas na integração de mídia a um sistema de gerenciamento de bancos de dados [1, 62]:

Integração virtual. Solução mais simples, mostra a integração não real dos dados multimídia dentro dos serviços do SGBD. Isso significa que o banco de dados simplesmente contém os nomes de arquivos e outras informações textuais que fazem referência aos dados multimídia armazenados e manipulados fora do controle do sistema.

Pseudo ou semi-integração. Quando o sistema provê uma semi-integração dos dados multimídia através de algum tipo de dado embutido como *binary large object* (blob), que permite armazenar dados multimídia mas deixa a responsabilidade de sua interpretação, manipulação e apresentação para a aplicação.

Integração real. O sistema é capaz de manipular dados multimídia de uma forma integrada. O usuário tem todo o suporte que necessita para armazenar, manipular, consultar e apresentar os dados multimídia usando o sistema de banco de dados.

Na primeira solução, o usuário de uma linguagem de programação (associada a um sistema de banco de dados) ou de um modelo de dados, constrói um sistema de informação multimídia escrevendo uma aplicação por cima do sistema de banco de dados. Porém, ele não tem suporte do sistema para o gerenciamento de dados multimídia. Esse é o caso onde o sistema não permite funções definidas pelos usuários.

Muitos dos sistemas atuais fazem uma semi-integração dos dados multimídia usando os objetos binários. Um objeto binário é um campo sem tipo, tipicamente grande, com comprimento variável. Quando um objeto for requisitado, ele é simplesmente entregue à aplicação como blocos

de objetos binários e a aplicação será responsável pelo processamento "inteligente"desses blocos binários.

Uma vantagem desse esquema é a integração mínima dos dados tradicionais com os dados abstratos (em particular, multimídia), proporcionando algum nível de acesso associativo a esses dois tipos de dados. Porém, um problema óbvio associado a essa abordagem é que o SGBD não conhece a estrutura do objeto binário, e por isso não pode aplicar técnicas de otimização para o processamento e recuperação dele, nem mecanismos para consulta baseada no conteúdo.

Na terceira solução, o sistema de banco de dados é capaz de gerenciar os dados multimídia e dar o suporte necessário para o desenvolvedor de aplicações multimídia. O projetista dessas aplicações não estará forçado a lidar com detalhes de implementação de conceitos multimídia básicos e primitivas de modelagem.

Para desenvolver um SGBDI de propósito geral foram analisados vários aspectos descritos a seguir.

2.1 Esforços Correntes para Desenvolvimento de Sistemas Gerenciadores

Nesta seção final são descritos os esforços correntes que poderão afetar o desenvolvimento de sistemas gerenciadores de base de objetos, com ênfase nos aspectos de padronização.

Padrão ODMG

O padrão ODMG (*Object Data Management Group*) [8] é uma extensão do padrão OMG (*Object Management Group*) [32] que define CORBA. ODMG é um consórcio, formado em 1991 por diversos fabricantes de sistemas gerenciadores de base de objetos, que tem por objetivo criar um padrão para bancos de dados orientados a objetos que permita ampliar o grau de portabilidade das aplicações desenvolvidas para os seus diferentes sistemas. Um banco de dados armazena objetos, permitindo que estes sejam compartilhados por múltiplos usuários e aplicações. É importante observar que não há uma padronização da implementação. Assim, os produtos de cada empresa não são idênticos; apenas oferecem uma interface comum para as aplicações.

O padrão ODMG está organizado em dois elementos principais: *framework* e *bindings*. Em *framework* definem-se aspectos comuns a todas as linguagens de programação, tais como a arquitetura, o modelo de objetos, a linguagem de declaração (ODL - *Object Definition Language*) e uma linguagem declarativa de consulta (OQL - *Object Query Language*).

O componente *binding* é o encarregado de estabelecer o mapeamento das construções do padrão (ODL e manipulação de objetos) para aquelas de linguagens de programação. Atualmente, C++ , Smalltalk e Java são contempladas no padrão.

A seguir são listados os principais elementos do *framework* do padrão ODMG.

- Modelo de Objetos (*Object Model*). O modelo de objetos de ODMG resume diferentes conceitos suportados por sistemas gerenciadores de base de objetos. Ele oferece objetos

e literais como primitivas de modelagem. Os literais podem ser atômicos (Long, Float, Octet, ...), coleção (Set, Bag, List e Array) ou estruturados (Date, Time, Timestamp, ...). Os objetos podem ser atômicos ou coleção. O estado de um objeto é definido por suas propriedades, que podem ser atributos ou relacionamentos. O comportamento do objeto é definido pelo conjunto de operações que podem ser executadas sobre ele. Todo objeto e literal possui um tipo. Os elementos de um mesmo tipo têm estrutura e comportamento comuns. A semântica deste modelo define as características, relacionamentos e identidade dos objetos.

- Linguagens de Especificação de Objetos (*Object Specification Languages*). A ODL (*Object Data Language*) e o OIF (*Object Interchange Format*) compõem as linguagens de especificação. Um banco de dados é baseado em um esquema que é definido em uma linguagem de definição de objetos e contém instâncias de tipos definidos por seu esquema.

- Linguagem de Definição de Objetos (ODL: *Object Definition Language*). Similar à linguagem de definição de dados (DDL) tradicional para bancos de dados relacionais. Define as características do objeto incluindo propriedades e operações, mas não inclui a definição de métodos para a implementação de operações. A especificação do modelo de objetos (utilizando a ODL) é independente da implementação (encasulamento). Cada linguagem faz um mapeamento entre as classes abstratas definidas no modelo de objetos e a implementação destas classes.

- Formato de Troca de Objetos (OIF: *Object Interchange Format*). Carrega um SGBDO. É uma linguagem de especificação utilizada para manipular o estado atual de um SGBDO e para caracterizar o estado de todos os objetos persistentes. Para isto, as instâncias dos objetos são especificadas usando os identificadores de objetos, os tipos dos objetos, os valores dos atributos e enlaces a outros objetos.

- Linguagem de Consulta (OQL: *Object Query Language*). Similar à SQL92. As extensões foram feitas encaminhadas à orientação a objetos, tais como: identidade de objetos, objetos complexos, polimorfismo, *late binding*, dentre outras. Cada consulta retorna um tipo de objeto.

No próximo capítulo apresentamos uma instanciação de nosso modelo de gerenciamento de imagens baseada no padrão ODMG. A vantagem de utilizar ODMG é poder aproveitar as vantagens dos sistemas de gerenciamento de objetos e explorar uma linguagem de consulta altamente flexível.

2.2 Trabalhos Relacionados

Um banco de dados é uma coleção de dados relacionados coerentemente representando um universo de informação determinado do mundo real. Eles são projetados, construídos e povoados com dados para um propósito específico. Podem ser de qualquer tamanho e de complexidade variável.

Um Sistema de Gerenciamento de Banco de Dados (SGBD) é um conjunto de programas que permitem criar, manter e acessar um banco de dados. O objetivo primário de um SGBD é prover um ambiente conveniente e eficiente para armazenar e recuperar informações de um banco de dados.

Nesta seção serão descritos alguns sistemas de gerenciamento de bancos de imagens como sistemas de propósito geral. Apresenta-se uma breve descrição dos sistemas relacionais estendidos e extensíveis, assim como dos sistemas orientados a objetos. São descritos sistemas orientados especificamente ao gerenciamento de imagens.

2.2.1 Sistemas de Bancos de Dados Estendidos e Extensíveis

Os modelos relacionais estendidos incorporam mecanismos não previstos na proposta original do modelo relacional. São caracterizados pela adição de componentes específicos de uma aplicação ao núcleo de um SGBD existente.

Os principais mecanismos descritos em diversas propostas incluem procedimentos armazenados, relações aninhadas, objetos binários (BLOBs) e extensões à linguagem de consulta. O poder de expressão da linguagem se incrementou devido ao fato dos usuários poderem especificar nomes de procedimentos para valores de atributos na formulação de uma consulta.

Hoje em dia, existem duas linguagens de consultas desenvolvendo-se rapidamente: a SQL-3 da ANSI X3H2 SQL committee e a OQL da ODMG (*Object Data Management Group*), mas ainda encontram-se em evolução. Alternativamente, os sistemas atuais utilizam uma SQL estendida, basicamente com suporte para tipos, funções e operadores definidos pelos usuários.

Outra abordagem para estender o modelo relacional são os sistemas de bancos de dados extensíveis. A idéia básica é oferecer facilidades para um usuário definir extensões específicas à sua aplicação. Novas características foram criadas para melhorar a representação dos dados, tais como: tipos de dados abstratos e definição de novas funções e operadores. Esses sistemas integram o gerenciamento de dados tradicionais e objetos complexos (dados espaciais, dados binários tais como áudio, vídeo e imagens).

A solução dos objetos binários, apesar de permitir o armazenamento de grandes volumes de dados, não oferece mecanismos para a interpretação desses dados, delegando tal tarefa exclusivamente à aplicação.

Alguns sistemas representativos de esta abordagem são:

PostgreSQL

O PostgreSQL[35] é um sistema (*object-relational*), de domínio público. PostgreSQL suporta estruturas de indexação para imagens como R-tree. PostgreSQL não tem definido ainda métodos de acesso para novos tipos de dados e operadores. No entanto, existe uma maneira de usar um tipo de dado ou operador definido pelo usuário com um índice, mas não existe um mecanismo simples para fazer isto. O mecanismo existente precisa modificar o catálogo do sistema de PostgreSQL. Usando as classes do catálogo é possível especificar para um tipo de dado um determinado índice, operadores e funções de suporte. Com estes mecanismos é possível conseguir índices B-trees, R-trees estendidos.

Informix

Informix [28, 29] pode suportar um índice R-tree para atributos que contêm dados espaciais tais como mapas ou diagramas. O módulo *spatial DataBlade* suporta objetos geométricos de duas dimensões (por exemplo: tipo de dado ponto, linha, polígono, círculo). Alguns dos operadores e funções definidos incluem *contained* e *overlap*. O módulo implementa um índice R-tree para a manipulação dos dados espaciais.

Oracle

Oracle Universal Server [33, 34] possui uma extensão aos dados espaciais chamada de *Spatial Data Option* (SDO). Uma implementação foi baseada na transformada Z para simular a árvore quad-tree. A última versão implementa SDO como uma quad-tree sobre uma árvore B-tree, isto é, utiliza-se uma B-tree para simular uma quad-tree [84].

2.2.2 Sistemas de Bancos de Dados Orientados a Objetos

O paradigma de orientação a objetos surge como alternativa adequada para a representação e manipulação dos dados integrado aos sistemas de bancos de dados.

Ainda não existe uma base teórica única (modelo de dados e linguagem de consulta) para os sistemas de bancos de dados orientados a objetos. Muitas propostas podem ser encontradas na literatura, mas não há um consenso de padronização.

Um sistema de banco de dados orientado a objetos precisa satisfazer dois critérios: ser um sistema de gerenciamento de banco de dados e ser um sistema orientado a objetos.

Para ser um sistema de gerenciamento, o sistema de bancos de dados orientados a objetos deve ser persistente, possuir gerenciamento de armazenamento secundário, concorrência e recuperação às falhas e facilidades de consultas. Para satisfazer o segundo critério, um sistema deve ser consistente com as linguagens de programação orientadas a objetos [93].

A orientação a objetos é uma tendência mundial em termos de programação e desenvolvimento de sistemas. Aplicados à área de bancos de dados, os conceitos de orientação a objetos levam à definição mais racional, mais próxima do mundo real, de modelos e estruturas de dados. Isto é especialmente benéfico no caso das imagens, uma vez que as informações que manipulam, devido às suas características especiais, são difíceis de modelar utilizando as técnicas tradicionais.

Alguns sistemas representativos de esta abordagem são:

O_2

O sistema O_2 é um sistema comercial de banco de dados orientado a objetos. Os objetos podem ser atômicos (inteiros, reais, booleanos, *strings* e *bits*) ou complexos, criados a partir de construtores de tuplas, conjuntos, *bags* e arranjos. A persistência é definida por referência; os objetos que são derivados de uma raiz persistente, direta ou indiretamente, são persistentes. A interação de uma aplicação com o sistema pode ocorrer através de interfaces de programação em C ou C++ , através de uma linguagem declarativa (O_2C) ou através de O_2SQL, uma extensão de SQL para trabalhar com objetos complexos.

ObjectStore

ObjectStore é um sistema comercial de banco de dados orientado a objetos. ObjectStore oferece suporte para identificação de objetos, atributos inversos, herança múltipla, acesso paginado a BLOBs e aplicações interativas para visualização de conteúdo e criação de esquemas. A manipulação de coleções é oferecida através de bibliotecas de tipos de coleções, que inclui conjuntos, *bags*, listas, arranjos e dicionários. Um tipo genérico, coleção, também é oferecido. A interação entre ObjectStore e a aplicação pode ocorrer através de uma linguagem de manipulação de dados ou através de interfaces de programação em C, em C++ e, em Java.

2.2.3 Sistemas Orientados a Imagens

A seguir são apresentados alguns sistemas representativos dos sistemas de manipulação de imagens.

Photobook [63] é uma ferramenta para consultar bancos de dados de imagens baseado nas características do conteúdo. A comparação entre as imagens é baseada em características associadas às imagens. Photobook possui um agente interativo inteligente que escolhe o modelo (cor, textura ou forma) mais adequado segundo os exemplos dos usuários, utilizando refinamento sucessivo. Algumas aplicações exemplos utilizam técnicas tais como o reconhecimento de faces ou a similaridade de texturas. O sistema inclui uma biblioteca de algoritmos de similaridade para comparar as imagens.

SEMCOG [47] (*SEMantic COGnition-based image retrieval*) foi desenvolvido na NEC (C&C Laboratories, USA). O objetivo principal do sistema é a exploração do conteúdo de um banco de dados de imagens através de técnicas de reformulações sucessivas que tornam o resultado mais próximo à expectativa do usuário. SEMCOG suporta a construção de consultas no nível de objetos nas imagens. É um sistema híbrido capaz de integrar em uma consulta predicados textuais, predicados utilizando exemplos visuais de imagens (*media-based*) e predicados baseados em relações espaciais. SEMCOG executa o reconhecimento semi-automático de objetos. Assim, o sistema é baseado em objetos e não oferece consultas sobre as características baseadas na imagem em si.

DISIMA (*DIStributed Image database MAnagement system*) [96] é um projeto que propõe um sistema de bancos de dados de imagens, definindo um modelo, uma arquitetura distribuída e uma linguagem de consulta. O modelo proposto é para o desenvolvimento de aplicações espaciais e geográficas. A linguagem de consulta, MOQL, captura as relações espaciais e temporais inerentes a muitos dos tipos de dados multimídia.

QBIC (*Query by Image Content*)[19, 44] é um sistema de recuperação de imagens baseado nas características do conteúdo das imagens. A extração das características é feita semi-automaticamente. O sistema oferece consultas sobre bancos de dados de imagens baseadas em imagens exemplos, desenhos definidos pelos usuários e padrões de texturas, cores e formas.

2.3 Impacto das Imagens sobre o Projeto de um Sistema de Gerenciamento de Bancos de Dados

Um sistema de gerenciamento de bancos de dados de imagens deve fornecer um ambiente adequado para o gerenciamento das imagens no banco de dados. Conseqüentemente, ele deve suportar um tipo de dado imagem e as funcionalidades de um sistema convencional. As características das imagens impõem novos requisitos.

A seguir são descritos os pontos que um sistema de imagens deve satisfazer tomando como base as funcionalidades suportadas em um sistema convencional de bancos de dados [2, 56]. As propriedades tradicionais dos sistemas de gerenciamento de bancos de dados são as seguintes .

- Independência dos dados. Separar o banco de dados e as funções para o gerenciamento dele dos programas de aplicações.
- Controle de concorrência. Assegurar a consistência do banco de dados através de regras, impondo um esquema de ordem de execução sobre as transações concorrentes.
- Persistência. Permitir aos dados sobreviver a diferentes transações e diferentes chamadas de programas.
- Privacidade. Restringir o acesso e a modificação dos dados armazenados no banco de dados.
- Controle de integridade. Assegurar estados consistentes do banco de dados através de restrições impostas sobre as transações.
- Recuperação a falhas. Evitar que os resultados de transações que falham deixem o banco de dados em um estado inconsistente.
- Controle de versão. Permitir a organização e manipulação de diferentes versões de dados persistentes, o que pode ser requerido por aplicações que desejam acessar estados anteriores de um dado.

A inclusão das imagens nos bancos de dados deve ser transparente para a aplicação e para o desenvolvedor da aplicação. O armazenamento de grandes volumes de dados que acarreta o trabalho com imagens é resolvido em muitos sistemas de bancos de dados com a inclusão dos objetos binários, mas isto não é suficiente para a manipulação das imagens. A representação do conteúdo de uma imagem em um sistema de bancos de dados impõe os seguintes requisitos [49].

Algoritmos e Estruturas de Representação

A extração das características do conteúdo das imagens exige algoritmos e técnicas eficientes de processamento de imagens. Os resultados destes algoritmos devem ser representados em estruturas específicas que permitam a sua manipulação eficiente.

Definição de Critérios de Similaridade

Diferentes medidas de similaridade são usadas para diferentes descritores. Por exemplo, a medida euclidiana pode ser utilizada para comparar representações baseadas em vetores, e a intersecção de histogramas pode ser usada para comparar histogramas de cores.

Expressão das Imagens nas Linguagens de Consultas

As linguagens de consultas devem fornecer recursos para representar as imagens que nem sempre podem ser expressas de forma textual. As comparações por similaridade devem ser manipuladas como parte da consulta. Existem várias abordagens na forma de expressar uma consulta.

- *Query by pictorial example.* A especificação da consulta é feita através de um modelo de imagem representando aquela que deseja recuperar.
- *Consultas textuais.* A linguagem oferece possibilidades de descrever uma imagem utilizando termos ou conceitos. Por exemplo, definindo recursos textuais tais como *SomeOrange*, *sunset* [57], dentre outros.
- *Consultas visuais (*sketch*).* Ferramentas gráficas são fornecidas ao usuário para representar as características que deseja recuperar. Por exemplo, na interface de consulta de QBIC as formas e posição dos objetos nas imagens podem ser representadas graficamente.

Suporte para a Indexação das Imagens

A execução eficiente de consultas sobre grandes bancos de dados de imagens baseadas em conteúdo depende da possibilidade de criar estruturas de indexação para as características das imagens extraídas do seu conteúdo. Dada a natureza inexata da recuperação, estas estruturas devem considerar o acesso rápido a imagens a partir do critério de similaridade adotado. Estruturas de indexação foram definidas em [49, 44, 27, 64].

Integração da Recuperação de Dados Tradicionais e Imagens

As linguagens de consultas devem suportar manipulação de imagens junto aos tipos de dados tradicionais. A integração de dados tradicionais e as representações das características do conteúdo das imagens nas consultas ajuda expressar consultas com maior valor semântico.

2.4 Tópicos Relacionados com o Gerenciamento de Imagens

Nesta seção é feito um breve apanhado dos tópicos que consideramos importantes a serem estudados no projeto de um SGBDI.

2.4.1 Recuperação de Imagens

A técnica mais comum para integrar imagens em um banco de dados tradicional é armazenar estes dados junto com textos descritivos ou palavras chave. Esta abordagem é exclusivamente

baseada em textos. As palavras chave são definidas pelos operadores, o processo de entrada dos descritores consome bastante tempo e a busca está sujeita a valores subjetivos e incompletos. A recuperação pode falhar se o usuário usar palavras diferentes das palavras chave armazenada. Adicionalmente, algumas propriedades visuais como textura e formas são difíceis ou quase impossíveis de descrever com palavras.

A única solução para estes problemas é o desenvolvimento de métodos automatizados capazes de analisar o conteúdo das imagens. As primeiras abordagens foram desenvolvidas fundamentalmente por pesquisadores da área de bancos de dados enquanto a última por especialistas da área de processamento de imagens. Nessa última abordagem, a recuperação de imagens baseada no seu conteúdo é feita através da extração de descritores do conteúdo utilizando técnicas de processamento de imagens.

2.4.1.1 Mecanismo de Retroalimentação

Satisfazer as necessidades dos usuários é o principal alvo na recuperação de informação. No processo de recuperação de imagens, uma consulta não é simples de expressar por duas razões:

- a subjetividade humana: diferentes pessoas ou a mesma pessoa em diferentes circunstâncias podem perceber o mesmo conteúdo visual de maneira diferente.
- a ligação entre os conceitos de alto nível de expressão e os descritores de baixo nível: o mapeamento entre conceitos e descritores do conteúdo nem sempre é uma tarefa simples. Por exemplo, de que maneira podemos ligar a detecção de um objeto utilizando processamento de cor e forma com seu conceito lógico ou nome, sem intervenção manual?

O mecanismo de retroalimentação ou refinamento (*relevance feedback*) é utilizado para envolver os usuários no processo de recuperação de resultados nas consultas.

O refinamento é o processo de automaticamente adaptar uma consulta usando a informação dada pelo usuário sobre a relevância de resultados anteriores, e assim a consulta adaptada é uma aproximação melhor à realidade do usuário. Neste processo, é interessante incluir um sistema de regras baseado em conhecimento. O conhecimento é necessário para interpretar as consultas, assim como para capturar o conteúdo semântico das imagens.

Dentre dos métodos para recuperar imagens baseado em conhecimento existem dois importantes [92]. O primeiro extrai manualmente as propriedades semânticas. Esse método é usado quando é difícil reconhecer o conteúdo da imagem. Assim como nos bancos de dados relacionais, com este método é difícil de manter a consistência nas descrições e não é prático para bancos de dados muito grandes.

O segundo método fornece uma base de conhecimento de onde se derivam os descritores semânticos associados às imagens. O conhecimento é utilizado para a extração dos descritores e a análise das consultas. A base de dados é difícil de manter semanticamente consistente. Por isto, deve existir uma dependência entre a base de conhecimento e o banco de dados. Uma solução é a definição de um sistema de regras.

2.4.1.2 Avaliação de Sistemas de Recuperação de Imagens

A principal medida para avaliar um sistema de recuperação de informação é a performance ou desempenho do sistema em relação aos resultados obtidos para as consultas dos usuários e na relação custo/tempo de execução.

Na hora de testar o grau de certeza de um sistema que usa a recuperação de imagens baseada em similaridade do conteúdo, a avaliação do sistema é feita através de uma análise manual do conjunto de todas as imagens relevantes a uma dada consulta no banco de dados (*conjunto relevante*), e comparado com o conjunto resultado que o algoritmo de processamento do sistema automaticamente recupera (*conjunto resultado*).

Para testar o sistema, escolhe-se um conjunto de consultas e aplicam-se os seguintes parâmetros [25]:

1. *recall.* Do conjunto relevante de uma consulta, o valor do *recall* é a porcentagem de imagens nesse conjunto que foi encontrado no conjunto resultado;

2. *precision.* Do conjunto resultado, o valor de *precision* é a porcentagem de imagens nesse conjunto que pertence ao conjunto relevante da consulta.

2.4.2 Modelagem das Imagens

Um modelo de dados captura os objetos, atributos, relacionamentos e as semânticas do conteúdo dos dados. Em um modelo de dados para as imagens podemos considerar armazenar as suas grandezas relevantes (por exemplo: as cores e os objetos), mas as características relevantes dependem dos requisitos de cada aplicação.

2.4.2.1 Características das Imagens

As características das imagens podem ser agrupadas como visuais e não visuais [5].

Características Não Visuais

A descrição das imagens é feita através de atributos alfanuméricos estruturados que servem como base às consultas.

As características não visuais das imagens estão compostas por atributos tradicionais que descrevem as imagens com palavras chave ou textos livres e atributos tradicionais que não descrevem o conteúdo de uma imagem.

No primeiro caso, quatro técnicas foram identificadas para representar essas características nos sistemas de recuperação de imagens [68]: subtítulos (usualmente, textos livres); palavras chave extraídas de um vocabulário controlado; descrições simbólicas envolvendo objetos, atributos e seus relacionamentos; e textos hipermídia que vinculam uma imagem com informações sobre ela.

No último caso encontram-se, por exemplo: atributos referentes à fonte, origem e a história da imagem (autor ou criador, a data de publicação, localização ou título); atributos de digitalização como resolução, tipo e tamanho do arquivo; dentre outros.

Características Visuais

As características visuais são aquelas extraídas do conteúdo das imagens utilizando técnicas de processamento de imagens. Incluem-se dentre elas: cor, textura, forma, objetos e relacionamentos espaciais.

As características visuais podem ser modeladas como uma hierarquia de abstrações. No primeiro nível encontram-se os *pixels* com informação sobre as cores ou o brilho. Por exemplo, existem diferentes esquemas de representação das cores incluindo: RGB (*Red-Green-Blue*), o sistema de cromatismo e luminosidade da Comisão Internacional sobre Iluminação (CIE) e Matiz-Saturação-Intensidade (HSI). Outras características como linhas, curvas e regiões de cores podem ser obtidas usando processamento.

Em um nível superior, essas características podem ser combinadas e interpretadas como objetos e seus atributos. A detecção de objetos envolve a verificação da presença de um objeto em uma imagem e a localização precisa dele para o reconhecimento.

No nível mais alto, encontram-se os conceitos lógicos com valor semântico formulados pelos usuários, envolvendo vários objetos e seus relacionamentos espaciais.

Um *descritor* das imagens é uma representação das características visuais. A representação das imagens pode ser física ou lógica. Um descritor físico é a representação no nível de *pixels*. Por exemplo, as representações matricial e vetorial são representações físicas. Na representação matricial, uma imagem é considerada como uma matriz de *pixels*, cada *pixel* é representado pela sua cor. Na representação vetorial, uma imagem é representada como um conjunto de equações matemáticas definindo objetos $n-$dimensionais com conhecimento explícito de sua estrutura e localizações.

Um descritor no nível lógico é a representação abstrata da representação física. O processamento desta representação é mais eficiente para alguns tipos de consultas. Alguns exemplos de descritores lógicos são o *Minimum Bounding Rectangle* (MBR) que encerra um dado objeto no mínimo retângulo que o contém completamente (útil em consultas com relacionamentos espaciais onde deseja-se averiguar se dois objetos se sobrepõem ou não); e *2D-Strings* que projeta no plano bi-dimensional os objetos de uma determinada imagem (útil em consultas com relacionamentos espaciais entre objetos) [36].

As características visuais possuem múltiplos descritores que as descrevem em diferentes perspectivas. Por exemplo, a cor pode ser representada com o histograma de cores ou suas aproximações *color moments* e *color set* [69]. Outro descritor pode ser o *color layout* (esquema de cores).

O processamento de imagens é computacionalmente caro, difícil e normalmente de domínio específico. A maioria das aplicações fazem um pré-processamento das características das imagens [44, 49, 57, 63, 76, 81], para diminuir o tempo do processamento das consultas. Os descritores das imagens são armazenados no banco de dados, mas eles precisam ser analisados por algum algoritmo que interprete o seu valor para efetivar as comparações. Muitas vezes são construídos índices para os descritores com o objetivo de aumentar a performance do processamento das consultas.

Toda informação extraída do conteúdo das imagens faz parte dos chamados metadados. Os metadados são informações adicionais associadas ao dado primário, neste caso a imagens. A seguir são descritas diferentes propostas para padronizar os metadados das imagens.

2.4.2.2 Padronização dos Metadados das Imagens

O esforço mais recente de padronizar o conteúdo dos dados multimídia é o chamado *Multimedia Content Description Interface* (MPEG-7) [30, 31]. O objetivo de MPEG-7 é descrever o conteúdo multimídia e representar a informação sobre o conteúdo em diferentes níveis de abstração. Os elementos da terminologia utilizada no MPEG-7 são:

- o dado multimídia (*data*);
- as características (*Feature*): característica relevante do dado possui um significado;
- os descritores (*Descriptor*, D): para definir a sintaxe e semânticas de cada característica multimídia (F);
- valor do descritor (*Descriptor Value, DV*): instanciação de um descritor;
- os esquemas de descrição(*Description Schemes*, DS): para especificar a estrutura e semânticas dos relacionamentos entre seus componentes, que podem ser D ou DS;
- descrição (*description*): consiste em uma estrutura (DS) e um conjunto de DV que descrevem o dado;
- descrição codificada (*coded description*): descrição codificada para satisfazer requisitos como: eficiência de compressão, acesso aleatório, etc.;
- a linguagem de definição dos descritores (*Description Definition Language*, DDL): utilizada para definir a sintaxe de MPEG-7 e a estrutura dos descritores, e criar novos esquemas e descritores (DS e D) e estender e modificar os existentes.

Diversos trabalhos como DIG35 [14], Dublin Core [20] e NISO [12] definem metadados sobre as imagens. DIG35 define um padrão de metadados para imagens digitais, para melhorar a interoperabilidade semântica entre dispositivos, serviços e software e para ser independente dos diferentes formatos das imagens. Este padrão está orientado a catálogos de imagens fotográficas. O Dublin Core é outro padrão que define metadados para a comunidade dos museus, fornecendo um vocabulário comum para a área. NISO é um comitê que desenvolve um dicionário de dados para instituições culturais e outras organizações interessadas em preservar coleções de imagens digitais. O propósito desses padrões é facilitar o desenvolvimento de aplicações para validar e migrar grandes repositórios de imagens digitais.

O esquema mais utilizado para representar os metadados é o XML (*eXtensible Markup Language*) [88]. O XML é um padrão aberto de formato para a descrição de dados (é uma meta-linguagem), que baseia seu formato em um arquivo texto, com estruturas organizadas hierarquicamente. Segundo a descrição do W3C, este formato permite troca de dados independente de plataforma. A linguagem XML permite aos desenvolvedores a criação de suas próprias estruturas de definição.

Um documento XML é formado por texto contendo caracteres de dado e informações de marcação explicitamente separadas. A entidade fundamental de um documento XML é o elemento. Os atributos constituem as características de um elemento. Outra informação fundamental é a definição de uma estrutura XML ou esquema. Em um esquema descrevem-se as

marcas, atributos e elementos que se podem utilizar, bem como as relações que existem entre eles. Para tanto, são utilizadas linguagens que definem regras para os documentos, são elas:

- DTD (*Data Type Definition*). É parte integrante da recomendação 1.0 do XML e não utiliza sintaxe XML o que acaba restringindo os elementos e atributos que podem ser utilizados na caracterização do documento.
- *XML Schema Definition.* O XML Schema é uma alternativa criada pela W3C, que incorpora a sintaxe XML e utiliza documentos XML para definir a estrutura. Com este esquema, é possível ter um melhor controle sobre a estrutura de dados. O XML Schema é um padrão mais abrangente que uma DTD, permitindo expressar tipos de dados, herança, tipos abstratos, unicidade e chave, dentre outros. Sua finalidade principal é a de permitir às aplicações descreverem documentos XML para que outras aplicações que utilizam esses documentos possam satisfazer/conferir a estrutura do documento.

2.4.3 Critério de Similaridade e Normalização

Os critérios baseados em igualdade e ordem não são adequados aos chamados tipos de dados complexos, ou não convencionais, como as imagens, que são estruturalmente mais sofisticados.

Para estes tipos não há sentido em realizar consultas como "obter as imagens cuja textura seja *igual* a esta". Dificilmente, as texturas de duas imagens serão exatamente iguais. Portanto, o critério mais adequado para casos assim é o de semelhança, sendo que a avaliação da similaridade existente entre duas imagens leva em consideração o significado do conteúdo que apresentam. Assim, a consulta acima faria mais sentido da seguinte forma: "obter as imagens cuja textura seja *similar* a esta".

Uma técnica comum para tratar a similaridade entre imagens é a extração de descritores das imagens. Diferentes medidas de similaridade são usadas para diferentes descritores. Por exemplo, a medida euclidiana é utilizada para comparar representações baseadas em vetores, e a intersecção de histogramas é usada para comparar histogramas de cores.

A extração de descritores é uma operação complexa que pode gerar vetores de descritores em diferentes faixas de valores, dependendo do algoritmo de extração e das representações das imagens. A *normalização de descritores* iguala aproximadamente as faixas dos elementos dos vetores para causar o mesmo efeito quando executa-se a similaridade. Em [3] são explicados cinco métodos de normalização para normalizar cada componente do descritor na faixa $[0, 1]$.

A avaliação da similaridade entre duas imagens é realizada através de funções que medem a "distância"ou dissimilaridade entre elas. Estas funções correspondem a algoritmos computacionais que recebem dois objetos de um mesmo domínio (mesmo tipo) e retornam o valor da distância entre eles. Formalmente, uma função de distância atende às seguintes propriedades (considerando x, y, e z objetos a serem comparados): simetria: $d(x, y) = d(y, x)$; e não negatividade: $0 \leq d(x, y) < \infty$. Além destas, se a desigualdade triangular: $d(x, y) \leq d(y, z) + d(z, y)$ também for respeitada, a função é considerada uma função de distância métrica, ou simplesmente métrica.

As métricas da família L_p trabalham com imagens compostas por n-atributos de valores numéricos. Considerando cada atributo como um eixo de ordenadas cartesianas, a imagem pode ser vista como um ponto no espaço de n-dimensões, também chamado de espaço vetorial. Assim,

métricas como a euclidiana, L_2, e Manhattam, L_1, são usadas para calcular a distância entre as imagens.

O método de decisão usado em consultas por similaridade leva em consideração uma imagem de referência e a distância entre ela e as armazenadas no banco de dados. A forma mais comuns de consulta por similaridade é a consulta dos k-Vizinhos mais Próximos (*k-Nearest Neighbor Query*), consulta que visa recuperar as k imagens mais próximas à imagem de referência. Nas medidas de similaridade geométricas como o do vizinho mais próximo, cada imagem no banco de dados é representada por seu vetor de características no espaço multi-dimensional de características. Dado um vetor x como entrada da consulta, o objetivo é achar quais os vizinhos mais próximos de x para uma medida de distância p. Os k vizinhos mais próximos são recuperados como os mais relevantes à consulta.

