UNIVERSIDADE DE SÃO PAULO

Escola de Artes, Ciências e Humanidades

Leila Cristina Carneiro Bergamasco

Recuperação de imagens por conteúdo utilizando Lógica *Fuzzy* - um estudo de caso sobre imagens faciais

São Paulo

Novembro de 2010

Universidade de São Paulo

Escola de Artes, Ciências e Humanidades

Leila Cristina Carneiro Bergamasco

Recuperação de imagens por conteúdo utilizando Lógica *Fuzzy* - um estudo de caso sobre imagens faciais

Monografia apresentada à Escola de Artes, Ciências e Humanidades, da Universidade de São Paulo, como parte dos requisitos exigidos na disciplina ACH 2018 – Projeto Supervisionado ou de Graduação II, do curso de Bacharelado em Sistemas de Informação.

São Paulo

Novembro de 2010

Universidade de São Paulo
Escola de Artes, Ciências e Humanidades

Leila Cristina Carneiro Bergamasco

Recuperação de imagens por conteúdo utilizando Lógica *Fuzzy* - um estudo de caso sobre imagens faciais

Orientação/Supervisor

_____

Fátima L. S. Nunes

Profa. Dra. Fátima L. S. Nunes

Banca Examinadora:

_____

Ariane Machado Lima

Profa. Dra. Ariane Machado Lima (Supervisor da Disciplina)

_____

São Paulo, Novembro de 2010

Dedicatória

Dedico essa monografia a todas as pessoas que estiveram envolvidas durante esse ano nesse meu pequeno, porém importante projeto, que finaliza uma etapa muito importante da minha vida.

Aos meus pais, que sempre me apoiaram incondicionalmente não somente nesse ano, mas durante todo o meu percurso, me erguendo quando a queda era inevitável, e sempre com uma palavra de consolo para que eu nunca desistisse dos meus sonhos. Esse projeto pode ser concluído porque sempre pude encontrar em casa um lugar para me concentrar, relaxar e ter fé que tudo ia dar certo.

Aos meus irmãos, meus avós e família, por sempre me incentivarem com risadas, carinho, pensamentos positivos e um bom almoço de domingo.

Ao Alex, uma pessoa que em pouco tempo se tornar essencial para que tudo se concretizasse, sempre com uma palavra de apoio, uma revisão gramatical, tudo o que estava em seu alcance para que eu seguisse em frente. Um verdadeiro amigo acima de tudo.

Ao Felipe, Edson e Fernando, que nas horas díficeis sempre tinham uma piada ou uma cerveja. Espero que vocês também sintam esse sentimento de dever cumprido quando entregarem as suas monografias.

À professora Fátima, que desde o ínicio se mostrou totalmente disponível para escutar minhas incertezas, me incentivar a fazer um bom projeto. Guardarei na lembrança, com muito carinho todas as nossas reuniões e conversas. Muito obrigada pela confinça!

A todos os professores de Sistemas de Informação dessa escola fantástica que eu tenho muito orgulho de pertencer. Sem eles isso não seria possível, a competência de cada um é extraordinária e a vontade deles de fazer do nosso curso, um curso modelo é enorme, fico honrada de poder ter visto várias dessas conquistas.

E a Deus, principalmente, só Ele ouviu todos os meus pedidos de que tudo desse certo. Não só hoje, como sempre.

Minha eterna gratidão a todos vocês.

Glossário

CBIR: *Content Based Image Retrieval* – Recuperação de imagens baseado em conteúdo.

O-FIm: *Oracle for Images* – Oráculo gráfico que permite a customização e uso de extratores e funções de similaridade para cada necessidade de busca de determinada imagem .

RGB: Sistema de cores composto por *Red* (Vermelho), *Green* (Verde), Blue (Azul).

CMY: Sistemas de cores composto por *Cyan* (Ciano), *Magenta* (Magenta), *Yellow* (Amarelo)

HSV: Sistema de cores composto por *Hue* (Matiz), *Saturation* (Saturação) e *Value* (Valor).

HSI: Sistema de cores composto por *Hue* (Matiz), *Saturation* (Saturação) e *Instensity* (Intensidade).

API: *Application Programming Interface* – Conjunto de classes e métodos já implementados na linguagem Java

JAI: *Java Advanced Imaging* – API do Java para processamento de imagens.

Resumo

A técnica de CBIR *(Content-Based Image Retrieval)* consiste em recuperar imagens de um banco de dados com base na similaridade entre uma imagem modelo e as imagens armazenadas. A lógica *Fuzzy*, tem por objetivo classificar essa similaridade considerando graus de pertinência, sendo que quanto maior o grau, maior a similaridade. O presente trabalho propõe a implementação e a análise da recuperação de imagens por conteúdo utilizando a técnica de Inteligência Artificial de lógica *Fuzzy*. A integração entre CBIR e lógica *Fuzzy* ocorrerá por meio da técnica de realimentação por relevância, na qual o usuário classifica cada imagem resultante como relevante ou não a sua busca, de modo que o algoritmo 'aprenda' e adeque seus graus de pertinência de acordo com as especificidades do usuário.

Palavras chaves: Recuperação de imagens baseada em conteúdo, CBIR, Lógica *Fuzzy*, O-FIm, Extratores

Abstract

The CBIR technique (Content-Based Image Retrieval) retrieves images from a database based on the similarity between a template image and stored images. Fuzzy Logic, aims to classify the similarity on degrees of relevance, the degree is proportional to the similarity. This paper proposes the implementation and analysis of content based image retrieval using the technique of Artificial Intelligence, the fuzzy logic. The integration of fuzzy logic and CBIR occur through the technique of relevance feedback, which the user rates each image resulting of your search as relevant or not, so that the algorithm 'learns' and suit their degrees of relevance according to the specific user.

Keywords: Content-Based Image Retrieval, CBIR, fuzzy logic, O-FIm, Features

Lista de Figuras

Lista de Tabelas

Sumário

1. Introdução

A tecnologia tem avançado em todas as camadas da sociedade e auxiliado diversos setores com a agilidade de resposta e precisão de seus resultados. Muitos desses setores, como a área médica e a área investigativa criminal, utilizam imagens como dados de entrada para obter resultados necessários às atividades rotineiras (MAXIMO, 2004).

A automatização do reconhecimento facial é uma dessas técnicas. Com o emprego massivo de formas digitais para validações de identidade, surgiu a necessidade de se criar sistemas que executassem essa validação a partir de uma imagem facial. Assim, a ideia principal é buscar em um banco de dados imagens similares a uma imagem fornecida como modelo. Apesar de ser um problema conceitualmente simples, automatizar este processo não é trivial.

Muito se tem estudado durante esses últimos anos sobre problemas dessa natureza, a fim de se obter respostas cada vez mais rápidas e precisas. Uma técnica que está sendo explorada ultimamente é a recuperação de imagens baseada em conteúdo (*Content-Based Images Retrieval* - CBIR). Por meio de extratores (algoritmos que extraem características das imagens) e funções de similaridade, sistemas desenvolvidos com a técnica de CBIR têm sido aplicados para permitir buscas eficientes e diferenciadas em bases de imagens (TORRES; FALCÂO, 2008).

Para aumentar a qualidade das respostas do sistema, é possível utilizar a lógica *Fuzzy*, uma técnica amplamente divulgada em Inteligência Artificial, que tem por objetivo diminuir o *gap* semântico entre o usuário e a lógica matemática do computador. Essa técnica permite quantificar a pertinência de um elemento em um conjunto de dados entre um intervalo de 0 a 1 usando uma base de regras existentes. A lógica F*uzzy*, encontrada na literatura também como lógica nebulosa, é utilizada para automação industrial, finanças e aplicações embarcadas (JUNGES, 2006).

2. Objetivos

2.1. Objetivo Geral

O presente trabalho teve por objetivo implementar um sistema de recuperação de imagens baseada em conteúdo, tendo a recuperação de faces como estudo de caso, utilizando a lógica *Fuzzy* para melhorar o índice de acerto das respostas ao usuário.

2.2. Objetivos Específicos

Para atingir o objetivo geral proposto, foram estabelecidos os seguintes objetivos específicos:

• estudar processamento de imagens aplicado a problema de reconhecimento de faces;

• estudar e implementar extratores dentro da concepção do *framework* 'O-FIm';

• estudar e implementar algoritmos que utilizem lógica *Fuzzy* para qualificar respostas;

• integrar a lógica *Fuzzy* com a técnica de realimentação por relevância para melhorar o índice de acerto do sistema.

2.3. Organização do trabalho

O presente trabalho está organizado da seguinte forma:

No capítulo 3 está a revisão bibliográfica feita durante todo o projeto, com detalhes sobre cada conceito utilizado para a finalização do mesmo. No capítulo 4 é explicada a metodologia utilizada e a implementação dos módulos de CBIR, *Fuzzy* e a técnica de realimentação por relevância. Por fim, nos capítulos 5, 6 e 7 são discutidos os casos de testes conduzidos e a conclusão que se chegou ao fim do projeto.

3. Revisão Bibliográfica

A revisão bibliográfica apresentada a seguir engloba todos os assuntos discutidos no decorrer do projeto separados em tópicos. No primeiro, chamado de Recuperação de Imagem por Conteúdo é exposto o conceito de extratores de características assim como as funções de similaridade e estruturas de indexação; o conceito de realimentação por relevância e gráficos de precisão e revocação que auxiliam no processo de análise de dados são discutidos logo a seguir; na sessão destinada para o *framework* 'O-FIm' é feita um revisão de suas características e objetivos e por fim, a lógica *Fuzzy* e as suas propriedades, como o processo de *fuzzyficação*, inferência e *defuzzyficação* são discutidos ao fim.

3.1. Recuperação de Imagem por Conteúdo

A técnica de CBIR, conhecida como reconhecimento de imagens baseada em conteúdo, tem por objetivo, a partir de uma imagem modelo, apresentar ao usuário as imagens mais relevantes que constam na base de dados disponível.

A busca pelas imagens mais relevantes ocorre por meio das funções de similaridade que são aplicadas nos vetores de características de cada imagem. Esse vetor é composto de dados matemáticos extraídos das imagens usando-se extratores. A partir do resultado proposto o usuário pode, por meio do sistema de CBIR definir quais as imagens são mais relevantes no resultado da busca e quais imagens não foram relevantes, na sua concepção. (TORRES; FALCÃO, 2008).

Essa avaliação permite diminuir o *gap* semântico existente em outras técnicas de reconhecimento de padrões. Ele ocorre devido ao algoritmo analisar somente a semelhança matemática entre as imagens, resultando algumas vezes em imagens totalmente diferentes segundo o ponto de vista do usuário. A Figura 1 ilustra essa relação.

Figura 1: Imagens matematicamente iguais e semanticamente diferentes

3.1.1. Extratores

Os extratores de caracteristicas, são algoritmo que extraem características matemáticas que, quando transformadas em números, representam o conteúdo da imagem. Os extratores podem ser globais, recuperando características da imagem inteira, ou locais, agindo em regiões específicas da imagem. As características extraídas formam um vetor de características que define matematicamente a imagem (SOUZA; SANTOS; GULIATO, 2008).

Os extratores podem ser do tipo cor, forma e textura.

Os extratores de cor são os descritores mais usados em sistemas de CBIR devido a sua robustez e certa independência em relação ao tamanho e orientação da imagem. É também um dos extratores em que o conceito é mais facilmente assimilado e, dessa forma, mais explorado (SILVA,2008).