A escolha da métrica deve considerar o tipo de imagem, as métricas associadas e o modelo que melhor se ajusta à aplicação. Esta tarefa deve ser auxiliada por um especialista no domínio, que avalia a qualidade dos resultados apresentados pela métrica.

2.4.4 Estruturas de Indexação e Métodos de Acesso

Os índices são estruturas de acesso usadas para aumentar a velocidade de recuperação em resposta às condições de busca. Um método de acesso é um conjunto de funções definidas para construir, ter acesso e manipular uma estrutura de indexação ou índice. Estas funções encasulam as operações tais como inserção, exclusão e busca de dados nos índices.

Para desenvolver um esquema de indexação para imagens, temos que identificar os métodos apropriados para indexar os metadados. As abordagens para indexar uma imagem podem ser classificadas como segue.

2.4.4.1 Indexação baseada em Conceito (*Concept-based Indexing*)

Existem várias propostas para classificar os atributos não visuais das imagens. No esquema de classificação de Shatford [68], os atributos podem ser categorizados como *Genéricos de*, *Específicos de* e *A respeito de*, com enfoque nas questões: Quem?, Que?, Onde? e Quando?.

Layne [68] faz outra classificação dos atributos para indexar as imagens. Os atributos são classificados em três classes: atributos biográficos (relativos à origem da imagem); atributos físicos (relativos ao formato da imagem); e atributos de relacionamento (relativo ao relacionamento entre imagens).

Em [42] é feita uma classificação dos atributos não visuais das imagens em atributos físicos (localização, armazenamento: tamanho, compressão), biográficos (autor, data, título) e informação associada (subtítulos, gravações relacionados à imagem).

A indexação por conceito também pode ser feita usando vocabulários controlados, usualmente tesauros com uma fonte de termos para diferentes tópicos. Existem tesauros especialmente projetados para imagens, tais como o Tesauro de Arte e Arquitetura (AAT), que consiste de 120.000 termos para a descrição de artes, arquitetura, história da arte e outros temas culturais, e possui trinta e três categorias hierárquicas para descrições das imagens [26]. O Tesauro de Materiais Gráficos da Biblioteca do Congresso (LCTGM I, II) é uma fonte de termos para tópicos encontrados em materiais gráficos tais como fotografias, desenhos, impressões, pôsteres

e desenhos arquitetônicos, e exclui nomes próprios de pessoas, organizações e eventos. Na introdução oferece alguns princípios para indexar imagens [68].

Uma outra maneira de indexação por conceito é utilizar textos livres para descrever as imagens. Logo, é necessário aplicar o processamento da linguagem natural. Os subtítulos são a fonte mais comum de descrever livremente as imagens. Algumas coleções comerciais de imagens em CD-ROM são acompanhadas com frases descritivas sobre as imagens [68].

2.4.4.2 Indexação baseada em Conteúdo (*Content-based Indexing*)

A execução eficiente de consultas sobre grandes bancos de dados de imagens baseadas no conteúdo depende da possibilidade de criar estruturas de indexação para os descritores. Dada a natureza inexata desse tipo de consultas, estas estruturas devem considerar o acesso rápido a imagens a partir do critério de similaridade adotado.

Uma abordagem comum para indexar as imagens é obter valores numéricos para n características e representar as imagens ou objetos como um ponto no espaço de n dimensões. Os métodos de acesso multi-dimensionais como os K-D-B-trees, quad-trees e R-trees[37] (e suas variações, ex. R*-tree, SS-tree, SR-tree) são utilizados para indexar e recuperar as imagens relevantes. No entanto, essas estruturas de indexação são ineficientes quando o número de dimensões é grande.

Como conseqüência, uma abordagem para solucionar esse problema é reduzir as altas dimensões e aplicar as técnicas de indexação multidimensional apropriadas. Existem duas abordagens principais para reduzir as dimensões dos vetores de características: *Karhunen-Löeve Transform* (KLT) e agrupamento (*clustering*) [16, 69].

Em [90] é apresentado um estudo do desempenho de estruturas de indexação multidimensional. Dentre elas pode-se mencionar VamSplit R-tree, Vam K-D tree, SS-tree e R*-tree. O artigo concluiu que o VAMSplit R-tree tem a melhor performance em aplicações que acessam a disco. Todas as estruturas obtiveram boa performance em acesso a memória principal (primária).

M-tree [94, 11] é outra estrutura de indexação que pode ser utilizada para a resolução eficiente de consultas por similaridade sobre objetos complexos (como imagens). A comparação pode ser feita usando qualquer medida de similaridade ou função de distância. O espaço métrico é genérico, de maneira que a proximidade é apenas definida por uma função de distância que satisfaça as propriedades de simetria, não negatividade e desigualdade triangular.

2.4.4.3 Combinação dessas duas abordagens

As duas abordagens anteriores são desenvolvidas paralelamente, com pouca interação, na literatura. Na nossa opinião, a construção de índices que as integram é a maneira mais completa de abordar a indexação de imagens, pois ela combina as propriedades semânticas, incluindo conceitos abstratos presentes nas imagens com a extração de características em baixo nível.

Uma descrição interessante do conteúdo das imagens pode ser feita utilizando *consultas por conceito descritivo* [57, 61]. O conceito descritivo pode ser expresso como combinação de características visuais extraídas das imagens. Por exemplo, podemos definir o conceito descritivo *pôr-de-sol*. O usuário pode definir este conceito incluindo nas restrições da sua consulta a palavra *ocaso* e/ou imagens com a cor laranja predominante na parte superior da imagem. Esses conceitos podem ser guardados no banco de dados para serem utilizados explicitamente

por outros usuários ou o sistema pode utilizar técnicas de aprendizagem ou retroalimentação para adaptar o conceito seguindo o critério do usuário.

2.4.5 Linguagem e Interface de Consulta

A estrutura complexa das imagens e sua semântica é de difícil representação pelas linguagens de consultas tradicionais. A propriedade visual inerente às imagens conduz a criar meios para que os usuários possam consultar o banco de dados utilizando representações visuais. Isto facilita a formulação das consultas sobre imagens.

A maioria dos sistemas suportam uma ou várias das seguintes classes de consultas.

- *Consultas seqüenciais.* Os usuários percorrem seqüencialmente todas as imagens do banco de dados.
- *Consultas por conceito.* Os usuários descrevem diretamente com palavras chave ou na linguagem natural as características semânticas do conteúdo das imagens buscadas. Essas características podem ser objetos e conceitos abstratos ou de alto conteúdo semântico.
- *Consultas por descritores.* Os usuários descrevem diretamente as características visuais do conteúdo das imagens buscadas (por exemplo: cores, texturas e formas).
- *Consultas por atributos tradicionais.* Os usuários especificam os atributos não visuais das imagens buscadas, tais como atributos físicos (localização, armazenamento: tamanho, compressão), biográficos (autor, data, título) ou outros associados (subtítulos, gravações).
- *Consultas por sketch/imagem.* O usuário oferece uma imagem como amostra ou desenha seus requisitos. A similaridade pode ser determinada baseada na característica escolhida pelo usuário.
- *Consultas por categorias.* A interface de consulta é organizada por categorias. O usuário navega pelas diferentes categorias ou escolhe uma a partir da qual expressa a consulta utilizando uma das classes anteriores.

Tipicamente, uma interface visual se divide em duas partes: a formulação das consultas e a apresentação dos resultados. Na formulação de consultas, uma aplicação precisa oferecer ferramentas para expressar as consultas de uma maneira mais próxima ao usuário final. Elas devem fornecer mecanismos para representar as propriedades semânticas que nem sempre podem ser expressas de forma textual. Usualmente, os elementos de uma interface visual são mapeados na linguagem de consulta. As consultas são processadas e a interface mostra as imagens do resultado. As interfaces de consultas correntes incluem mecanismos de retroalimentação, envolvendo os usuários no processo de recuperação.

2.4.5.1 Primitivas de Consultas sobre Imagens

As consultas sobre imagens podem ser feitas como uma combinação das seguintes primitivas descritas em [5].

- Consulta simples a uma característica visual. Por exemplo:

  `Recupera as imagens com 25% de vermelho e 40% de azul.`

- Consulta com uma combinação de características. Por exemplo:

  `Recupera as imagens com 25% de vermelho e 40% de uma textura de árvore.`

- Consulta de característica com localização. Por exemplo:

  `Recupera as imagens com céu azul na metade superior e verde na metade inferior.`

- Consulta por exemplo (*Query by Example* - QBE). O sistema gera um conjunto de imagens de maneira aleatória. O usuário escolhe uma delas e recupera imagens similares. A similaridade pode ser determinada baseada na característica escolhida pelo usuário. Por exemplo:

  `Recupera as imagens que contêm texturas similares a esta.`

  O usuário também pode usar uma região de uma imagem exemplo, utilizando ferramentas gráficas para descrever as propriedades do conteúdo que deseja recuperar.

- Objeto vs. Imagem. Ao invés de descrever a imagem, o usuário descreve características dos objetos na imagem. Por exemplo:

  `Recupera as imagens que contêm um carro azul localizado no centro.`

- Consultas utilizando os atributos definidos pelo usuário. Por exemplo:

  `Recupera as imagens onde a localização seja São Paulo e a data junho de 1972.`

- Consultas utilizando objetos, seus atributos e relacionamentos entre objetos. Por exemplo:

  `Recupera as imagens onde há um homem e uma criança à sua direita.`

- Consultas por conceito. Os conceitos são baseados nos descritores. Por exemplo: um conceito *céu* pode ser definido como aquelas imagens com a cor azul predominante na metade superior.

2.5 Sistemas de Recuperação de Imagens

Para projetar um sistema de gerenciamento de bancos de dados de imagens que realmente ajude no desenvolvimento de sistemas de recuperação de imagens é necessário fazer um estudo das características, requisitos, e abordagens de um estoque variado de aplicações de domínio específico, tentando enxergar os pontos que elas têm em comum.

A seguir fazemos uma revisão de sistemas e algoritmos de domínios específicos que têm sido estudados ao longo de nosso projeto. Destacamos as abordagens que esses sistemas oferecem para: modelo de dados, linguagem e interface de consulta, critérios de similaridade, indexação e mecanismos de retroalimentação das consultas.

DISIMA

Em [96] é descrito o sistema DISIMA. A imagem é definida em duas camadas. A primeira camada define a imagem como a unidade básica do modelo. Uma imagem é definida com quatro propriedades: o identificador; o conjunto das representações em formatos tais como `gif, jpg`; o conjunto de dados alfanuméricos associados e o conteúdo definido como o conjunto dos objetos relevantes. A segunda camada captura o conteúdo das imagens. Uma imagem é vista do ponto de vista físico e lógico. Do ponto de vista físico, uma imagem é o conjunto dos objetos relevantes com seus relacionamentos espaciais. Um objeto, do ponto de vista lógico, é usado para dar semântica a um objeto físico. O modelo fornece ferramentas para manipular hierarquias de objetos (físicos e lógicos) e hierarquias definidas pelos usuários [60]. DISIMA desenvolveu uma linguagem de consulta baseada na linguagem OQL do padrão ODMG, MOQL (*Multimedia OQL*) [45, 46]. MOQL possui uma interface visual, *VisualMOQL* [59]. A linguagem pretende ser geral no tratamento dos diferentes dados multimídia e para diferentes áreas de aplicações. Captura, principalmente, as relações espaciais e temporais inerentes a muitos dos tipos de dados multimídia e inclui funções para a apresentação das consultas. VisualMOQL traduz uma consulta visual na linguagem de consulta interna (MOQL). Implementa as mesmas consultas que MOQL e combina QBE (*Query By Example*), consultas por semântica das características visuais e consultas por atributos textuais. O usuário especifica a consulta selecionando as classes de imagens e os objetos envolvidos. Vários níveis de retroalimentação são oferecidos dependendo do tipo de consulta e do nível de precisão da consulta que o usuário desejar. Entretanto, MOQL se ocupa, principalmente, dos relacionamentos espaciais e temporais nas imagens. Em [55] é proposta uma estrutura de indexação, chamada de 2D-h tree, que consiste de duas estruturas. A primeira (2-D-S-tree) organiza as representações espaciais entre objetos nas imagens, sendo uma extensão da 2-D string [36]. A segunda (h-structure) organiza o relacionamento hierárquico entre objetos. A indexação em DISIMA é feita utilizando a estrutura 2D-h tree.

SEMCOG

O sistema SEMCOG possui uma linguagem de consulta denominada CSQL (*Cognition-SQL*) e uma interface de consulta visual chamada de IFQ (*Interface Feature Query*) [47, 89]. IFQ é baseada em objetos e suas relações espaciais. Tenta combinar a linguagem natural do usuário com a linguagem precisa do computador. O usuário, de maneira visual, constrói a consulta que é traduzida para a linguagem CSQL. Em uma consulta é possível definir um peso aos

objetos para definir a importância dos predicados, e é possível pesquisar atributos associados às imagens. A retroalimentação é feita através de reformulação da consulta. CSQL define três tipos de predicados: *semantic-based* (is, is-a, s-like = semantic-like), *cognition-based* (i-like = image-like, contains) e, *scened-based* (to-the-right-of, in, dentre outros). No entanto, as consultas são principalmente baseadas nos objetos das imagens, e não consideram a imagem tratada como uma única entidade.

CHITRA

O sistema CHITRA [54, 66] definiu o FOQL (*Fuzzy OQL*) [53] como uma extensão ao OQL do padrão ODMG. Permite definir conceitos abstratos (como *sunset*) em termos das características das imagens (como textura e cor) e a retroalimentação das consultas. A extensão *fuzzy* da linguagem permite obter os resultados com uma determinada relevância para a consulta (por exemplo: select distinct [0.8] I.imagename), assim como permite especificar o número de imagens relevantes a serem recuperadas (select distinct [1:10] I.imagename). Possui um mecanismo explícito para preferências nas restrições colocando pesos nos predicados (por exemplo: where [0.8] I.colormatch(J) & [0.2] I.texturematch(J) - a soma dos pesos nos predicados = 1). FODL (*Fuzzy ODL*) é a linguagem de definição de dados do sistema, similar à tradicional mas utiliza o modelo de objetos *fuzzy* definido no sistema com o suporte a alguns tipos de dados *fuzzy*. A definição de tipos de dados *fuzzy* captura melhor as comparações entre as imagens baseadas em similaridade, mas o sistema é endereçado à solução do processamento e otimização de consulta e não fornece ferramentas para a modelagem, representação e apresentação visual das imagens.

QBIC

No sistema QBIC (*Query by Image Content*)[19, 44], a extração das características é feita semi-automaticamente. O sistema oferece consultas sobre bancos de dados de imagens baseadas em imagens exemplos, desenhos definidos pelos usuários e padrões de texturas, cores e formas. No modelo de dados, existem dois tipos de dados principais: as imagens (ou cenas) e os objetos das imagens. Uma cena tem zero ou mais objetos. Os objetos são identificados manualmente ou semi-automaticamente. As características das imagens para capturar os seus atributos são computadas durante a etapa de entrada de dados no banco. As consultas são baseadas nas cenas (*scene-based*) ou nos objetos (*object-based*) e são processadas por métodos que oferecem aproximações baseados em similaridade. A interface com o usuário é gráfica e permite aos usuários especificar visualmente consultas por exemplos, refinamento dos resultados e navegação no banco de dados. No entanto, não suporta consultas que integrem as características do conteúdo com os diversos tipos de dados tradicionais.

MARS (*Multimedia Authoring Retrieval Service*)

O sistema MARS (*Multimedia Authoring Retrieval Service*) [40] desenvolveu outra extensão à OQL, POQL^{MM} [39]. POQL^{MM} é uma linguagem de propósito geral orientada a documentos multimídia estruturados. As principais características desta linguagem são: a incorporação de expressões (*regular paths*) para manipular documentos estruturados; a integração de vários operadores de extração de características para os diferentes tipos de mídia; e o operador combine

que permite a combinação de diferentes critérios de similaridade. Desenvolveram uma interface visual que traduz a consulta a POQL^{MM}. A interface apresenta três seções: seleção por atributos das imagens; busca baseada em textos associados às imagens; e busca baseada no conteúdo das imagens, onde pode ser representado um esquema (*sketch*) ou um exemplo (QBE), considerando relacionamentos espaciais, histograma de cores, textura e forma. As expressões em POQL^{MM} acabam sendo complexas e a interface visual possui elementos que podem não ser compreensivel em nível de usuário final.

lambda-DB

Um SGBDO baseado no padrão ODMG (embora não específico para imagens) é o sistema lambda-DB [18]. Sua linguagem de consulta é a λ-OQL e desenvolveu uma interface de consulta visual chamada de VOODOO (*Visual Object-Oriented Database language for ODMG OQL*) [17]. A pesquisa está centrada nos mecanismos de otimização das consultas OQL. As consultas em VOODOO são mapeadas em OQL, e formam uma árvore que reflete a estrutura do esquema de um banco de dados, onde cada referência a classes e tipos no esquema é expandida. Uma consulta é formulada mapeando os nós da árvore, correspondentes à solicitação, em valores.

MARS (*Multimedia Analysis and Retrieval System*)

Em [70, 71, 87] é proposta a arquitetura do sistema MARS (*Multimedia Analysis and Retrieval System*). O trabalho descreve as ferramentas de extração de características, o modelo de objetos e o modelo de recuperação com um mecanismo de retroalimentação. O modelo de objetos descreve o conteúdo multimídia (imagens) e os esquemas de indexação utilizando como base a interface MPEG-7. O modelo inclui vários descritores para cada característica (cor, textura, forma, *layout*, etc.), para suportar a subjetividade dos usuários. Um método de retroalimentação é proposto em [72, 73] que associa pesos às imagens do resultado, baseado nos critérios dos usuários.

Cypress

Cypress [41] é uma nova versão do CHABOT [57]. O projeto foi definido de domínio específico com o objetivo de armazenar e recuperar imagens de uma grande coleção do *Department of Water Resources* (DWR) no Estado da Califórnia, USA. A aplicação utiliza um SGBD extensível (Postgres), para integrar sistemas de bancos de dados relacionais com técnicas de análise do conteúdo das imagens. A análise do conteúdo das imagens é baseada em *color blobs*, que são agrupamentos de cores dentro de uma imagem que podem corresponder a objetos, por exemplo: carro vermelho ou peixe amarelo. Define consultas por conceito. Uma consulta por conceito envolve informações contextuais como "céu azul". O conceito "céu azul"inclui imagens com a palavra "céu"no campo de descrição e imagens com predominância da cor azul. A interface com o usuário para consultas tem acesso através da Web.

Piction

Piction [81, 82] analisa o papel das informações textuais como informação colateral para a extração de características do conteúdo das imagens em um ambiente de banco de dados integrando textos e imagens. O sistema utiliza os subtítulos das fotografias para identificar pessoas nas fotos e emprega duas técnicas principais: o processamento da linguagem natural para obter informação dos subtítulos das fotos e o reconhecimento de objetos para localização de faces. PICTION usa as restrições espaciais (geométricas ou topológicas), características (propriedades dos objetos; por exemplo: gênero, cor do cabelo) e contextuais (por exemplo: as pessoas na foto mencionadas no subtítulo, e outras características da cena como o lugar - apartamento, aeroporto, etc.) derivadas dos subtítulos para rotular rostos gerados pelo localizador de faces. O sistema aplica-se para projetos de domínio específico: fotografias com subtítulos. As técnicas de análise do conteúdo foram feitas somente para o reconhecimento de faces humanas. Ambos os processamentos da linguagem e de imagens são complexos.

IDQS

IDQS (*Image Database Query System*) [91] foi projetado para explorar o conhecimento dos usuários utilizando técnicas de retroalimentação iterativa sobre as informações relevantes em tempo de execução. Após a tentativa inicial de recuperação, o refinamento é dado pela aceitação ou rejeição desses resultados. Esta informação é usada como uma coleção de exemplos positivos e negativos que são analisados e classificados. O processo iterativo faz o sistema aprender cada vez mais com as informações dos usuários e melhorar o processo de recuperação. IDQS consiste de três etapas: um pré-processamento para estabelecer a segmentação e os vetores das características para cada imagem, as seções de consultas com ciclos de retroalimentação, e um treinamento *off-line* para construir uma biblioteca com informações referentes aos objetos envolvidos no processo de recuperação.

Blobworld

Blobworld [7] é um sistema de recuperação de imagens baseado no conteúdo. As imagens são automaticamente segmentadas em regiões que correspondem aproximadamente a objetos ou partes de objetos. O usuário escolhe, a partir de uma imagem inicial, uma região e indica o grau de importância para a consulta. Logo, indica a importância da textura, cor, localização e forma da região. Nas consultas podem ser utilizadas mais de uma região.

DrawSearch

DrawSearch [74] é um sistema protótipo que utiliza técnicas de retroalimentação para facilitar a interação com o usuário. DrawSearch tem duas interfaces. A primeira permite consultas baseadas em cor e forma. O usuário faz um desenho (*sketch*) colocando cores nas linhas e regiões fechadas. O segundo subsistema permite consultar escolhendo uma textura.

MIR

MIR (*Multimodal Information Retrieval System*) [83] combina várias técnicas de processamento de textos e de imagens para induzir descrições semânticas das imagens. Utiliza o processamento da linguagem natural para extrair descrições semânticas como: localizações, relacionamentos espaciais entre pessoas e características que ajudem no reconhecimento de faces. Faz reconhecimento de faces em um modelo de três contornos (linha do cabelo, contornos esquerdo e direito).

NETRA

Em NETRA [50], as imagens são automaticamente segmentadas em regiões de cor homogênea. Para cada região são extraídas a cor, textura, forma e relacionamentos espaciais. As consultas podem ser feitas utilizando exemplos de imagens ou especificando cores ou relacionamentos.

VisualSeek

Em VisualSeek [78] as imagens são segmentadas em regiões de cores dominantes. As consultas consistem em achar as imagens que contêm a organização mais similar de regiões similares. Utilizando a interface, o usuário desenha as regiões, o relacionamento espacial e as dimensões das mesmas, e escolhe a cor para cada região.

Sistemas na Web

Com o advento das tecnologias de redes, muitos sistemas de banco de dados possuem arquitetura cliente-servidor e/ou desenvolvem interfaces para serem utilizados na Web. Além disso, muitas aplicações estão voltadas a explorar as capacidades da Web. A informação espalhada por toda a Internet é uma fonte inesgotável que os sistemas estão tentando aproveitar. Os sistemas aqui descritos consideram a Internet como um grande repositório de imagens.

WebSEEk [79] fornece consultas baseadas em textos e cores para as imagens e vídeo coletadas via Web. Classifica as imagens em categorias temáticas. Cria um dicionário escolhendo os termos significativos (apreciação manual) que aparecem no URL.

WebSeer [86] submete as imagens a testes de cores para classificá-las em fotografias ou desenhos. As fotografias são sujeitas a um detector de faces. O sistema extrai palavras chave sobre os textos associados tais como a referência HTTP e o título da página.

Istorama [43] é um sistema de busca pela Internet. Coleta as imagens percorrendo a rede. As imagens são segmentadas em regiões utilizando descritores de cor, textura e forma. Os descritores são armazenados em um banco de dados junto com outras informações tais como URL, data da transação e um *thumbnail*.

Amore (*Advanced Multimedia Oriented Retrieval Engine*) [51] indexa os sítios Web por textos e imagens contidas ou referenciadas nos documentos HTML. As imagens são segmentadas em regiões homogêneas de cores. As consultas também podem especificar um URL.

Outros Sistemas de Recuperação de Imagens

O projeto em [13] aborda o problema da introdução de facilidades para a recuperação de imagens de sensoriamento remoto, segundo as características do seu conteúdo em sistemas de informações

geográficas. O conteúdo da imagem é modelado como um conjunto de regiões homogêneas, ou seja, este conteúdo é obtido a partir de uma segmentação prévia da imagem. Textura e cor são as principais características de conteúdo. Cada região segmentada é descrita utilizando múltiplas representações de uma característica. Um padrão define a visão subjetiva de um usuário sobre um conceito presente nas imagens, tais como um tipo de vegetação, tipo de solo, etc. O padrão é construído a partir de várias amostras que o usuário extrai de imagens e que ele associa a um mesmo conceito. Uma consulta é definida como um conjunto de padrões. Uma imagem resultante de uma consulta deve conter regiões similares a cada um dos padrões. A função de similaridade considera múltiplas representações. Esta função se define como a combinação de dois elementos: a similaridade intra-padrão que estabelece qual a similaridade da imagem com um padrão, considerando todas as regiões segmentadas da imagem e todas as formas de representação; e a similaridade global que estabelece um único valor de similaridade de uma imagem em relação à consulta, integrando os diferentes valores de similaridade intra-padrão da imagem com cada um dos padrões. O mecanismo de retroalimentação integrado ao modelo de similaridade permite ao usuário refinar a consulta em múltiplas iterações com o sistema. Com estas iterações o sistema redefine um conjunto de parâmetros do modelo de similaridade que permitem uma aproximação ao modelo subjetivo de similaridade do usuário.

A abordagem de Sheikholeslami e Zhang [76] propõe analisar e identificar as características das imagens e determinar a sua importância para a recuperação por similaridade. O sistema suporta métodos para agrupar as imagens em um banco de dados baseados na suas similaridades e predefine ícones de imagens chamados de ícones de grupos (*cluster icons*). Os vetores de características das imagens são categorizados em grupos baseados na sua similaridade com este conjunto de ícones de imagens predefinidos. Para recuperar as imagens de um banco de dado similares a uma dada imagem em uma consulta, a mesma é comparada com os ícones dos grupos. Somente as imagens nos grupos correspondentes aos ícones similares serão analisadas. Um agrupamento efetivo das imagens no banco de dados pode reduzir grandemente o número de imagens a ser analisadas. Em particular, para esse projeto foram extraídas diferentes características da textura para construir os vetores no espaço de características. Os ícones e, portanto, os grupos de imagens são dependentes da aplicação e são determinados por um especialista.

A abordagem de Picard em [65] descreve suas pesquisas sobre qual modelo usar na análise de imagens em uma biblioteca de imagens e como combinar estes modelos para satisfazer as solicitações dos usuários. Um dos problemas no processamento do conteúdo das imagens é quando o conteúdo da imagem é difícil de descrever objetivamente. Dentre as soluções propostas nesta pesquisa, o usuário não precisa selecionar parâmetros do modelo senão simplesmente escolher imagens exemplos. A pesquisa se baseia na subjetividade das solicitações dos usuários e possui dois componentes: o modelo de inferência e combinação e o modelo de instrução e generalização. O primeiro infere automaticamente a melhor combinação de modelos para representar os dados de interesse a um usuário. A análise tenta inferir as características das imagens comuns às imagens escolhidas por um usuário e as utiliza para predizer outras imagens de interesse para ele. A inferência é sempre baseada no conjunto atual de exemplos positivos e negativos dados por um usuário. O modelo de instrução e generalização baseia-se na subjetividade na modelagem; o sistema aprende continuamente durante a interação com cada usuário. O conhecimento aprendido em um problema pode ser usado para resolver outro problema.

Em [6] é proposto um modelo canônico para classes de objetos em imagens aéreas. Este

modelo é motivado pela observação de regiões geográficas de interesse que são caracterizadas por coleções de padrões de texturas que se correspondem com processos geográficos que as geram. Zitnick e Kanade [95] apresentam um algoritmo para a obtenção de mapas de disparidade com oclusão explicitamente detectada. Hallinan e Jackway [38] descrevem um algoritmo de seleção de características que utilizam um algoritmo genético para escolher um subconjunto de características associadas a pesos para uma rede de classificação de três camadas com retroalimentação. Marcas de água digitais foram propostas em [21] para a autenticação e impressões digitais de vídeos e imagens e para a verificação da integridade da mídia visual.

Em [80, 85] é proposto um esquema para recuperar imagens por conteúdo usando classes pré-determinadas de imagens. Para cada classe é determinada uma imagem representativa. A construção das classes é feita automaticamente com mínima mediação manual (as imagens representativas iniciais são escolhidas manualmente). As características analisadas das imagens foram o histograma de cores e a textura. A solicitação é comparada com as imagens representativas de todas as classes, e uma ordem das classes é determinada baseada na distância da solicitação a cada imagem representativa. A busca é feita nas classes mais próximas na ordem de comparação.

Em [77], cada imagem passa por uma classificação de três etapas. Na primeira etapa as imagens são classificadas em tipos (por exemplo: fotos em cores, gráficos e fotos em escala de cinza), baseado na análise da cor. Na segunda etapa as imagens são classificadas por composição em cenas, texturas, silhuetas e centro-borda (*center-surround*), aplicando técnicas de análise mais complexas como a extração de regiões. Na última etapa as imagens são classificadas em classes semânticas (por exemplo: praias, prédios e natureza), usando classificadores especializados que foram treinados com imagens que foram classificadas utilizando a informação textual relacionada (por exemplo, os *hiperlinks* na Web).

Natsev et al. [52] propõem WALRUS (WAveLet-based Retrieval of User-specified Scenes), um algoritmo de recuperação por similaridade que escala e traslada objetos dentro de uma imagem. Cada imagem é separada em regiões. A medida de similaridade entre dois pares de imagens é definida para ser a fração da área das duas imagens cobertas pelas regiões coincidentes das imagens.

Em [9] é apresentada uma abordagem onde a indexação é feita por múltiplas características das imagens, mas é criada uma estrutura de indexação para cada característica, utilizando o método de acesso VamSplit R-tree. Durante o processamento das solicitações, as imagens resultantes de consultar cada característica separadamente são combinadas para obter uma única ordem global. Várias técnicas são propostas para fazer a combinação de resultados. Nesta abordagem, diferentes características podem ter diferentes pesos ou diferentes métricas, de acordo com a relevância de cada característica para a consulta.

Em [42] é apresentada uma estrutura de indexação de dez níveis para representar a informação visual de uma imagem. A estrutura é representada por uma pirâmide, onde os quatro níveis mais altos representam as características sintáticas das imagens, ou seja, o processamento a baixo nível, tal como a distribuição espacial dos objetos e das cores. Os restantes níveis (níveis inferiores da pirâmide) representam o conteúdo semântico das imagens. Uma estrutura de três níveis é apresentada para a informação não visual. A integração destas duas estruturas para representar a informação semântica é feita através da definição de uma tabela de informação semântica.

[87] faz uma revisão de trinta e nove sistemas. Cada sistema é descrito baseado em caracte-

Sistema	Descritores	Similaridade	Indexação	Aplicações
QBIC	Cor: vetor de 3 dim. da média das cores no espaço Munsell	Distância euclidiana ponderada	multi-*	aplicações IBM, por exemplo: DB2
	histograma de cores no espaço RGB com vetor de dim. 256	Medida quadrática	vetor de dim. 64 em *R-tree*; multi-*	
	Textura: vetor de 3 dim. CCD	Distância euclidiana ponderada	multi-*	
	Forma: vetor com os seguintes elementos: área, circularidade, orientação dos eixos, etc	Distância euclidiana ponderada	multi-*; vetor de 64 dim. → transformada *KL* → *R-tree*	
	Relacionamentos espaciais: localização do centro do objeto na imagem	Segmentação em regiões, cada região é espacialmente analisada		
	Sketch: mapa de baixa resolução 64x64			
	Palavras chaves (separadamente)			
	O processamento pode ser para objetos, regiões ou imagem.			
Amore	Cor, forma: segmentação em 8 regiões por cores homogêneas	Região a região: Forma: pixels sobrepostos	Sistema COIR de SEMCOG	SRI de arte *Arthur*, Centro de Tecnologia Cultural na California.
	Palavras chaves	Cor: distância no espaço HLS		
Blobworld	Cor: histograma de 218 dim. no espaço Lab por região	Distância quadrática	SVD para reduzir a dim. do vetor → *R*-tree*	Coleção de 10,000 fotos.
	Textura: (contraste, anisotropy) por região	Distância euclidiana		
	Forma: (area, centro, orientação)	Distância euclidiana entre os centros		
	Combinação: as distâncias são combinadas em uma distância final			

Sistema	Descritores	Similaridade	Indexação	Aplicações
VisualSeek	Textura: vetor de 9 dim. (Somente CBVQ)	Combinação: matriz para textura e cor, a partir da qual é computada uma distância.	índice por região	Dois sistemas para busca de imagens e vídeos na Web, Universidade de Columbia, NY.
	Cor: : segmentação e histograma de cores			
	Forma: MBR, área e centro de regiões	Distância euclidiana	MBR: *R-tree*; Centros: quadtree	
	Combinação: a soma das distâncias ponderadas dá uma distância combinada			
Drawsearch	Cor: imagem dividida em 4x4. Para cada subimagem: média de cores em 48 dim.	Cor e forma: métrica do cosseno para os vetores		SRI por *sketch* e refinamento sucessivo com interface na Web.
	Forma: os elementos do vetor descriptor são a parte real dos 100 coeficientes de *Fourier*	Combinação: soma ponderada das distâncias entre o vetor de cor e o descritor da forma		
	Textura: segmentação usando *Gaussian Markovian Random Fields*	Distância euclidiana		
Cypress Chabot	Cor: busca por *color blobs*, correspondência com objetos (carro vermelho) Consultas por conceito	Cor: métrica de distância Combinação: textos & *color blobs*		Coleção de fotos de recursos naturais (DWR-Califórnia)
MARS[1]	Cor: histograma sobre espaço HSV	Interseção de histogramas		Imagens de artefatos da África antiga do Museu de História Cultural de UCLA.
	Textura: vetor de 3 dim. CCD	Distância euclidiana		
	Esquema (*Layout*): 5x5 subimg e p/ cada, histograma HS de 2 dim. (cor, textura)	Distância entre duas imagens: soma das 5x5 similaridades		
	Forma: MFD	Distância euclidiana		
	Palavras chaves			

Sistema	Descritores	Similaridade	Indexação	Aplicações
MIR	Processamento da Linguagem Natural para extrair descrições semânticas como: localizações, relacionamentos espaciais entre pessoas e características que ajudem no reconhecimento de faces. Reconhecimento de faces: modelo de 3 contornos (linha do cabelo, contornos esquerdo e direito)	Combinação: Soma parametrizada das similaridades dos diferentes processamentos (textos e CIs)		BD1: 5,000 imagens acompanhadas de textos, fornecidas pela *United Press International*. BD2: fotos de clientes da Kodak com anotações. BD3: documentos da Web.
	Cor: histograma	Distância euclidiana		
NETRA	Segmentação de imagens em regiões de cores homogêneas. Para cada região:			Alexandria Digital Library (ADL).
	Cor: vetor de *n* dim. (*n* = número de cores para representar a região)	Distância euclidiana no espaço RGB	*SS-tree* separado	
	Forma: FFT de 3 vetores usados para representar a forma das regiões	L_1*-distance*	*SS-tree* separado	
	Textura: vetor que representa a transformada de *Gabor*	Distância euclidiana	*SS-tree* separado	
WISE ou WBIIS	Esquema de Cor: vetor calculado a partir da: • desviação padrão • transformada de *wavelets* de *Daubechies*	Passo 1: desvio padrão para a imagem da consulta comparada com as imagens no BD;		Atende aprendizagem e pesquisa em projetos de arte liberal, na Biblioteca da Universidade de Standford.
		Passo 2: para as imgs dentro do critério, distância euclidiana entre os coeficientes da transformada.		

Sistema	Descritores	Similaridade	Indexação	Aplicações
DISIMA	Objetos e relacionamentos espaciais: • deteção de faces usando MBR. Palavras chaves: anotações sobre as faces.	Predicado *contains* (objeto contido em imagem?). Distâncias de cor e textura sobre imagens. Distâncias sobre objetos. Buscas tradicionais sobre textos. Combinações de todos.	*2D-h-tree* (B^+*-tree* sobre *2D-String*)	SRI distribuído com suporte para consultas sobre BD espaciais e geográficos.
SEMCOG	Objetos e relacionamentos espaciais: Sistema COIR: identifica regiões (objetos) Catálogo: semântica s/ componentes das imagens	COIR: identifica relacionamentos espaciais dos objetos e semântica usando o catálogo	*2D-String*	Amore
CHITRA	Cor: histograma de 768 elementos Textura: vetor de dim. 48 com funções de *Gabor*	Alguma critério de similaridade para cada CI.	Algoritmo melhorado de Fagin para otimizar acessos ao BD.	Testes com BD de 1,000 imagens.
MARS[2]	QBE: cor (histograma), textura e forma. *Sketch*: representando relacionamentos espaciais. Atributos textuais e outros tipos de mídia.	Cor, forma: distância euclidiana. Combinação: soma das distâncias ponderadas.	Estrutura de indexação armazena vetores de CIs. Combinado: LSD^h*-tree* adaptado	Sistema de Recuperação de diferentes tipos de mídia: textos, imagens, etc.

[1] MARS (*Multimedia Analysis and Retrieval Systems*)
[2] MARS (*Multimedia Authoring Retrieval Service*)

rísticas, apresentação, consulta, indexação, aplicação, critério de similaridade. Outros sistemas e algoritmos são descritos em [23, 22, 10, 9, 15, 67, 90].

2.5.1 Quadro Resumo

Nesta seção apresentamos um quadro resumo com as propriedades de 14 SRIs, subconjunto dos trabalhos descritos até aqui. As linhas da tabela formam a informação das aplicações enquanto as colunas fazem uma classificação baseada nos seguintes pontos:

- **Descritores**: Características visuais da imagem de baixo ou alto nível e seu processamento para extrair a informação;
- **Similaridade**: os critérios de similaridade para cada descritor, utilizados para comparar a imagem da consulta (ou *sketch*, textura, cor, etc.) com as imagens no banco;
- **Indexação**: a estrutura usada para indexar os descritores;
- **Aplicações**: as áreas de aplicação onde estes sistemas são utilizados.

As consultas em cada sistema são baseadas nos descritores e as comparações são feitas utilizando as operações de similaridade correspondente. No caso dos atributos tradicionais e as palavras chave, as consultas são feitas da maneira tradicional.

O fato de ter itens que não foram preenchidos no quadro significa que não se encontraram as informações correspondentes para esses sistemas ou os sistemas não desenvolveram as técnicas correspondentes.

A maioria dos SRIs extraem as características visuais das imagens e/ou utilizam palavras chave para fazer buscas em uma coleção. Como podemos observar na tabela, para cada característica visual extraída se aplica um critério de similaridade e em alguns casos, existe uma estrutura de indexação. Alguns sistemas combinam vários descritores para dar um critério de similaridade e/ou um método de acesso final (ou combinado). Para isto, aplica-se peso (que determina a importância do descritor na consulta) para cada descritor envolvido na métrica combinada para comparar duas imagens. Sistemas como QBIC (multi-*), Blobworld (critério de similaridade que combina todas os descritores das características extraídas: cor, forma e textura), Drawsearch (que combina dois dos descritores extraídos: o de cor e o de forma) são exemplos de sistemas que utilizam combinações das similaridades dos descritores nas consultas.

Outro comportamento comum aos sistemas analisados é que alguns sistemas como o WISE, implementam vários passos para determinar as imagens similares para uma determinada consulta, ou para criar uma estrutura de indexação (QBIC, Blobworld).

Por último, com o propósito de ter um estoque variado de aplicações, escolhemos sistemas que trabalham em diferentes domínios de aplicação, tais como: processamento da linguagem natural, detecção de faces para imagens acompanhadas de textos (em jornais), bancos de dados espaciais ou geográficos, coleções de arte, história, fotos, aplicações para bibliotecas e coleções de imagens distribuídas na Web.

Como conclusão do estudo e análise dos sistemas de recuperação de imagens, obtemos dois resultados interessantes: uma classificação dos algoritmos que intervêm no processo de recuperação de imagens e uma arquitetura geral que caracteriza os sistemas de recuperação de imagens.

2.5.2 Algoritmos de Recuperação de Imagens

Depois de estudar os sistemas de recuperação de imagens podemos concluir que os algoritmos que utilizam estas aplicações podem ser classificados em três tipos fundamentais.

1. *Algoritmos de extração de descritores*: são as operações de segmentação e de extração de descritores das imagens.
2. *Algoritmos de comparação ou similaridade*: são as operações que comparam as características ou informações semânticas de duas ou mais imagens.
3. *Algoritmos de indexação*: a cada característica da imagem pode ter associado um método de acesso (por questões de otimização).

Uma vez definido um descritor para as imagens é preciso definir um operador de similaridade que faz as comparações entre as imagens. Para acelerar o processo das comparações podem ser construídos índices e os métodos apropriados para indexar os descritores e fazer as buscas a partir do critério de similaridade adotado.

2.5.3 Arquitetura de um Sistema de Recuperação de Imagens

Nesta seção apresentamos uma proposta nossa de uma arquitetura que descreve muitos dos sistemas de recuperação de imagens e expressa seus principais componentes.

A figura 2.1 mostra a arquitetura de um sistema típico de recuperação de imagens. Os sistemas podem não incluir todos os passos presentes nesta arquitetura.

A arquitetura possui duas fases. A primeira fase refere-se à etapa de pré-processamento das imagens, onde ocorre o povoamento dos bancos de dados. Os descritores de uma imagem são extraídos e armazenados em um banco de dados, e são criadas as estruturas de indexação.

Na segunda etapa, o usuário final de uma aplicação expressa uma consulta que vai ser processada. Uma vez expressada, a consulta é processada. No caso da consulta por *sketch*/imagem, os descritores apropriados são extraídos e comparados aos armazenados no banco de dados (utilizando as estruturas de indexação) e assim as imagens similares são recuperadas e os resultados são visualizados. A partir da visualização dos primeiros resultados, os usuários podem refinar a consulta.

2.6 Usuários das Aplicações

A parte mais importante nas aplicações de recuperação de imagens são os usuários. Esses sistemas, como qualquer outra aplicação de banco de dados, possuem diferentes tipos de usuários, diferenciados pela maneira deles interagirem com o sistema.

Os desenvolvedores ou programadores de aplicações são profissionais de computação que interagem com o sistema através de chamadas à linguagem de manipulação dos dados, que podem estar incluídas dentro de programas escritos em uma linguagem anfitriã (C, Pascal, Cobol). Outros usuários do sistema não precisam escrever programas, mas interagem com o sistema através de chamadas a aplicações feita pelos programadores ou submetendo consultas em uma linguagem de consulta interativa.

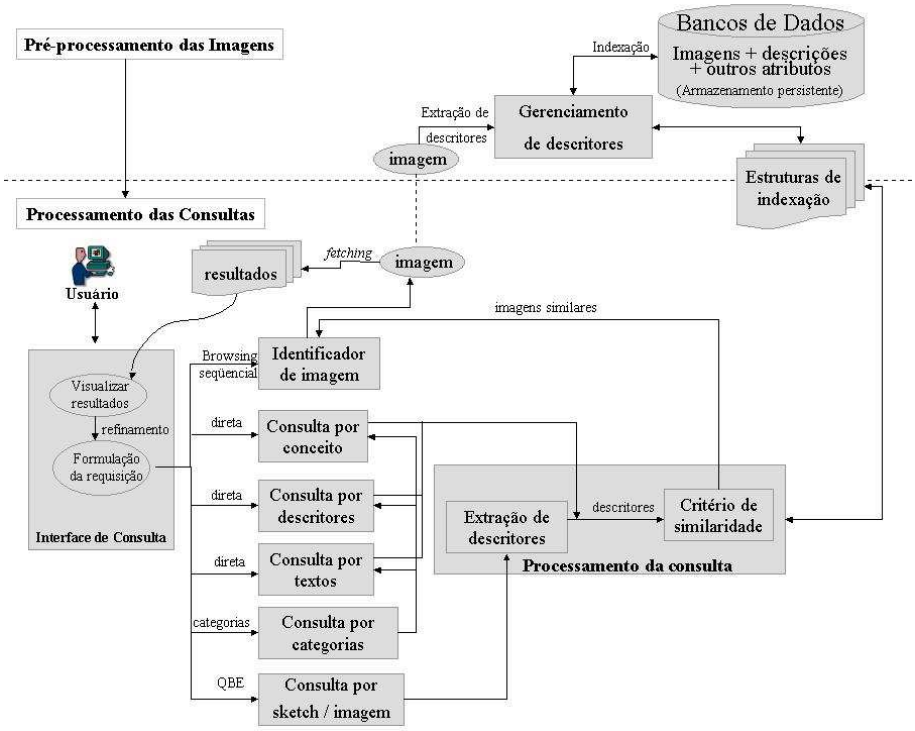


Figura 2.1: Arquitetura que descreve os Sistemas de Recuperação de Imagens.

Nem todos os usuários têm o mesmo nível de acesso à informação. Assim, os usuários se classificam em diferentes grupos. Cada grupo vai ter um conjunto de capacidades sobre o banco de dados. O nível de acesso ao banco de dados pode ser classificado em dois subconjuntos.

- Os que podem alterar informação no banco de dados. Formado por dois grupos:

  Desenvolvedor: é a pessoa que tem esse controle central sobre o sistema. Algumas funções do desenvolvedor são apresentadas a seguir.

  - Definição do esquema do banco de dados
  - Definição de estruturas de armazenamento e métodos de acesso
  - Modificação do esquema e da organização do banco de dados
  - Definição das restrições de acesso ao banco de dados
  - Especificação das restrições de integridade dos dados

  Operadores: pode alterar o conteúdo do banco.

- Os que somente podem consultar. Formado por um grupo:

  Usuários: os usuários podem ser classificados em diferentes grupos para estabelecer diferentes níveis de consulta. Isto é, nem todos os usuários podem consultar toda a informação.

2.7 Sumário

Dificilmente seria possível desenvolver um sistema de gerenciamento que atendesse adequadamente às necessidades de todas aplicações de bancos de dados de imagens, pois estas variam muito de acordo com o caráter da aplicação.

Neste capítulo, descrevemos os requisitos que as aplicações de imagens impõem aos sistemas de gerenciamento de bancos de dados. Apresentamos um estudo das aplicações e definimos os pontos comuns a esses sistemas, que podem ser atendidos por um sistema de gerenciamento de bancos de dados de imagens de propósito geral. Esses pontos são levados em consideração em nossa abordagem a partir do próximo capítulo.

Capítulo 3

Modelo para o Gerenciamento de Imagens sobre Bancos de Dados

3.1 Introdução

As imagens impõem novas condições às aplicações e aos sistemas de bancos de dados, que não estavam presentes nas aplicações tradicionais. A inclusão das imagens no projeto de uma aplicação introduz desafios tais como: definição dos algoritmos e estruturas dos descritores das imagens, definição de critérios de similaridade, expressão dos descritores nas linguagens de consultas, suporte para a indexação dos descritores, e integração da recuperação de dados tradicionais e descritores.

Neste capítulo apresentamos nosso modelo de gerenciamento de imagens sobre bancos de dados. Atendendo esses requisitos das aplicações de recuperação de imagens, a descrição do nosso modelo de gerenciamento de imagens inclui os tópicos a seguir.

1. Esquema para o gerenciamento de imagens: o esquema em camadas que define o gerenciamento de imagens sobre bancos de dados.

2. Modelo do dado imagem: propõe um modelo das imagens, suas características, comportamentos e relacionamentos.

3. Modelo para a recuperação de imagens: propõe um modelo que representa a similaridade das imagens nas consultas.

4. Modelo de consultas: definição de um modelo fuzzy para comparar as imagens.

5. Mecanismo de retroalimentação das consultas: propõe um mecanismo geral baseado no modelo fuzzy de consultas e no modelo de recuperação de imagens.

6. Modelo baseado no padrão ODMG: nossa proposta ao modelo de gerenciamento de imagens é especificamente definida baseada no padrão de dados ODMG. Esta proposta é somente uma abordagem para desenvolver nosso modelo geral. O objetivo da utilização do ODMG é padronizar a manipulação de bancos de dados e ampliar o grau de portabilidade das

aplicações que poderão ser desenvolvidas utilizando nosso modelo. Propõe novos tipos de dados para o modelo de objetos. Utiliza e estende as linguagens de definição e consulta (ODL-I e OQL-I), definindo: operadores de similaridade, operadores lógicos fuzzy, relacionamentos espaciais entre os objetos, e conceitos.

Ao longo do capítulo, a apresentação do texto é acompanhada de exemplos.

Os princípios de projeto que nosso modelo deve adotar para poder definir um sistema de gerenciamento de bancos de dados de imagens. Os princípios de projeto podem ser resumidos como a seguir.

- *Generalidade.* Um SGBDI deve oferecer as facilidades necessárias para implementar aplicações de bancos de dados de imagens independentemente do domínio da aplicação.
- *Flexibilidade e Adaptabilidade.* O SGBDI é mais flexível na medida que expressa maior número e tipos de consulta sobre os bancos de dados. Também, pode oferecer ao desenvolvedor de aplicações meios simples e ferramentas para desenvolver aplicações de domínio específico.
- *Extensibilidade.* O modelo de dados, os tipos de dados, a linguagem de consulta e qualquer funcionalidade que o modelo proponha deve oferecer meios para serem estendidos com novos elementos.

3.2 Esquema para o Gerenciamento de Bancos de Dados de Imagens

Nesta seção apresentamos o esquema que define nossa idéia sobre o gerenciamento de imagens sobre bancos de dados. A figura 3.1 mostra o esquema de três camadas. A camada 1 representa os sistemas gerenciadores de bancos de dados. Os SGBDs armazenam os bancos de dados das imagens. A adição de uma camada intermediária (camada 2) entre a aplicação (camada 3) e os sistemas de bancos de dados é necessária para dar o suporte que as aplicações de gerenciamento multimídia, em particular de imagens, requerem.

A camada intermediária possui três componentes.

1. *Módulo de Definição de Imagens (MDI).* Permite especificar o esquema do banco de dados de imagens.
2. *Módulo de Manipulação de Imagens (MMI).* Permite aos usuários acessar e manipular um banco de dados organizado pelo esquema de dados apropriado. Uma vez que o banco de dados é povoado, os usuários precisam meios para manipulá-lo. A manipulação de imagens compreende as operações apresentadas a seguir.
   - Inserção de imagens no banco de dados.
   - Remoção de imagens do banco de dados.
   - Modificação da informação armazenada no banco de dados.

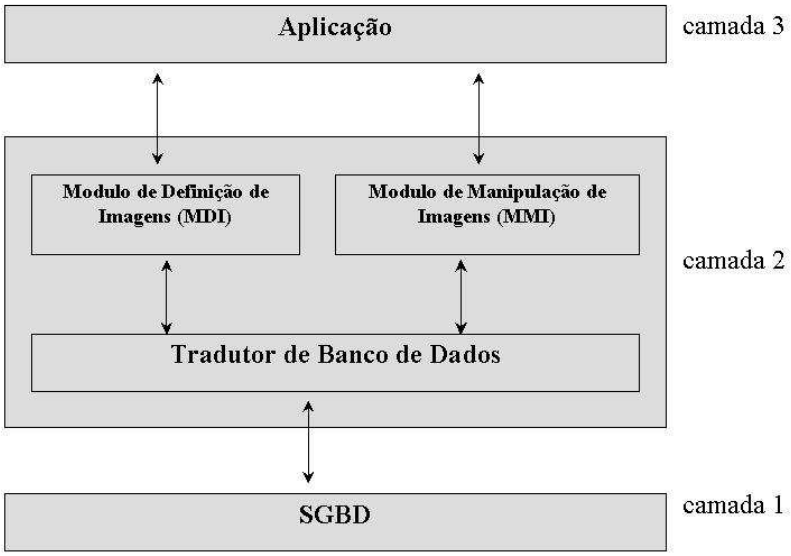


Figura 3.1: Esquema em três camadas que define o gerenciamento de bancos de dados de imagens.

- Recuperação de imagens do banco de dados.

3. *Tradutor.* Oferece as funcionalidades para interagir com o sistema de bancos de dados na camada inferior. Dado que nem todos os SGBDs possuem a mesma interface, a função do Tradutor é mapear as solicitações e obter os resultados dos dados que se requisitem. Com a existência deste componente, uma mudança de SGBD não causará mudanças no modelo.

A seguir descrevemos o modelo das imagens que vai ser utilizado pela segunda camada do esquema de gerenciamento para a declaração e recuperação das imagens.

3.3 Modelo das Imagens

Para projetar um modelo que suporte o gerenciamento de imagens, primeiro necessitamos formalizar como uma imagem é modelada. Nesta seção, apresentamos uma proposta para modelar as imagens, suas características e comportamento. O modelo do dado imagem é enfocado a partir dos seus atributos e descritores.

3.3.1 Atributos

As características não visuais das imagens são os atributos alfanuméricos estruturados que servem como base às consultas. Definamos aqui o conjunto A de características não visuais de uma imagem. Para isto, sejam os seguintes conjuntos.

SA: Conjunto de atributos tradicionais que descrevem as imagens com palavras chave ou textos livres.
NSA: Conjunto de atributos tradicionais que não descrevem o conteúdo de uma imagem

O conjunto A dos atributos não visuais de uma imagem é o conjunto de todos os valores resultantes da união de SA e NSA,

$$A = SA \cup NSA$$

Por exemplo:

$$sa \subset SA = \{description\}$$

$$nsa \subset NSA = \{author, date\}$$

$$a \subset A = \{description, author, date\}$$

3.3.2 Descritores

Os descritores das imagens são as representações das características visuais. Definamos as representações de uma imagem.

I: Conjunto de todas as representações visuais de uma imagem, em formatos como `.gif`, `.tiff`, `.jpg`.

R: Conjunto de todas as representações resultado de operações de processamento das imagens. Para isto,

FR: Conjunto de todas as representações físicas das imagens (representação no nível de *pixels*, por exemplo, as representações matricial e vetorial).
ILR: Descritores Globais. Conjunto de todas as representações lógicas sobre a imagem tratada como uma única entidade.
OLR: Descritores Locais. Conjunto de todas as representações lógicas sobre os objetos da imagem.

Uma representação lógica é obtida da extração de uma ou mais características visuais da imagem a partir de uma representação física, ou seja,

Dado F: Conjunto de características visuais (extraídas da imagem), tais como a cor, textura, forma, objetos e relações espaciais,

$$F = \{f_i\},\ i = 1..k$$

$$FR \xrightarrow{f_i} \{LR_{ij}\} ,$$

onde $\{LR_{ij}\}$ é o conjunto dos descritores de f_i. Por exemplo, o histograma de cores, *color moments*, ou *color layout* (esquema de cores) são representações da cor. Os conjuntos ILR e OLR contêm todos os descritores globais e locais para todas as f_i,

$$ILR = \{ILR_{ij}\}$$

$$OLR = \{OLR_{ij}\}$$

Então, o conjunto de representações lógicas de uma imagem, LR, é a união dos conjuntos das representações lógicas das imagens e seus objetos.

LR: Conjunto de todas as representações lógicas sobre a imagem, tal que

$$LR = ILR \cup OLR \text{ e,}$$

o conjunto de representações de uma imagem está composto, no mínimo, por uma representação física e as representações lógicas.

$$R = FR - \{\emptyset\} \cup LR$$

3.3.3 Imagem

Por definição do modelo, a descrição de uma imagem depende dos conjuntos (I, A, F, R). Uma imagem precisa ter, no mínimo, uma representação visual e uma representação física. As representações lógicas dependem de cada aplicação.

Uma imagem *IMG* pode ser definida como o conjunto de todas as representações, tal que:

$$IMG = I \cup A \cup R$$

IMG contém todos os atributos e descritores de uma imagem. Cada aplicação pode escolher representar as características que se ajustam a seus requisitos, como sendo um subconjunto de *IMG*.

Seja *img* uma das descrições em *IMG*; *img* pode descrever, por exemplo, uma classe de imagens que contenha um histograma de cores como descritor da cor e descritores de objetos e relacionamentos espaciais, da seguinte maneira:

$$a \subset A = \{description, author, date\}$$

$$i \subset I = \{file\}$$

$fr \subset FR = \{raster\}$

$f \subset F = \{color, spatialrelationships, imageobjects\}$

$ilr \subset ILR = \{ColorHistogram, 2DString\}$

$olr \subset OLR = \{MBR\}$

$r \subset R = \{ColorHistogram, 2DString, MBR\}$

$img \subset IMG = \{file, raster, ColorHistogram, 2DString, MBR, description, author, date\}$

Uma instanciação de img seria:

$$img_1 = \{image.gif, fraster, vHC_1, v2DStr_1, vMBR_1, reunião, smg, 25-05-2001\}$$

Em um banco de dados, as imagens podem ser representadas da seguinte maneira:

$t_i = \{id, img_i\}$, onde

id é o identificador único da tupla ou objeto.

Portanto, no MDI, a declaração de uma coleção de imagens dentro de um banco de dados é feita especificando uma descrição baseada nos conjuntos do modelo de imagens mais um identificador único da imagem (id, I, A, R).

Este modelo é baseado na descrição dos dados multimídia definido pelo padrão MPEG-7. O modelo não depende de um domínio específico de aplicações. Para identificar o padrão MPEG-7 no nosso modelo, podemos associar os conjuntos definidos à terminologia do MPEG-7.

Mapeamento em MPEG-7
$IMG = (I, A, R)$

data: I;
feature: F, A;
D: R;
descriptor value: instanciação de R;
DS: img;
description: instanciação de img (img_1);
coded description: indexação;
DDL: por exemplo, ODMG - ODL-I.

3.4 Modelo de Recuperação de Imagens

Uma vez definido um modelo para representar as imagens, propomos um modelo para a recuperação de imagens. Este modelo inclui o modelo de imagens junto com o conjunto de operações de similaridade das imagens.

Uma representação das imagens tem sempre associado uma operação de similaridade e, opcionalmente, um índice que armazena os descritores para acelerar o processo das comparações.

As operações de similaridade são utilizadas para determinar se duas imagens são similares ou não. Podemos definir o conjunto das operações de similaridade, S, como

$$S = \{S_{ij}\}$$

onde S_{ij} é o conjunto das operações de similaridade para comparar as representações R_{ij}.

Dado um descritor das imagens $LR_{i1} \in R$, para a característica visual f_i, e seus valores para duas imagens $LR_{i1}(img_1)$ e $LR_{i1}(img_2)$, então,

$$LR_{i1}(img_1) \;\; s_{i1} \;\; LR_{i1}(img_2) \;\rightarrow\; \delta,$$

onde o critério de similaridade utilizado para comparar as duas imagens é $s_{i1} \in S$, e δ é o valor de similaridade entre as duas imagens.

A recuperação de imagens baseada em descritores do conteúdo pode ser representada como o modelo de imagem acima em junção com o conjunto de operações de similaridade S.

Modelo de Recuperação de Imagens: (I, A, R, S)

Os índices são mecanismos que otimizam a recuperação, mas não fazem parte essencial da recuperação das imagens. Por esta razão, eles não foram incluídos no modelo de recuperação. Entretanto, os índices ajudam no processamento das solicitações. Por esta razão, definimos uma tupla formada por referências a algoritmos de recuperação de imagens, chamada de *tupla de operações*. Uma tupla contém associações de um algoritmo de extração de descritores, uma operação de similaridade e um índice.

3.4.1 Tuplas de Operações

O objetivo de definir a tupla de operações é facilitar o processamento das solicitações. Além dos três tipos de algoritmos mencionados na seção 2.5.2, acrescentamos mais um tipo de algoritmo:

- *Algoritmos de normalização*: estas operações normalizam as operações de similaridade para obter um valor no intervalo $[0, 1]$, pois é necessária a compatibilidade com o modelo de consultas fuzzy (proposto na próxima seção).

Uma tupla de operações é formada por quatro elementos, e definida como:

$$id = \{< op_{extrdesc}, op_{similar}, op_{norma}, op_{index} >\}$$

O identificador da tupla, id, serve para referenciar os algoritmos nas solicitações, assim estas últimas são feitas de maneira uniforme, sem chamadas explícitas a funções. Os identificadores devem ser palavras fáceis de lembrar.

Cada aplicação precisa especificar as tuplas de operações para processar suas imagens. Uma aplicação pode definir tantas tuplas de operações quanto necessárias para a recuperação de imagens, por exemplo:

Aplicação 1 define

$$id_1 = \{< op_{extrdesc_1}, op_{similar_1}, op_{norma_1}, op_{index_1} >\}$$

$$id_2 = \{< op_{extrdesc_2}, op_{similar_1}, op_{norma_1}, op_{index_2} >\}$$

$$id_3 = \{< op_{extrdesc_3}, op_{similar_2}, null, op_{index_1} >\}$$

A aplicação do exemplo define três grupos de operações. O primeiro grupo de identificador id_1 diz que imagens no banco de dados da aplicação vão ser processadas com o algoritmo de extração de descritores $op_{extrdesc_1}$, inseridas no índice op_{index_1}, e depois comparada com outras imagens utilizando as operações de similaridade e normalização $op_{similar_1}$ e op_{norma_1}.

Os algoritmos de normalização não são obrigatórios na definição dos conjuntos, pois as operações de similaridade podem retornar seus valores no intervalo $[0, 1]$. Os algoritmos de indexação são utilizados opcionalmente pelas aplicações. Por exemplo, na terceira tupla de operações da *aplicação 1*, identificada por id_3, a operação de normalização é definida como nula.

A definição das tuplas de operações é simples e permite diversas combinações dos algoritmos. Ainda neste capítulo veremos como é feita a declaração das tuplas pelas aplicações e como elas são utilizadas nas consultas.

O MMI permite aos usuários consultar um banco de dados. A recuperação de imagens em um banco de dados é baseada no nosso modelo de recuperação. O modelo de recuperação de imagens é utilizado pelo modelo fuzzy de consultas e o mecanismo de retroalimentação de consultas, descritos a seguir.

3.4.2 Modelo Fuzzy para as Consultas

As imagens não podem ser envolvidas em uma comparação exata entre elas ou seus descritores. A resolução de uma consulta é feita usando uma definição de critérios de similaridade que implica em uma recuperação imprecisa. Nosso modelo introduz características fuzzy na linguagem de consulta.

Os sistemas de recuperação de imagens precisam desenvolver mecanismos para capturar essa natureza fuzzy inerente às imagens. Uma abordagem para representar a similaridade entre as imagens pode ser a utilização da teoria dos conjuntos fuzzy. A teoria fuzzy pode complementar a busca por critérios, assim melhorando os resultados da consulta.

Os conceitos da teoria fuzzy empregados no desenvolvimento do modelo de similaridade são explicados no anexo B.

Modelo Fuzzy

Para definir o modelo de consultas fuzzy, vamos definir a representação de uma consulta na álgebra relacional, como segue.

$$< conjunto\,de\,resultados > \leftarrow \sigma_{<criterio\,de\,selecao>}(colecao\,de\,imagens)$$

$$< conjunto\,de\,resultados > ::= colecao\,de\,imagens$$

$$< criterio\,de\,selecao > ::= < predicado > \ (operador\_logico < predicado >)^{+}$$

$$operador\_logico \in \{AND, OR, NOT\}$$

$$< predicado > ::= < predicado\_tradicional > \mid < predicado\_fuzzy >$$

$$< predicado\_tradicional > ::= < nome\,de\,atributo > \ operador\_comparacao \\ < constante | nome\,de\,atributo >$$

$$operador\_comparacao \in \{=, >, <, \neq, \geq, \leq\}$$

$$< predicado\_fuzzy > ::= < modificador > < operacao\_similaridade >$$

$$< operacao\_similaridade > ::= < identificador > (< parametros >)S$$

$$< parametros > ::= < constante | nome\,de\,atributo > [, < parametros >]$$

$$< operacao\_similaridade > \in S$$

Podemos dizer que o conjunto de resultados de uma consulta, U, tem o mesmo modelo das imagens, pois ele é também uma coleção de imagens.

$$U = \{img_i\}, \quad U \subseteq IMG$$

Nesta seção vamos definir os predicados fuzzy de uma consulta. O conjunto U será nosso universo de discurso e será fuzzy. Para isto, é definida a pertinência de um elemento img_i no conjunto U. Uma forma de se indicar essa pertinência é através de uma *função de pertinência* $\mu_U(img_i)$, cujo valor indica se o elemento img_i pertence ou não ao conjunto U. Cada elemento do conjunto tem associado o grau de pertinência da imagem à consulta, de maneira que:

$$U = \{< img_i \ , \ \mu_U(img_i) >\}$$

Definimos as operações de similaridade, em S, como operações que retornam valores fuzzy. Lembrando que δ é o valor de similaridade entre duas imagens, podemos definir δ como um valor fuzzy ($\delta \in [0, 1]$). Logo, definimos o grau de pertinência como equivalente ao valor resultado da comparação.

$$\mu_U(img_i) = \delta,$$

Os resultados de uma consulta são ordenados em ordem decrescente do grau de similaridade.

Os resultados dos predicados tradicionais podem ser facilmente mapeáveis a esta definição fuzzy, como a seguir.

$$\mu_U(img_i) = \begin{cases} 1, & se < predicado\_tradicional > \ para\ img_i \ \ verdadeiro \\ 0, & se < predicado\_tradicional > \ para\ img_i \ \ falso \end{cases}$$

3.4.2.1 Modificadores Fuzzy

Para expressar as consultas de uma maneira mais próxima às necessidades do usuário, definimos a variável lingüística *modifiers* associada a um conjunto de modificadores. Esses modificadores qualificam os predicados da consulta, restringindo o conjunto resultado a subconjuntos de maneira conveniente para o usuário. Os modificadores em uma consulta são utilizados a critério do usuário e dão força semântica aos resultados.

A variável lingüística *modifiers* toma valores no intervalo $[0, 1]$. Seja u o domínio *modifiers* do universo U de resultados. A variável pode ser definida como um conjunto de termos:

$$T(u = modifiers) = \{termo_1\ , ...,\ termo_k\ , ...,\ termo_q\}$$

A seguir apresentamos dois exemplos que podem representar os valores lingüísticos no domínio u.

Exemplo 1

$$T(u = modifiers) = \{low\_similar\ ,\ some\_similar\ ,\ high\_similar\}$$

Este conjunto de modificadores é adaptável à semântica que as consultas podem oferecer. O conjunto de modificadores pode ser extensível, de maneira que novos valores lingüísticos possam ser acrescentados ao domínio da variável e reajustados ao intervalo $[0, 1]$. Por exemplo, se uma aplicação resolve definir um novo valor quite_similar, pode redefinir a variável lingüística *modifiers* da seguinte maneira:

Exemplo 2

$$T(modifiers) = \{low\_similar\ ,\ some\_similar,\ quite\_similar\ ,\ high\_similar\}$$

Os exemplos anteriores mostram dois conjuntos de termos que podem representar a variável *modifiers*. O conjunto de termos pode ser redefinido a critério das aplicações ou dos usuários das aplicações. Uma consulta utilizando os modificadores é apresentada a seguir.

$$\sigma_{high\_similar\ shapesimilarto("tree.jpg") \wedge\ some\_similar\ texturesimilarto("leaves.jpg")}(\text{IMAGES})$$

Valores dos Modificadores

Os valores dos modificadores podem ser escolhidos de maneira conveniente, para selecionar determinados subconjuntos do universo. Os critérios utilizados para dar valor aos modificadores devem levar em consideração o valor semântico que eles acrescentam às consultas.

A abordagem escolhida para dar valor aos modificadores é definida pelas aplicações ou pelos usuários das aplicações. A seguir apresentamos duas propostas.