Pode-se optar por implementar extratores de cor seguindo qualquer um dos modelos existentes: orientados a hardware (RGB, CMY) ou orientados ao usuário (HSV, HSI). Os mais usados atualmente são os modelos RGB e HSV. No que tange à técnica escolhida para extrair características, podemos citar os histogramas como sendo os mais usados para esse tipo de problema, eles retornam as frequências de cada cor na imagem, permitindo a comparação matemática delas. Quanto mais diferentes os histogramas, mais diferentes as imagens (SILVA, 2008). A Figura 2 ilustra as cores de uma imagem representadas em um histograma.

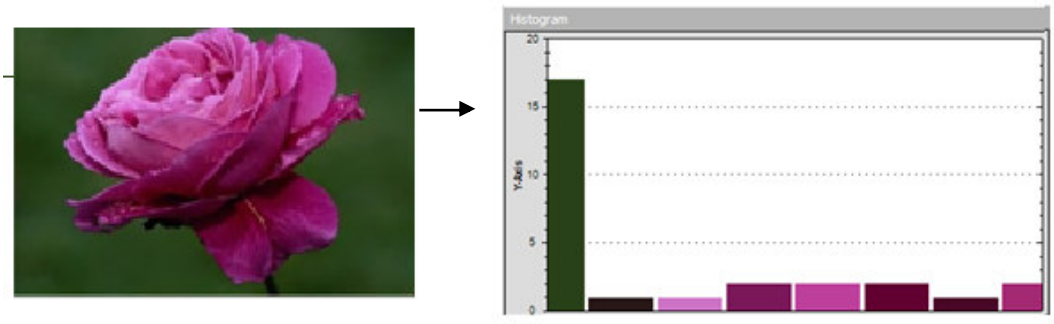


Figura 2: Imagem e seu respectivo histograma

Os extratores de textura caracterizam matematicamente o arranjo estrutural da imagem. Isso significa que propriedades como orientação, regularidade e granularidade podem ser medidas a partir de então. Esse tipo de descritor está

tendo maior visibilidade devido aos resultados satisfatórios que ele tem alcançado com a recuperação de imagens.

Há duas abordagens quanto ao tipo de análise feita: uma é a estatística, aconselhada para imagens com necessidade de um processamento mais rigoroso nas regiões de textura, e a outra, a estrutural, recomendada para imagens com regiões de textura uniformes e bem definidas (TORRES; FALCÃO, 2008).

A matriz de co-ocorrência, ilustrada na Figura 3, é uma técnica do tipo estatística e muito utilizada para extrair as características de textura. Consiste em demonstrar o relacionamento espacial entre os *pixels* e, a partir daí, por meio de fórmulas como a de entropia, contraste e energia, por exemplo, calcular as probabilidades de regiões de textura e obter o vetor de características (SILVA, 2008).

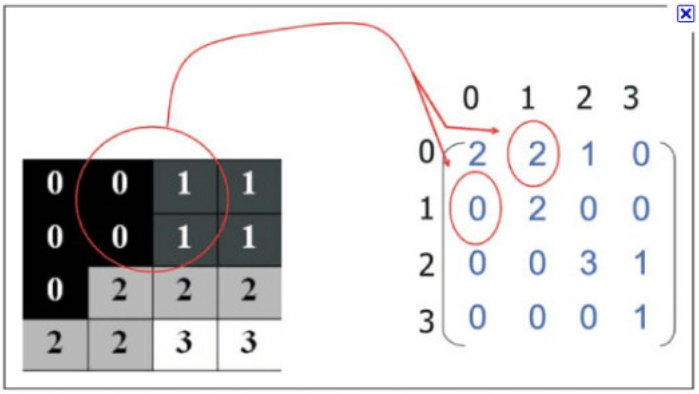


Figura 3: Extração da matriz de ocorrência na direção 0º (MARTINS, 2005).

Por fim, os extratores de forma são considerados os mais difíceis de implementar devido à necessidade de pré-processamento para a segmentação da imagem para somente depois aplicar o extrator, já que ele só funciona adequadamente em imagens nas quais os objetos de interesse estão devidamente separados do fundo da figura (ASLANDOGAN; YU, 1999).

Uma das técnicas de extração de características de forma são os momentos invariantes de Hu que procuram expressar matematicamente a partir de uma imagem em níveis de cinza, através de seus 7 momentos invariantes, sua variação quanto às propriedades de forma, rotação e escala (OLIVEIRA, 2005).

3.1.2. Funções de Similaridade

As funções de similaridade buscam demonstrar matematicamente o quanto uma imagem é similar a outra. Esse processo envolve a escolha de uma ou mais funções de distâncias métricas e sua aplicação nos vetores de características das imagens obtidos por meio dos extratores.

Algumas das funções mais utilizadas em sistemas CBIR são as distâncias de *Minkowski,* compostas por *Manhattan* e Euclidiana.(BALAN, 2007).

A distância *Manhattan,* é obtida através da soma das diferenças absolutas entre as suas coordenadas. A Equação (1) apresenta a definição desta métrica usada, onde R_i é uma das coordenada do ponto R e S_i é uma das coordenadas do ponto S:

$$Dist_{Manhattan}(R,S) = \sum_{i=0}^{N-1} |R_i - S_i| \quad (1)$$

A distância Euclidiana, é a raiz da somatória das diferenças absolutas entre suas coordenadas ao quadrado, conforme mostrado na Equação (2):

$$Dist_{Euclidiana}(R,S) = \sqrt{\sum_{i=0}^{N-1} (R_i - S_i)^2} \quad (2)$$

Na figura abaixo é possível identificar a diferença entre os dois métodos de cálculo de distância.

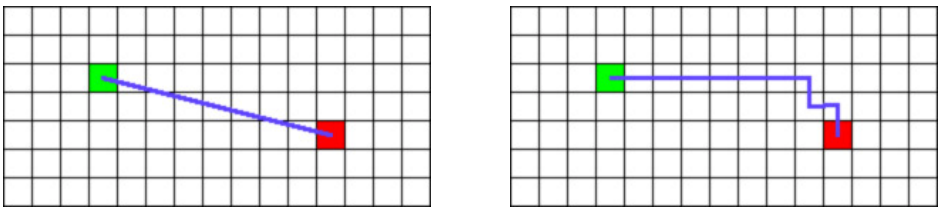


Figura 4: Distância Euclidiana e Distância *Manhattan*

3.1.3. Estruturas de indexação

O principal objetivo das estruturas de indexação é tornar rápida e eficiente a busca por registros em uma base de dados.

No caso dos sistemas CBIR, esse é um tópico que vem avançando em discussões atualmente, devido ao seu processamento ser considerado computacionalmente caro. Isso se deve pelo motivo da própria estrutura do CBIR, onde uma imagem de consulta tem suas características extraídas e comparadas por meio de uma ou mais funções de similaridade com todos os outros registros da base de dados (MATOS, 2008).

A abordagem mais simplista é armazenar sem nenhum critério todos os vetores de características em um banco de dados ou matriz e comparar um a um cada registro. A desvantagem desta abordagem é o custo caro dessa operação para uma base de dados grande. Por esse motivo, é comum encontrar na literatura trabalhos desenvolvendo e otimizando o tempo de resposta de uma consulta em um sistema CBIR. Podemos citar, por exemplo, a técnica de índice invertido, na qual os vetores de características são agrupados por faixas, de modo que imagens com vetores similares fiquem agrupados em uma única lista invertida e em uma futura consulta somente algumas faixas precisem ser analisadas, diminuindo assim o tempo de resposta (MATOS,2008).

3.1.4. Realimentação por relevância

A técnica de realimentação por relevância tem dois objetivos principais: reduzir o *gap* semântico entre a análise visual de alto nível feita pelo usuário e a análise de baixo nível feita por meio dos extratores e funções de similaridade; e reduzir a subjetividade da percepção humana, uma vez que diferentes usuários podem ter diferentes percepções de relevância para a mesma imagem (MARQUES, 2006).

Esse objetivo é alcançado através do próprio usuário, que categoriza as imagens resultantes de determinada consulta como sendo relevantes ou não. Dessa forma o algoritmo ‘aprende’ quais propriedades visuais são melhores para aquela determinada consulta. O aprendizado se dá através de pesos que são

acrescidos nos vetores de características como um todo ou em cada uma das coordenadas desse vetor (SILVA, 2009).

3.1.5. Avaliação de um sistema de CBIR

Para medir a eficácia do sistema de CBIR são utilizados constantemente os gráficos de Precisão e Revocação.

O gráfico de Precisão, Fórmula (3), indica o índice de imagens relevantes que foram recuperadas na busca, já a medida Revocação, Fórmula (4) exibe a porcentagem de imagens relevantes que estão entre as recuperadas (TORRES; FALCÃO, 2008).

$$Precisão = \frac{\{imagens\ relevantes\} \cap [imagens\ recuperadas]}{\{imagens\ recuperadas\}} \quad (3)$$

$$Revocação = \frac{\{imagens\ relevantes\} \cap [imagens\ recuperadas]}{\{imagens\ relevantes\}} \quad (4)$$

Nos gráficos, normalmente a curva mais alta indica um descritor mais eficaz que está sendo avaliado.

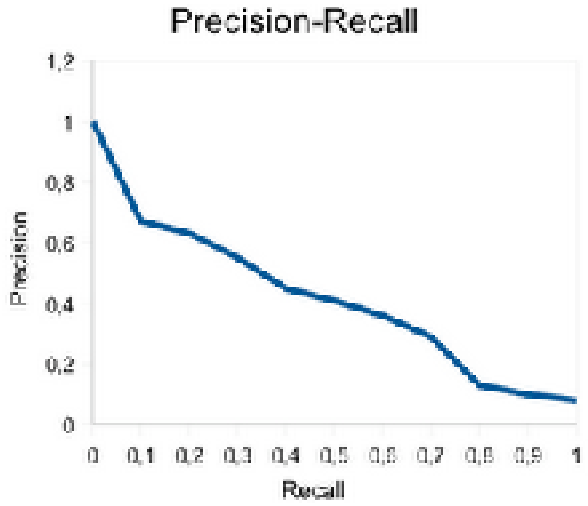


Figura 5: Imagem de gráfico demonstrando a relação das duas medidas.

3.2. *Framework* 'O-FIm'

O *framework* 'O-FIm' é um oráculo gráfico que utiliza os conceitos do CBIR para utilização em testes de programas com saída gráfica, avaliando o funcionamento dos mesmos.

Entede-se por oráculo um mecanismo que pode ser tanto *hardware* ou *software* que avalia se a saída de um programa e seu funcionamento foram adequados ou não (OLIVEIRA, 2008).

No contexto de CBIR, o 'O-FIm' possibilita ao usuário escolher quais extratores e funções de similaridade devem ser aplicadas a determinada imagem. O usuário pode ainda implementar novos extratores e/ou funções, seguindo os padrões de codificação da ferramenta.

O 'O-FIm' é composto por alguns componentes, a saber: *plugins*, *parser*, núcleo e descritores de oráculos .

Os *plugins* são interfaces que contêm métodos que auxiliam na construção de extratores e funções de similaridade padronizando-os. O *parser* é responsável por reconhecer os extratores, funções e parâmetros escolhidos e traduzir para a linguagem de baixo nível da ferramenta. O núcleo permite a instalação e remoção de *plugins,* possibilitando a customização da busca por parte do usuário. Por fim, o descritor de oráculos é um arquivo texto gerado com as especificações que o usuário fez para determinada busca.