Abordagem 1: α-cut

Um critério que pode ser utilizado para definir o conjunto dos modificadores sobre o universo dos resultados é o procedimento α-cut, onde são definidos subconjuntos não-fuzzy do universo cujos elementos possuem o valor da função de pertinência maior ou igual aos valores α_i dos modificadores. Os valores dos modificadores podem ser definidos como:

Exemplo 1

$$T(u) = \begin{cases} low\_similar \;\; \alpha_1 = 0 \\ some\_similar \;\; \alpha_2 = 0.3 \\ high\_similar \;\; \alpha_3 = 0.7 \end{cases}$$

Para o segundo exemplo, os valores dos diferentes α_i podem ser redefinidos como:

Exemplo 2

$$T(u) = \begin{cases} low\_similar \;\; \alpha_1 = 0 \\ some\_similar \;\; \alpha_2 = 0.3 \\ quite\_similar \;\; \alpha_3 = 0.5 \\ high\_similar \;\; \alpha_4 = 0.8 \end{cases}$$

Abordagem 2: Intervalos

Os valores dos modificadores podem ser definidos como intervalos que formam subconjuntos não-fuzzy do universo cujos elementos possuem o valor da função de pertinência dentro do intervalo dos modificadores. Os valores dos modificadores podem ser definidos como:

Exemplo 1

$$T(u) = \begin{cases} low\_similar,\ \mu_U(img) \in [0..0.3] \\ some\_similar,\ \mu_U(img) \in [0.3..0.7] \\ high\_similar,\ \mu_U(img) \in [0.7, 1] \end{cases}$$

Para o segundo exemplo, os intervalos podem ser redefinidos como:

Exemplo 2

$$T(u) = \begin{cases} low\_similar,\ \mu_U(img) \in [0..0.3] \\ some\_similar,\ \mu_U(img) \in [0.3..0.5] \\ quite\_similar,\ \mu_U(img) \in [0.5, 0.8] \\ high\_similar,\ \mu_U(img) \in [0.8, 1] \end{cases}$$

A definição de intervalos para os valores dos modificadores é uma abordagem que se ajusta melhor no valor semântico que eles têm para as consultas. Vejamos a seguir, o significado do modificador some_similar para as duas abordagens.

$$\sigma_{some\_similar\ \ texturesimilarto(\text{"}leaves.jpg\text{"})}(\text{IMAGES})$$

O exemplo seleciona as imagens que sejam moderadamente similares na textura à imagem leaves.jpg. Na primeira abordagem, as imagens do resultado da consulta precisam ser similares à dada imagem a partir do $\alpha_2 = 0.3$ (intervalo de $[0.3, 1]$). Quer dizer que as imagens mesmo com uma alta similaridade (modificador high_similar) são selecionadas. Logo, o significado do modificador some_similar neste caso é o de selecionar as imagens cuja similaridade com a imagem exemplo seja major do que a média. A segunda abordagem restringe esses valores às imagens cuja similaridade esteja mesmo na média (intervalo de $[0.3, 0.7]$).

Desenvolvemos uma proposta para a abordagem dos modificadores como sub-intervalos na faixa [0,1]. A seguir descrevemos como os modificadores modificam o valor de pertinência de uma imagem no conjunto de resultados.

Mapeamento dos Modificadores nos Predicados das Consultas

Sejam $P_j,\ j = 1..m$, os m predicados de uma consulta. Para cada predicado P_j, há um subconjunto de resultados U_j.

A avaliação de um predicado fuzzy é feita na seguinte ordem. O primeiro passo corresponde à execução da operação de similaridade para as imagens no banco de dados, obtendo o conjunto de resultados parciais U_j.

$$U_j = \{< img_i, \mu_{U_j}(img_i) >\}$$

A seguir, deve ser feito um mapeamento dos valores da similaridade de cada resultado com o modificador correspondente. Nossa proposta é a seguinte: a função de pertinência de uma imagem reajusta o valor de pertinência das imagens em U_j, para tomar valores dentro do sub-intervalo do modificador específico ao predicado P_j.

Uma variável lingüística pode ser definida conforme a percepção do usuário. Definamos, então, a função de pertinência como dependendo dos valores de similaridade e os valores dos modificadores.

Dado o termo $T(u_k)$ do domínio da variável *modifiers*, $T(u_k)$ é definido como $[\alpha_0, \gamma_0] \subseteq [0,1]$, então,

$$\mu_{U_j}(img_i) = \begin{cases} 0, & se\ \mu_{U_j}(img_i) < \alpha_0\ \vee\ \mu_{U_j}(img_i) > \gamma_0 \\ f(\mu_{U_j}(img_i)), & se\ \alpha_0 \leq \mu_{Uj}(img_i) \leq \gamma_0 \end{cases}$$

Exemplo de Função de Pertinência para $T(u)$

Considere o seguinte exemplo. Dependendo do número de termos da variável lingüística *modifiers*, a função de pertinência é definida a seguir.

1. **Se o número de termos do domínio é 1**: a função de pertinência não altera os resultados da operação de similaridade, como mostra o gráfico da figura 3.2.

2. **Se o número de termos do domínio é 2**: quer dizer que temos dois sub-intervalos na faixa [0,1]. A função de pertinência é definida, a partir dos resultados da operação de similaridade, como:

   Intervalos:
   $termo_1 = [\alpha_0, \gamma_0]$,
   $termo_2 = [\alpha_1, \gamma_1]$,
   $\alpha_0 = 0, \gamma_1 = 1$

   Se o modificador é termo$_1$:

$$\mu_{U_j}(img_i) = \begin{cases} 0, & se\ \mu_{U_j}(img_i) > \gamma_0 \\ \frac{\gamma_0 - \mu_{U_j}(img_i)}{\gamma_0}, & se\ 0 \leq \mu_{Uj}(img_i) \leq \gamma_0 \end{cases}$$

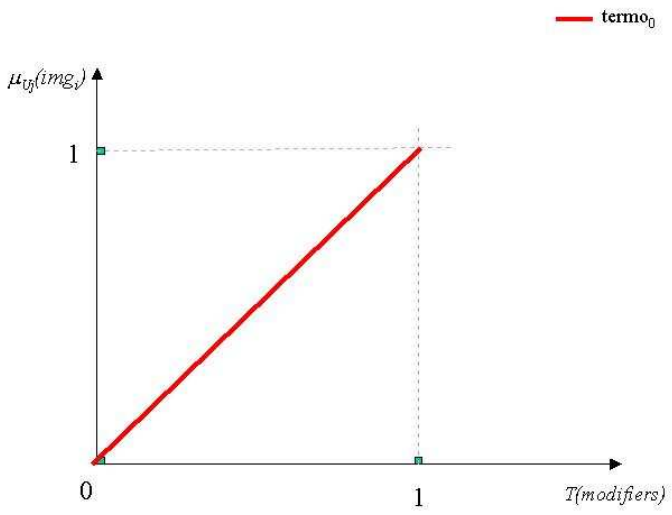


Figura 3.2: Gráfico da variável lingüística *modifiers* quando a dimensão do T(u) é 1.

Se o modificador é termo$_2$:

$$\mu_{U_j}(img_i) = \begin{cases} 0, & se\ \mu_{U_j}(img_i) < \alpha_1 \\ \frac{\mu_{U_j}(img_i) - \alpha_1}{1-\alpha_1}, & se\ \alpha_1 \leq \mu_{Uj}(img_i) \leq 1 \end{cases}$$

O gráfico da figura 3.3 mostra o comportamento dos modificadores dentro do intervalo dos termos. O modificador reajusta os valores da similaridade dentro do seu intervalo para devolver valores no intervalo [0,1].

3. **Se o número de termos do domínio é maior do que 3**: quer dizer que temos q sub-intervalos na faixa [0,1].

   Seja α_i e γ_i os valores mínimos e máximos dos sub-intervalos, tal que

   $$[\alpha_0 = 0, \gamma_0] \ldots [\alpha_k, \gamma_k]$$

   $$[\alpha_{k+1}, \gamma_{k+1}] \ldots [\alpha_q, \gamma_q = 1] \quad k = 1..q$$

   Definimos as seguintes restrições sobre as quotas dos sub-intervalos:

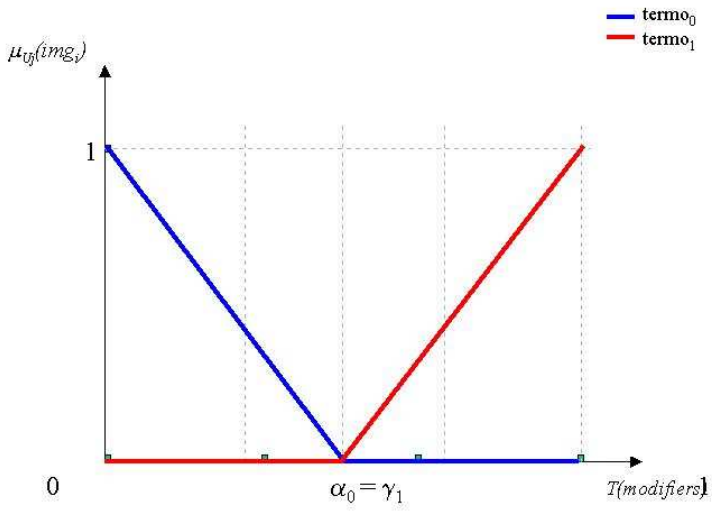


Figura 3.3: Gráfico da variável lingüística *modifiers* quando a dimensão do T(u) é 2.

$$\alpha_k < \gamma_k,\ \alpha_k < \alpha_{k+1},\ \gamma_k < \gamma_{k+1}$$

A função de pertinência é definida, a partir dos resultados da operação de similaridade, como:

Se o modificador é $termo_0$:

$$\mu_{U_j}(img_i) = \begin{cases} 0, & se\ \mu_{U_j}(img_i) > \gamma_0 \\ \frac{\gamma_0 - \mu_{U_j}(img_i)}{\gamma_0}, & se\ 0 \leq \mu_{Uj}(img_i) \leq \gamma_0 \end{cases}$$

Se os modificadores são: $termo_1$ **até** $termo_{q-1}$:

$$\mu_{U_j}(img_i) = \begin{cases} 0, & se\ \mu_{U_j}(img_i) < \alpha_k\ \vee\ \mu_{U_j}(img_i) > \gamma_k \\ \frac{\mu_{U_j}(img_i) - \alpha_k}{\frac{\gamma_k - \alpha_k}{2}}, & se\ \alpha_k \leq \mu_{Uj}(img_i) \leq \alpha_k + \frac{\gamma_k - \alpha_k}{2} \\ \frac{\gamma_k - \mu_{U_j}(img_i)}{\frac{\gamma_k - \alpha_k}{2}}, & se\ \alpha_k + \frac{\gamma_k - \alpha_k}{2} \leq \mu_{Uj}(img_i) \leq \gamma_k \end{cases}$$

Se o modificador é $termo_q$:

$$\mu_{U_j}(img_i) = \begin{cases} 0, & se\ \mu_{U_j}(img_i) < \alpha_q \\ \frac{\mu_{U_j}(img_i) - \alpha_q}{1-\alpha_q}, & se\ \alpha_q \leq \mu_{Uj}(img_i) \leq 1 \end{cases}$$

Exemplo do Mapeamento dos valores de similaridade na Função de Pertinência do Modificador

A seguir apresentamos um exemplo completo de como é feito o mapeamento dos valores de similaridade, a partir da função de pertinência acima explicada. Definimos a variável *modifiers* como:

$$T(u) = \begin{cases} low\_similar,\ \mu_U(img) \in\ [0..0.5] \\ some\_similar,\ \mu_U(img) \in\ [0.3..0.8] \\ high\_similar,\ \mu_U(img) \in\ [0.7, 1] \end{cases}$$

Observe como os intervalos podem ou não ser sobrepostos, dependendo da definição das aplicações. Então, a função de pertinência para os modificadores fica:

$\text{termo}_0 = low\_similar$

$$\mu_{U_j}(img_i) = \begin{cases} 0, & se\ \mu_{U_j}(img_i) > 0.5 \\ \frac{0.5 - \mu_{U_j}(img_i)}{0.5}, & se\ 0 \leq \mu_{Uj}(img_i) \leq 0.5 \end{cases}$$

$\text{termo}_1 = some\_similar$

$$\mu_{U_j}(img_i) = \begin{cases} 0, & se\ \mu_{U_j}(img_i) < 0.3\ \vee\ \mu_{U_j}(img_i) > 0.8 \\ \frac{\mu_{U_j}(img_i) - 0.3}{0.25}, & se\ 0.3 \leq \mu_{Uj}(img_i) \leq 0.55 \\ \frac{0.8 - \mu_{U_j}(img_i)}{0.25}, & se\ 0.55 \leq \mu_{Uj}(img_i) \leq 0.8 \end{cases}$$

$\text{termo}_2 = high\_similar$

$$\mu_{U_j}(img_i) = \begin{cases} 0, & se\ \mu_{U_j}(img_i) < 0.7 \\ \frac{\mu_{U_j}(img_i) - 0.7}{0.3}, & se\ 0.7 \leq \mu_{Uj}(img_i) \leq 1 \end{cases}$$

O gráfico da variável *modifiers* para esse exemplo é apresentado na figura 3.4.

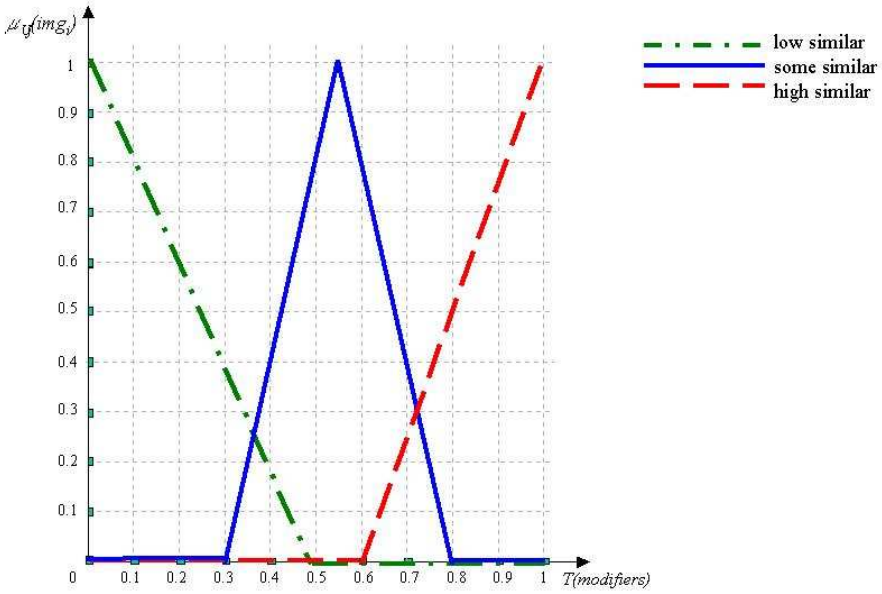


Figura 3.4: Gráfico da variável lingüística *modifiers* para um exemplo de três termos.

Os predicados da consulta são:

```
predicado1 = high_similar shapesimilarto("tree.jpg")
predicado2 = some_similar texturesimilarto("leaves.jpg")
```

e, U_1 e U_2 são os conjuntos fuzzy, resultados de executar as operações de similaridade.

$$U_1 = \{< I_1, 0.3 >, < I_2, 0.46 >, < I_3, 0.78 >, < I_4, 0.84 >\}$$
$$U_2 = \{< I_1, 0.45 >, < I_2, 0.35 >, < I_3, 0.67 >, < I_4, 0.2 >\}$$

Para o primeiro predicado, o modificador é *high_similar*. Então,

$$U_1 = \{< I_1, 0 >, < I_2, 0 >, < I_3, 0.27 >, < I_4, 0.47 >\}$$

Para o segundo predicado, o modificador é *some_similar*. O conjunto de resultados fica:

$$U_2 = \{< I_1, 0.6 >, < I_2, 0.2 >, < I_3, 0.52 >, < I_4, 0 >\}$$

A inclusão de novos termos ou a alteração da função de pertinência dos modificadores somente pode ser feita pela aplicação. A inclusão de novos termos pode implicar na redefinição da função de pertinência. O usuário escolhe os termos lingüísticos para os predicados. Os valores dos intervalos dos termos modificadores também podem ser redefinidos a critério do usuário. Ainda neste capítulo veremos como é feita a declaração e definição dos modificadores fuzzy pelas aplicações.

3.4.2.2 Resolução dos Operadores Lógicos

Seja U, o conjunto dos resultado de uma consulta, tal que U seja uma combinação dos conjuntos $\{U_j\}$. Os operadores lógicos AND e OR unem os predicados em um único critério final.

Os operadores lógicos fuzzy precisam ser definidos para poder combinar os predicados das consultas. Por exemplo, o operador AND pode ser a intersecção dos subconjuntos $\{U_j\}$, tal como mostra (a) e o operador OR pode ser a união dos subconjuntos $\{U_j\}$, tal como mostra (b).

(a) operador AND: $\mu_U(x) = min[\mu_{\{U_j\}}(x)]$

(b) operador OR: $\mu_U(x) = max[\mu_{\{U_j\}}(x)]$

De maneira geral, definimos os operadores lógicos AND e OR como funções cujos parâmetros são os valores de similaridade, μ_{U_j}, das imagens para os predicados P_j. Os predicados tradicionais são incluídos na função do operador. Os valores dos predicados tradicionais são interpretados como:

$$\mu_{U_j}(x) = \begin{cases} 0, & se\ falso \\ \\ 1, & se\ verdadeiro \end{cases}$$

A função para os operadores pode ser representada como:

$$\delta_{img_i} = f(\mu_{U_1}(img_i), \mu_{U_2}(img_i), ..., \mu_{U_n}(img_i))$$

A função f define o operador lógico que resolve a união dos predicados. O valor de δ_{img_i} representa a similaridade da combinação dos predicados para a imagem img_i. Na resolução da consulta anterior, onde $U = U_1\ AND\ U_2$, seria:

$$U_1\ AND\ U_2 = U_1 \bigcap U_2 = min[\mu_{\{U_j\}}]$$

$$\delta_{(I_1,I_2,I_3,I_4)} = f(\mu_{U_1}, \mu_{U_2}) = min[\mu_{\{U_j\}}]$$

$$U = \{< I_1, 0 >, < I_2, 0 >, < I_3, 0.27 >, < I_4, 0 >\}$$

Uma consulta utiliza uma função do operador lógico para resolver os predicados, mas para diferentes consultas podem ser utilizados diferentes funções, de maneira que para:

para Q_1 : o operador AND é definido como $f_1(\mu_{U_1}(x), \mu_{U_2}(x), ..., \mu_{U_n}(x))$

para Q_2 : o operador AND é definido como $f_2(\mu_{U_1}(x), \mu_{U_2}(x), ..., \mu_{U_n}(x))$

Desta maneira as aplicações podem ser capazes de fazer consultas com diferentes interpretações dos operadores lógicos, o que sugere a possibilidade de testes sobre os diferentes resultados. Ainda neste capítulo veremos como é feita a declaração dos operadores lógicos pelas aplicações.

O operador NOT pode ser definido como o modificador low_similar em uma consulta. A seguir é proposto um mecanismo de retroalimentação das consultas.

3.4.3 Mecanismo de Retroalimentação das Consultas

Todo sistema que permite fazer consultas deve oferecer alguma técnica para refinar os resultados. O mecanismo de retroalimentação leva em consideração as interpretações das imagens pelos diferentes usuários. Um conjunto de imagens resultante de uma recuperação pode mudar de um usuário a outro para uma dada consulta. O sistema pode solicitar aos usuários o refinamento iterativo da consulta desde o resultado inicial e assim, aprender incrementalmente a perspectiva de um usuário para conseguir melhores resultados individuais.

No processo de recuperação de imagens, é muito importante que cada consulta possa ser refinada em busca de resultados ótimos. O mecanismo de retroalimentação está sendo muito utilizado por muitos sistemas na recuperação de imagens. A seguir propomos uma abordagem para refinar as consultas das aplicações. O mecanismo mantém o caráter de generalidade que levamos em consideração ao longo do desenvolvimento do nosso trabalho.

Nossa abordagem

O objetivo do mecanismo de retroalimentação é atribuir flexibilidade ao sistema para que os usuários possam refinar suas consultas, representando sua própria interpretação. O mecanismo de retroalimentação é baseado no modelo fuzzy de consultas e no modelo de recuperação de imagens.

As imagens do resultados de uma consulta são envolvidas no processo de refinamento do próximo passo. O usuário pode escolher um conjunto de imagens. Para cada imagem, o usuário pode atribuir os descritores com seus critérios de similaridade, redefinir os valores dos modificadores fuzzy e as funções dos operadores lógicos. Adicionalmente, o usuário pode redefinir os critérios sobre os predicados da consulta prévia. A partir dessas modificações, o sistema compõe a nova consulta com os novos predicados.

No processo de recuperação da consulta, os seguintes passos são repetidos até que os resultados satisfaçam os usuários:

1. Seja uma consulta inicial, Q_0, e U_0 as imagens resultado de Q_0. Definimos:

$Q_k = \sigma_{P_k}(colecao)$, consulta corrente no processo de recuperação

$P_k = \{p_{kj}\},\ j = 1..m$, conjunto de predicados para a consulta Q_k

$p_{kj} =< modificador_j > s_j,\ \ s_j \in S$, esquema do predicado fuzzy p_{kj}

$U_k = \{img_i\},\ i = 1..n$, o conjunto de imagens resultado da consulta Q_k, n é o número de imagens no banco de dados.

2. O usuário:

   (a) escolhe as imagens em U_k úteis para o próximo passo de refinamento,

   (b) define os critérios sobre elas:

   Seja $U'_k,\ U'_k \subseteq U_k$, o conjunto de imagens escolhidas pelo usuário.

   i. Para toda img_i que pertence ao conjunto U'_k, o usuário define os descritores envolvidos na redefinição da consulta, e portanto, o critério de similaridade.

   ii. O usuário ajusta os modificadores fuzzy para os predicados de similaridade, ou seja, pode escolher os termos lingüísticos para os predicados. Os valores dos intervalos dos termos modificadores também podem ser redefinidos a critério do usuário.

   (c) Se é de interesse do usuário, cada predicado $p_{kj} \in P_k$ pode ser redefinido, aplicando os passos em 2.(b).

   $p_{k+1j} =< modificador'_j > s'_j$

   (d) O usuário configura a junção dos predicados, escolhendo os operadores lógicos. A função dos operadores pode ser redefinida para a nova consulta.

   (e) Se nenhuma imagem satisfaz a busca, o usuário deve fazer uma nova consulta que redefina os critérios de busca.

3. Todos os predicados configuram a consulta Q_{k+1}

   $p_{k+1\,j} =< modificador_j > s_j\ j = 1..m'$

4. A nova consulta, Q_{k+1}, é processada, começando novamente o processo de refinamento no passo 2.

Neste processo de refinamento, o sistema verifica se os resultados de cada fase coincidem. Se isto ocorrer, a retroalimentação não está encaminhando resultados melhores. Portanto, o refinamento deve satisfazer que:

$$U_{k+1} - U_k \neq \{\emptyset\}$$

3.4.3.1 Exemplo

Seja Q_0

$$Q_0 = \sigma_{some\_similar\ shapesimilarto("tree.jpg") \wedge\ some\_similar\ texturesimilarto("leaves.jpg")}(\text{IMAGES})$$

$$p_{01} = some\_similar\ shapesimilarto("tree.jpg")$$

$$p_{02} = some\_similar\ texturesimilarto("leaves.jpg")$$

$$U_0 = \{img_1, img_2, img_3, img_4, img_5\},\ n = 5$$

2.(a) O usuário escolhe as imagens que vão ser utilizadas na retroalimentação. Por exemplo:

$$U_0' = \{img_1, img_4, img_5\}$$

2.(b) Para cada imagem em U_0', o usuário define os critérios de busca escolhendo descritores e modificadores:

p_{12}: high_similar shapesimilarto(img_1)

p_{13}: high_similar colorsimilarto(img_4)

p_{14}: some_similar texturesimilarto(img_5)

2.(c) Cada predicado da consulta inicial Q_0 é analisado:

```
p01:  some_similar shapesimilarto("tree.jpg")  ->
p11: high_similar shapesimilarto("tree.jpg")

p02:  some_similar texturesimilarto("leaves.jpg") -> descartado
```

2.(d) Junção dos predicados:

$$p_{11}\ AND\ p_{12}\ AND\ p_{13}\ AND\ p_{14}$$

O usuário redefine a função dos operadores lógicos como sendo:

```
AND -> AND_fuzzy_min
OR  -> OR_fuzzy_max
```

3. Logo, Q_1 é:

$$\sigma_{high\_similar\ shapesimilarto("tree.jpg") \wedge\ high\_similar\ shapesimilarto(img_1)\ \wedge\ high\_similar\ colorsimilarto(img_4)\ \wedge\ high\_similar\ texturesimilarto(img_5)}(\text{IMAGES})$$

3.5 Modelo de Manipulação de Imagens sobre Banco de Dados

O MMI permite aos usuários manipular um banco de dados. A manipulação (inserção, modificação e remoção) de imagens do banco de dados é feita, basicamente, da maneira tradicional. Entretanto, a declaração da tupla de operações facilita a manipulação das imagens sobre o banco de dados.

Uma aplicação pode definir a seguinte tupla de operações:

$$id_1 = \{< op_{extrdesc_1}, op_{similar_1}, op_{norma_1}, op_{index_1} >\}$$

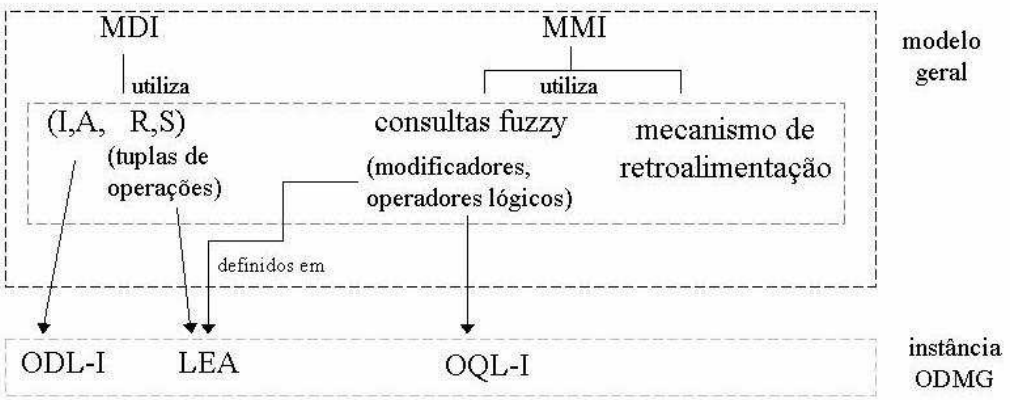
Quando uma solicitação da aplicação faz referência a um identificador (id_1), ocorre:

- *na inserção de imagens*: o algoritmo de extração de descritores, $op_{extrdesc_1}$, é automaticamente executado para essa imagem e ela é incluída no índice. (insert(img,id_1))
- *nas consultas*: os algoritmos de similaridade e normalização (se houver), $op_{similar_1}$, op_{norma_1}, e o método de busca para op_{index_1} são utilizados para processar a consulta.
- *na criação do banco de dados*: a estrutura de indexação é automaticamente criada (se houver), utilizando o método correspondente para op_{index_1}.

A tupla de operações tem o propósito de ajudar na manipulação e recuperação das imagens para as aplicações. Os algoritmos das imagens são executados automaticamente segundo a solicitação.

Neste trabalho não aprofundamos na pesquisa da especificação de um modelo para manipular as imagens sobre os bancos de dados, mas sim em um modelo de recuperação das imagens. Nesta seção somente propomos uma maneira de processar as solicitações de imagens para as aplicações.

O esquema mostra os relacionamentos das seções para melhor entendimento do modelo. Até este ponto do capítulo, definimos um modelo geral de gerenciamento para imagens (quadro em preto). Na próxima seção, vamos definir uma instância desse modelo baseada no padrão ODMG (quadro em cinza).



3.6 Modelo baseado no Padrão ODMG

Como caso específico do modelo de gerenciamento de bancos de dados de imagens, nesta seção apresentamos uma abordagem baseada no padrão de dados ODMG.

O objetivo na utilização do ODMG é padronizar a manipulação de bancos de dados e ampliar o grau de portabilidade das aplicações. Todas as aplicações baseadas nesse padrão poderão ser desenvolvidas utilizando nosso modelo.

Utilizando o padrão ODMG, podemos concretizar o modelo das imagens, a declaração e manipulação dos bancos de dados de imagens, assim como a linguagem de consulta.

A figura 3.5 mostra o esquema do gerenciamento utilizando a abordagem do padrão ODMG. O Módulo de Definição de Imagens (MDI) utiliza a ODL-I para fazer as declarações de BDs. O modelo de objetos do ODMG foi estendido para suportar os conceitos fuzzy relacionados com as imagens. A ODL-I é, essencialmente, a mesma ODL do ODMG mas utiliza o nosso modelo de objeto. Definimos a Linguagem de Especificação das Aplicações (LEA) para especificar as particularidades das aplicações sobre o esquema de bancos de dados.

O Módulo de Manipulação de Imagens (MMI) é encarregado da recuperação e manipulação dos dados no banco de dados. A linguagem de consulta OQL-I é a OQL estendida para suportar nosso modelo fuzzy de consultas. Propomos um conjunto de operadores e definimos a possibilidade de criar conceitos.

3.6.1 Exemplo

O seguinte banco de dados vai ser utilizado para exemplificar a definição e manipulação de imagens com o padrão ODMG.

Banco de dados: Cadastro de Pessoas e Assinaturas para Bancos

Descrição: a aplicação cadastra as pessoas e o arquivo digital de suas assinaturas em um banco de dados. O objetivo principal é verificar se a assinatura de uma pessoa corresponde

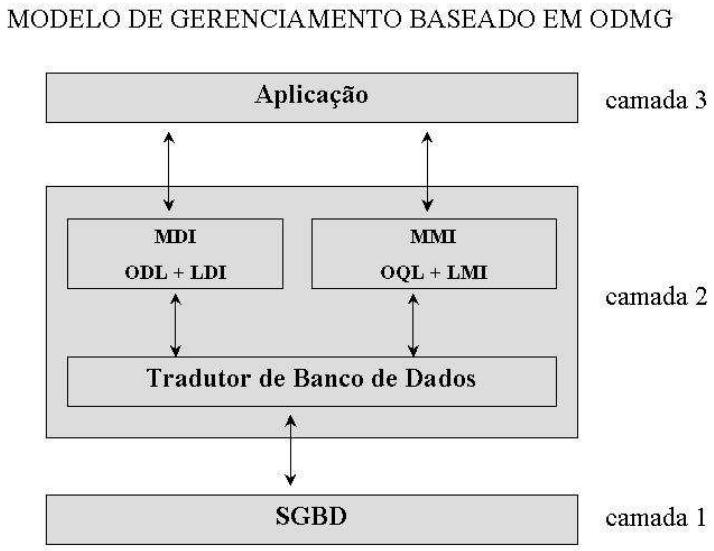


Figura 3.5: Modelo em camadas baseado no padrão ODMG.

com a armazenada no banco de dados quando um usuário faz uma transação. As operações correspondentes para reconhecer assinaturas similares são desenvolvidas para a aplicação.

Coleções: Pessoa, Assinatura, Foto, Conta_Corrente

O esquema da figura 3.6 mostra o relacionamento entre as coleções do banco de dados.

Cada coleção tem as seguintes propriedades.

Propriedades: Pessoas
nome
endereço (rua, número, bairro, cidade, cep)
identidade
CPF
renda
conta_corrente
assinatura
foto

Propriedades: Conta_Corrente
numero_conta
limite_credito
saldo
numero_cartao

Propriedades: Assinaturas
assinatura
reconhecimento de assinatura

Propriedades: Foto
foto
reconhecimento de Face

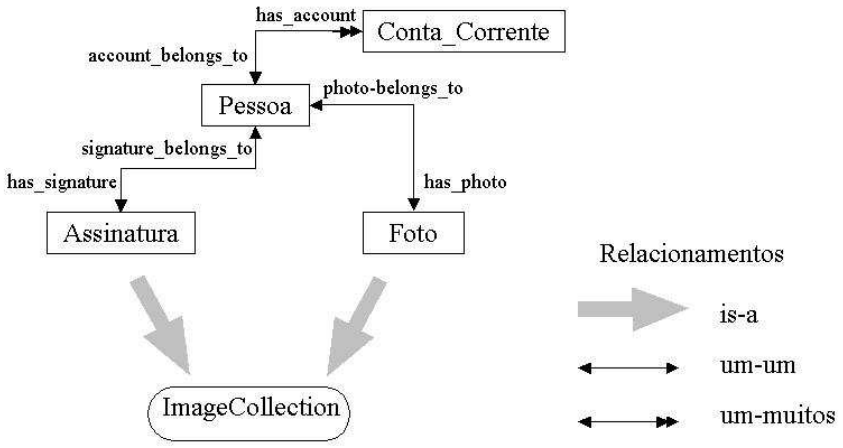


Figura 3.6: Exemplo: Banco de dados de Cadastro de Pessoas e Assinaturas para Bancos.

3.6.2 Modelo de Objetos para Imagens

O modelo de objetos da ODMG oferece objetos e literais como primitivas de modelagem. Os literais podem ser atômicos (Long, Float, Octet, ...), coleção (Set, Bag, List e Array) ou estruturados (Date, Time, Timestamp, ...). Os objetos podem ser atômicos ou coleção.

O modelo de objetos foi estendido para suportar novos tipos de dados relacionados com as imagens e o modelo de consultas fuzzy.

- FuzzyDT: um literal Float que toma valores na faixa [0..1]. Nas consultas, as operações de similaridade retornam um valor FuzzyDT.

- FuzzySet, FuzzyBag, FuzzyList: são coleções fuzzy que representam os resultados de uma consulta.

  O resultado de uma consulta OQL (select-from-where) sempre vai ser uma coleção. Dependendo da consulta, ela será um set, bag ou list. Uma coleção fuzzy pode ser definida como uma coleção, onde cada imagem tem associado o valor da similaridade, assim como no nosso modelo de consultas, os resultados são pares $< img, \delta >$.

  Se, na consulta, usa-se a palavra chave distinct, obtemos um FuzzySet, em outro caso, obtemos um FuzzyBag. Se a cláusula order-by é usada, obtemos uma FuzzyList.

Também definimos uma interface ImageCollection. Esta interface possui um atributo do tipo DigitalImage. Este atributo armazena a imagem como objeto binário. A definição de outros elementos na interface pode causar perda da generalidade do modelo. As classes que sejam coleções de imagens herdam desta interface.

```
typedef blob DigitalImage;

interface ImageCollection {
(extent images)

    attribute DigitalImage imageraw;

};
```

3.6.3 Declaração dos Bancos de Dados de Imagens

A declaração dos bancos de dados das imagens é definida a partir de duas linguagens de especificação: ODL-I e LEA. Um banco de dados é baseado em um esquema que é definido na ODL-I. A Linguagem de Declaração de Imagens (LEA) é desenvolvida como parte de nosso modelo. Ela define as particularidades das aplicações sobre o esquema de bancos de dados.

3.6.3.1 ODL-I

A ODL-I é, essencialmente, a mesma ODL do ODMG mas utiliza o nosso modelo de objeto. A linguagem ODL-I define o esquema do banco de dados com as imagens, atributos e relacionamentos. Os atributos definidos para as coleções de imagens são as características não visuais das imagens, referentes ao conjunto A do modelo.

As coleções de imagens herdam da interface ImageCollection. O conjunto I, das representações visuais de uma imagem, é instanciado pelo atributo do tipo DigitalImage de ImageCollection.

Uma breve descrição da gramática da ODL é apresentada a seguir. A sintaxe completa pode ser encontrada em [8].

```
<specification>::= <definition> | <definition> <specification>

<definition>::= <type_dcl>; | <const_dcl>; | <except_dcl>; |
                <interface>; | <module>; | <class>;

<module>::= module <identifier> {<specification>}

<class>::= <class_dcl>

<class_dcl>::= <class_header> {<interface_body>}

<class_header>::= class <identifier>
                    [extends <scoped_name>]
                    [<inheritance_spec>]
                    [<type_property_list>]
```

```
<inheritance_spec>::= : <scoped_name> [, <inheritance_spec>]

<interface_body>::= <export> | <export> <interface_body>

<export>::= <type_dcl>; | <const_dcl>; | <except_dcl>; |
            <attr_dcl>; | <rel_dcl>; | <op_dcl>;

<attr_dcl>::= [readonly] attribute <domain_type> <attr_list>

<rel_dcl>::= relationship <target_of_path> <identifier>
             inverse <inverse_traversal_path>

...
```

A declaração do esquema de banco de dados do exemplo é como segue.

```
module DBBankExample {

        struct Address {string street, string complement,
                        string CEP, string city, string state;};

        class Person
        ( extent persons
          key RG         )
        {
                attribute string RG;
                attribute string name;
                attribute string CPF;
                attribute Address address;
                attribute float salary;

                relationship set<BankAccount> has_account
                        inverse BankAccount::account_belongs_to;
                relationship ImgPhoto has_photo
                        inverse ImgPhoto::photo_belongs_to;
                relationship ImgSignature has_signature
                        inverse ImgSignature::signature_belongs_to;

                void move(in string new_address);
        };
```

```
        class BankAccount
        (extent accounts)
        {
                attribute string account;
                attribute double credit_limit;
                attribute double remainder;
                attribute set<double> transactions;
                attribute string card_number;

                relationship Person account_belongs_to
                        inverse Person::has_account
        };

        class ImgSignature : ImageCollection
        (extent signatures)
        {
                attribute string imgname;

                relationship Person signature_belongs_to
                        inverse Person::has_signature

        };

        class ImgPhoto : ImageCollection
        (extent photos)
        {
                attribute date photoshoot;
                attribute string imgname;

                relationship Person photo_belongs_to
                       inverse Person::has_photo
        };
};
```

3.6.3.2 Linguagem de Especificação das Aplicações

A Linguagem de Especificação das Aplicações (LEA) é uma linguagem de especificação desenvolvida como parte de nosso modelo. Ela foi projetada para definir as particularidades das aplicações sobre o esquema de bancos de dados.

A LEA define as tuplas de operações, os modificadores fuzzy e os operadores lógicos fuzzy. Os conjuntos R e S do modelo de recuperação de imagens são representados com as tuplas de operações.

Os modificadores, os operadores lógicos, as operações de similaridade e os índices são uti-

lizados nas consultas OQL-I. A linguagem OQL-I é descrita na próxima seção.

A gramática da LEA é descrita a seguir.

```
<section>::= <fuzzy_modifiers> | <fuzzy_logical_operators> |
             <image_operation_tuples>



<fuzzy_modifiers>::= "[fuzzy_modifiers]" <fm_body>

<fm_body>::= modifier is <identifier> values <modifier_term_list>

<modifier_term_list>::= <identifier> as <expression>
                                      modifyingBy <function_identifier>
                        [; <modifier_term_list>]

<expression>::= minvalue=<float_value>, maxvalue=<float_value>



<logical_operators>::= "[logical_operators]" <flo_body>

<flo_body>::= operator <identifier> is <operator_identifier> |
              operator <identifier> is <boolean_logical_operator>
              [, <flo_body>]

<boolean_logical_operator>::= BOOLEAN_AND | BOOLEAN_OR |
                              BOOLEAN_NOT



<image_operation_tuples>::= "[image_operation_tuples]" <imt_body>

<imt_body>::= image_operation_tuple is <identifier>
              as <image_operation_list>
              for <image_collections> [; <imt_body>]

<image_operation_list>::= descriptor_extraction is <operation_identifier>,
                          similarity is <operation_identifier>,
                          [normalization is <operation_identifier>],
                          [indexing is <operation_identifier>]

<image_collections>::= <image_collection> [, <image_collections>]
```

Um arquivo de especificação das aplicações contém três partes principais: a especificação dos modificadores fuzzy, a especificação dos operadores lógicos e a especificação das tuplas de operações.

A seguir vamos declarar as particularidades da aplicação sobre o bancos de dados DBBankExample.

Modificadores Fuzzy

Os modificadores tomam valores no intervalo $[0, 1]$. A aplicação definiu um modificador fuzzy MODIFIER1 com três termos e seus correspondentes valores em intervalos.

A cláusula <function_identifier> especifica a função de pertinência do termo modificador, utilizando um identificador. A criação do formato dos identificadores das funções de pertinência é delegada aos desenvolvedores do modelo, que podem criar um repositório com as funções de pertinência.

```
[fuzzy_modifiers]

modifier is MODIFIER1 values
    low_similar  as minvalue=0, maxvalue=0.3
                 modifyingBy F11;
    some_similar as minvalue=0.3, maxvalue=0.7
                 modifyingBy F12;
    high_similar as minvalue=0.7, maxvalue=1
                 modifyingBy F13;
```

Operadores Lógicos

Os operadores lógicos podem ser algoritmos ou o operador tradicional booleano. Uma aplicação pode definir um ou vários operadores lógicos.

A cláusula <operator_identifier> é o identificador único dos algoritmos. A criação do formato dos identificadores dos algoritmos é delegada aos desenvolvedores do modelo, que podem criar um repositório com as implementações dos operadores lógicos.

A aplicação definiu os operadores booleanos tradicionais e dois algoritmos para funcionar como operadores lógicos.

```
[logical_operators]

operator AND_fuzzy_min is LOG_OP_UID1,
operator AND is AND_BOOLEAN,
operator OR_fuzzy_max is LOG_OP_UID2,
```

```
operator OR is OR_BOOLEAN
```

Tuplas de Operações

Uma aplicação pode definir uma ou várias tuplas de operações. A análise de cada coleção de imagens no banco de dados pode ser feita por uma tupla de operações diferente. Uma coleção pode ter associadas várias tuplas de operações. Isto garante uma parte da flexibilidade do modelo.

A cláusula <image_collections> especifica as coleções de imagens para as quais a tupla está sendo definida.

A cláusula <operation_identifier> é o identificador único para cada algoritmo. Uma implementação do modelo pode criar um repositório com os diferentes algoritmos que compõem as tuplas de operações. A criação do formato dos identificadores dos algoritmos é tarefa das implementações do modelo.

Duas tuplas de operações com identificadores de tuplas sign e face-match foram indicadas para as coleções ImgSignature e ImgPhoto, respectivamente. A maneira de exemplo, nas tuplas de operações foram colocadas as interfaces dos algoritmos, mas na linguagem de especificação é necessário colocar um identificador único que destaque o algoritmo. As tuplas de operações podem ser definidas como:

```
sign = {<float signProcess(out CI),
        FuzzyDT detectSignMatch(in ImgAssinatura assinatura),
        null, null>}

face-match = {<float faceExtract(out faceCI),
               FuzzyDT facesimilarto(in ImgPhoto photo),
               null, null>}
```

A especificação das tuplas do exemplo é como a seguir.

```
[image_operation_tuples]

image_operation_tuple is sign as
        descriptor_extraction is IMG_OP_UID1,
        similarity is IMG_OP_UID2
    for ImgSignature;

image_operation_tuple is face-match as
        descriptor_extraction is IMG_OP_UID3,
        similarity is IMG_OP_UID4
```

```
for ImgPhoto
```

3.6.4 Linguagem de Consulta: OQL-I

Nesta seção descrevemos a linguagem de consulta. OQL-I suporta nosso modelo fuzzy de consultas. Propomos um conjunto de operadores que suportam as consultas primitivas para imagens. OQL-I tem uma instrução básica para as consultas:

query ::= **select** [distinct] projection_attributes
from query [[as] identifier] , query [[as] identifier]
where query]
group by partition_attributes]
order by sort_criterion , sort_criterion]

onde os atributos de projeção (*projection_attributes*) são uma lista dos nomes dos atributos cujos valores serão recuperados na consulta. Na cláusula **from** delimitam-se os conjuntos de objetos ou consultas (*query*) que vão ser questionados. Na cláusula *where*, a consulta é uma expressão condicional de busca que identifica os objetos a serem recuperados. As cláusulas **group by** e **order by** têm a mesma semântica que em SQL.

O nosso trabalho estende esta linguagem. A maioria das extensões que introduzimos à linguagem foram feitas na cláusula **where** com novos operadores: modificadores fuzzy, predicados de relacionamentos espaciais e o predicado *contains* (*contains-predicate*). A definição de conceitos é outra característica de nosso modelo para acrescentar semântica às solicitações.

3.6.4.1 Operadores de Similaridade

Nas consultas, a chamada aos operadores de comparação é feita utilizando o identificador da tupla com os parâmetros da operação de similaridade correspondente.

A nova solução oferece uma uniformidade na linguagem e faz com que seja no nível de usuário final. Os usuários fazem referência aos identificadores que devem ser palavras relacionadas e fáceis de lembrar. Uma consulta OQL-I pode ser:

```
Consulta 1:  SELECT DISTINCT I FROM I Assinaturas
             WHERE  some_similar I.sign("joao.jpg");
```

O exemplo busca as assinaturas no banco de dados que sejam medianamente similares com a assinatura de João em "joao.jpg"(utilizando o modificador fuzzy). sign é o identificador de uma tupla de operações em *ImgAssinatura* e a operação correspondente, detectSignMatch, vai ser executada para processar a consulta.

3.6.4.2 Relacionamentos Espaciais dos Objetos

Incluímos um conjunto de operadores e funções que expressam, nas consultas, a maioria dos relacionamentos espaciais entre objetos ou regiões de uma imagem. O conjunto de operadores

espaciais foi tomado da linguagem MOQL [46] e é apresentado no anexo C.

Uma vez definidas as primitivas de relacionamentos espaciais, operadores como between podem ser definidos também a partir delas, como:

$$< obj_1 > \ between \ < obj_2 \mid obj_3 > =$$

$$< obj_1 > \ left \ < obj_2 > \ AND \ < obj_1 > \ right \ < obj_3 > \ OR$$

$$< obj_1 > \ left \ < obj_3 > \ AND \ < obj_1 > \ right \ < obj_2 >$$

3.6.4.3 Outros Operadores

Outros operadores são incorporados para melhor atender as necessidades dos usuários nas solicitações. O predicado *contains* foi tomado da linguagem MOQL.

Operador	Descrição
$< img_1 \mid obj_1 > contains < obj_2 >$	*contains-predicate* analisa se o obj_2 está em $img_1 \mid obj_1$ ou não.
[n]*select...*	Limita o número de resultados a ser recuperados a n. Ex.: [20]*select...*, mostra as primeiras vinte imagens similares.

3.6.4.4 Definição de Conceitos

Conceitos lógicos com valor semântico podem ser formados a partir dos atributos tradicionais e os descritores das imagens nos diferentes níveis de abstração. Esses conceitos são armazenados persistentemente em um catálogo de conceitos.

A concepção de um catálogo de conceitos tem como objetivo acrescentar o valor semântico às consultas, de maneira que os conceitos possam capturar a subjetividade dos diferentes usuários.

Um conceito é criado utilizando a definição de consultas nomeadas da OQL, que pode definir, também, operadores e funções.

define_query ::= define identifier as query

Por exemplo, podemos definir:

```
Ex.1: DEFINE girafa AS
        SELECT DISTINCT I FROM I Images
        WHERE high_similar I.texture_match("giraffe.gif")

Ex.2: DEFINE bluesky AS
        SELECT DISTINCT I FROM I Images
        WHERE high_similar I.region.color_match("blue") AND
              I.region north I
```

Para utilizar os conceitos, o usuário somente precisa referenciá-lo nas consultas.

```
Ex.3: girafa;

Ex.4: SELECT DISTINCT I FROM I Images, I1 Images.region
          WHERE bluesky AND
            high_similar I1.color_match("green") AND
            I1 south I
```

Para nosso exemplo em 3.6.1, podemos declarar a função assinatura_coincidente para referenciar nas consultas. A função, dado um número de conta e uma assinatura, verifica a assinatura do dono da conta:

```
Ex.5: DEFINE matching_signature(string acc,ImgSignature is) AS
         SELECT DISTINCT P.name FROM P Person, A P.has, C P.has_account
         WHERE C.account = acc AND high_similar A.sign(is)
```

O operador sign é o identificador associado à operação detectSignMatch.

3.6.5 Resolução de Primitivas de Consultas

Na seção 2.4.5 é descrito um conjunto de consultas consideradas as primitivas do amplo estoque de consultas sobre as imagens. Cada uma destas primitivas são resolvidas utilizando OQL-I e os nossos novos operadores.

- Consulta simples a uma característica visual. Por exemplo:

```
Recupera as imagens com 25% de vermelho e 40% de azul.

Consulta 1:    SELECT DISTINCT I FROM I Images
               WHERE  some_similar I.color_match("red") AND
                 some_similar I.color_match("blue")
```

- Consulta com uma combinação de características. Por exemplo:

```
Recupera as imagens com 25% de vermelho e 40% de uma textura de árvore.

Consulta 2:  SELECT DISTINCT I FROM I Images
             WHERE  some_similar I.color_match("red") AND
                  some_similar I.texture_match("arvore.gif")
```

- Consulta de característica com localização. Por exemplo:

```
Recupera as imagens com céu azul na metade superior e verde na
metade inferior.

Consulta 3:   SELECT DISTINCT I FROM I Images, I1 Images.region
              WHERE bluesky AND
                 high_similar I1.color_match("green") AND
                 I1 south I
```

- Consulta por exemplo (*Query by Example* - QBE). Por exemplo:

```
Recupera as imagens que contêm texturas similares a esta
("arvore.gif").

Consulta 4:   SELECT DISTINCT I FROM I Images
              WHERE  high_similar I.texture_match("arvore.gif")
```

- Objeto vs. Imagem. Por exemplo:

```
Recupera as imagens que contêm um carro azul localizado no centro.

Consulta 5:   SELECT DISTINCT I FROM I Images, C Carros
              WHERE I contains C AND  high_similar C.color_match("blue") AND
                 C.region(I).centroid  centroid  I
```

- Consultas utilizando os atributos definidos pelo usuário. Por exemplo:

```
Recupera as imagens onde a localização seja São Paulo e a data
junho de 1972.

Consulta 6:   SELECT DISTINCT I FROM I Images
              WHERE   I.location = "SAO PAULO" AND
                I.date.month = "06" AND I.date.year = "1972"
```

- Consultas utilizando objetos, seus atributos e relacionamentos entre objetos. Por exemplo:

```
Recupera as imagens onde há um homem e uma criança à sua direita.

Consulta 7:   SELECT DISTINCT I
              FROM I Images, P1 Persons, P2 Persons
              WHERE   I contains P1 AND I contains P2 AND
                P1.region(I) left P2.region(I) AND
                P1.age > 16 AND p2.age < 12
```

- Consultas por conceito. Por exemplo: o conceito *céu* é definido como a cor azul predominante na metade superior da imagem, já definido nesta seção como:

```
DEFINE bluesky AS SELECT DISTINCT I FROM I Images
WHERE  some_similar I.region.color_match("blue") AND
       I.region north I
```

```
Consulta 8:   bluesky;
```

3.7 Sumário

Este capítulo apresentou um modelo para gerenciamento de bancos de dados de imagens. O gerenciamento de imagem se define como uma camada intermediária entre as aplicações e os sistemas de bancos de dados. O modelo consta de um modelo do dado imagem, um modelo de recuperação de imagens, um modelo fuzzy de consultas e um mecanismo de retroalimentação.

Modelos das Imagens e de Recuperação de Imagens

Podemos destacar as propriedades dos modelos de imagens e de recuperação:

- Suporte de múltiplos descritores para as características das imagens.
- Os atributos não semânticos são representados no modelo e junto aos descritores descrevem as imagens.
- Suporte à similaridade entre as imagens.

O modelo não depende de um domínio específico de aplicações. Cada aplicação pode escolher representar as características que se ajustam a seus requisitos. O modelo de imagens proposto é baseado nas descrições do padrão MPEG-7.

Modelo baseado no padrão ODMG

O modelo, baseado no padrão ODMG, faz a declaração dos bancos de dados utilizando as linguagens ODL-I e LEA. Um banco de dados é baseado em um esquema definido na ODL-I. Na LEA, definimos as particularidades das aplicações sobre o esquema de bancos de dados. Novos tipos de dados foram definidos no modelo de objetos para suportar o modelo de consultas fuzzy.

A linguagem de consulta OQL é estendida, definindo novos operadores.

Construímos a sentença OQL-I de cada primitiva de consulta para imagens, o que demonstra que a linguagem de consulta suporta uma ampla variedade de consultas para imagens. A figura 3.7 mostra as diferentes classes de consultas para imagens. Conceitos de alto nível de semântica podem ser formados a partir dos atributos tradicionais e os descritores das imagens nos diferentes níveis de abstração.

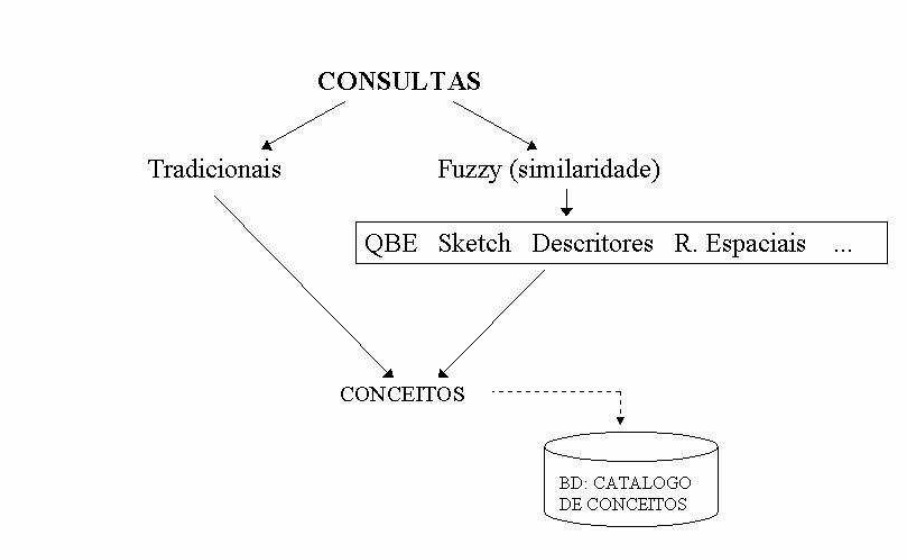


Figura 3.7: Diferentes classes de consultas sobre as imagens.

Características relevantes do Modelo de Gerenciamento

Nosso modelo para o gerenciamento de bancos de dados de imagens oferece flexibilidade e generalidade suficiente para suportar a diversificação de domínios e aplicações, e para lidar com a subjetividade dos usuários. Podemos salientar as características mais importantes do modelo:

- Modelagem das imagens: as imagens, suas propriedades, e semântica foram modeladas, baseado no padrão MPEG-7.
- Modelo para a recuperação de imagens: representa a similaridade das imagens nas consultas.
- Tuplas de operações: simplifica a manipulação dos algoritmos para as aplicações e os usuários.
- Modelagem fuzzy das consultas: define um modelo fuzzy para consultar imagens, expressando melhor a natureza inexata das imagens.

  Define os modificadores fuzzy que qualificam os predicados da consulta;

  Define os operadores lógicos fuzzy.
- Grande estoque de consultas: oferece um amplo estoque de consultas, tentando atingir todas as possíveis consultas sobre imagens:

  as consultas podem definir relacionamentos espaciais e conceitos;

  as consultas integram informação semântica, não semântica e abstração de informação em vários níveis.

- Mecanismo de retroalimentação nas consultas: é simples, flexível, mantém o caráter de generalidade do nosso sistema e preocupa-se por levar em consideração a subjetividade dos usuários.

O texto foi complementado com exemplos ao longo do capítulo.

Capítulo 4

AMID: Sistema para o Gerenciamento de Aplicações de Bancos de Dados de Imagens

4.1 Introdução

Neste capítulo descrevemos, AMID (*Architecture for Management of Image Databases*), nosso sistema para o gerenciamento de imagens sobre bancos de dados. O sistema é baseado no modelo de gerenciamento proposto no capítulo anterior.

4.2 Arquitetura Geral do Sistema Amid

Esta seção apresenta a arquitetura baseada no esquema de gerenciamento de imagens.

O esquema da figura 4.1 mostra os seis módulos da arquitetura. Cada módulo é explicado a seguir.

1. Interface de AMID com as aplicações. Aceita as solicitações enviadas pelas aplicações e retorna os resultados.

2. Processador de solicitações. Analisa as operações enviadas, sejam de declaração de bancos de dados, inserção, modificação, exclusão ou recuperação de imagens. Faz o processamento utilizando os algoritmos e métodos de acesso, comunicando-se com o banco de dados através do Tradutor e retorna uma resposta para a aplicação.

3. Serviços. As funcionalidades envolvidas no gerenciamento de imagens podem ser integradas ao sistema como serviços. A idéia é ter um conjunto de serviços orientados à manipulação de imagens. Por exemplo, um serviço integrado é o repositório de algoritmos e a manipulação do mesmo.

4. Repositório de Algoritmos. Integra as diversas operações sobre imagens, de maneira que elas estejam disponíveis para ser consultadas e utilizadas por diferentes usuários e

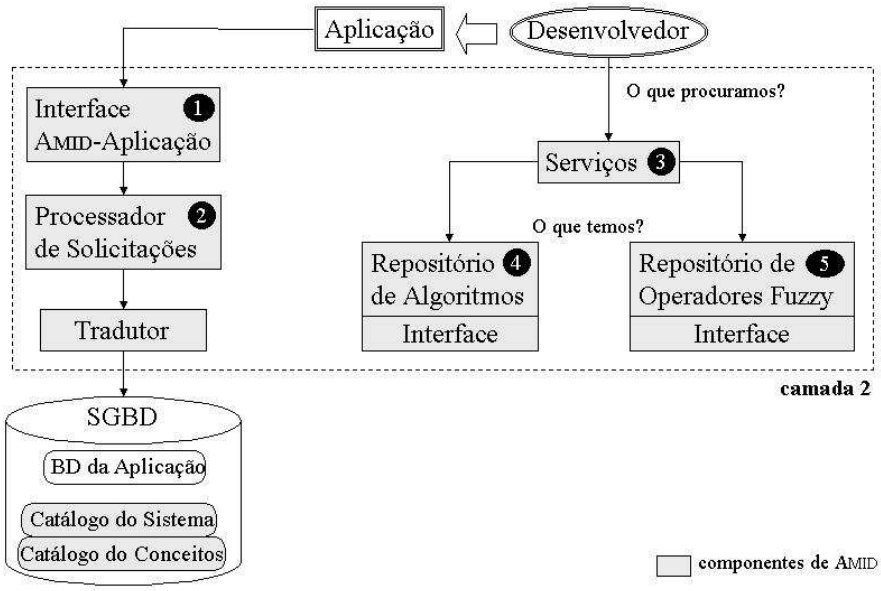


Figura 4.1: Arquitetura do Sistema AMID.

aplicações. Possui uma interface usada pelos usuários para consultar os algoritmos. Uma implementação deve fornecer aplicativos para a manutenção do repositório.

5. Repositório de Operadores Lógicos Fuzzy. Integra diferentes interpretações para os operadores lógicos. Possui uma interface usada pelos usuários para manipular e consultar as implementações dos operadores lógicos.

6. Catálogo de Sistema. Armazena os metadados sobre os esquemas dos bancos de dados e as especificações das aplicações para processar as imagens. Contém um catálogo com os conceitos definidos pela aplicação.

Os serviços definem ferramentas orientadas à manipulação de imagens. Os serviços definidos em AMID são os repositórios de algoritmos e de operadores lógicos. Novos serviços podem ser acrescentados ao sistema.

Uma abordagem e implementação do repositório de algoritmos são apresentadas no próximo capítulo. O repositório de operadores lógicos pode ser implementado de maneira similar.

O sistema AMID torna-se um sistema de gerenciamento distribuído através da rede, se os componentes da arquitetura são executados em diferentes computadores na rede.

4.3 Interface de Amid com as aplicações

Definimos um conjunto de serviços na interface para que as aplicações possam manipular seus bancos de dados.

- Serviços de criação dos bancos de dados: o compilador ODL e o parser LEA. A figura 4.4 mostra como a partir do arquivo ODL é criado o esquema do bancos de dados. Com a informação no arquivo de especificações da aplicação é criado o catálogo de sistema. O sistema possui dois operadores lógicos (união e intersecção) e uma função de pertinência para os modificadores (o exemplo descrito no capítulo anterior) implementados. Caso a aplicação não especifique novas implementações, o sistema assume os próprios modificadores e operadores lógicos na execução das solicitações.

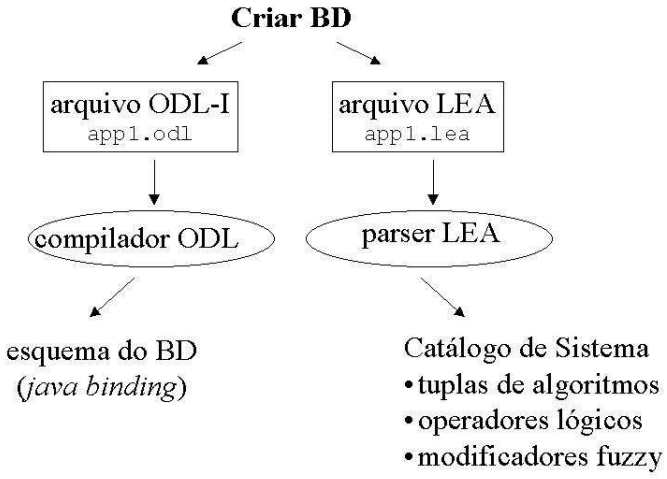


Figura 4.2: Serviços de criação dos bancos de dados.

- Serviços de Interface: conexão ao banco de dados e encerramento da conexão.
- Serviços para solicitações: operações de manipulação (inserir, apagar, modificar imagens) e consultas OQL-I ao banco de dados. Um subconjunto das solicitações OQL-I foram implementadas.
- Serviços de Suporte: manipulação do catálogo de sistema.

4.3.1 Serviços de Interface: Conexão ao Banco de Dados

O objetivo de atribuir identificadores únicos aos recursos em um sistema é possibilitar sua independência de localização, assim como sua longevidade de referência.

Un Identificador Universal de Recursos (URI - *Uniform Resource Identifier*) é uma cadeia de caracteres que identifica recursos na Internet. O URI é uma forma de encapsular um nome em um espaço de nomes cadastrados, e etiquetá-lo com o espaço de nomes, produzindo um membro do conjunto universal.

A sintaxe universal permite o acesso a objetos disponíveis, utilizando protocolos existentes, e é capaz de ser estendida. A especificação da sintaxe URI não diz a respeito das propriedades

dos nomes e dos espaços de nomes que mapeia. As propriedades encontram-se na especificação dos protocolos e as convenções de cada esquema.

O URI mais comum é o Localizador Uniforme de Recursos (URL - *Uniform Resource Locator*), o qual identifica um endereço de domínio de Internet. Em geral, a sintaxe de um URL é:

```
esquema : partes-especificas-do-esquema
```

O esquema identifica o tipo de URL. Por exemplo, o seguinte URL identifica o esquema ou protocolo HTTP:

```
http://www.unicamp.br
```

No sistema AMID, uma aplicação precisa identificar seus bancos de dados. Para isto, nos definimos um identificador de nomes único para cada esquema de bancos de dados.

Estrutura do Identificador

O identificador dos bancos de dados em AMID é um URL da seguinte forma:

```
amid:protocolo-interface-conexao:protocolo-SGBD:database

amid:protocolo-interface-conexao:protocolo-SGBD://host/database

amid:protocolo-interface-conexao:protocolo-SGBD://host:port/database
```

Sendo que:

amid é o esquema do sistema,

protocolo-interface-conexao é o protocolo da interface de conexão com o SGBD,

protocolo-SGBD é a interface do SGBD,

host é o servidor, e

port é o número do porto.

database é o banco de dados,

Por exemplo:

```
amid:jdbc:postgresql:DBBankExample
```

A primeira vez que a aplicação interage com o sistema é para criar o banco de dados. Uma vez criado, toda vez que a aplicação faz referência ao banco de dados precisa conectar-se ao URL do esquema de AMID.

4.3.2 Aplicações

Uma aplicação manipula e consulta os bancos de dados utilizando os serviços da interface de AMID. A figura 4.3 mostra as operações disponíveis para as aplicações.

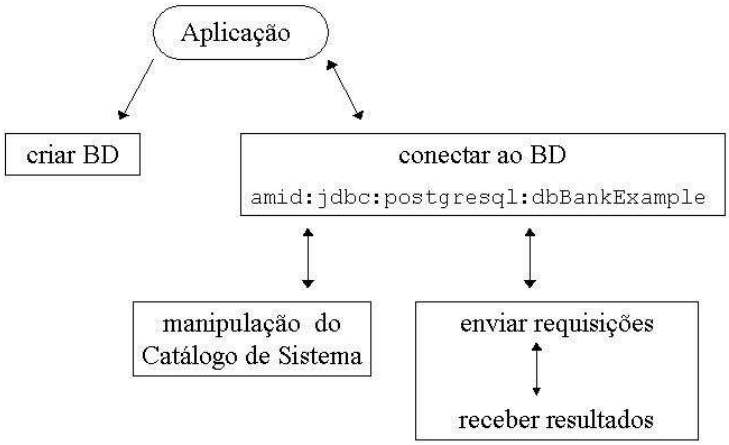


Figura 4.3: Serviços de AMID para as aplicações.

4.4 Processador das Solicitações

O processador de solicitações tem a função de conferir a estrutura de uma solicitação e executá-la, retornando os resultados à aplicação. A figura 4.4 mostra o plano de execução de uma solicitação em AMID.

Uma solicitação é processada da seguinte maneira:

1. O sistema confere a estrutura sintática da solicitação.
2. O sistema confere a estrutura semântica da solicitação.

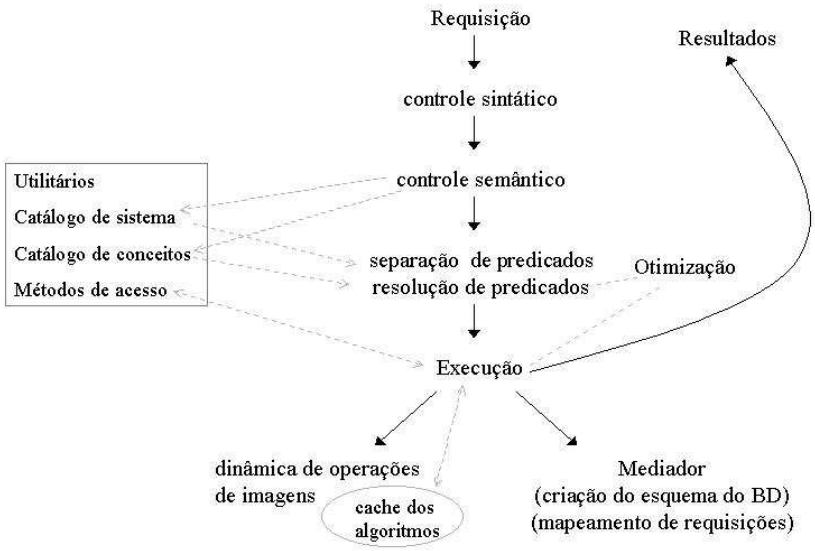


Figura 4.4: Processamento das solicitações.

- Utiliza o catálogo de conceitos para conferir conceitos.
- Utiliza o catálogo de sistema para conferir os identificadores das tuplas de operações, dos operadores lógicos e dos modificadores fuzzy.
- Localiza os algoritmos para mapear os parâmetros das operações.

3. Separa os predicados tradicionais dos predicados fuzzy.
4. Resolve os predicados com a substituição de conceitos, as operações de similaridade e os operadores lógicos.
5. O sistema executa a solicitação.
   - Uma consulta com os predicados tradicionais é enviada ao SGBD, utilizando o Tradutor que retorna os resultados.
   - Com os resultados parciais, a consulta fuzzy é processada.
   - Os algoritmos de similaridade são carregados em memória e executados para as imagens do resultado parcial (na ausência de índices).
   - Uma vez obtidos todos os pares (imagem, valor de similaridade), para cada predicado é feito o mapeamento com os modificadores fuzzy, segundo a função de pertinência especificada pela aplicação.
   - O sistema aplica os operadores lógicos para fazer a junção dos predicados.
6. O resultado final é enviado à aplicação.