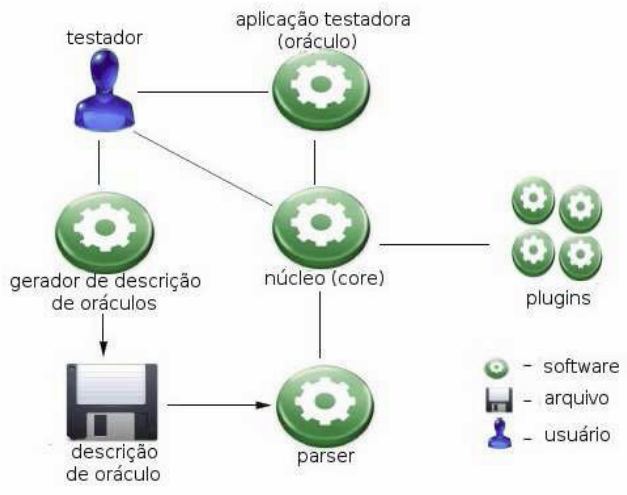


Figura 6: Arquitetura do *framework* 'O-FIm' (OLIVEIRA,2008)

3.3. Lógica *fuzzy*

A lógica *fuzzy*, também conhecida como lógica nebulosa, é uma nova proposta que procura atender o aspecto vago deixado pela teoria clássica dos conjuntos em relação a informações com dados imprecisos e incertos (SANDRI; CORREA, 1999).

Ela é muito bem sucedida em sistemas complexos de controle de processos e decisões. Alguns exemplos de uso se encontram em aréas industriais como na fabricação de eletrodomésticos, nas áreas automotivas e no controle de semáforos.

Esses controladores de processos e decisões são chamados de sistemas *fuzzy,* que são compostos por três processos base: *fuzzyficação*, inferência e *defuzzyficação*. A Figura 7 ilustra o esquema utilizado em um sistema *fuzzy* (JUNGES, 2006).

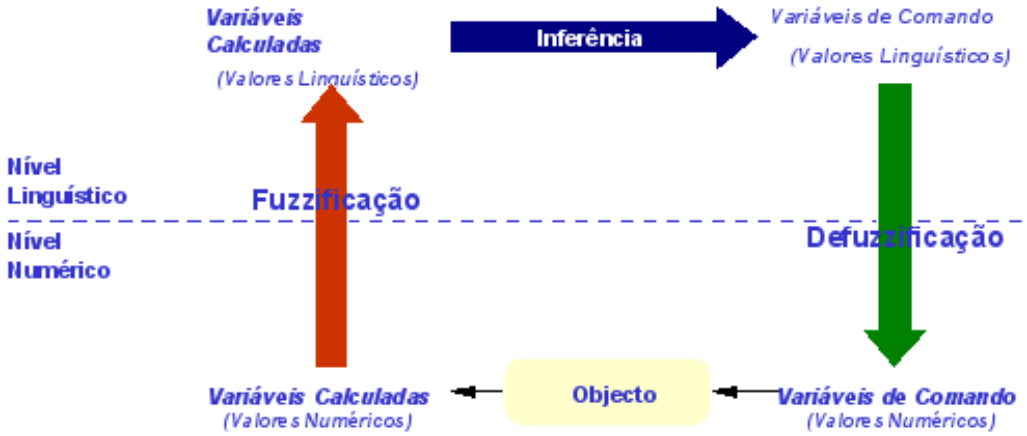


Figura 7: Esquema lógico de um sistema *fuzzy* (JUNGES,2006).

3.3.1. Conceitos básicos

A grande diferença de um conjunto *fuzzy* com os conjuntos clássicos é o fato da pertinência de um elemento ser ou não ser pertencente a eles. Enquanto na teoria clássica, aqui chamada por conjuntos '*crisp*', um elemento pertence ou não a um determinado conjunto, ou seja, tem valores 0 ou 1 de pertinência, nos conjuntos *fuzzy* um determinado elemento pode pertencer com grau em um intervalo de 0 a 1

Na Figura 8 é apresentado um exemplo para ilustrar como a lógica *fuzzy* consegue expressar melhor uma dada informação imprecisa.

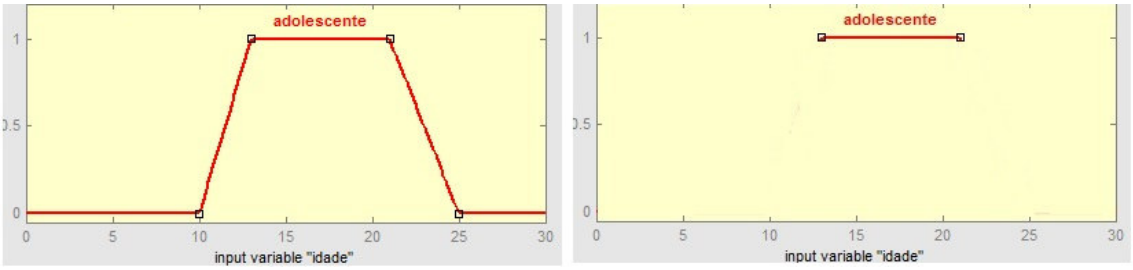


Figura 8: A variável adolescente e sua pertinência nos conjuntos *fuzzy* e *crisp*

Por meios dos gráficos da Figura 8, é possível perceber que caso o valor 12 anos 10 meses fosse dado ao sistema, ele diria no caso dos conjuntos *crisp* essa pessoa não é um adolescente, já no conjunto *fuzzy,* ele é mais 'tolerante' e retorna que sim, essa pessoa é um adolescente com um grau de pertinência 0.8, ou seja, essa pessoa é '*quase* um adolescente'.

As funções de pertinência são usadas para calcular o grau que uma determinada variável pertence ou não ao conjunto relacionado. As funções mais utilizadas são a trapezoidal, a triangular e a gaussiana. A escolha de qual função usar é feita normalmente de forma empírica, analisando casos passados semelhantes e vendo qual obteve melhor resultado. (SANDRI; CORREA, 1999)

Similarmente à teoria dos conjuntos, há também nos conjuntos *fuzzy* operações entre diferentes conjuntos tais como união e intersecção. Essas operações são implementadas por uma família de operadores chamados *s-norma* e *t-norma* respectivamente. Na Figura 9 se encontram a relação de dois conjuntos *fuzzy* A1 e A2 e suas respectivas *t-normas* e *s-normas* ilustradas.

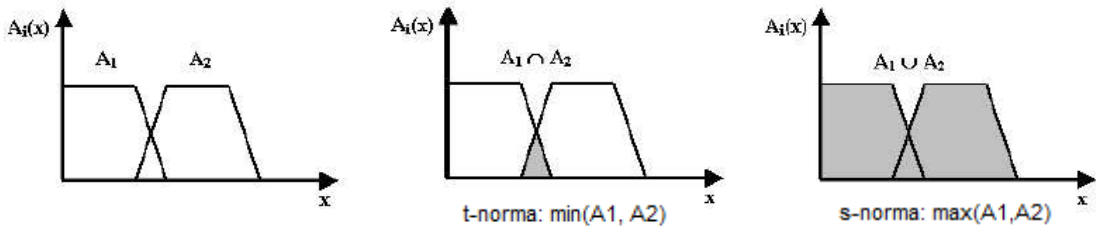


Figura 9: Representação gráfica de uma t-norma e s-norma (PERES, 2009)

Outra diferença entre os conjuntos *crisp* e os conjuntos *fuzzy,* é a forma como os controladores são construídos. Enquanto em um conjunto *crisp* ele é expresso por expressões algébricas matemáticas, no conjunto *fuzzy* são usadas as variáveis linguísticas que buscam expressar a rotina e a experiência humana. (PEDRYCZ, GOMIDE, 1998)

Uma variável linguística é uma quádrupla $< X, T(X), \Omega, M >$, onde: X é o nome da variável; $T(X)$ é o conjunto de rótulos para a variável X; Ω é o universo de discurso de X e M é uma função de pertinência que associa cada elemento do universo de X ao seu rótulo (SANDRI; CORREA, 1999).

É possível ter, por exemplo, a variável lingüística temperatura, na qual X corresponde à 'temperatura', $T(X)$ é o conjunto de variáveis: {muito baixa, baixa, agradável, alta, muito alta}, Ω é o intervalo [-10℃, 40℃] que em um gráfico cartesiano é identificado com os valores [-10, 40] e M são as funções (trapezodais, triangulares, gaussianas, etc..) escolhidas para cada caso. Na Figura 10 está o gráfico demonstrando a situação.

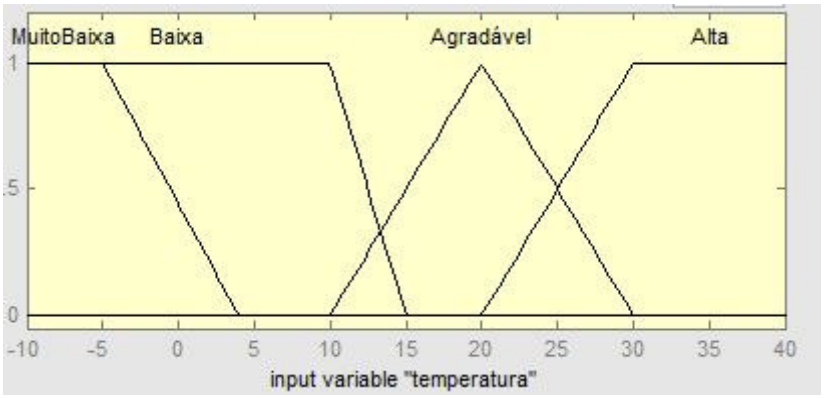


Figura 10: Demonstração da variável linguística 'temperatura'

O processo de *fuzzyficação* consiste em transformar os valores de entrada, que são valores *crisp,* em valores *fuzzy* (SANDRI; CORREA, 1999). No exemplo anterior, dada uma temperatura de 25 graus, é verificado o grau de pertinência em cada variável lingüística e dessa forma não é mais analisada a informação pelo seu valor matemático, mas sim como parte integrante de um dos conjuntos de variáveis. Assim 25 graus pode ser "uma temperatura agradável" e uma "temperatura alta" com grau de pertinêncencia menor.

Na inferência *fuzzy,* são aplicadas as regras definidas do tipo "Se <premissa> então <conclusão>". Esse processo é o mais importante de um sistema *fuzzy* devido ao fato que nele a lógica *fuzzy* é amplamente utilizada por meio dos operadores de implicação. Para cada dado de entrada é feita uma análise em cada regra definida e as conclusões obtidas individualmente são agrupadas para obtenção de um resultado global.