4.5 Tradutor

O objetivo do Tradutor é oferecer funcionalidades para interagir com o SGBD que armazena as imagens. Dado que nem todos os SGBDs possuem a mesma interface, a função do Tradutor é mapear as solicitações e obter os resultados dos dados solicitados. Com a existência deste componente, uma mudança de SGBD não causará mudanças no modelo.

O acesso aos dados varia dependendo do sistema de banco de dados, por exemplo, as aplicações usam o API do JDBC para ter acesso aos dados nos sistemas relacionais. JDBC manipula instruções SQL, mas a sintaxe e o formato das instruções SQL podem variar dependendo do sistema de bancos de dados.

4.6 Catálogo do Sistema

Tradicionalmente, os catálogos ou dicionários dos sistemas de bancos de dados armazenam os metadados sobre a estrutura dos esquemas dos bancos de dados, os tipos de dados, estruturas de indexação e operadores.

O catálogo do sistema foi concebido para guardar as especificações das aplicações feitas na linguagem de especificação das aplicações. O catálogo desempenha uma função importante dentro do sistema. As informações armazenadas servem para que o sistema confira o processamento correto das imagens. Os identificadores das operações garantem a execução dos algoritmos.

Quando uma aplicação envia os arquivos com o esquema do banco de dados e as especificações, o sistema cria o catálogo para a aplicação. A figura 4.5 mostra a estrutura geral do catálogo de sistema.

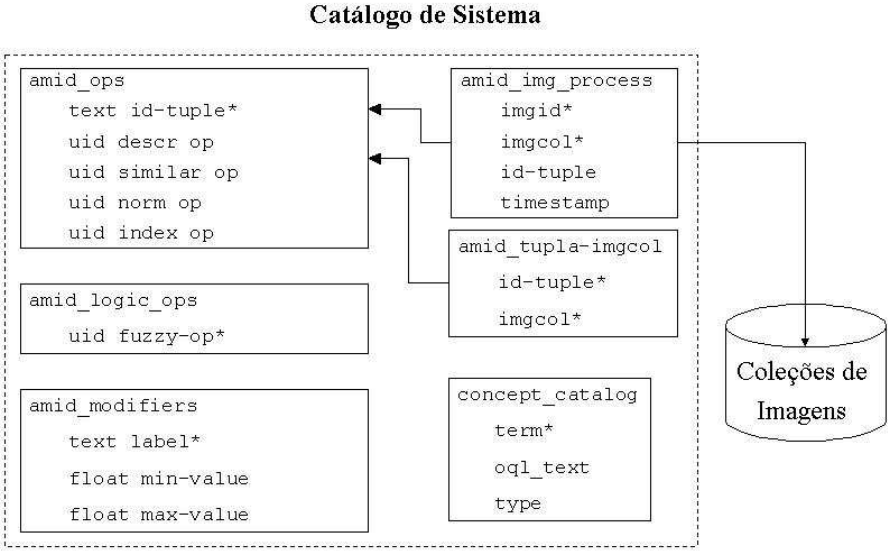


Figura 4.5: Esquema do Catálogo de Sistema de AMID.

O catálogo do sistema armazena as referências aos operadores lógicos, os valores dos modificadores e as tuplas de operações.

A referência ao operador lógico é o identificador único dentro do repositório de operadores. Para os modificadores, são guardados os nomes dos termos e, para cada termo os valores mínimo e máximo do intervalo.

Para cada tupla de operações se armazena a referência aos algoritmos e as coleções de imagens que são processadas para cada tupla. A referência ao algoritmo é o identificador único dentro do repositório de algoritmos. Os identificadores dos algoritmos têm o formato definido no capitulo 5.

Quando uma imagem é inserida no banco de dados, o sistema armazena no catálogo o identificador da tupla que processou a imagem junto com o identificador da mesma.

O catálogo está disponível para consultar os metadados dos bancos de dados. As tuplas de operações definidas e outras informações podem ser consultadas. Assim, podem ser feitas consultas sobre o processamento para uma imagem. Questões como: quantas/quais imagens foram analisadas extraindo o descritor x, ou qual descritor foi extraído para a imagem i, podem ser respondidas utilizando esta ferramenta.

O catálogo de sistema possui um catálogo de conceitos. Ele é inicializado vazio quando o sistema cria o catálogo de sistema para o banco de dados da aplicação. O catálogo de conceitos é explicado a seguir.

4.6.1 Catálogo de Conceitos

O catálogo de conceitos é simples. Ele armazena o nome do conceito, o tipo, e sua definição na linguagem OQL-I. Os conceitos podem ser termos, operadores, ou funções.

O catálogo de conceitos vai ser povoado na medida que os conceitos sejam definidos pela aplicação e pelos usuários.

4.7 Usuários do Sistema Amid

Existem vários tipos de usuários quando trabalhamos com as aplicações de bancos de dados. Esses usuários são: os desenvolvedores da aplicação, os operadores encarregados de povoar e manter os bancos de dados e os clientes (usuário final) para os quais as aplicações são desenvolvidas.

O sistema AMID é voltado a facilitar o trabalho dos desenvolvedores de aplicações de recuperação de imagens. Existem duas maneiras desses usuários interagir com AMID.

Desenvolvimento de Aplicações. Estes usuários utilizam o sistema para implementar sua aplicação de recuperação de imagens. A interface de AMID com a aplicação oferece o suporte para criar o esquema dos bancos de dados de imagens e manipular as solicitações.

Serviços. Os desenvolvedores de aplicações utilizam os serviços para ver quais são as funcionalidades oferecidas pelo sistema. Em particular, podem utilizar a interface do repositório de algoritmos para fazer consultas sobre o mesmo.

Para os usuários das aplicações o sistema oferece a linguagem OQL-I para consultar os bancos de dados e o mecanismo de retroalimentação de consultas.

Os desenvolvedores das aplicações são encarregados de criar uma interface visual para as aplicações, pois é específico da própria aplicação a maneira de apresentar as informações, projetar, organizar ou classificar as consultas para os seus usuários.

4.8 Detalhes de Implementação

O sistema é implementado na linguagem Java. O SGBD utilizado para criar os esquemas de bancos de dados e armazenar as imagens é o PostgreSQL.

Nossa implementação do sistema foi desenvolvida como um subconjunto dos componentes, operações e serviços descritos neste capítulo.

O sistema implementou duas funções fuzzy para os operadores lógicos (OR: máximo e AND: mínimo) e uma função de pertinência para os modificadores como descrito no exemplo do capítulo anterior.

O repositório de algoritmos foi implementado como explicado no próximo capítulo, utilizando um servidor Web.

Interface de Amid

O serviço de criação dos bancos de dados foi implementado como uma ferramenta simples que cria o banco de dados da aplicação no PostgreSQL e povoa o banco de imagens e seus dados tradicionais, sempre utilizando o Tradutor.

Os serviços de interface e solicitações foram implementados. Uma aplicação utiliza essas interfaces fazendo um chamado às classes que as implementam da seguinte maneira:

Serviços de Interface

```
class AppInterface

    public void setConnection (String driver, String url, String user,
                               String password);
    public void closeConnection();
```

Os parâmetros da conexão são:

driver: postgresql.Driver

url: o identificador dos bancos de dados em AMID por exemplo, amid:jdbc:postgresql:DBBankExample.

user, password: algumas aplicações podem impor restrições aos usuários utilizando o sistema de identificação do sistema gerenciador de bancos de dados.

Servicos de Solicitações

```
class AppInterface

    public ResultsManager sendRequest (String oqlStr);
```

Os parâmetros da solicitação são:

oqlStr é uma cadeia de caracteres que contém a expressão OQL-I da solicitação. A implementação foi feita para consultas simples OQL-I da forma select-from-where, onde os predicados da cláusula where são predicados fuzzy da forma: <modificador> <operação de similaridade>.

ResultManager é a classe que manipula a estrutura FuzzyResultSet criada para armazenar as imagens com o valor de pertinência à consulta. Este manipulador define operações de união (OR) e intersecção (AND) sobre os conjuntos de resultados, como sendo uma implementação dos operadores lógicos fuzzy.

Processador de Solicitações

O plano de execução foi implementado como descrito em **??**. Desenvolvemos um parser para processar um subconjunto de solicitações. A implementação foi feita para consultas simples OQL-I da forma select-from-where, onde os predicados da cláusula where são predicados fuzzy da forma: <modificador> <operação de similaridade>.

Para suportar a execução das funções de similaridade, utilizamos os mecanismos de introspecção de Java. Esses mecanismos permitem o acesso a detalhes internos de classes e seus métodos para invocar, carregar em memória e executar os algoritmos. O valor da similaridade associado a cada imagem é armazenado na estrutura FuzzyResultSet.

Mecanismos de Otimização

Alguns mecanismos simples de otimização foram utilizados com vista a melhorar a eficiência do processamento das solicitações.

O primeiro passo na execução de uma consulta é resolver os predicados tradicionais. Assim, a execução dos predicados fuzzy é feita para aquelas imagens que satisfazem os predicados tradicionais, restringindo o espaço de execução.

O sistema pode criar um cachê de memória para os algoritmos da aplicação que estejam sendo mais utilizados. Se existem, os métodos de acesso serão utilizados para acelerar as consultas.

As operações de similaridade devem ser executadas localmente no servidor da consulta, pois elas vão ser executadas tantas vezes quanto imagens no banco de dados da aplicação.

Tradutor

Para implementar o Tradutor utilizamos um *pattern* de projeto, conhecido como DAO (*Data Access Object*). DAO abstrai e encapsula todo o acesso à fonte de dados. Ele manipula a conexão

com o banco de dados para obter e armazenar dados. DAO oculta os detalhes de implementação dos clientes. A interface aos clientes não muda segundo implementações.

Catálogos

O catálogo de sistema e o catálogo de conceitos são criados no sistema gerenciador de bancos de dados da aplicação, neste caso, o PostgreSQL.

Para toda aplicação, o catálogo de sistema contém as tuplas de operações definidas pela aplicação e os operadores lógicos e os modificadores definidos pelo sistema. Se a aplicação definiu outros operadores e modificadores, eles também são inseridos no catálogo da aplicação.

AMID cria o catálogo de conceitos para cada aplicação. Nesta implementação, os conceitos são considerados termos. O nome do conceito é a chave do catálogo. Portanto, ao introduzir um conceito cujo nome já está armazenado, o sistema assume a redefinição do conceito. Outras implementações mais complexas podem ser propostas para o catálogo de conceitos.

4.9 Sumário

Este capítulo apresentou AMID, um sistema de gerenciamento de imagens sobre bancos de dados baseado no modelo descrito no capítulo 3. Os componentes da arquitetura do sistema foram descritos. Foi explicado como as aplicações interagem com o sistema para manipular os bancos de dados. Os comentários da implementação do sistema AMID foram descritos na última seção.

As características relevantes de nosso sistema são:

- Generalidade: o sistema se ocupa de definir soluções de propósito geral.
- Independência de SGBD: com o Tradutor, AMID não precisa se preocupar com os detalhes de implementação de um SGBD específico.
- Serviços: conjunto de serviços orientados à manipulação de imagens.
- Catálogo do sistema: AMID possui o catálogo dos metadados da aplicação e do banco de dados. Dito catálogo armazena os conceitos (definidos pelos usuários e as aplicações) utilizados nas consultas.

O repositório de algoritmos é um serviço do sistema que oferece ao desenvolvedor a possibilidade de consultar os diferentes algoritmos utilizados na recuperação de imagens. A descrição, a arquitetura e a implementação do repositório são apresentadas no próximo capítulo.

Capítulo 5

Repositório de Algoritmos de Recuperação de Imagens

5.1 Introdução

Neste capítulo apresentamos o módulo do sistema AMID correspondente ao Repositório Distribuído de Algoritmos para a Recuperação de Imagens.

Problema:	Dada a grande variedade de aplicações de imagens, o número de operações para a busca por conteúdo das imagens cresce devido à quantidade de aplicações de domínio específico. Cada aplicação constrói várias técnicas para explorar as imagens de acordo com as suas necessidades.
Solução:	O objetivo do Repositório é integrar as mais diversas operações sobre imagens, de maneira que elas estejam disponíveis para serem consultadas e utilizadas por diferentes aplicações.

A criação do repositório envolve os tópicos a seguir.

- Definição de Metadados para os Algoritmos: conjunto padrão de metadados, aplicáveis aos algoritmos de recuperação de imagens.
- Arquitetura do Repositório de Algoritmos: arquitetura geral para a manipulação dos algoritmos pelo repositório.
- Implementação do Repositório de Algoritmos: implementação de uma versão utilizando as tecnologias da Web, baseada na arquitetura do repositório.

5.2 Objetivos do Repositório de Algoritmos

Os objetivos da criação do Repositório de Algoritmos para a Recuperação de Imagens são os seguintes:

- Reunir as operações de imagens.
- Manter a independência de domínio, sem importar o domínio onde a operação foi desenvolvida e nem precisam ser operações de domínio global.
- Disponibilizar as operações a múltiplos usuários.
- Reusar ou compartilhar as operações por múltiplas aplicações.
- Inserir, manter, buscar e recuperar (consultar) operações de imagens como suporte ao desenvolvimento das aplicações de recuperação de imagens.
- Fazer o repositório independente de linguagem de programação. O repositório pode estar formado por operações implementadas em diferentes linguagens de programação.
- Fazer valer o Repositório como uma ferramenta fundamental no desenvolvimento de aplicações.

Dada a variedade e disponibilidade de operações sobre imagens, uma oportunidade de expansão das aplicações é criada. O grande volume de operações de recuperação de imagens fornece uma oportunidade para manipulá-las. Para isto, é necessário criar os mecanismos que permitam armazenar, catalogar, indexar, pesquisar e recuperar operações.

No contexto das operações de imagens, o uso dos metadados permite a identificação, classificação das operações. Com isto, os usuários tiram proveito das operações existentes no repositório e podem consultar livremente para buscar o que precisarem.

Muitos trabalhos estão voltados para organizar as informações das imagens propriamente ditas. Isto inclui o processamento de imagens (edição e transformação de imagens). No entanto, as operações que contribuem na pesquisa e recuperação de imagens (segmentação, extração de descritores, indexação e similaridade) não estão envolvidas nessas pesquisas.

Nosso trabalho fornece suporte de metadados para essas últimas operações que são de grande importância no processo de recuperação de imagens.

5.3 Definição de Metadados para os Algoritmos

Os objetivos de fornecer uma coleção de metadados para os algoritmos de recuperação de imagens são os seguintes.

- Definir um conjunto padrão de metadados, aplicáveis aos algoritmos de recuperação de imagens.
- Fornecer suporte semântico uniforme à manipulação de algoritmos de imagens.
- Fornecer suporte à extensibilidade do conjunto proposto de metadados, oferecendo a possibilidade de acrescentar outros dados.

Os metadados para os algoritmos de recuperação de imagens foram definidos baseados em conceitos genéricos divididos em blocos conceituais. Cada bloco descreve um aspecto único dos algoritmos. Esta classificação permite que extensões possam ser feitas em um determinado bloco (ou criar um novo) sem mexer na arquitetura geral e garantindo a estrutura semântica. Na figura 5.1 definimos quatro blocos de metadados descritos nas próximas seções.

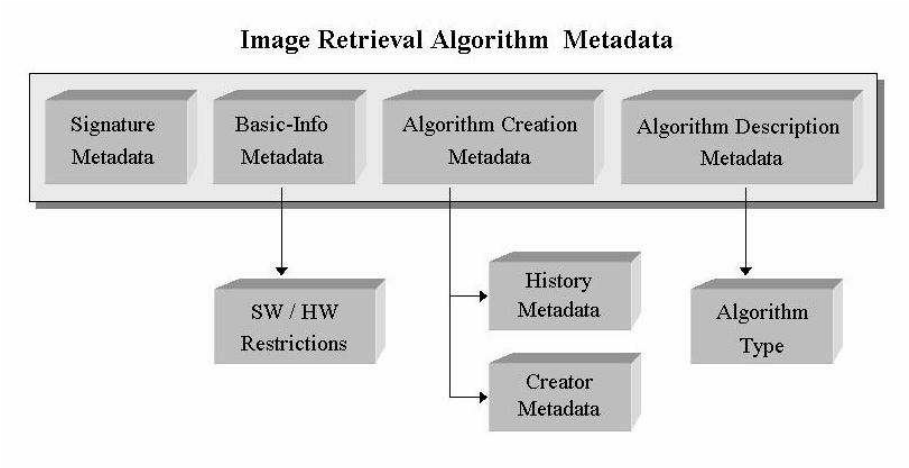


Figura 5.1: Arquitetura dos Metadados dos Algoritmos.

Cada campo do conjunto de metadados vem acompanhado de uma breve descrição e sua representação XML. Cada bloco de metadados possui sua definição de esquema em XML e acompanha um exemplo. As razões fundamentais para escolhermos este padrão são as seguintes. O padrão XML foi adotado seguindo uma tendência mundial que permite comunicação mais fácil via Internet e com sistemas legados; XML é altamente extensível e fornece uma infra-estrutura sólida para implementar metadados. O anexo D apresenta a especificação completa do esquema XML para os algoritmos de recuperação de imagens.

5.3.1 Metadados da Interface

A interface é uma informação requerida para cada algoritmo, pois ela mostra a forma de representação dos algoritmos.

- Interface do algoritmo: nome e parâmetros de entrada/saída da implementação. Esta interface é usada para chamar e executar o algoritmo.

Interface dos Algoritmos

Todos os algoritmos têm uma interface ou assinatura do tipo:

<nome da operação> (<lista de parâmetros>): <descritor>

Os algoritmos de extração de descritores possuem uma das duas interfaces a seguir.

Interface Global: <nome da operação> (imagem): <descritor>

Interface Local:
<nome da operação> (imagem, segmento): <lista de descritores>

Os algoritmos de comparação ou similaridade possuem a interface a seguir.

<nome da operação> (<descritor1>, <descritor2>): <valor de similaridade>

5.3.1.1 Definição de Esquema

```
<xs:complexType name="locationType">
  <xs:sequence>
    <xs:element name="NAMESPACE" type="xs:anyURI"/>
    <xs:element name="FILENAME" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="signatureType">
  <xs:sequence>
    <xs:element name="LOCATION" type="locationType" minOccurs="0"/>
    <xs:element name="METHOD" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs: element name="SIGNATURE" type="signatureType"/>
```

Exemplo

<SIGNATURE><METHOD>find_matches (vector1, vector2): [0,1]</METHOD></SIGNATURE>

5.3.2 Metadados Básicos dos Algoritmos

Esta seção define a informação básica de um algoritmo. Contém a informação sobre a localização dos arquivos para executar o algoritmo.

- Versão dessa implementação.
- Localização física: localização original onde se encontra o código do algoritmo, tais como referência ao URL, nome e caminho do arquivo local, etc.

- Localização física dos arquivos fonte: localização original onde se encontram os arquivos fonte do algoritmo, tais como referência ao URL, nome e caminho do arquivo local, etc.
- Requisitos de execução: limitações de hardware para executar o algoritmo, por exemplo, processador, espaço em disco rígido, memória requerida.
- Requisitos de ambiente: limitações de software para executar o algoritmo, por exemplo, sistema operacional, bibliotecas requeridas, etc.
- Linguagem de programação utilizada: por exemplo, Java, C++, C, etc.
- Versão da linguagem de programação dessa implementação: por exemplo, JDK 1.1.8.

5.3.2.1 Definição de Esquema

```
<xs:element name="BASIC_INFO_METADATA">
   <xs:complexType>
      <xs:attribute name="VERSION" type="xs:string"/>
      <xs:sequence>
         <xs:element name="MAIN_FILENAME" type="locationType"/>
         <xs:element name="COMPLEMENTARY_FILENAME" type="locationType"
                                    minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="PROGRAMMING_LANG" type="xs:string"/>
         <xs:element name="PL_VERSION" type="xs:string" minOccurs="0"/>
         <xs:element name="REQUIREMENTS" minOccurs="0"/>
         <xs:element name="SRC_LOCATION" type="xs:anyURI"  minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="REQUIREMENTS">
  <xs:complexType>
      <xs:sequence>
         <xs:element name="SOFTWARE" minOccurs="0"/>
         <xs:element name="HARDWARE" minOccurs="0"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="SOFTWARE">
  <xs:complexType>
      <xs:sequence>
         <xs:element name="PLATFORM" type="xs:string" minOccurs="0"/>
         <xs:element name="LIBRARY"  type="xs:string" minOccurs="0"/>
      </xs:sequence>
```

```
  </xs:complexType>
</xs:element>

<xs:element name="HARDWARE">
  <xs:complexType>
      <xs:sequence>
         <xs:element name="PROCESSOR" type="xs:string" minOccurs="0"/>
         <xs:element name="HD_SPACE"  type="xs:string" minOccurs="0"/>
         <xs:element name="RAM"       type="xs:string" minOccurs="0"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>
```

5.3.2.2 Exemplo

```
<BASIC_INFO_METADATA VERSION="1.0">
  <MAIN_FILE_NAME>ex1/ex1.class</MAIN_FILE_NAME>
  <PROGRAMMING_LANG>java</PROGRAMMING_LANG>
  <PL_VERSION 1.2/>
  <SRC_LOCATION>
    http://www.dt.fee.unicamp.br/pool/src/ex1/
  </SRC_LOCATION>
  <REQUIREMENTS>
    <SOFTWARE>
      <PLATFORM any/>
    </SOFTWARE>
    <HARDWARE>
      <RAM 16Mb/>
    </HARDWARE>
  </REQUIREMENTS>
</BASIC_INFO_METADATA>
```

5.3.3 Metadados sobre a Criação de Algoritmos

Os metadados nesta seção referem-se a informação relacionada à origem e criação do algoritmo.

- Data de criação.
- Histórico de uso.
- Desenvolvedores dos algoritmos.

5.3.3.1 Histórico de Uso

Este bloco possui informação sobre as ligações do algoritmos com sistemas e outros algoritmos.

- Sistema que desenvolveu este algoritmo, se existir essa ligação: por exemplo, o multi-* é um algoritmo de indexação desenvolvido pelo sistema QBIC [44].

Relacionamento com outros algoritmos. É utilizado para descrever os relacionamentos significativos com outros algoritmos , podem ser classificados em (esta classificação é baseada em [20]):

Tipo de relacionamento	Relacionamento	Inverso
Parte lógica de outra operação	Faz parte de outra operação (IsPartOf)	Lista de operações que fazem parte dela (HasPart)
Estado histórico	É uma versão de outra operação (IsVersionOf)	Tem uma outras versões (HasVersion)
Derivada ou adaptada	É baseada em outra operação (HasSourceFrom)	É a base para outras operações (IsSourceFor)
Requer operações para seu funcionamento	Requer outras operações (Requires)	É requerida por outras operações (IsRequiredBy)

5.3.3.2 Metadados dos Desenvolvedores

Este bloco contém informação sobre o grupo de pessoas que desenvolveram e/ou contribuíram à criação e implementação do algoritmo.

- Autores.

  Contribuintes.

  Informação de contato: por exemplo, e-mail de cada um.

5.3.3.3 Definição de Esquema

```
<xs:element name="CREATION_METADATA">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="CREATION_DATE" type="xs:dateTime"   minOccurs="0" />
         <xs:element name="USE_HISTORY"                        minOccurs="0" />
         <xs:element name="CREATOR_METADATA"                   minOccurs="0" />
      </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="USE_HISTORY">
   <xs:complexType>
      <xs:sequence>
        <xs:element name="SYSTEM" type="xs:string" minOccurs="0"/>
        <xs:element name="RELATIONSHIPS"  minOccurs="0"  maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="RELATIONSHIPS">
```

```
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:anyURI">
         <xs:attribute name="TYPE" type="xs:string" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Os valores sugeridos para os tipos de relacionamentos foram descritos acima.

```
<xs:element name="CREATOR_METADATA">
  <xs:complexType>
      <xs:choice maxOccurs="unbounded"/>
         <xs:element ref="PERSON"/>
         <xs:element ref="ORGANIZATION"/>
      </xs:choice>
  </xs:complexType>
</xs:element>

<xs:element name="PERSON">
   <xs:complexType>
      <xs:attribute name="TYPE" type="xs:string"/>
      <xs:sequence>
         <xs:element name="PNAME"  type="xs:string"/>
         <xs:element ref="ORGANIZATION"  minOccurs="0"/>
         <xs:element ref="CONTACT_INFO"  minOccurs="0"  maxOccurs="10"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="ORGANIZATION">
  <xs:complexType>
      <xs:sequence>
         <xs:element name="ORG_NAME" />
         <xs:element ref="CONTACT_INFO" minOccurs="0"  maxOccurs="10"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="CONTACT_INFO">
  <xs:complexType>
```

```
      <xs:sequence>
         <xs:element name="ADDRESS"  type="xs:string"/>
         <xs:element name="EMAIL"  type="xs:string"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>
```

Os tipos sugeridos para as pessoas relacionadas com o algoritmo são: autores, contribuintes. As palavras chave sugeridas para isto são: "author", "contributor".

5.3.3.4 Exemplo

```
<CREATION_METADATA>
 <CREATION_DATE>2002-12-07T12:50:13</CREATION_DATE>
 <USE_HISTORY>
  <RELATIONSHIPSTYPE = "IsVersionOf≫
   ex20/ex20.class      </RELATIONSHIPS>
 </USE_HISTORY>
 <CREATOR_METADATA>
  <PERSON DESCRIPTION="author">
  <!- person: also contributor ->
   <NAME>Sahudy</NAME>
   <ORGANIZATION>Unicamp</ORGANIZATION>
   <CONTACT_INFO>
    <EMAIL>sahudy@dt.fee.unicamp.br</EMAIL>
   </CONTACT_INFO>
  </PERSON>
 </CREATOR_METADATA>
</CREATION_METADATA>
```

5.3.4 Metadados para a Descrição dos Algoritmos

Este bloco descreve o conteúdo dos algoritmos, oferecendo uma documentação sobre eles.

- Tipo de algoritmo: o objetivo do algoritmo é um dentre: segmentação, extração de descritores, indexação, comparação por similaridade, ou normalização.
- Operação de propósito geral: pode ser utilizada em qualquer imagem com resultados aceitáveis.
- Domínio de aplicação: o domínio de aplicação para a qual o algoritmo foi implementado. Por exemplo: medicina, arte, etc.
- Palavras chave: termos relevantes sobre o algoritmo.
- Descrição da operação: descrição geral do processamento feito pelo algoritmo.
- Descrição por passos gerais do algoritmo: cadeia de ações que formam o algoritmo.
- Tipos de dados envolvidos: descrição dos tipos de dados definidos pelos usuários.

- Comentários: aqui pode ser especificado outras propriedades que não foram consideradas neste bloco.

Usualmente, as operações de similaridade, extração de descritores e normalização possuem uma única interface que é usada como referência à operação, a partir da qual ela é executada. As operações de indexação possuem quatro interfaces indispensáveis. Elas são: interface de criação da estrutura de indexação, duas interfaces de inserção/exclusão dos elementos no índice, e a interface de busca que é utilizada na consulta. Cada uma será executada no momento apropriado. Dependendo do tipo de algoritmo, os metadados são descritos como uma das próximas seções.

5.3.4.1 Algoritmos de Extração de Descritores

Estes metadados são aplicáveis quando um algoritmo tem como objetivo a extração de descritores do conteúdo das imagens.

- características do conteúdo da imagem: características processadas pelo algoritmo, por exemplo, cor, textura, forma, ou uma combinação delas.
- descritores das características da imagem: por exemplo, Gabor filter para textura, MBR, 2D-String para objetos e relacionamentos espaciais.

5.3.4.2 Algoritmos de Comparação por Similaridade

Este bloco é aplicável aos algoritmos de comparação por similaridade.

- método de decisão: por exemplo, vizinho mais próximo (*nearest neighbor*), baseado em probabilidade (*likelihood-based*).
- medida/métrica da similaridade: por exemplo, distância Euclidiana, quadrática, etc.

5.3.4.3 Algoritmos de Normalização

Este bloco descreve a informação sobre os algoritmos de normalização.

- métrica da normalização: medida utilizada para normalizar um descritor de características, por exemplo,*rank normalization.*

5.3.4.4 Algoritmos de Indexação

Estes metadados descrevem os algoritmos de indexação.

- Estrutura de indexação: por exemplo, R-tree.
- Métodos de acesso. A indexação tem quatro interfaces primárias. Elas são: criação da estrutura de indexação, métodos para adicionar, comparar e remover elementos. Dado que todo algoritmo tem uma interface do método principal, nos algoritmos de indexação tomamos o método de criação do índice como o método principal.

5.3.4.5 Definição de Esquema

```
<xs:element name="CONTENT_METADATA">
  <xs:complexType>
      <xs:sequence>
         <xs:element ref="ALGORITHM_TYPE"/>
         <xs:element name="APP_DOMAIN" type="xs:string" minOccurs="0"/>
         <xs:element name="GENERAL_PURPOSE" type="xs:boolean" minOccurs="0"/>
         <xs:element name="KEYWORD" type="xs:string" maxOccurs="unbounded"/>
         <xs:element name="DESCRIPTION" type="xs:string"/>
         <xs:element ref="STEP_DESCRIPTION" minOccurs="0"/>
         <xs:element ref="UD_DATATYPES" minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="COMMENTS" type="xs:string"  minOccurs="0"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="ALGORITHM_TYPE">
  <xs:complexType>
      <xs:choice>
         <xs:element ref="SIMILARITY"/>
         <xs:element ref="DESCRIPTOR_EXTRACTION"/>
         <xs:element ref="INDEXING"/>
         <xs:element ref="NORMALIZATION"/>
      </xs:choice>
  </xs:complexType>
</xs:element>

<xs:element name="STEP_DESCRIPTION">
  <xs:complexType>
      <xs:sequence>
         <xs:element name="STEP" type="xs:string" maxOccurs="unbounded"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="UD_DATATYPES">
  <xs:complexType>
      <xs:sequence>
         <xs:element name="NAME" type="xs:string"/>
         <xs:element name="TYPE" type="xs:string"/>
         <xs:element name="DESCRIPTION" type="xs:string"/>
      </xs:sequence>
```

```
  </xs:complexType>
</xs:element>

<xs:element name="DESCRIPTOR_EXTRACTION">
  <xs:complexType>
      <xs:sequence>
         <xs:element name="IMAGE_FEATURE" type="xs:string"/>
         <xs:element name="DESCRIPTOR" type="xs:string"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="SIMILARITY">
  <xs:complexType>
      <xs:sequence>
         <xs:element name="DECISION_METHOD" type="xs:string"/>
         <xs:element name="METRIC" type="xs:string"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="NORMALIZATION">
  <xs:complexType>
      <xs:sequence>
         <xs:element name="METRIC"  type="xs:string"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="INDEXING">
  <xs:complexType>
     <xs:sequence>
        <xs:element name="INDEXING_STRUCTURE"              type="signatureType"/>
        <xs:element name="ADD_ACCESS_METHOD_SIGNATURE"     type="signatureType"/>
        <xs:element name="QUERY_ACCESS_METHOD_SIGNATURE"   type="signatureType"/>
        <xs:element name="DELETE_ACCESS_METHOD_SIGNATURE"  type="signatureType"/>
     </xs:sequence>
  </xs:complexType>
</xs:element>
```

5.3.5 Exemplos de Representação de Algoritmos

A seguir apresentamos a representação XML de dois algoritmos. O primeiro exemplo é um algoritmo de similaridade que dado dois descritores de imagens aplica a distância euclidiana e calcula o grau de similaridade entre as imagens. O segundo exemplo é um algoritmo para extrair os descritores de uma imagem, nesse caso o histograma de cores.

Exemplo 1: Algoritmo de Similaridade

```
<?xml version="1.0" ?>

<ALGORITHM_METADATA xmlns="http://www.dt.fee.unicamp.br/roira"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.dt.fee.unicamp.br/roira roira.xsd"
 LOCATION="http://www.dt.fee.unicamp.br/roira/pool/ex1">

<SIGNATURE>find_matches (vector1, vector2): [0,1]</SIGNATURE>

<BASIC_INFO_METADATA VERSION="1.0">
    <MAIN_FILE_NAME>EucDist.class</MAIN_FILE_NAME>
    <PROGRAMMING_LANG>java</PROGRAMMING_LANG>
    <PL_VERSION>1.2</PL_VERSION>
    <SRC_LOCATION> http://www.dt.fee.unicamp.br/pool/src/ex1/</SRC_LOCATION>
</BASIC_INFO_METADATA>

<CREATION_METADATA>
    <CREATION_DATE>2002-12-07T12:50:13</CREATION_DATE>
    <CREATOR_METADATA>
        <PERSON DESCRIPTION="author">
        <!-- person: also contributor -->
            <NAME>Sahudy</NAME>
            <ORGANIZATION>Unicamp</ORGANIZATION>
            <CONTACT_INFO>
                <EMAIL>sahudy@dt.fee.unicamp.br</EMAIL>
            </CONTACT_INFO>
        </PERSON>
    </CREATOR_METADATA>
</CREATION_METADATA>

<CONTENT_METADATA>
    <ALGORITHM_TYPE>
        <SIMILARITY>
            <DECISION_METHOD>nearest neighbor</DECISION_METHOD>
```

```
            <METRIC>Euclidian distance</METRIC>
        </SIMILARITY>
    </ALGORITHM_TYPE>
    <GENERAL_PURPOSE>true</GENERAL_PURPOSE>
    <APP_DOMAIN>general</APP_DOMAIN>
    <KEYWORD>similarity</KEYWORD>
    <KEYWORD>Euclidian distance</KEYWORD>
    <DESCRIPTION>Given two descriptor vectors, the Euclidian
        distance is applied, and  its return value is the
        similarity degree between these two images</DESCRIPTION>
    <STEP_DESCRIPTION>
        <STEP>Given vector1 = (x1,x2,...,xn),
                    vector2 = (y1,y2,...,yn)</STEP>
        <STEP>d = sqrt(sqr(y1-x1) + sqr(y2-x2) + ...)</STEP>
        <STEP>normalize d</STEP>
        <STEP>return d in [0,1]</STEP>
    </STEP_DESCRIPTION>
</CONTENT_METADATA>

</ALGORITHM_METADATA>
```

Exemplo 2: Algoritmo de Extração de Descritores

```
<?xml version="1.0" ?>

<ALGORITHM_METADATA xmlns="http://www.dt.fee.unicamp.br/roira"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.dt.fee.unicamp.br/roira roira.xsd"
 LOCATION="http://www.dt.fee.unicamp.br/roira/pool/ex2">

<SIGNATURE>color_histogram (image): vector</SIGNATURE>

<BASIC_INFO_METADATA VERSION="1.0">
    <MAIN_FILE_NAME>Histogram.class</MAIN_FILE_NAME>
    <PROGRAMMING_LANG>java</PROGRAMMING_LANG>
    <PL_VERSION>1.2</PL_VERSION>
    <REQUIREMENTS>
        <SOFTWARE>
            <PLATFORM>any</PLATFORM>
        </SOFTWARE>
        <HARDWARE>
            <RAM>16Mb</RAM>
```

```
        </HARDWARE>
    </REQUIREMENTS>
</BASIC_INFO_METADATA>


<CREATION_METADATA>
    <CREATION_DATE>2002-12-07T12:50:13</CREATION_DATE>
    <CREATOR_METADATA>
        <PERSON DESCRIPTION="author">
        <!-- person: also contributor -->
            <NAME>Sahudy</NAME>
            <ORGANIZATION>Unicamp</ORGANIZATION>
            <CONTACT_INFO>
                <EMAIL>sahudy@dt.fee.unicamp.br</EMAIL>
            </CONTACT_INFO>
        </PERSON>
    </CREATOR_METADATA>
</CREATION_METADATA>

<CONTENT_METADATA>
    <ALGORITHM_TYPE>
        <DESCRIPTOR_EXTRACTION>
            <IMAGE_FEATURE>color</IMAGE_FEATURE>
            <DESCRIPTOR>histogram</DESCRIPTOR>
        </DESCRIPTOR_EXTRACTION>
    </ALGORITHM_TYPE>
    <GENERAL_PURPOSE>true</GENERAL_PURPOSE>
    <APP_DOMAIN>general</APP_DOMAIN>
    <KEYWORD>color histogram</KEYWORD>
    <DESCRIPTION>Given an image, the color feature
        is extracted as a histogram and its return value
        is a descriptor vector with dimension 3: (R,G,B)</DESCRIPTION>
    <STEP_DESCRIPTION>
        <STEP>Initialize H(R,G,B) = 0, H histogram matrix
            R(i,j), G(i,j) , B(i,j): pixel value (i,j) in [0..256]
            for the {R,G,B} components</STEP>
        <STEP>Double loop for all image pixels (i,j), calculating:
            H(R(i,j), G(i,j) , B(i,j) ) = H ( R(i,j),G(i,j) , B(i,j)) + 1</STEP>
        <STEP>returns H(R,G,B)</STEP>
    </STEP_DESCRIPTION>
</CONTENT_METADATA>

</ALGORITHM_METADATA>
```

5.4 Definição do Esquema para Consultas

Uma consulta tem um formato diferente para ser apresentada. Baseada na linguagem SQL, propomos uma definição de esquema XML para transportar as consultas do cliente ao servidor.

5.4.1 Definição de Esquema

```
<xs: element name="QUERY_TRANSPORT">
   <xs:complexType>
      <xs: attribute name="VERSION" type="  xs:string" />
      <xs:sequence>
         <xs: element name="OUTPUT_PARAMETERS" type="xs:string" />
         <xs: element name="RESTRICTIONS"      type=' xs:string" />
         <xs: element name="ORDER_BY"          type="xs:string" />
         <xs: element name="GROUP_BY"          type="xs:string" />
      </xs:sequence>
   </xs:complexType>
</xs: element>
```

5.4.2 Exemplo

```
<QUERY_TRANSPORT>
   <OUTPUT_PARAMETER>algorithm_signature</OUTPUT_PARAMETER>
   <RESTRICTIONS>algorithm_type=descriptor_extraction
     and image_feature=color</RESTRICTIONS>
</QUERY_TRANSPORT>
```

5.5 Arquitetura do Repositório de Algoritmos

A figura 5.2 mostra a arquitetura do repositório.

A arquitetura do repositório possui três componentes principais: os bancos de dados, o servidor e a interface visual. O repositório consta de um conjunto de bancos de dados distribuídos na rede. O acesso aos bancos de dados do repositório é sempre através do servidor. As solicitações são enviadas ao servidor, o qual as executa, acessando de maneira transparente aos bancos de dados.

O formato XML dos metadados dos algoritmos facilita aos usuários a especificação dos algoritmos e serialização através da rede. Os dados apresentam uma estrutura regular, com uma granularidade bem fina, facilmente mapeável a um esquema de banco de dados.

Um cliente do repositório pode importar ou consultar algoritmos. Ele tem duas opções para utilizar o repositório. A primeira é enviar os dados em formato XML diretamente ao sevidor

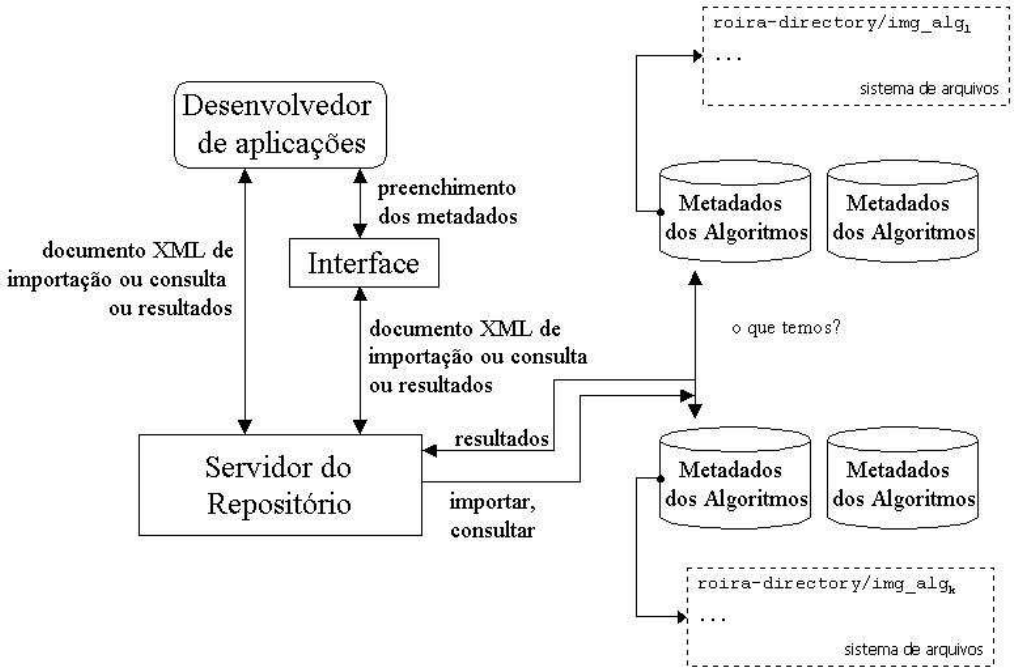


Figura 5.2: Arquitetura do Repositório de Algoritmos.

do repositório. A segunda é utitilizar a interface visual oferecida pelo repositório. Neste último caso, a informação é traduzida em um documento XML e enviado ao servidor.

O servidor lê os documentos de acordo com as seguintes ações:

- Para importar ou recuperar resultados: documentos XML com a definição de esquema do Metadata-Algorithm (seção 5.3).
- Para fazer consultas: documento XML com a definição de esquema de consultas (seção 5.4).

O servidor tem a função de receber os documentos XML, ler o seu conteúdo e mapear na linguagem do banco de dados. Ao inserir um algoritmo, um identificador único global (UID) é criado para ele. Os dados de localização do algoritmo (URL) são utilizados para buscar os arquivos relacionados e armazená-los no banco de dados do repositório.

Para utilizar os algoritmos nas aplicações de recuperação de imagens, o desenvolvedor precisa apenas especificar o UID do algoritmo na declaração XML da aplicação.

5.6 Implementação do Repositório de Algoritmos

O Sistema de Gerenciamento de Banco de Dados utilizado para esta primeira implementação do Repositório é o MySQL. O esquema de bancos de dados é simples e facilmente mapeável sobre a estrutura do esquema XML.

Os documentos XML vão ser utilizados como transporte dos dados entre o cliente e banco de dados. Para utilizar XML como transporte, temos de garantir que toda a troca de informação vai ter um formato XML. Para importar ou recuperar resultados, os documentos XML apresentam a definição de esquema do exemplo da seção 5.3.5.

As consultas feitas nas linguagens de consulta tradicionais precisam ter um formato XML para serem transportadas pela rede, consistente com a arquitetura. Na seção 5.4 é apresentado um esquema simples para transportar consultas SQL/OQL em XML.

5.6.1 Armazenamento dos Algoritmos

Os metadados dos algoritmos são armazenados persistentemente no banco de dados, mas os algoritmos são armazenados por referência. Os arquivos que formam a execução do algoritmo são armazenados no sistema de arquivos.

O sistema possui um serviço para criar a localização de um algoritmo, apartir de um contexto ou caminho inicial ($HOME$/roira/pool/).

5.6.2 Geração de Identificafores para os Algoritmos

Um identificador de recursos (URI), menos comum, é o Nome Uniforme de Recursos (URN - *Universal Resource Name*). Um URN é diferente de um URL pois seu propósito principal é etiquetar persistentemente um recurso com um identificador. Este identificador é obtido de um conjunto de espaços de nomes definidos, cada um deles tem estabelecida sua própria estrutura de nomes e procedimentos de atribuição.

No repositório de algoritmos, o sistema precisa atribuir um identificador único para cada algoritmo de maneira que a chamada aos algoritmos garanta unicidade.

Estrutura do Identificador

Uma convenção de nomes é o conjunto de regras sintáticas que governam a geração de um nome. Essas regras determinam se um nome é válido ou não no contexto em que o nome é usado.

O identificador dos algoritmos do repositório de AMID é um URN da seguinte forma:

```
amid:roira:location:main-filename

location = URL:roira-directory
```

Sendo que,

amid:roira é o esquema do repositório,

URL é a localização do arquivo na rede,

roira-directory é uma variável que contém o diretório completo que o sistema cria para colocar o algoritmo, e

main-filename é o nome do arquivo principal para executar o algoritmo.

Por exemplo:

```
amid:roira:localhost:roira/pool/ex1:ex1.class
```

O diretório completo de um algoritmo é um caminho de diretório que está formado por um contexto inicial (roira/pool/) mais uma localização particular (ex1/) gerada para o algoritmo.

Podemos dizer que o UID de um algoritmo está formado pelo endereço que o localiza no sistema de arquivos.

A garantia de unicidade do UID consiste em que o próprio sistema gera o identificador. Inconsistência como a duplicação de um algoritmo é verificada utilizando o campo MAIN_FILENAME do documento XML. Esse campo tem dois elementos, a localização URL e o nome do arquivo. O endereço na Internet garante a unicidade deste recurso.

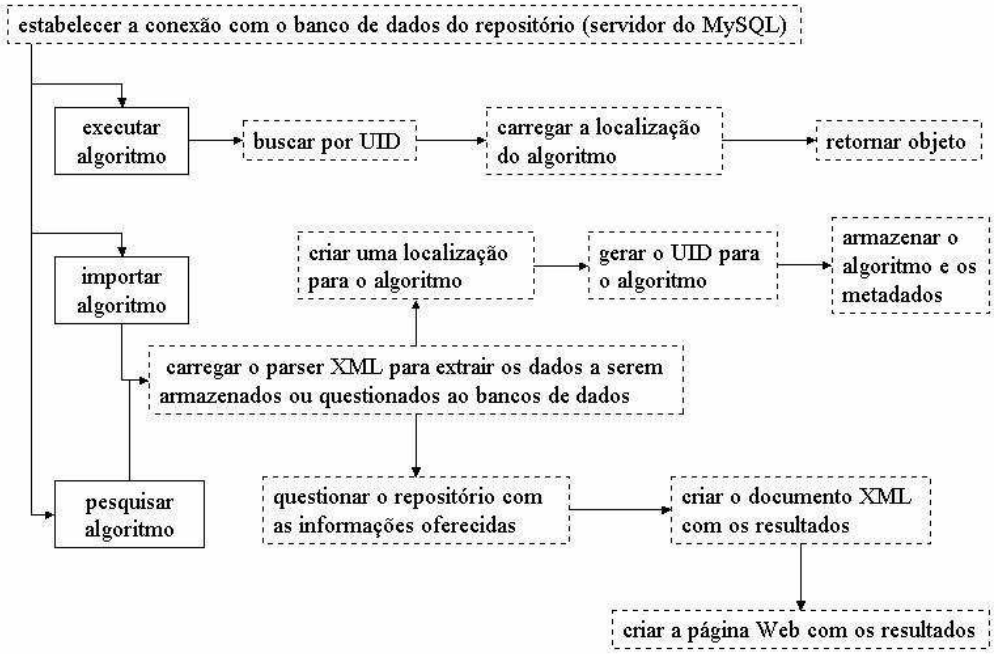
Nesta implementação, o repositório é desenvolvido localmente, onde os arquivos e o banco de dados encontram-se na mesma localização. Portanto, a localização de um algoritmo o do seguinte tipo:

```
location = roira-directory
```

5.6.3 Servidor

Nesta implementação, o servidor do sistema é um servidor Web. A tecnologia utilizada para analisar as páginas são os *servlets*. Os *servlets* são a resposta da tecnologia Java à programação CGI. São programas que se executam em um servidor Web e constroem páginas Web.

Quando as informações são fornecidas, elas são submetidas ao servidor web e o servlet é encarregado de processar os dados e criar a página Web de resultados. De maneira geral, no servidor é realizada a seguinte seqüência de ações.



No nosso caso, o sistema vai precisar mapear os algoritmos com os metadados armazenados no banco de dados. O UID armazenado no banco de dados tem a informação necessária para localizar o algoritmo na rede e no sistema de arquivos correspondente. Um serviço que mapeia os identificadores e as localizações foi criado, pois a referência da localização é parte do UID.

5.6.4 Interface Visual

A interface do repositório é implementada como uma aplicação Web. A página web inicial é mostrada na figura 5.3. O acesso ao repositório pode ser de dois modos. No primeiro modo, um documento XML pode ser diretamente importado no repositório.

A figura 5.4 mostra a página que um cliente pode usar para importar um algoritmo como documento XML, assim como os arquivos envolvidos na execução do algoritmo. O segundo modo é utilizar uma interface visual preenchendo os campos correspondentes para os algoritmos.

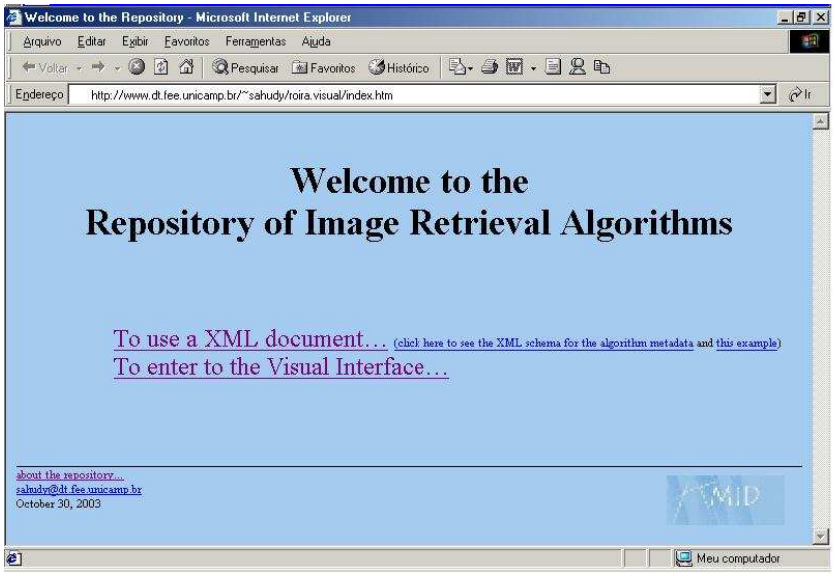


Figura 5.3: Página Web inicial da Interface do Repositório.

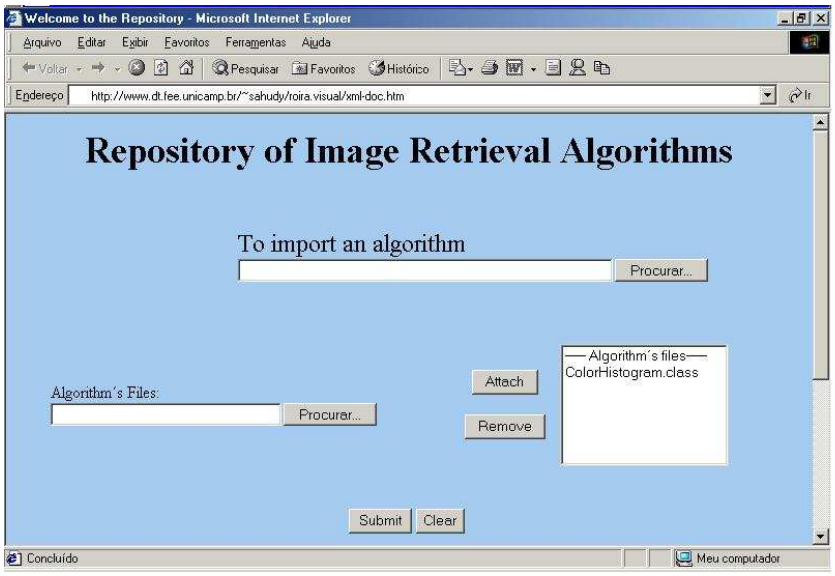


Figura 5.4: Página Web para importar um documento XML.

5.6.4.1 Importar Algoritmos

Ao importar uma operação no Repositório, os metadados devem ser preenchidos para serem incorporados no banco de dados junto com a operação, como mostra a figura 5.5.

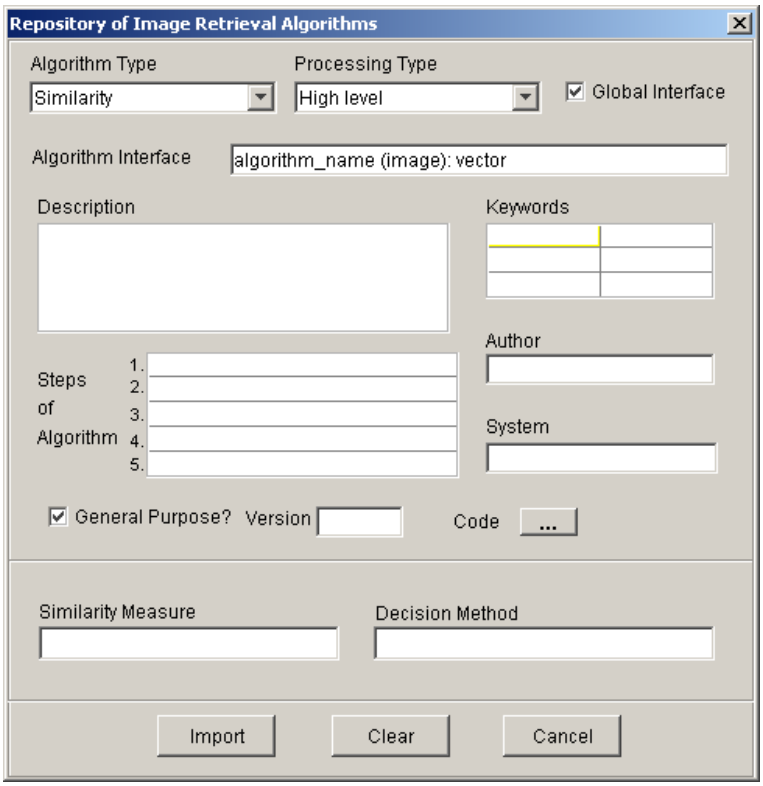


Figura 5.5: Interface para importar uma operação no Repositório de Operações.

Alguns metadados podem ser derivados de informações solicitadas nesta interface. Por exemplo: a localização original, o tamanho e a linguagem de programação da operação podem ser obtidos quando o código executável é importado; e se a operação é de propósito geral, não é preciso questionar sobre a temática de domínio específico associada.

Quando um conjunto de arquivos é selecionado como código executável da operação, a interface pede para escolher a classe principal. Se a linguagem de programação é Java, os possíveis métodos da interface da operação na classe principal são detectados. Desses métodos vai ser selecionado o correto.

5.6.4.2 Pesquisar sobre os Algoritmos

Na consulta, estão disponíveis duas opções: *browsing* ou acesso seqüencial às operações e, uma interface para pesquisar as operações utilizando restrições sobre os metadados, como mostra a figura 5.6.

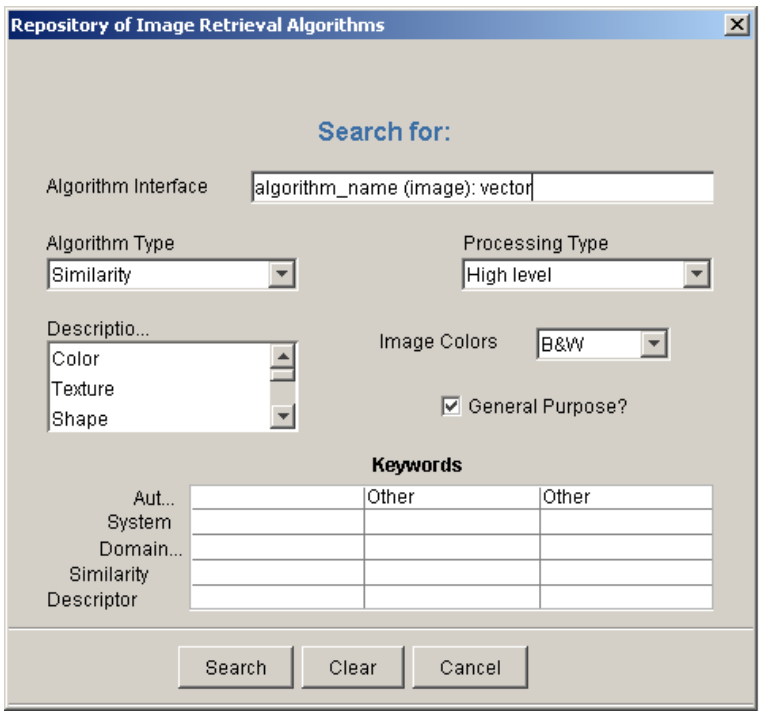


Figura 5.6: Interface para pesquisar uma operação no Repositório de Operações.

Uma vez definida a consulta, ela é enviada ao servidor que, com a utilização de índices tradicionais, busca no banco de dados a informação adequada.

A figura 5.7 mostra a tela de apresentação dos resultados de uma pesquisa, muito similar a tela do acesso seqüencial. Os resultados são apresentados parcialmente, somente o identificador, a interface, o tipo e a temática da operação. Para ter informações detalhadas (todos os metadados) sobre uma operação, marca-se o botão de seleção à esquerda da operação desejada e o botão *Show selection*.

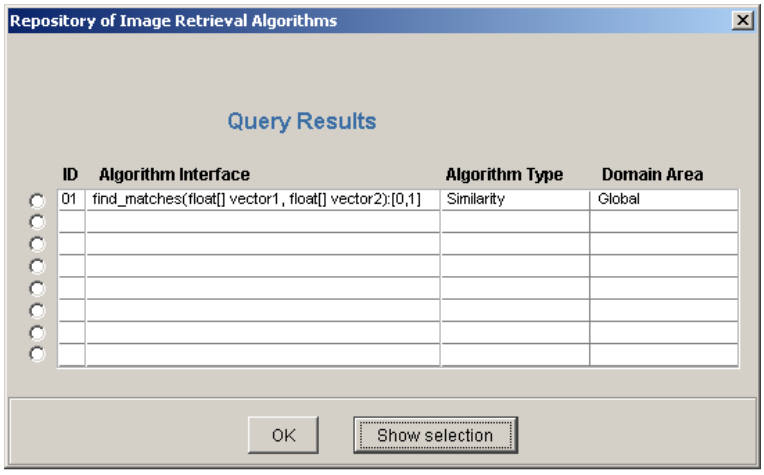


Figura 5.7: Interface para apresentar os resultados de uma pesquisa no Repositório de Operações.

Neste caso, estamos procurando pelo algoritmo de similaridade do exemplo acima. A tela de resultados tem um botão "Show selection"para mostrar mais informações sobre o algoritmo escolhido.

5.7 Sumário

Este capítulo descreve o Repositório de Algoritmos, sua arquitetura e uma implementação. O propósito do repositório é integrar as operações sobre imagens, de maneira que elas estejam disponíveis para serem consultadas e compartilhadas pelos usuários.

Uma definição de metadados para os algoritmos de recuperação de imagens e sua representação no padrão XML é apresentada. O XML é um padrão de formatação de dados que visa a extensibilidade dos dados que são representados no seu esquema.

A arquitetura do repositório apresenta três componentes fundamentais. O conjunto de bancos de dados é distribuído na rede. O servidor é o encarregado de processar todas as solicitações. A interface visual oferece uma maneira melhor de consultar o repositório.

A implementação do repositório foi desenvolvida na linguagem Java, com a tecnologia para interagir com o servidor Web. As telas da interface visual e as páginas Web são mostradas neste capítulo.

Outras operações de imagens como as operações de processamento (edição ou transformação das imagens) e compressão podem ser integradas ao repositório.

Capítulo 6

Conclusões e Trabalhos Futuros

Dificilmente seria possível desenvolver um sistema gerenciador de banco de dados especializado que atendesse adequadamente às necessidades de todas aplicações de bancos de dados de imagens, pois estas variam muito de acordo com o caráter da aplicação.

Este trabalho apresentou um modelo e um sistema de gerenciamento de imagens sobre bancos de dados. O modelo de gerenciamento é projetado como uma camada intermediária entre as aplicações e os sistemas de gerenciamento de bancos de dados. Integra as vantagens que oferecem os bancos de dados com a manipulação das imagens em um SGBDI. Baseia-se em padrões que ajudem à flexibilidade e portabilidade de aplicações tais como: o MPEG-7 para o modelo de dados, o ODMG para o modelo de consultas e o XML para padronizar os metadados dos algoritmos.

A nossa abordagem abrange um conjunto de tópicos relevantes às aplicações de recuperação de imagens, que não são oferecidas em todo seu conjunto pelos sistemas de gerenciamento já existentes. O nosso sistema aproveita características desenvolvidas por diferentes abordagens mas que todas elas não são vistas integradas em nenhum outro sistema. A seguir um resumo das contribuições do nosso trabalho.

- Busca representar as particularidades de cada aplicação dependente de domínio, definindo soluções de propósito geral.
- Modela a recuperação de imagens considerando múltiplos descritores das características visuais em junção com os atributos tradicionais.
- Modela as consultas expressando a natureza inexata das imagens e da comparação das imagens por similaridade.
- Estende a linguagem OQL a OQL-I para expressar um amplo estoque de consultas sobre imagens.
- Define as tuplas de operações que simplifica a manipulação dos algoritmos para as aplicações e os usuários.
- Define um conjunto de metadados para a manipulação dos algoritmos de recuperação de imagens.

- Define um catálogo do sistema que armazena e permite consultar meta-informação sobre o gerenciamento das aplicações.

O nosso sistema é voltado a facilitar o trabalho dos desenvolvedores de aplicações. O desenvolvedor tem duas maneiras de interagir com o sistema: utilizando a interface de AMID com a aplicação que oferece o suporte para declarar e manipular os bancos de dados de imagens; e os serviços orientados à manipulação de imagens como o repositório de algoritmos.

Os usuários das aplicações tiram proveito de nosso sistema devido a disponibilidade de um grande número de consultas sobre imagens, abstração de informação em vários níveis e conceitos que elevam o caráter semântico das consultas. Também, o mecanismo de retroalimentação proposto é simples e flexível, considera a subjetividade dos usuários que podem alterar os descritores, modificadores lingüísticos e operadores lógicos para cada consulta.

Trabalho Futuro

Vários tópicos relacionados ao gerenciamento de bancos de dados ainda não conseguiram soluções definitivas no projeto de nosso trabalho. A seguir propomos alguns deles que seriam interessantes de desenvolver para fazer de nossa abordagem uma solução ainda mais forte.

- Modelo de manipulação de imagens: especificar um modelo para manipular (inserir, modificar e remover) as imagens sobre os bancos de dados.
- Catálogo de conceitos: a criação de um catálogo de conceitos fuzzy para ser manipulado utilizando a linguagem OQL-I.
- Mecanismo de otimização eficiente para o processamento das consultas.
- Aspectos de implementação: implementar o mecanismo de retroalimentação de consultas é um aspecto interessante para que os usuários possam fazer testes sobre os bancos de dados.

Apêndice A

Glossário de Termos

Termo	Significado
AMID*	Architecture for Management of Image Databases
BD	Banco de Dados
BR	Bound Rectangle
CCD	Coarseness, Contrast, Directionality
DDL	Data Definition Language
DNS	Internet Domain Name System
FFT	Fast Fourier Transform
JDBC	Java DataBase Connectivity
LEA*	Linguagem de Especificação das Aplicações
MBR	Minnimum Bound Rectangle
MDI*	Módulo de Definição de Imagens
MFD	Modified Fourier Descriptor
MMI*	Módulo de Manipulação de Imagens
MPEG-7	Multimedia Content Description Interface
ODL	Object Definition Language
ODL-I*	Object Definition Language for Image
OML	Object Manipulation Language
ODMG	Object Data Management Group
OIF	Object Interchange Format
OMG	Object Management Group
OQL	Object Query Language
OQL-I*	Object Query Language for Image
QBE	Query By Example
SGBD	Sistema de Gerenciamento de Bancos de Dados
SGBDI	Sistema de Gerenciamento de Bancos de Dados de Imagens
SGBDO	Sistema de Gerenciamento de Bancos de Dados orientado a Objetos
SQL	Structure Query Language
SRI	Sistema ou aplicação de Recuperação de Imagens
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
XML	eXtensible Markup Language

*Termo específico ao nosso trabalho

Apêndice B

Tópicos Relevantes sobre a Lógica Fuzzy

Neste anexo apresentamos conceitos da teoria fuzzy que são relevantes no desenvolvimento do modelo de dados da nossa abordagem ao gerenciamento de imagens (ver capítulo 4).

A Lógica Fuzzy é uma técnica que incorpora a forma humana de pensar em um sistema. Sua característica especial é a de representar uma forma inovadora de manuseio de informações imprecisas. Fornece um método de traduzir expressões verbais, vagas, imprecisas e qualitativas, comuns na comunicação humana, em valores numéricos.

A lógica clássica (aristotélica) é bivalente, onde se reconhece somente dois valores: verdadeiro ou falso. A lógica fuzzy é multivalente, reconhece uma multitude de valores, assegurando que a verdade é uma questão de ponto de vista, definindo o grau de certeza em um intervalo numérico [0,1], onde a certeza absoluta é representada pelo valor 1.

A seguir são apresentados alguns conceitos básicos da teoria fuzzy considerados relevantes para nosso trabalho, acompanhados de exemplos [75].

Conjunto Fuzzy e Função de Pertinência

Por inferência lógica, a entrada ou condição e a saída ou conseqüência são associadas por regras de raciocínio, com graus de certeza no intervalo [0,1]. As operações de teoria de conjuntos fornecem um mapeamento entrada-saída. Para isto, é definida a pertinência de um elemento x em um conjunto A, indicada pelo símbolo $\in$: $x \in A$. Uma forma de se indicar essa pertinência é através de uma *função de pertinência* $\mu_A(x)$, cujo valor indica se o elemento x pertence ou não ao conjunto A. No exemplo seguinte, a função $\mu_A(x)$ é bivalente:

$$\mu_A(x) = \begin{cases} 1, x \in A \\ 0, x \notin A \end{cases}$$

Seja E um conjunto e x um elemento de E; então, o *subconjunto fuzzy* A de E é um conjunto de pares ordenados;

$$\{< x, \mu_A(x) >\}, \ \forall x \in \ E$$

Se $\mu_A(x)$ toma seus valores em um conjunto M, chamado por conjunto de pertinência, pode se afirmar que x toma seus valores em M através da função $\mu_A(x)$:

$$x \xrightarrow{\mu_A(x)} M$$

No exemplo anterior, $M = \{0, 1\}$ e a função de pertinência é booleana (binária). Se $M = [0, 1]$, o subconjunto A é um conjunto fuzzy e a função de pertinência é também fuzzy. Um conjunto é definido por seu vetor de pertinência, calculado a partir dos valores individuais de cada elemento do conjunto.

Universo de discurso

O *universo de discurso*, U, é um conjunto de valores finitos. Assim, todo conjunto fuzzy seria na realidade sempre um subconjunto.

Número fuzzy

Um *número fuzzy*, X, que está em um conjunto fuzzy em um universo de discurso U, possui algumas restrições que seguem:

- X deve ser normal: $max\ \mu_X(u) = 1, \ u \in \ U$
- X deve ser convexo: as funções de pertinência de números ou conjuntos fuzzy devem possibilitar unicidade na avaliação numérica do valor no eixo horizontal.

Variáveis Lingüísticas

Uma variável lingüística, u, no universo de discurso U, é definida em um conjunto de termos, nomes ou rótulos, $T(u)$, com cada valor sendo um número fuzzy definido em U. Por exemplo, se u for *altura*, então seu conjunto de termos $T(u)$ poderia ser:

$$T(altura) = \{baixo, mediano, alto\}$$

sobre o universo de discurso $U = [0, 2.50]$, onde *baixo, mediano, alto* são termos lingüísticos da grandeza *altura*. A figura B.1 representa a definição dos subconjuntos fuzzy definidos sobre a variável lingüística *altura*.

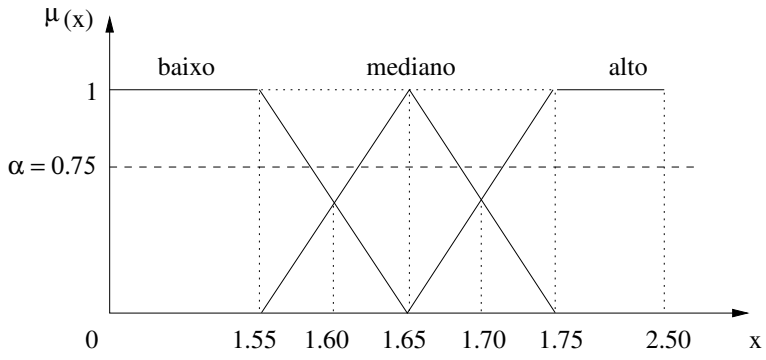


Figura B.1: Definição dos subconjuntos fuzzy sobre a grandeza *altura*, U = [0,2.50].

Conjunto Suporte e α-cut

O suporte de um conjunto fuzzy A é um subconjunto do conjunto universo U, tal que

$$Supp(A) = \{x \mid x \in\ U\ and\ \mu_A(x) > 0\}$$

O conjunto A poderia ser descrito como:

$$A = \{\mu_A(x)/x \mid x \in\ Supp(A)\}$$

Quer dizer que, somente os elementos x cuja função de pertinência $\mu_A(x)$ é maior do que zero pertencem ao conjunto A. Para o universo U = *altura*, um subconjunto A pode ser a variável lingüística *mediano* (A = [1.55,1.75]) . Como mostra a figura B.1, o conjunto Supp(mediano) = [1.56,1.74], onde a função de pertinência $\mu_{A=mediano}(x) > 0$.

Um conceito importante é o α-cut, definido como um conjunto não-fuzzy do universo cujos elementos possuem o valor da função de pertinência maior ou igual a um valor α.

$$A_\alpha = \{x \mid \mu_A(x) \geq\ \alpha\}$$

O *alpha*-cut de um conjunto é um subconjunto do conjunto suporte. Os valores de α podem ser escolhidos de maneira conveniente, para selecionar determinados subconjuntos do universo. Na figura B.1 é definido $\alpha\ = 0.75$. O subconjunto definido sobre o valor lingüístico *alto*, cuja $\mu_{alto}(x) \in [0.75, 1]$ pode ser interpretado como o conjunto *muito alto*, e assim por diante para os outros termos.

Divisa (*hedge*) lingüística

O operador de concentração concentra os elementos fuzzy reduzindo o grau de pertinência proporcionalmente para elementos de baixo grau de pertinência. O operador de dilatação dilata os elementos fuzzy incrementando o grau de pertinência para os elementos com baixo grau de

pertinência.

Esses operadores são utilizados para representar divisas lingüísticas que agem como modificadores das variáveis lingüísticas representadas nos conjuntos fuzzy. O operador de concentração pode ser representado como o exponente μ^2 e o operador de dilatação como $\mu^{1/2}$.

As divisas lingüísticas e o α-cut são operações que podem ser utilizadas como modificadores, para restringir o conjunto fuzzy em subconjuntos, como pode se observar na figura B.2.

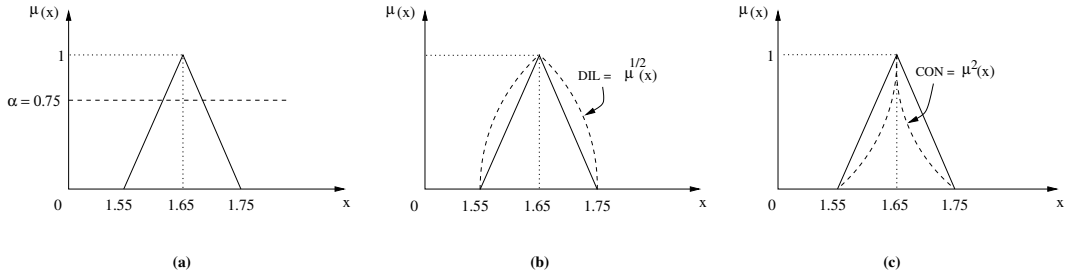


Figura B.2: Operadores fuzzy. (a) α-cut. (b) Operador de Dilatação. (c) Operador de Concentração.

O *alpha*-cut do esquema em (a) define um subconjunto a partir de $\alpha = 0.75$. No caso (b) da figura, o subconjunto definido aumenta o grau de pertinência dos números fuzzy ao conjunto $(\mu_{mediano}(x))$. No gráfico (c), o subconjunto definido diminui o grau de pertinência dos números fuzzy ao conjunto $(\mu_{mediano}(x))$.

Operações entre conjuntos fuzzy

A seguir, as operações entre conjuntos fuzzy serão definidas. Dado U o universo de discurso, U pode ser composto de subconjuntos fuzzy, tal que $U = \{U_i\}$. Então, a intersecção dos subconjuntos $\{U_i\}$, tal como mostra (a) (operador AND) e a união dos subconjuntos $\{U_i\}$, tal como mostra (b) (operador OR).

(a) operador AND: $\mu_{\{U_i\}}(x) = U_1 \bigcap U_2 = min[\mu_{U_i}]$

(b) operador OR: $\mu_{\{U_i\}}(x) = U_1 \bigcup U_2 = max[\mu_{U_i}]$

Apêndice C

Especificação da Linguagem de Consulta

O conjunto de operadores espaciais foi tomado da linguagem MOQL [46].

Expressões Espaciais