No processo de *defuzzyficação* é implementado um algoritmo para extração de um valor *crisp* a partir de um conjunto de dados nebulosos. Alguns exemplos são: maior dos máximos, menor dos mínimos, média dos máximos e centróide. Este último é amplamente usado e está demonstrado na Equação 5, onde μ corresponde ao grau de pertinência de determinado valor *x*:

$$Centróide = \frac{\mu x_0 + \mu x_1 + \cdots + \mu x_n}{x_0 \mid x_1 \mid \cdots \mid x_n} \quad (5)$$

No Anexo 1 está um exemplo retirado de Laufner (2003), para avaliação do valor de uma gorjeta, baseado na qualidade do serviço e sabor da comida utilizando um sistema *Fuzzy.*

3.4. Considerações Finais

Com as informações adquiridas sobre o sistema de CBIR, lógica *fuzzy*, realimentação por relevância e o *framework* 'O-FIm' foi possível elaborar uma estratégia de metodologia de implementação e realizar o estudo de caso de forma que os resultados fossem apoiados por uma base sólida de conhecimento.

Devido a nessidade de aprendizagem de vários assuntos distintos e totalmente novos como processamento de imagens e oráculos gráficos , a revisão bibliográfica foi extensa e bastante longa, existindo durante todo o projeto de acordo com o módulo que estava sendo implementado.

4. Metodologia

O objetivo desse projeto, como mencionado no Capítulo 2, é implementar um sistema de reconhecimento facial. O sistema deve partir de uma imagem modelo, escolhida pelo usuário, extrair suas características e comparar com os vetores de características das imagens existentes no banco de dados, e retornar as imagens mais similares. A partir desse conjunto de resposta o usuário avalia cada imagem retornada com o valor de 0-10 sendo a nota 0 como sendo uma imagem sem nenhuma relação com a imagem modelo, na sua concepção, e 10 como sendo uma imagem praticamente igual à usada como referência. Com esses valores o sistema ajusta o valor do peso de cada imagem no banco de dados e uma nova busca é feita retornando o novo resultado.

4.1. Tecnologias utilizadas

Para a implementação do sistema foi utilizada a linguagem Java e algumas de suas bibliotecas, como o JAI (*Java Advanced Imaging)* que destina-se ao processamento e representação de imagens de grande porte através de métodos que realizam leitura e criação de pixels, segmentação de imagens entre outras funcionalidades (SANTOS, 2008).

Outra API bastante utilizada foi a *Raster,* que similarmente a JAI, também processa imagens porém é utilizada normalmente em imagens mais simples. Sua execução é mais rápida porém possui menos funcionalidades que a JAI.

4.2. Estrutura do sistema

Para o desenvolvimento do sistema, foram criados 3 módulos e o seu desenvolvimento foi distribuído durante o ano do projeto, conforme apresentado na Figura 11.

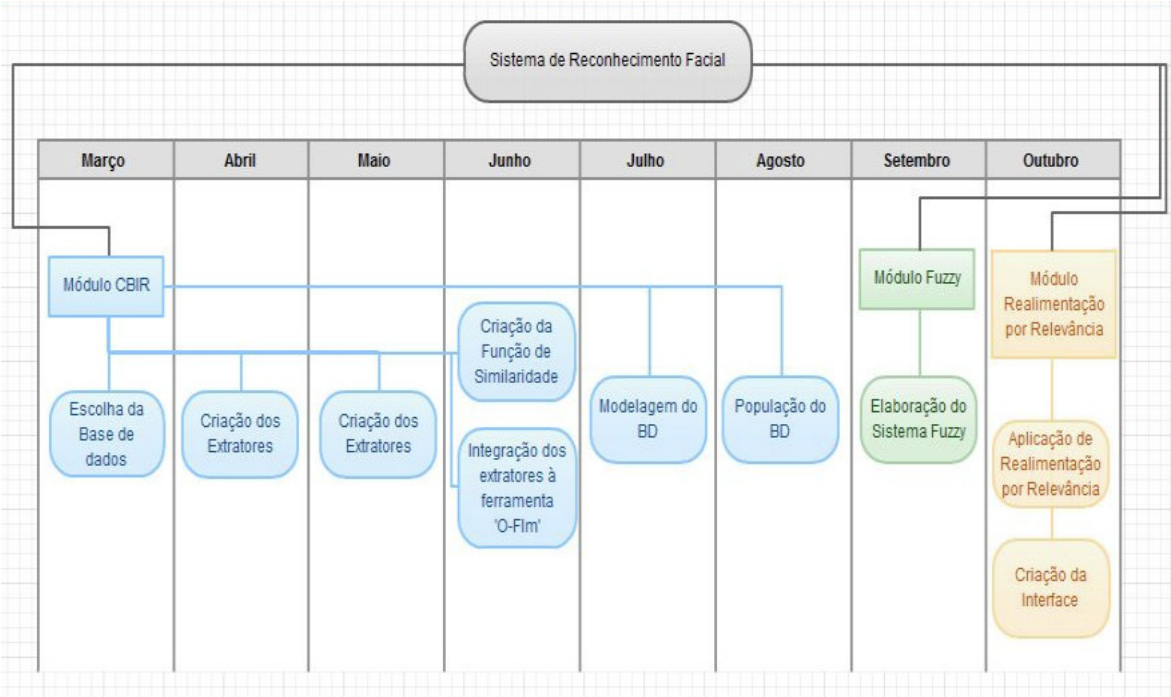


Figura 11: Diagramação dos módulos do sistema

4.2.1. Módulo CBIR

A primeira etapa desse módulo consistiu em escolher uma base de dados para que fosse possível extrair mais facilmente características e testar a eficiência do sistema. Foi escolhido um cenário controlado, com fotografias de rostos de homens e mulheres de diferentes raças na fase adulta. A base de dados é da Universidade de Essex (ESSEX UNIVERSITY).

Em seguida, foram escolhidos e implementados os algoritmos para a extração de características das imagens. Foram elaborados 17 extratores contemplando extratores de forma, cor e textura. Na Tabela 1 se encontra a relação deles.

Tabela 1: Conjunto dos extratores implementados

Conjunto dos extratores implementados		
Tipo	**Extrator**	**Resumo**
Cor	Média do Histograma em Nivel de Cinza	Calcula através do histograma, a média da frequência dos *pixels*. **Objetivo**: obter uma informação relativa ao tom da imagem (mais clara ou mais escura)
	Média do Histograma em RGB	Calcula através do histograma, a média da frequência dos *pixels*. **Objetivo:** obter uma informação relativa ao tom da imagem (mais clara ou mais escura)
	Desvio Padrão do Histograma em Nivel de Cinza	Calcula através do histograma, o desvio padrão da frequência dos *pixels*. **Objetivo:** obter a informação da variação de cores em uma imagem (se houve muita ou pouca variação)
	Desvio Padrão do Histograma em RGB	Calcula através do histograma o desvio padrão da frequência dos *pixels*. **Objetivo:** obter a informação da variação de cores em uma imagem (se houve muita ou pouca variação)
	Obliquidade do Histograma em Nível de Cinza	Calcula através do histograma, o desvio padrão da frequência dos *pixels*. **Objetivo:** obter a informação de quanto é a variação das cores
	Obliquidade do Histograma em RGB	Calcula através do histograma, o desvio padrão da frequência dos *pixels*. **Objetivo:** obter a informação de quanto é a variação das cores
	Média dos Pixels em Nivel de Cinza	Calcula através da varredura individual dos *pixels*, a média de suas cores . **Objetivo:** obter uma informação relativa ao tom da imagem (mais clara ou mais escura)
	Média dos Pixels em RGB	Calcula através da varredura individual dos *pixels*, a média de suas cores . **Objetivo:** obter uma informação relativa ao tom da imagem (mais clara ou mais escura)

	Desvio Padrão dos Pixels em Nivel de Cinza	Calcula através da varredura individual dos *pixels*, o desvio padrão de suas cores . **Objetivo**:obter a informação da variação de cores em uma imagem (se houve muita ou pouca variação)
	Desvio Padrão dos Pixels em RGB	Calcula através da varredura individual dos *pixels*, o desvio padrão de suas cores . **Objetivo**: obter a informação da variação de cores em uma imagem (se houve muita ou pouca variação)
	Obliquidade dos Pixels em Nível de Cinza	Calcula através da varredura individual dos *pixels*, o desvio padrão de suas cores . **Objetivo:** obter a informação de quanto é a variação das cores
	Obliquidade dos Pixels em RGB	Calcula através da varredura individual dos *pixels*, o desvio padrão de suas cores . **Objetivo:** obter a informação de quanto é a variação das cores
Textura	Contraste	Por meio da matriz de co-ocorrência, usando os parâmetros de análise dos pixel imediatamente vizinhos e à esquerda, para a verificação da probabilidade de ocorrência de determinado padrão de cor. Com essa probabilidade é calculado o contraste. **Objetivo**: Determinar a variação dos *pixels*.
	Entropia	Por meio da matriz de co-ocorrência, usando os parâmetros de análise dos pixel imediatamente vizinhos e à esquerda, para a verificação da probabilidade de ocorrência de determinado padrão de cor. Com essa probabilidade é calculada a entropia. **Objetivo**: Determinar a dispersão dos *pixels*.
Forma	Escala	Através dos momentos invariantes de Hu, calculados a partir do centro de gravidade da imagem é possível calcular o grau de escala da imagem. **Objetivo:** Por meio do índice de escala, comparar duas imagens e ver a diferença entre os tamanho dos rostos

	Rotação	Através dos momentos invariantes de Hu, calculados a partir do centro de gravidade da imagem é possível calcular o grau de rotação da imagem. **Objetivo:** Achar os eixos de inércia da imagem para comparação.
	Translação	Através dos momentos invariantes de Hu, calculados a partir do centro de gravidade da imagem é possível calcular o grau de translação da imagem. **Objetivo:** Encontrar os prolongamentos e orientações da forma do rosto.

Esses algoritmos foram aplicados nas imagens com suas cores em níveis de cinza, propicia uma relativa normalização das cores e resultados mais confiáveis. Nos extratores de cor também foram usadas suas cores originais usando o formato RGB. O código fonte dos extratores se encontra no Apêndice A.

Após implementados os extratores foi feita a integração deles com o *framework* 'O-FIm', a fim de facilitar a visualização de similaridades entre as imagens. Com a facilidade de customização na escolha dos extratores foi possível verificar como eles se comportavam trabalhando em conjunto ou separados.

O *framework* já possui interfaces e métodos definidos. No caso dos extratores, há na interface *IExtractor* e o método *computeValue* que é o responsável por efetuar a extração de características de uma determinada imagem, dada como parâmetro. O mesmo ocorre para a interface *ISimilarity,* que é responsável pela compilação das funções de similaridade.

Para armazenar os valores dos vetores de características foi modelado um banco de dados, conforme a Figura 12, de modo que fosse possível armazenar em uma única tabela, *image_has_feature,* a identificação da imagem, o extrator utilizado e o respectivo valor obtido. Dessa forma, ao selecionar uma imagem, todos os extratores ligados a ela já estão calculados para serem utilizados no cálculo da função de similaridade, que para o projeto foi escolhida a distância Euclidiana, já fornecida pelo *framework* 'O-FIm'.