```
spatial_expression ::= spatial_function_expression
comparison_operator
                      spatial_function_expression

spatial_expression ::= spatial_object spatial_predicate
spatial_object

spatial_function_expression ::= spatial_term
                      [arithmetic_operator spatial_function_expression]

spatial_term ::= float_literal | integer_literal | spatial_object
               | spatial_function (query)

spatial_function ::= nearest | farthest | length | slope |
centroid | area
             | mbr | distance | intersect | interior | exterior | perimeter

spatial_predicate ::= generic_spatial_predicate |
topological_predicate
                     | directional_predicate | picture_predicate

generic_spatial_predicate ::= nearest | farthest | intersect |
inside
                             | cross | centroid | midpoint
```

	point	line	region
point	nearest, farthest	within, midpoint	centroid, inside
line	cross	intersect	inside, cross
region	cover	cover, cross	topological_predicate directional_predicate

Tabela C.1: Predicados Espaciais

```
topological_predicate ::= equal | touch | inside | contains |
overlap
                        | disjoint | cover | coveredBy

directional_predicate ::= positional_predicate | depth_predicate
                        | simple_directional_predicate
                        | complex_directional_predicate

positional_predicate ::= left | right | above | below

depth_predicate ::= back | front

simple_directional_predicate ::= north | south | west | east
                       | northwest | northeast | southwest | southeast

complex_directional_predicate ::= depth_predicate -
positional_predicate
                          | depth_predicate - simple_directional_predicate

picture_predicate ::= equal | coincident | subpicture | similar

spatial_object ::= query | point (integer_literal ,
integer_literal)
                 | line (point , point) | window (point , point)
                 | circle (point , radius)
```

Para análise semântica dos predicados e das funções, apresentamos as tabelas C.1 e C.2. Foram definidas três primitivas espaciais: *point*, *line* e *region*.

Os operandos para os predicados espaciais precisam ser do mesmo tipo ou compatíveis; por exemplo: o predicado *nearest* pode ser aplicado somente a dois pontos, e o predicado *within* somente a um ponto e uma linha.

O retorno de uma função espacial refere-se ao tipo do objeto retornado pela função. A coluna *value* especifica que a função retorna um valor escalar, por exemplo: area(region) = valor.

Retorno	point	line	region	value
point	nearest, farthest		region	
line	intersect	intersect	region	length, slope
region	centroid		interior, exterior, mbr	area, perimeter

Tabela C.2: Funções Espaciais

Apêndice D

Especificação do XML-Schema para os Algoritmos

Nesta seção especificamos o esquema XML completo dos metadados dos algoritmos de recuperação de imagem.

```
<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.dt.fee.unicamp.br/roira"
xmlns="http://www.dt.fee.unicamp.br/roira>

<xs:complexType name="locationType">
  <xs:sequence>
    <xs:element name="NAMESPACE" type="xs:anyURI"/>
    <xs:element name="FILENAME" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="signatureType">
  <xs:sequence>
    <xs:element name="LOCATION" type="locationType" minOccurs="0"/>
    <xs:element name="METHOD" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="ALGORITHM_METADATA"/>

 <xs:complexType>

  <xs:attribute name="NAMESPACE" type="xs:anyURI" use="required"/>
```

```
  <xs:sequence>

    <xs:element name="SIGNATURE" type="signaturetype"/>

    <xs:element name="BASIC_INFO_METADATA">

    <xs:element name="CREATION_METADATA" minOccurs="0">

    <xs:element name="CONTENT_METADATA" minOccurs="0">

  </xs:sequence>

 </xs:complexType>

</xs:element>

<xs:element name="BASIC_INFO_METADATA">
   <xs:complexType>
      <xs:attribute name="VERSION" type="xs:string"/>
      <xs:sequence>
         <xs:element name="MAIN_FILENAME" type="locationType"/>
         <xs:element name="COMPLEMENTARY_FILENAME" type="locationType"
                                 minOccurs="0" maxOccurs="unbounded"/>
         <xs:element name="PROGRAMMING_LANG" type="xs:string"/>
         <xs:element name="PL_VERSION" type="xs:string" minOccurs="0"/>
         <xs:element name="REQUIREMENTS" minOccurs="0"/>
         <xs:element name="SRC_LOCATION" type="xs:anyURI"  minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="REQUIREMENTS">
  <xs:complexType>
      <xs:sequence>
         <xs:element name="SOFTWARE" minOccurs="0"/>
         <xs:element name="HARDWARE" minOccurs="0"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="SOFTWARE">
  <xs:complexType>
      <xs:sequence>
```

```
         <xs:element name="PLATFORM" type="xs:string" minOccurs="0"/>
         <xs:element name="LIBRARY"  type="xs:string" minOccurs="0"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="HARDWARE">
  <xs:complexType>
      <xs:sequence>
         <xs:element name="PROCESSOR" type="xs:string" minOccurs="0"/>
         <xs:element name="HD_SPACE"  type="xs:string" minOccurs="0"/>
         <xs:element name="RAM"       type="xs:string" minOccurs="0"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="CREATION_METADATA">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="CREATION_DATE" type="xs:dateTime"   minOccurs="0" />
         <xs:element name="USE_HISTORY"                        minOccurs="0" />
         <xs:element name="CREATOR_METADATA"                   minOccurs="0" />
      </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="USE_HISTORY">
   <xs:complexType>
      <xs:sequence>
        <xs:element name="SYSTEM" type="xs:string" minOccurs="0"/>
        <xs:element name="RELATIONSHIPS"  minOccurs="0"  maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="RELATIONSHIPS">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:anyURI">
         <xs:attribute name="TYPE" type="xs:string" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="CREATOR_METADATA">
  <xs:complexType>
      <xs:choice maxOccurs="unbounded"/>
         <xs:element ref="PERSON"/>
         <xs:element ref="ORGANIZATION"/>
      </xs:choice>
  </xs:complexType>
</xs:element>

<xs:element name="PERSON">
   <xs:complexType>
      <xs:attribute name="TYPE" type="xs:string"/>
      <xs:sequence>
         <xs:element name="PNAME"  type="xs:string"/>
         <xs:element ref="ORGANIZATION"  minOccurs="0"/>
         <xs:element ref="CONTACT_INFO"  minOccurs="0"  maxOccurs="10"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>

<xs:element name="ORGANIZATION">
  <xs:complexType>
      <xs:sequence>
         <xs:element name="ORG_NAME" />
         <xs:element ref="CONTACT_INFO" minOccurs="0"  maxOccurs="10"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="CONTACT_INFO">
  <xs:complexType>
      <xs:sequence>
         <xs:element name="ADDRESS"  type="xs:string"/>
         <xs:element name="EMAIL"  type="xs:string"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="CONTENT_METADATA">
  <xs:complexType>
      <xs:sequence>
         <xs:element ref="ALGORITHM_TYPE"/>
         <xs:element name="APP_DOMAIN" type="xs:string" minOccurs="0"/>
```

```
        <xs:element name="GENERAL_PURPOSE" type="xs:boolean" minOccurs="0"/>
        <xs:element name="KEYWORD" type="xs:string" maxOccurs="unbounded"/>
        <xs:element name="DESCRIPTION" type="xs:string"/>
        <xs:element ref="STEP_DESCRIPTION" minOccurs="0"/>
        <xs:element ref="UD_DATATYPES" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="COMMENTS" type="xs:string"  minOccurs="0"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="ALGORITHM_TYPE">
  <xs:complexType>
      <xs:choice>
        <xs:element ref="SIMILARITY"/>
        <xs:element ref="DESCRIPTOR_EXTRACTION"/>
        <xs:element ref="INDEXING"/>
        <xs:element ref="NORMALIZATION"/>
      </xs:choice>
  </xs:complexType>
</xs:element>

<xs:element name="STEP_DESCRIPTION">
  <xs:complexType>
      <xs:sequence>
        <xs:element name="STEP" type="xs:string" maxOccurs="unbounded"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="UD_DATATYPES">
  <xs:complexType>
      <xs:sequence>
        <xs:element name="NAME" type="xs:string"/>
        <xs:element name="TYPE" type="xs:string"/>
        <xs:element name="DESCRIPTION" type="xs:string"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="DESCRIPTOR_EXTRACTION">
  <xs:complexType>
      <xs:sequence>
        <xs:element name="IMAGE_FEATURE" type="xs:string"/>
        <xs:element name="DESCRIPTOR" type="xs:string"/>
```

```
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="SIMILARITY">
  <xs:complexType>
      <xs:sequence>
         <xs:element name="DECISION_METHOD" type="xs:string"/>
         <xs:element name="METRIC" type="xs:string"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="NORMALIZATION">
  <xs:complexType>
      <xs:sequence>
         <xs:element name="METRIC"  type="xs:string"/>
      </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="INDEXING">
  <xs:complexType>
     <xs:sequence>
        <xs:element name="INDEXING_STRUCTURE"              type="signatureType"/>
        <xs:element name="ADD_ACCESS_METHOD_SIGNATURE"     type="signatureType"/>
        <xs:element name="QUERY_ACCESS_METHOD_SIGNATURE"   type="signatureType"/>
        <xs:element name="DELETE_ACCESS_METHOD_SIGNATURE"  type="signatureType"/>
     </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Bibliografia

[1] Karl Aberer and Wolfgang Klas. The Impact of Multimedia Data on Database Management System. Technical Report TR-92-065, International Computer Science Institute (ICSI), Berkeley, CA, USA, 1992.

[2] Donald A. Adjeroh and Kingsley C. Nwosu. Multimedia database management - requirements and issues. *IEEE Multimedia*, 4(3), Julho-Setembro 1997.

[3] Selim Aksoy and Robert M. Haralick. Feature normalization and likelihood-based similarity measures for image retrieval. *Pattern Recognition Letters*, 22(5):563–582, 2001.

[4] P. Androutsos, A. Kushki, K.N. Plataniotis, and A.N. Venetsanopoulos. Fuzzy agregations of palette colors for hybrid querying of fine art image databases. In *Proceedings of the* 14^{th} *International Conference on Digital Signal Processing*, pages 115–120, 2002.

[5] Y. Alp Aslandogan and Clement T. Yu. Techniques and systems for image and video retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):56–63, Janeiro/Fevereiro 1999.

[6] S. Bhagavathy, S. Newsam, and B. Manjunath. Modeling object classes in aerial images using texture motifs. In *Proceedings of the International Conference on Pattern Recognition*, August 2002. http://citeseer.nj.nec.com/bhagavathy02modeling.html.

[7] Chad Carson, Megan Thomas, Serge Belongie, Joseph M. Hellerstein, and Jitendra Malok. BlobWorld: a system for region-based image indexing and retrieval. In *Proceedings of the Third International Conference on Visual Information Systems*, Junho 1999.

[8] R.G.G. Cattell, Douglas Barry, and et al. *The Object Data Standard: ODMG 3.0.* Morgan Kaufmann, 2000.

[9] A. Celentano and S. Sabbadin. Multiple features indexing in image retrieval systems. In *Proceedings of the First European Workshop on Content-Based Multimedia Indexing, CBMI 99*, 1999.

[10] K. Chakrabarti and S. Mehrotra. High dimensional feature indexing using hybrid trees. In *Proceedings of the* 15^{th} *International Conference on Data Engineering (ICDE)*, 1999.

[11] Paolo Ciaccia and Marco Patella. Searching in metric spaces with user-defined and approximate distances. *ACM Transactions on Database Systems (TODS)*, 27(4):398–437, December 2002.

[12] NISO Standards Committee. Data Dictionary: Technical Metadata for Digital Still Images, Julho 2000. URL: www.niso.org.

[13] Luis Mariano del Val Cura. *Recuperação por Conteúdo de Imagens em Sistemas de Informação Geográficos*. PhD thesis, Universidade Estadual de Campinas (UNICAMP), 2002.

[14] Digital Imaging Group DIG35. Metadata for Digital Images, Abril 2001. URL: www.digitalimaging.org.

[15] Rinie Egas, Dionysius P. Huijsmans, Michael S. Lew, and Nicu Sebe. Adapting k-d trees to visual retrieval. In *Visual Information and Information Systems*, pages 533–540, 1999.

[16] Christos Faloutsos and Douglas W. Oard. A survey of information retrieval and filtering methods. Technical Report CS-TR-3514, Information Filtering Project, University of Maryland, College Park, 1995. URL: citeseer.nj.nec.com/faloutsos96survey.html.

[17] Leonidas Fegaras. VOODOO: A visual object-oriented database language for ODMG OQL. In *ECOOP Workshop on Object-Oriented Databases*, pages 61–72, 1999. citeseer.nj.nec.com/fegaras99voodoo.html.

[18] Leonidas Fegaras, Chandrasekhar Srinivasan, Arvind Rajendran, and David Maier. lambdaDB: An ODMG-Based Object-Oriented DBMS. In *SIGMOD Conference*, 2000. citeseer.nj.nec.com/355051.html.

[19] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yauker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–31, Setembro 1995.

[20] Consortium for the Computer Interchange of Museum Information (CIMI). Guide to Best Practice: Dublin Core, Abril 2000. URL: www.cimi.org.

[21] J. Fridrich. Visual hash for oblivious watermarking. In *Proceedings of SPIE Photonic West Electronic Imaging*, 2000. http://citeseer.nj.nec.com/fridrich00visual.html.

[22] C.S. Fuh, S.W. Cho, and K. Essig. Hierarchical color image region segmentation for content-based image retrieval system. *IEEE Transactions on Image Processing*, 9(1):156–162, Janeiro 2000.

[23] T. Gevers and A.W.M. Smeulders. Pictoseek: Combining color and shape invariant features for image retrieval. *IEEE Transactions on Image Processing*, 9(1):102–119, Janeiro 2000.

[24] M.O. Güld, B.B. Wein, D. Keysers, C. Thies, M. Cohnen, H. Schubert, and T.M. Lehmann. A distributed architecture for content-based image retrieval in medical applications. In *Proceedings of the 2^{nd} International Workshop on Pattern Recognition in Information Systems*, pages 299–314, 2002.

[25] Yihong Gong. *Intelligent Image Databases. Towards Advanced Image.* Kluwer Academic Publishers, 1998.

[26] Abby Goodrum. Image information retrieval: An overview of current research. *Special Issue on Information Science Research*, 3(2), 2000. URL: citeseer.nj.nec.com/goodrum00image.html.

[27] R. Gray. Content-based image retrieval: Color and edges. Technical report, Department of Computer Science, Darmouth College, 1995.

[28] Informix Development Group. Datablade developer's kit user's guide. http://www.informix.com/answers/oldsite/answers/english/dbmd91wn.htm, Julho 1997.

[29] Informix Development Group. Extending informix-universal server: Data types. http://www.informix.com/answers/oldsite/answers/english/ius91ux.htm, Março 1997.

[30] MPEG Requirements Group. MPEG−7 Context, Objectives and Technical Roadmap, v. 12, Julho 1999. URL: www.darmstadt.gmd.de/mobile/MPEG7/Documents.html.

[31] MPEG Requirements Group. MPEG−7 Requirements Document, v.14, Março 2001. URL: www.darmstadt.gmd.de/mobile/MPEG7/Documents.html.

[32] Object Management Group. OMG homepage. http://www.omg.org/, 2002.

[33] Oracle Development Group. Oracle7 server concepts. http://www.oracle.com/support/products/docs/html/a32534-1.pdf, Fevereiro 1996.

[34] Oracle Development Group. Oracle8i objects and extensibility option. http://www.oracle.com/database/, Fevereiro 1999.

[35] The PostgreSQL Development Group. Postgresql programmer's guide. Ed. by Thomas Lockhart, Fevereiro 1998.

[36] Venkat N. Gudivada, V. V. Raghavan, and K. Vanapipat. Unified approach to data modeling and retrieval for a class of image database applications. In S. Jajodia and V. S. Subrahmanian, editors, *Multimedia Database Systems: Issues and Research Directions*, pages 37–78. Springer-Verlag, 1995.

[37] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of ACM-SIGMOD International Conference on Management of Data*, pages 47–57, Junho 1984.

[38] Jennifer Hallinan and Paul Jackway. Co-operative evolution of a neural classifier and feature subset. *Lecture Notes in Computer Science*, 1585:397–404, 1999.

[39] Andreas Henrich and Günter Robbert. $POQL^{MM}$: A query language for structured multimedia documents. In *Proceedings 1^{s}t International Workshop on Multimedia Data and Document Engineering (MDDE'01)*, pages 17–26, July 2001. citeseer.nj.nec.com/henrich01poqlmm.html.

[40] Andreas Henrich and Gunter Robbert. MARS: A retrieval service for multimedia authoring environments. In *ADBIS-DASFAA Symposium*, pages 88–98, 2000. citeseer.nj.nec.com/410587.html.

[41] Nancy A. Van House, Mark H. Butler, Virginia E. Ogle, and Lisa Schiff. User-Centered Iterative Design for Digital Libraries. The Cypress Experience. *D-Lib Magazine*, 1996.

[42] A. Jaimes and S. Chang. A conceptual framework for indexing visual information at multiple levels. In *Proceedings of IS&T/SPIE Internet Imaging*, Janeiro 2000. URL: citeseer.nj.nec.com/jaimes00conceptual.html.

[43] Ioannis Kompatsiaris, Evangelia Triantafyllou, and M. G. Strintzis. A World Wide Web Region-Based Image Search Engine. In *Proceedings of the IWDC*, 2002.

[44] Denis Lee, Ron Barber, Wayne Niblack, Myron Flickner, Jim Hafner, and Dragutin Petkovic. Complex queries for a query-by-content image database. Technical report, IBM Research Division, Fevereiro 1994.

[45] John Z. Li, M. Tamer Özsu, and Duane Szafron. MOQL: A Multimedia Object Query Language. In *Third International Workshop on Multimedia Information Systems*, pages 19–28, Setembro 1997. citeseer.nj.nec.com/li97moql.html.

[46] John Z. Li, M. Tamer Özsu, and Duane Szafron. Multimedia Extensions to Database Query Languages. Technical Report TR-97-01, Department of Computing Science, University of Alberta, Janeiro 1997.

[47] Wen-Syan Li, K. Selçuk Candan, Kyoji Hirata, and Yoshinori Hara. A hybrid approach to multimedia database systems through integration of semantics and media-based search. In *WWCA'97, Proceedings*, volume 1274, pages 182–197. Springer-Verlag, 1997.

[48] Y. Liu, N.A. Lazar, and W.E. Rothfus. Semantic-based biomedical image indexing and retrieval. In *Proceedings of the International Conference on Diagnostic Imaging and Analysis (ICDIA 2002)*, 2002.

[49] Hongjun Lu, Beng-Chin Ooi, and Kian-Lee Tan. Efficient image retrieval by color contents. In *Proceedings of 1th International Conference ADB-94. Applications of Databases*, pages 95–108. Lecture Notes in Computer Sciences 819, 1994.

[50] Wei-Ying Ma and B. S. Manjunath. Netra: A toolbox for navigating large image databases. *Multimedia Systems*, 7(3):184–198, 1999.

[51] S. Mukherjea, K. Hirata, and Y. Hara. AMORE: a World Wide Web Image Retrieval Engine. *World Wide Web, Baltzer*, 2(3):115–132, 1999.

[52] Apostol Natsev, Rajeev Rastogi, and Kyuseok Shim. WALRUS: a similarity retrieval algorithm for image databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 395–406. ACM Press, 1999.

[53] Surya Nepal, M.V.Ramakrishna, and James A.Thom. A Fuzzy Object Query Language (FOQL) for Image Databases. In *Proceedings of the 6th International Conference on Database Systems for Advanced Applications*, 1999. URL: citeseer.nj.nec.com/90616.html.

[54] Surya Nepal and M. V. Ramakrishna. Query processing issues in image (multimedia) databases. In *ICDE*, pages 22–29, 1999. citeseer.nj.nec.com/nepal99query.html.

[55] Youping Niu, M. Tamer Özsu, and Xiaobo Li. $2d - h$ trees: An index scheme for conten-based retrieval of images in multimedia systems. In *Proceedings of IEEE International Conference on Intellingent Processing Systems (ICIPS'97)*, 1997. URL: www.cs.ualberta.ca/ database/research/ImageDB/publications.html.

[56] K.C. Nwosu, B. Thuraisingham, and P. Bruce Berra. *Multimedia Database Systems. Design and Implementation Strategies.* Kluwer Academic Publishers, 1996.

[57] Virginia E. Ogle and M. Stonebraker. Chabot: Retrieval from a relational database of images. *IEEE Computer*, 28(9):40–48, Setembro 1995.

[58] V. Oria, M. Ozsu, X. Li, L. Liu, J. Li, Y. Niu, and P. Iglinski. Modeling images for content-based queries: The DISIMA approach. In *Proceedings of 2nd International Conference of Visual Information Systems, San Diego, California*, pages 339–346, Dezembro 1997. URL: citeseer.nj.nec.com/oria97modeling.html.

[59] Vincent Oria, M. Tamer Ozsu, Bing Xu, L. Irene Cheng, and Paul Iglinski. VisualMOQL: The DISIMA Visual Query Language. In *IEEE International Conference on Multimedia Computing and Systems*, volume 1, pages 536–542, Junho 1999. citeseer.nj.nec.com/oria99visualmoql.html.

[60] Vincent Oria and M. Tamer Özsu. Views or points of view on images. *International Journal of Image and Graphics*, 2002.

[61] Michael Ortega, Yong Rui, Kaushik Chakrabarti, Kriengkrai Porkaew, Sharad Mehrotra, and Thomas S. Huang. Supporting ranked boolean similarity queries in MARS. *Knowledge and Data Engineering*, 10(6):905–925, 1998. URL: citeseer.nj.nec.com/article/ortega98supporting.html.

[62] Paul Pazandak and Jaideep Srivastava. Evaluating Object DBMSs for Multimedia. *IEEE Multimedia*, 4(3), Julho-Setembro 1997.

[63] A. Pentland, Rosalind W. Picard, and S. Sclaroff. Photobook: Tools for content-based manipulation of image databases. In *Proceedings of SPIE Conference on Storage and Retrieval of Image and Video Databases II*, pages 34–47, San Jose, CA, fevereiro 1994.

[64] E. Petrakis and C. Faloutsos. Similarity searching in large image databases. Technical report, Department of Computer Science, University of Maryland, 1994. CS-TR-3388.

[65] Rosalind W. Picard, Thomas P. Minka, and Martin Szummer. Modeling user subjectivity in image libraries. In *Proceedings of the IEEE International Conference on Image Processing (ICIP'96)*, volume 2, pages 777–780, Lausanne, Switzerland, Setembro 1996.

[66] M. V. Ramakrishna and S. Nepal. Similarity query processing in image databases. Technical Report 2000/61, School of Computer Science and Software Engineering, Monash University, 2000. citeseer.nj.nec.com/ramakrishna00similarity.html.

[67] M.V. Ramakrishna and Santha Sumanasekara. Indexing multidimensional feature data, 1999.

[68] E. Rasmussen. Indexing images. *Annual Review of Information Science and Technology*, 32:169–196, 1997.

[69] Y. Rui, T. Huang, and S. Chang. Image retrieval: current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10(4):39–62, Abril 1999. URL: citeseer.nj.nec.com/rui99image.html.

[70] Y. Rui, T. Huang, and S. Mehrotra. Content-based image retrieval with relevance feedback in MARS. In *Proceedings of IEEE International Conference on Image Processing*, 1997. URL: citeseer.nj.nec.com/rui97contentbased.html.

[71] Y. Rui, T. Huang, and S. Mehrotra. MARS and its applications to MPEG, Julho 1997. URL: citeseer.nj.nec.com/rui97mars.html.

[72] Yong Rui, Thomas S. Huang, and Sharad Mehrotra. Relevance feedback techniques in interactive content-based image retrieval. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 25–36, 1998.

[73] Yong Rui, Thomas S. Huang, and Sharad Mehrotra. A novel relevance feedback technique in image retrieval. In *Proceedings of the Seventh ACM International Conference on Multimedia*, pages 67–70, 1999.

[74] Eugenio Di Sciascio, G. Mingolla, and Marina Mongiello. Content-Based Image Retrieval over the Web Using Query by Sketch and Relevance Feedback. In *Proceedings of the Third International Conference on Visual Information and Information Systems (VISUAL'99), Lecture Notes in Computer Science*, volume 1614, pages 123–130, 1999.

[75] Ian S. Shaw and Marcelo Godoy Simões. *Controle e Modelagem Fuzzy.* Edgard Blücher ltda., 1999.

[76] Gholamhosein Sheikholeslami and Aidong Zhang. Feature visualization and analysis for image classification and retrieval. In *Proceedings of 2^{nd} International Conference on Visual Information Systems, San Diego*, Dezembro 1997.

[77] J. Smith and S. Chang. Multi-stage classification of images from features and related text. In *Proceedings of $4^{t}h$ Europe EDLOS Workshop*, Agosto 1997. URL: citeseer.nj.nec.com/smith97multistage.html.

[78] J. R. Smith and S. F. Chang. Visualseek: A fully automated content-based image query system. In *Proceedings of ACM Multimedia Conference*, pages 87–98, Boston, USA, 1996.

[79] J. R. Smith and S. F. Chang. Visually searching the Web for content. *IEEE Multimedia Magazine*, 4(3):12–20, 1997.

[80] Subramanya Sanan Srakaew. Content-based retrieval of images using pre-determined image classes, 1998. URL: citeseer.nj.nec.com/322515.html.

[81] Rohini K. Srihari. Automatic indexing and content-based retrieval of captioned images. *IEEE Computer*, 28(9):49–56, Setembro 1995.

[82] Rohini K. Srihari and Z.F. Zhang. Exploiting multimodal context in image retrieval. *Library Trends*, 48(2):496–520, 1999.

[83] Rohini K. Srihari, Zhongfei Zhang, and Aibing Rao. Intelligent indexing and semantic retrieval of multimodal documents. *Information Retrieval*, 2(2):245–275, 2000.

[84] Michael Stonebraker and Paul Brown. *Object-Relational DBMSs: Tracking The Next Great Wave*. Morgan Kaufmann, 1999.

[85] S. Subramanya. A scheme for automated classification of images. In *International Conference on Computer Applications in Industry and Engineering (CAINE-98), Las Vegas*, Novembro 1998. URL: citeseer.nj.nec.com/article/subramanya98scheme.html.

[86] Michael J Swain, Charles Frankel, and Vassilis Athitsos. WebSeer: An image search engine for the World Wide Web. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, 1997.

[87] Remco C. Veltkamp and Mirela Tanase. Content-based image retrieval systems: A survey. Technical Report UU-CS-2000-34, Department of Computing Science, Utrecht University, Outubro 2000. citeseer.nj.nec.com/373932.html.

[88] W3C. Extensible Markup Language (XML) 1.0 (Second Edition), 2000. http://www.w3.org/TR/REC-xml.

[89] Kyoji Hirata e Yoshinori Hara Wen-Syan Li, K. Selçuk Candan. Facilitating multimedia database exploration through visual interfaces and perpetual query reformulations. In *Proceedings of the 23rd VLDB Conference*, pages 538–547, Athens, Greece, 1997.

[90] David A. White and Ramesh Jain. Similarity indexing: Algorithms and performance. In *Proceedings of SPIE Storage and Retrieval for Image and Video Databases IV*, 1996. URL: citeseer.nj.nec.com/white96similarity.html.

[91] M. E. J. Wood, N. W. Campbell, and B. T. Thomas. Iterative refinement by relevance feedback in content-based digital image retrieval. In *Proceedings of ACM Multimedia 98*, pages 13–20, Bristol, UK, Setembro 1998.

[92] Atsuo Yoshitaka and Tadao Ichikawa. A survey on content-based retrieval for multimedia databases. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):81–93, Janeiro/Fevereiro 1999.

[93] Carlo Zaniolo, Stefano Ceri, Christos Faloutsos, Richard T. Snodgrass, V. S. Subrahmanian, and Roberto Zicari. *Advanced Database Systems.* Morgan Kaufmann Publishers, 1997.

[94] Pavel Zezula, Paolo Ciaccia, and Fausto Rabitti. M-tree: A dynamic index for similarity queries in multimedia databases. Technical Report 7, HERMES ESPRIT LTR Project, 1996. URL: http://www.ced.tuc.gr/hermes/.

[95] C. Lawrence Zitnick and Takeo Kanade. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):675–684, 2000.

[96] M. Tamer Özsu and et al. DISIMA: A Distributed Image Database Management System. Technical Report STR181014, University of Alberta, 2000.