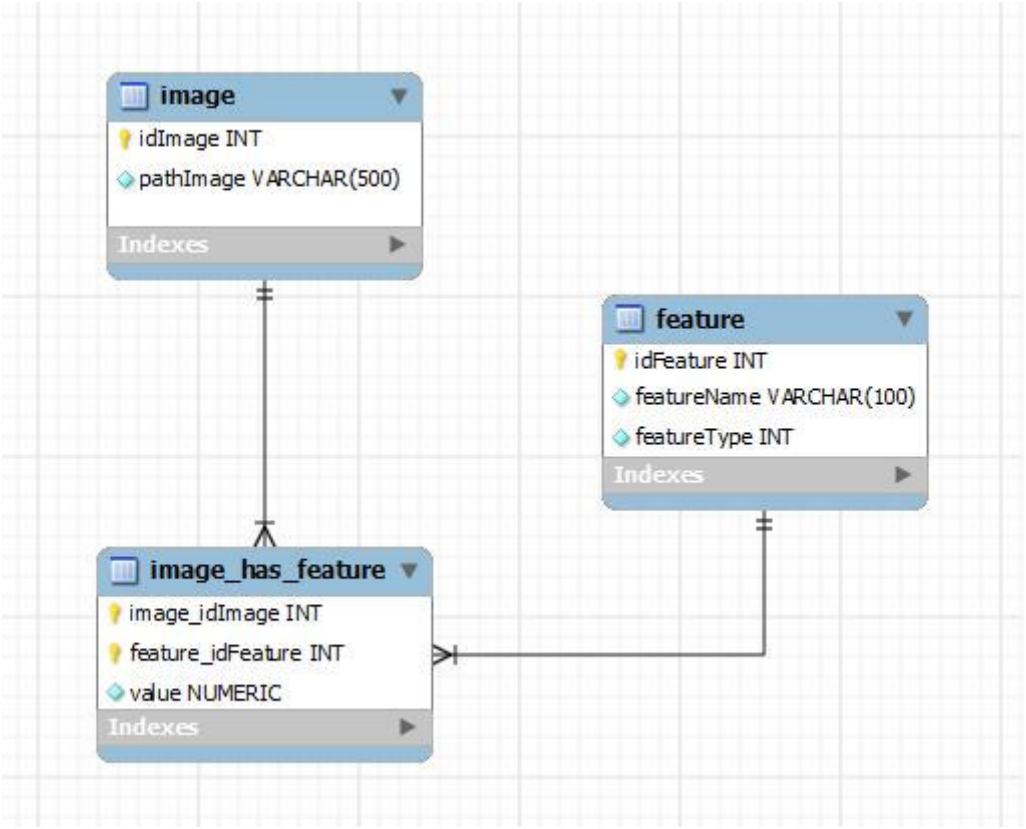


Figura 12: Imagem representativa da modelagem do Banco de dados

4.2.2. Módulo *Fuzzy*

Neste módulo foram definidas as atividades do processo de *fuzzyficação*, detalhadas a seguir.

- Definição do problema: Interpretar as notas dadas às imagens retornadas de forma que indiquem a relevância da imagem perante a imagem modelo fornecida. Essas notas são atríbuidas diversas vezes e espera-se que o algoritmo 'aprenda' durante o treinamento, a identificar imagens mais relevantes a cada noa interação;

- Definição das variáveis Fuzzy: Notas: {Baixa, Média, Alta}; Relevância: {Pouco relevante, relevante, muito relevante.};

- Definições de função de pertinência: Uso das funções do triângulo e trapézio para cálculo das funções de pertinência. A função triangular é usada para variáveis: Nota Baixa e Relevante e função trapezoidal para as demais;

A etapa de inferência, na qual onde ocorre todo o processo lógico do sistema. Para facilitar a construção dessa etapa, é dividida duas fases:

- Definição das regras: se nota baixa então relevância baixa; Se nota média então relevante; Se nota alta então muito relevante;

- Matriz de Regras: Ilustrada na Figura 13.

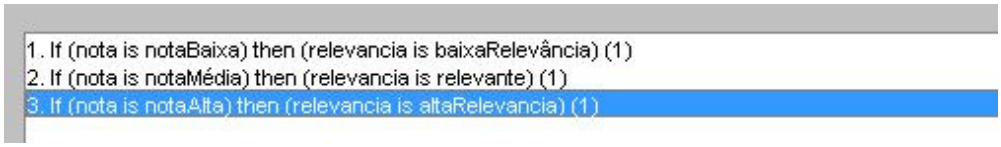


Figura 13: Matriz de regras construída para o sistema *Fuzzy*.

O objetivo do sistema final é determinar os pesos apropriados para as nota atribuídas pelo usuário às imagens candidatas. Cada fotografia possui um peso que é ajustado a cada iteração de treinamento, de acordo com a resposta do sistema *Fuzzy*. O valor do peso encontrado é, então, multiplicado pela distância Euclidiana da respectiva imagem, o que implica que pesos maiores causarão um aumento substancial na distância da imagem. Esta ação levará conseqüentemente, em uma próxima iteração, a não retornar como sendo uma das imagens candidatas. Dessa forma, o algoritmo a margem de erros causada pelo uso dos extratores puramente matemáticos e faz com que a busca por imagens candidatas fique mais refinada.

A seguir, os passos do sistema são apresentados por meio de gráficos.

1º Passo: as imagens na primeira iteração são reportadas aos usuários com base somente na distância Euclidiana entre as imagens e a imagem modelo. São apresentadas as cinco imagens candidatas com menor distância Euclidiana, que supõe-se serem as mais similares.

Os usuários atribuem notas através de um campo texto de 0-10, sendo 0 para a imagem menos relevante, e 10 para a mais relevante.

2º Passo: Cada nota é processada em cada uma das três regras definidas, verificando-se a pertinência encontrada em cada uma delas e a respectiva área de relevância, somando-se a área de cada regra ao final do processamento. A Figura 14 apresenta um exemplo para uma nota igual a 3.

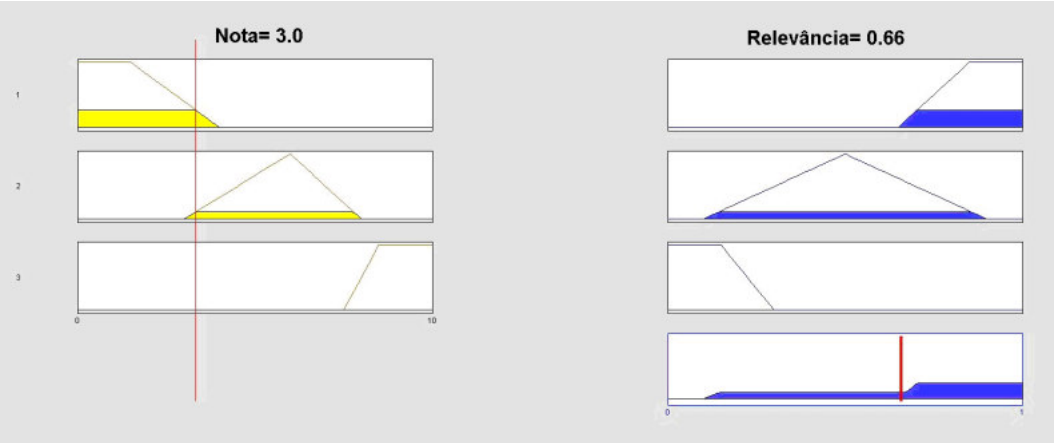


Figura 14: Relação entre a nota atribuída e o peso dado pelo sistema *Fuzzy*.

Para a área final encontrada é aplicado o algoritmo centróide que identifica o valor do eixo de x do gráfico de relevância que mais se enquadra na área demarcada. Com isso, é calculado um peso adequado para ser ajustado. No caso dos exemplos acima, para a nota 3 o peso é igual à 0,66.

4.2.3. Módulo Realimentação por Relevância

Para implementar a Realimentação por Relevância foi construída uma interface que permite ao usuário escolher a imagem modelo e, em seguida as cinco mais parecidas com um são apresentadas para avaliação. A Figura 15 demonstra a interface implementada.

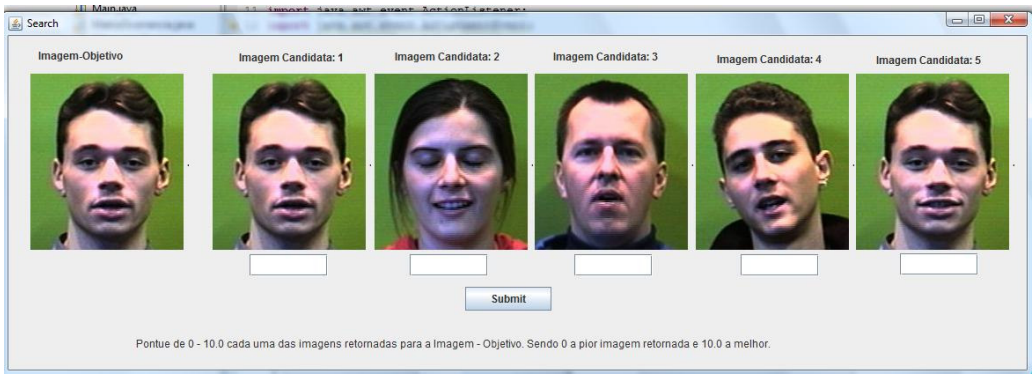


Figura 15: Interface do sistema

A realimentação por relevância acontece quando, a partir da nota astribuída pelo usuário, o algoritmo identifica quais imagens pode descartar e quais deve usar como referência em uma próxima busca. A interação com o usuário termina

somente quando ele julga que todas as imagens candidatas atendem a sua solicitação.

Em nível matemático, o aprendizado ocorre através da multiplicação do valor do peso encontrado na lógica *fuzzy* (lembrando que, quanto maior o peso menor a relevância) pela distância Euclidiana da imagem. Com essa multiplicação, o valor da distância tende a aumentar ou permanecer igual. Se o valor aumentar significa que seu peso foi alto, portanto ocorre relevância baixa . Assim, possivelmente em uma próxima interação essa mesma imagem não esteja entre as cinco menores distâncias e não retorne mais ao usuário.

A Figura 16 ilustra o fluxograma de interação do usuário com o sistema de reconhecimento facial.

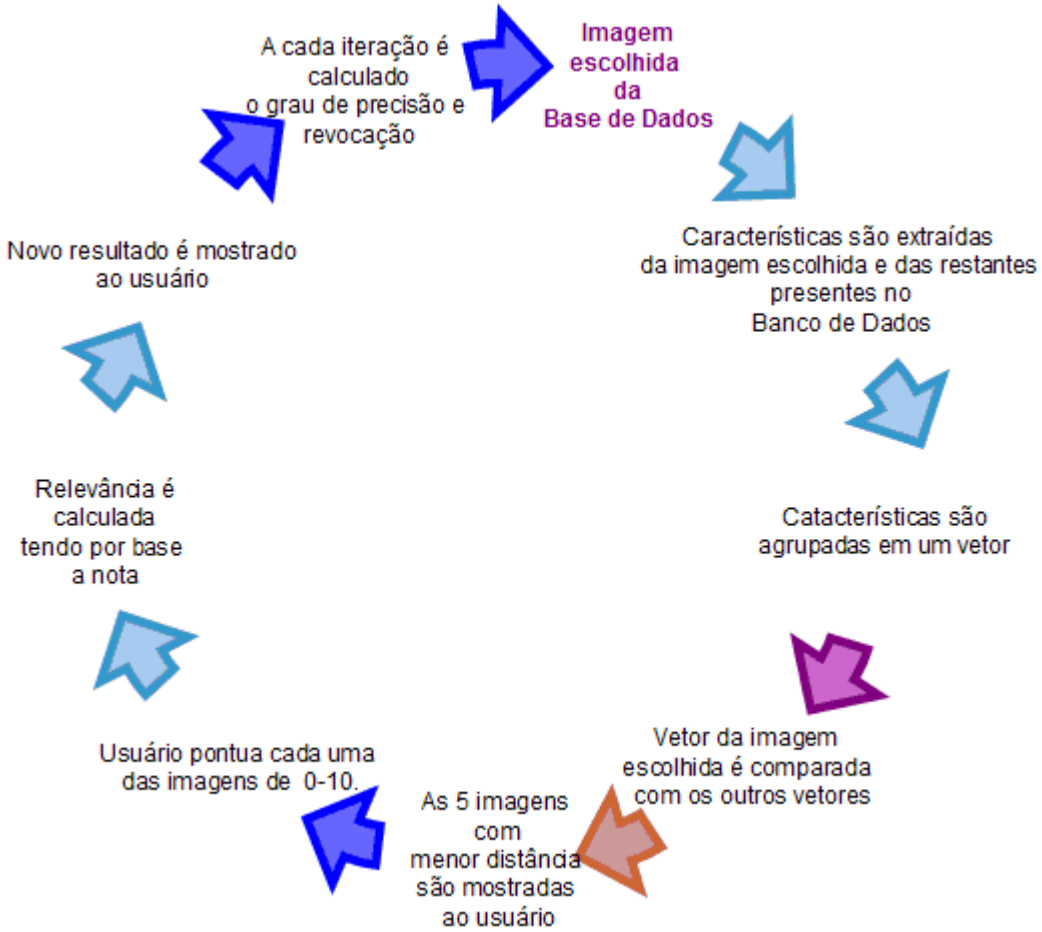


Figura 16: Interação do usuário com sistema de reconhecimento

5. Resultados

Para o estudo de caso, foram escolhidos três rostos, sendo duas mulheres (*R1* e *R2*) e um homem (*R3*) , conforme Figura 17:

Figura 17: Rostos usados para o teste.

Para cada um desses rostos há outras quatro imagens similares, apenas alterando a expressão facial. Elas estão em uma base de dados com outras 40 imagens de outras pessoas. O objetivo do teste foi verificar se através da realimentação por lógica *fuzzy*, o sistema é capaz de selecionar as três imagens corretamente para cada rosto buscado.

É possível verificar pela Tabela 2 o comportamento das distâncias Euclidianas calculadas , tomando por base a imagem *R1* e as cinco distâncias que foram retornadas na primeira interação, onde *I1* é a imagem candidata mais parecida e *I5* a quinta imagem mais parecida.

Tabela 2: Distância Euclidiana entre as cinco imagens candidatas retornadas

Distância Euclidiana entre as cinco imagens candidatas retornadas					
	I1	I2	I3	I4	I5
I1	0	0	0,39205	0,40531	0,41861
I2	0,39173	0	0,39235	0,33007	0,41982
I3	0,39205	0,39235	0	0,40026	0,42475
I4	0,40531	0,33007	0,40026	0	0,39065
I5	0,41861	0,41982	0,42475	0,39065	0

Na Figura 18 é possível verificar que algumas das imagens retornadas não são similares à imagem *R1*.

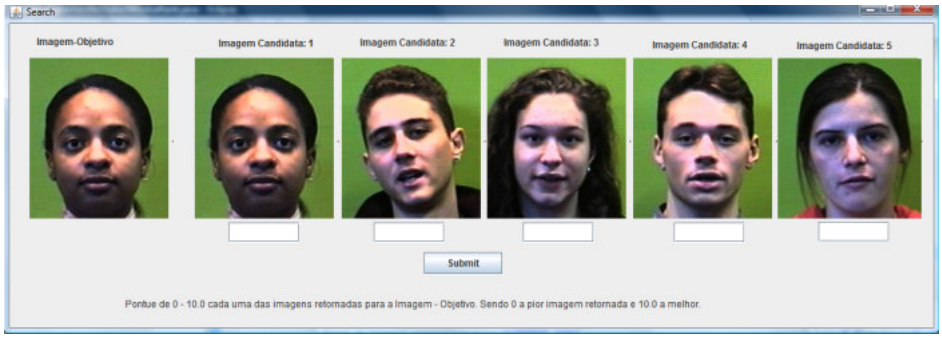


Figura 18: Primeira interação com a imagem *R1*.

Figura 19: Sexta interação com a imagem *R1*.

Na Tabela 3 foi feito um estudo comparativo do vetor de características da imagem *R1* com cada imagem candidata resultante, com o objetivo de verificar quais extratores impactaram na distância Euclidiana e tentar concluir se houve algum extrator 'ofensor' para a busca.

Tabela 3: Relação das distância dos extratores de R1 e das imagens candidatas

Relação das distância dos extratores de R1 e das imagens Candidatas					
	Imagens - Candidatas				
Extratores	1	2	3	4	5
Média do Histograma em Nivel de Cinza	0,0000	0,071	0,006	0,039	0,036
Média do Histograma em RGB	0,0000	0,578	0,003	6,417	3,533
Desvio Padrão do Histograma em Nivel de Cinza	0,0000	2,372	2,593	0,864	2,372
Desvio Padrão do Histograma em RGB	0,0000	0,130	0,002	0,030	0,044
Obliquidade do Histograma em Nível de Cinza	**0,0000**	**38,583**	**12,505**	**78,023**	**118,812**
Obliquidade do Histograma em RGB	**0,0000**	**0,000**	**0,000**	**142,881**	**37,992**
Média dos Pixels em Nivel de Cinza	0,0000	0,071	0,006	0,039	0,036
Média dos Pixels em RGB	0,0000	28,930	45,328	40,039	178,834
Desvio Padrão dos Pixels em Nivel de Cinza	0,0000	17,961	35,033	40,997	17,961
Desvio Padrão dos Pixels em RGB	0,0000	0,130	0,002	0,030	0,044
Obliquidade dos Pixels em Nível de Cinza	**0,0000**	**36,871**	**188,852**	**54,031**	**58,956**
Obliquidade dos Pixels em RGB	**0,0000**	**498,909**	**467,524**	**140,083**	**150,487**
Contraste	0,0000	0,007	0,011	0,016	0,007

Entropia	0,0000	0,005	0,005	0,008	0,005
Escala	0,0000	0,013	0,052	0,060	0,062
Rotação	0,0000	0,016	0,063	0,074	0,077
Translação	0,0000	0,006	0,104	0,083	0,044

Pela Tabela 3 vemos que alguns extratores como obliquidade produziram dados discrepantes e que colaboraram para a distância Euclidiana não retornar as imagens mais relevantes.

É sabido que esse tipo de extrator é mais indicado quando há imagens em cenários mais complexos, ao contrário da nossa base de testes que possuem todas fundo verde e iluminação controlada. Porém, depedendo do sistema não é possível o usuário escolher os extratores corretos, ou até mesmo, nem é possível conhecer quais extratores são os mais indicados para determinada situação.

Aplicando a realimentação por relevância por meio de notas variando de 0 a 10 e o processamento de ajustes de peso por meio da lógica *fuzzy*, o usuário consegue minimizar o erro causado pelos extratores e refinando a sua busca.

Isso pode ser constatado por meio do gráfico mostrado pela Figura 19, onde para a imagem *R1*, houve a necessidade de apenas 7 interações para a busca das outras 5 imagens que eram similares à ela. Para a imagem *R2*, foram feitas 10 interações e para a imagem R4 apenas 6 avaliações.

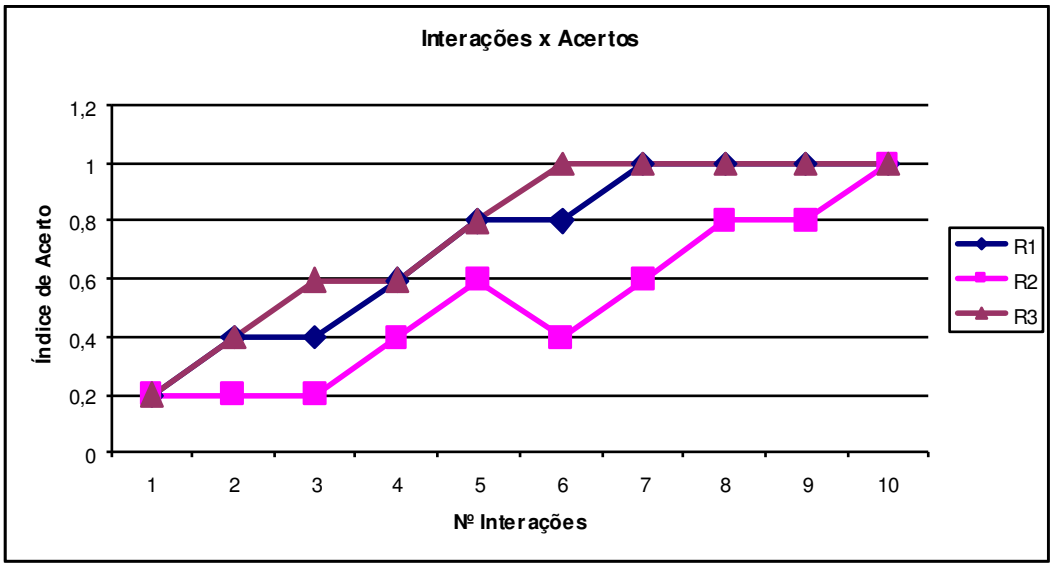
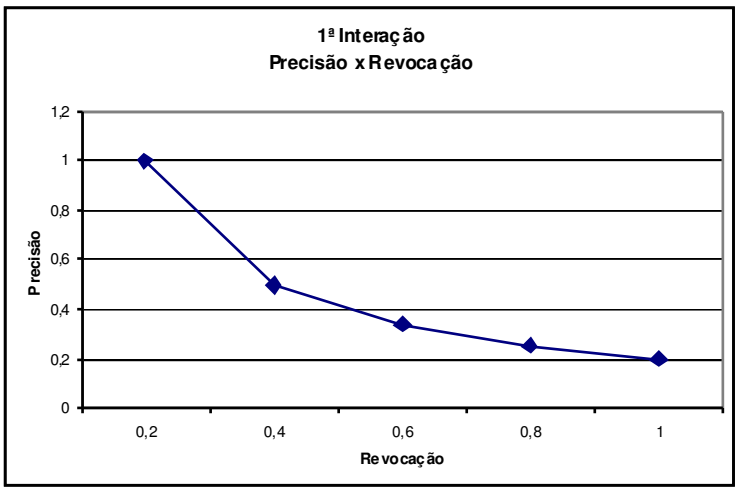
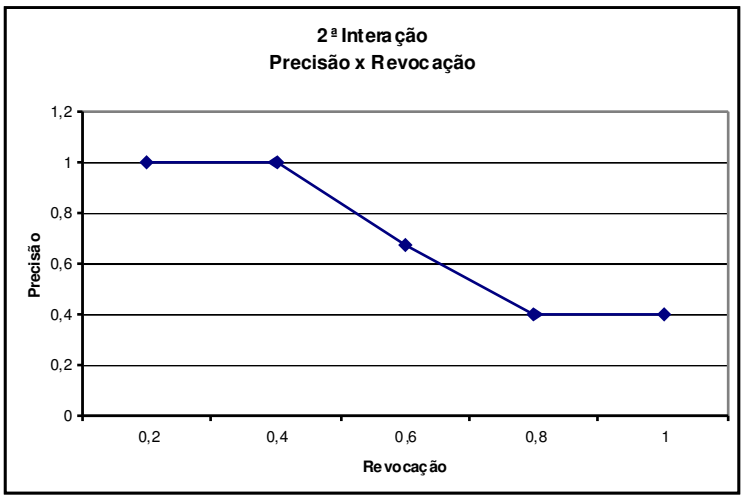
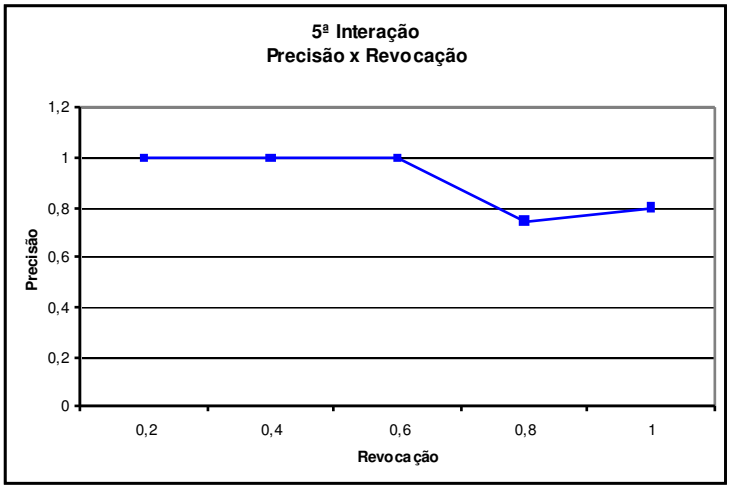


Figura 20: Gráfico da relação entre interações e acertos

Os gráficos de precisão e revocação da imagem *R1* está demonstrado na Figura 20. Vemos que no decorrer das interações as áreas dos gráficos foi aumentando, indicando que houve uma melhora nos resultados das buscas conforme os usuários avaliavam as imagens candidatas.







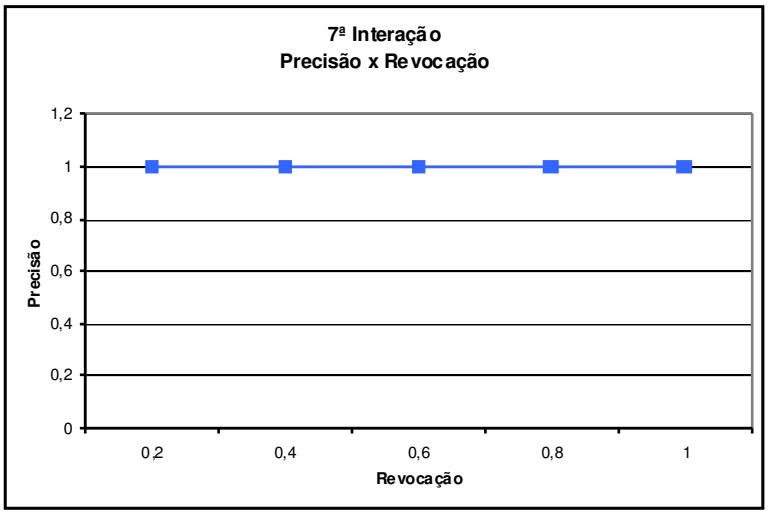


Figura 21: Gráficos de Precisão x Revocação das imagens de teste

6. Discussão

Alguns pontos foram levantados no decorrer do projeto, principalmente na fase de testes e conclusão do mesmo.

O primeiro deles é sobre os extratores e funções de similaridade. Como visto anteriormente, houve extratores que prejudicaram a distância Euclidiana com valores que se diferenciavam em grande escala dos demais. No sistema foram utilizados extratores de todas as categorias (cor, textura e forma), porém há alguns estudos que apontam para a relação da eficiência de determinados extratores no contexto em que são inseridos.

Outro ponto em destaque foi a função de similaridade usada. A distância Euclidiana retorna um valor global, reflexo do conjunto de todos os extratores, isso também ajudou a trazer resultados menos coerentes ao usuário. Uma vez que o extrator estava retornando resultados ruins, o valor da distância Euclidiana também piorava.

O uso da técnica de realimentação por relevância em conjunto com a lógica *fuzzy* produziu resultados bastantes satisfatórios, porém o ajuste de pesos ocorreu no valor da distância Euclidiana de cada imagem candidata, ou seja ele foi alterado globalmente, não atacando um extrator 'ofensor' que talvez estivesse prejudicando a busca. Essa mudança no algoritmo poderia ter acelerado e melhorado a qualidade dos resultados do sistema, sendo indicada como uma possível continuação do projeto.

7. Conclusão

O objetivo do projeto era o reconhecimento facial utilizando um sistema de CBIR utilizando-se a lógica *Fuzzy*. Para alcançá-lo foi necessário o estudo e implementação de conceitos de processamento de imagem, inteligência artificial e realimentação por relevância. Ao seu término foi possível a constatação de que o método de realimentação por relevância utilizando a lógica *fuzzy* teve um resultado bem satisfatório, retornando em poucas interações todas as imagens pertencentes ao grupo de testes. Apesar de alguns extratores terem interferido negativamente no resultado da distância Euclidiana, a realimentação por relevância conseguiu suprir essa deficiência e ter bons resultados.

Desse modo foi visto o quão importante é a utilização de métodos que aproximam o usuário do processo de busca, diminuindo o *gap* semântico que ainda existe entre a percepção do usuário e a lógica matemática da computação. É possível que ambas trabalhem juntas e produzam resultados cada vez mais significativos.

É totalmente pertinente que em estudos futuros a questão do uso de técnicas de realimentação e inteligência artificial possam ser estendidas para a melhoria dos próprios extratores. Pode-se, por exemplo, analisar cada posição do vetor de características, dar-lhe um peso e ele ser ajustado durante as interações, identificando extratores ofensores e capitalizando os que estão dando bons resultados.

8. Referências Bibliográficas

ASLANDOGAN, Y.A.; YU, C.T. **Techniques and Systems for Image and Video Retrieval**, IEEE Transactions on Knowledge and Data Engineering, Vol. 11, No. 1, Fevereiro, 1999.

BALAN, A. G. R. **Métodos adaptativos de segmentação aplicados à recuperação de imagem por conteúdo.** 2007. 183f. Tese (Doutorado em Ciência da Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e Computação, Universidade de São Paulo, São Paulo. cap. 2, p. 7-15.

ESSEX UNIVERSITY. Collection of Facial Images. Inglaterra, 2008. Disponível em: http://www.face-rec.org. Acessada em: Março, 2010

JUNGES, L.C.D. **Introdução a Lógica Fuzzy**. 2006. Disponível em: <s2i.das.ufsc.br/tikiwiki/apresentacoes/logica-fuzzy.pdf> Acesso em: 26 Ago. 2010.

LAUFNER, R. P. **Sistemas Fuzzy com Matlab.** 2003. Disponível em: < http://www.scribd.com/doc/22431838/Fuzzy-Matlab>. Acessado em : 03 Nov. 2010

MARQUES, J. **Realimentação de relevância para recuperação por conteúdo de imagens médicas visando diminuir a descontinuidade semântica.** 2006. Dissertação (Mestrado em Ciências da Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e Computação, Universidade de São Paulo, São Paulo.

MARTINS, E. R. da S. et al . **Caracterização de lesões intersticiais de pulmão em radiograma de tórax utilizando análise local de textura**. Radiol Bras, São Paulo, v. 38, n. 6, Dezembro, 2005 . Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-39842005000600008&lng=en&nrm=iso>. Acessado em: 14 Out. 2010.

MATOS. T. A. **Índice Invertido para Recuperação de Imagens Baseada em Conteúdo.** In: Congresso Nacional de Matemática Aplicada, 31. 2008, Bélem. **Anais...** Minas Gerais: Faculdade de Ciência da Computação, Universidade Federal de Uberlândia, 2008.

MÁXIMO, A. **A importância do mapeamento da criminalidade utilizando-se tecnologia de sistema de informação geográfica para auxiliar a segurança pública no combate à violência**. 2004. 101f. Dissertação (Mestrado em Engenharia de Produção e Sistemas) – PPGEP, Universidade Federal de Santa Catarina, Florianópolis. cap. 1, p. 11-13.

OLIVEIRA, L. E. S. **Inteligência Computacional – Extraindo Características**. In: Curso de Especialização em Inteligência Computacional, 1. 2005, Paraná. **Resumos...** Paraná: Pontifícia Universidade Católica, 2005.

OLIVEIRA, R. A. P. **Estrutura para Utilizaçào de CBIR em Oráculos Gráficos**. 2008. 89f. Monografia (Bacharel em Ciência da Computação) - Centro Universitário Eurípides de Marília – UNIVEM, São Paulo. cap. 2, p. 38-68.

PEDRYCZ, W.; GOMIDE F. **An introduction of Fuzzy Sets: Analysis and Design**. Mit Press, 1998.

PERES, S. M. **Teorias do Conjunto Fuzzy.** 2009. Disponível em: < http://www.each.usp.br/sarajane/SubPaginas/disciplinas.htm> Acessado em: 04 Out. 2010.

SANDRI, S.; CORREA, C. **Lógica Nebulosa.** In: Escola de Redes Neurais, 5. 1999, São Paulo. **Anais...** São Paulo: Instituto de Tecnologia e Aeronáutica – ITA, 1999.

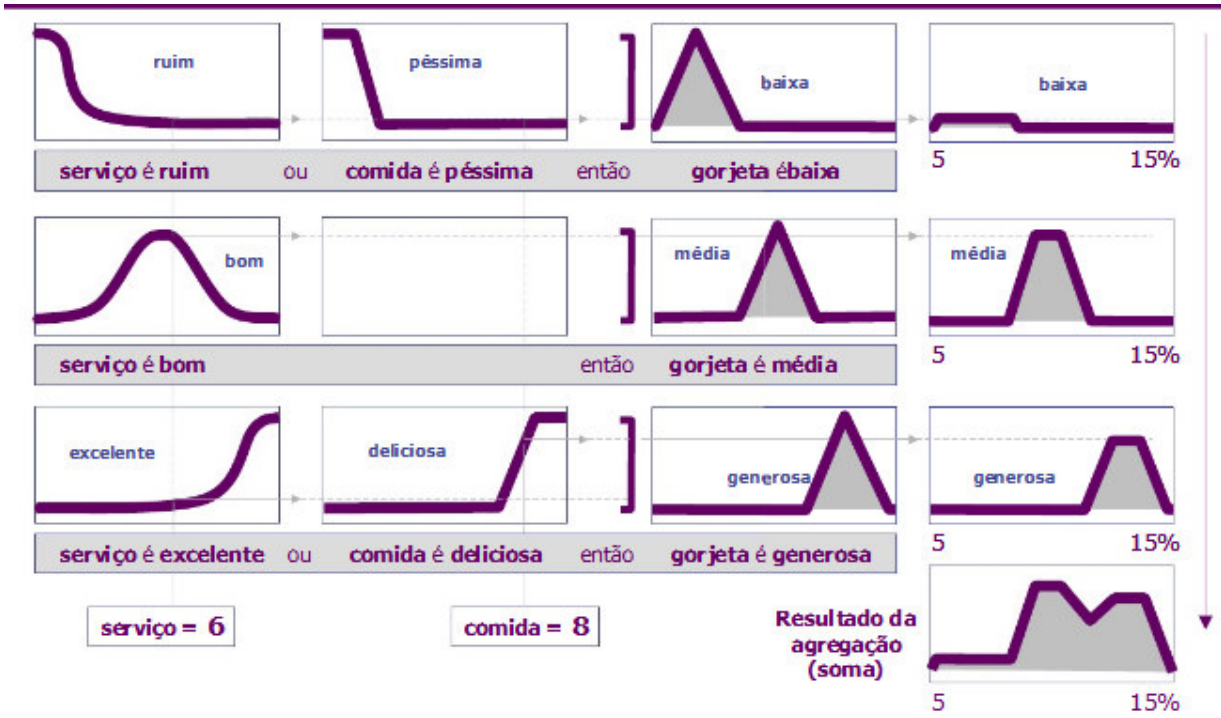
SANTOS, R. JAI: **Java Advanced Imaging.** In: Workshop de Visão Computacional, 4. 2008, São Paulo. **Anais...** São Paulo: Escola de Engenharia de São Carlos – USP, São Paulo, 2008.

SILVA, A.T da. **Descritores de Imagem. (Review).** 2008. Disponível em: < http://calhau.dca.fee.unicamp.br/wiki/images/0/06/Ats_ProjetoFinal.pdf> Acessado em: 24 Set. 2010.

SOUZA, L.B.L. de; SANTOS, J.C. de S; GULIATO, D. **Um extrator de características baseado em complexidade aplicado à classificação de tumores de mama.** In: Seminário de Iniciação Científica, 12. 2008, Uberlândia. **Anais...** Minas Gerais: Universidade Federal de Uberlância, 2008

TORRES, R. da S.; FALCÃO, A. X. **Recuperação de Imagens Baseadas em Conteúdo**. In: Workshop de Visão Computacional, 4. 2008, Bauru. **Anais...** São Paulo: Universidade Estadual de São Paulo - UNESP, 2008.

ANEXO A – Exemplo de Inferência *fuzzy*



APENDICE A- Código dos extratores

Média, Desvio e Obliquidade do Histograma RGB:

```
public double[] mediaDesvioRGB(String path){

      double sumR=0, sumG=0, sumB =0;
      double mediaInR =0, mediaInG=0, mediaInB=0 ;
      mediaR =0;
      mediaG=0;
      mediaB=0;
      double sumDesvioR = 0, sumDesvioG = 0, sumDesvioB = 0;
      desvioR=0;
      desvioG=0;
      desvioB=0;
      double sumInclR=0, sumInclG=0, sumInclB=0;
      inclR=0;
      inclG=0;
      inclB=0;

      double[] vetor = new double[6];
      PlanarImage image = JAI.create("fileload",path);
      ParameterBlock pb = new ParameterBlock();
       pb.addSource(image);
       pb.add(null); // The ROI.
       pb.add(1); // Samplings.
       pb.add(1);
       pb.add(new int[]{256}); // Num. bins.
       pb.add(new double[]{0}); // Min value to be considered.
       pb.add(new double[]{256}); // Max value to be considered.

      /*Cria o histograma*/
       PlanarImage temp = JAI.create("histogram", pb);
       Histogram h = (Histogram)temp.getProperty("histogram");

           for (int j = 0; j < 256; j++) {
             sumR = sumR + (h.getBinSize(0, j)*j);
           }
           for (int j = 0; j < 256; j++) {
               sumG = sumG + (h.getBinSize(1, j)*j);
           }
           for (int j = 0; j < 256; j++) {
               sumB = sumB + (h.getBinSize(2, j)*j);
           }

       mediaInR = sumR/(image.getHeight()*image.getWidth());
       mediaInG = sumG/(image.getHeight()*image.getWidth());
       mediaInB = sumB/(image.getHeight()*image.getWidth());

       mediaR = normalizeCores(mediaInR);
       mediaG = normalizeCores(mediaInG);
       mediaB = normalizeCores(mediaInB);

   for (int i = 0; i < h.getNumBands(); i++) {
        for (int j = 0; j < h.getNumBins(i); j++) {
          sumDesvioR =sumDesvioR + Math.pow(Math.abs((h.getBinSize(0,
j)-mediaR)),2);            ;
```

```
          sumDesvioG =sumDesvioG + Math.pow(Math.abs((h.getBinSize(1,
j)-mediaG)),2);
          sumDesvioB =sumDesvioB + Math.pow(Math.abs((h.getBinSize(2,
j)-mediaB)),2);
        }
   }
   if(sumDesvioR == 0){
      desvioR = 0.0;
      desvioG =
Math.sqrt(sumDesvioG/(image.getHeight()*image.getWidth()-1));
      desvioB =
Math.sqrt(sumDesvioB/(image.getHeight()*image.getWidth()-1));
   }else if(sumDesvioG == 0){
      desvioR =
Math.sqrt(sumDesvioR/(image.getHeight()*image.getWidth()-1));
      desvioG = 0.0;
      desvioB =
Math.sqrt(sumDesvioB/(image.getHeight()*image.getWidth()-1));
   }else if(sumDesvioB == 0){
      desvioR =
Math.sqrt(sumDesvioR/(image.getHeight()*image.getWidth()-1));
      desvioG =
Math.sqrt(sumDesvioG/(image.getHeight()*image.getWidth()-1));
      desvioB = 0.0;
   }else{
      desvioR =
Math.sqrt(sumDesvioR/(image.getHeight()*image.getWidth()-1));
      desvioG =
Math.sqrt(sumDesvioG/(image.getHeight()*image.getWidth()-1));
      desvioB =
Math.sqrt(sumDesvioB/(image.getHeight()*image.getWidth()-1));
   }

   for (int i = 0; i < h.getNumBands(); i++) {
        for (int j = 0; j < h.getNumBins(i); j++) {
          sumInclR =sumInclR + Math.pow(Math.abs((h.getBinSize(0, j)-
mediaR)),3);            ;
          sumInclG =sumInclG + Math.pow(Math.abs((h.getBinSize(1, j)-
mediaG)),3);
          sumInclB =sumInclB + Math.pow(Math.abs((h.getBinSize(2, j)-
mediaB)),3);
        }
   }
   if(sumInclR == 0){
      inclR = 0.0;
      inclG = Math.cbrt(sumDesvioG/(image.getHeight()*image.getWidth()-
1));
      inclB = Math.cbrt(sumDesvioB/(image.getHeight()*image.getWidth()-
1));
   }else if(sumInclG == 0){
      inclR = Math.cbrt(sumInclR/(image.getHeight()*image.getWidth()-
1));
      inclG = 0.0;
      inclB = Math.cbrt(sumInclB/(image.getHeight()*image.getWidth()-
1));
   }else if(sumInclB == 0){
      inclR = Math.cbrt(sumInclR/(image.getHeight()*image.getWidth()-
1));
      inclG = Math.cbrt(sumInclG/(image.getHeight()*image.getWidth()-
1));
      inclB = 0.0;
```

```
   }else{
      inclR = Math.cbrt(sumInclR/(image.getHeight()*image.getWidth()-
1));
      inclG = Math.cbrt(sumInclG/(image.getHeight()*image.getWidth()-
1));
      inclB = Math.cbrt(sumInclB/(image.getHeight()*image.getWidth()-
1));
   }

   vetor[0] = inclR;
   vetor[1] = inclG;
   vetor[2] = inclB;

   return vetor;
```

Média, Desvio e Obliquidade do Histograma em Nível de Cinza:

```
public double[] mediaDesvioCinza(String path){
      double sum = 0;
      double mediaIn =0;
      media = 0;
      desvio = 0;
      incl=0;
      double sumDesvio = 0;
      double sumIncl = 0;

      /* Transforma em cinza*
       * Equação de Luminância:
       * Relação entre XYZ e RGB+1. Saída = uma banda, por isso uma
linha*/
      double[][] bandCombine = {{0.5f,0.71f,0.25f,0.0f}};
      double[] vetorCaracteristicas = new double [2];
      PlanarImage image = JAI.create("fileload",path);

      /*Transformando em níveis de cinza*/
      PlanarImage imageBand = JAI.create("BandCombine",image,
bandCombine);
      ParameterBlock pb = new ParameterBlock();
       pb.addSource(imageBand);
       pb.add(null); // The ROI.
       pb.add(1); // Samplings.
       pb.add(1);
       pb.add(new int[]{256}); // Num. bins.
       pb.add(new double[]{0}); // Min value to be considered.
       pb.add(new double[]{256}); // Max value to be considered.

      /*Cria o histograma*/
       PlanarImage temp = JAI.create("histogram", pb);
       Histogram h = (Histogram)temp.getProperty("histogram");

      for (int i= 0; i < h.getNumBands(); i++) {
         for (int j = 0; j< h.getNumBins(i); j++) {
               sum = sum + (h.getBinSize(i, j)*j);
         }
      }
      mediaIn = (sum/(image.getHeight()*image.getWidth()-1));
      media = normalizeCores(mediaIn);

       for (int i= 0; i < h.getNumBands(); i++) {
           for (int j = 0; j< h.getNumBins(i); j++) {
```

```
            sumDesvio =sumDesvio + Math.pow((Math.abs(h.getBinSize(i,
j)-media)),2);
          }
      }
      desvio = Math.sqrt(sumDesvio/256);

       for (int i= 0; i < h.getNumBands(); i++) {
           for (int j = 0; j< h.getNumBins(i); j++) {
            sumIncl =sumIncl + Math.pow((Math.abs(h.getBinSize(i, j)-
media)),3);
          }
      }
      incl = Math.cbrt(sumIncl/256);

      vetorCaracteristicas[0]=media;
      vetorCaracteristicas[1]=desvio;

      return vetorCaracteristicas;
   }
```

Média, Desvio e Obliquidade dos *Pixels* em RGB:

```
public double[] mediaDesvioRGB(String path){
      double sumR=0, sumG=0, sumB =0;

      double mediaInR, mediaInG, mediaInB;
      double sumDesvioR = 0, sumDesvioG = 0, sumDesvioB = 0;
      double sumInclR=0, sumInclG=0, sumInclB=0;

      double [] vetor = new double[6];
      PlanarImage image = JAI.create("fileload",path);
      Raster rs = image.getData();
      for (int x = 0; x <image.getWidth(); x++) {
              for (int y = 0; y < image.getHeight(); y++) {
                  rs.getPixel(x, y, cores); // captura da combinação de
cor do pixel  "Cores é um VETOR"
                  sumR =sumR+ cores[0];
                  sumG =sumG+ cores[1];
                  sumB =sumB+ cores[2];
              }
      }

      mediaInR = sumR/(image.getHeight()*image.getWidth());
      mediaInB = sumB/(image.getHeight()*image.getWidth());
      mediaInG = sumG/(image.getHeight()*image.getWidth());

      mediaR = normalizeCores(mediaInR);
      mediaG = normalizeCores(mediaInG);
      mediaB = normalizeCores(mediaInB);

      for (int x = 0; x < image.getWidth(); x++) {
           for (int y = 0; y < image.getHeight(); y++) {
            sumDesvioR =sumDesvioR + Math.pow(Math.abs((cores[0]-
mediaR)),2);            ;
            sumDesvioG =sumDesvioG + Math.pow(Math.abs((cores[1]-
mediaG)),2);
            sumDesvioB =sumDesvioB + Math.pow(Math.abs((cores[0]-
mediaB)),2);
          }
      }
```

```
      desvioR =
Math.sqrt(sumDesvioR/(image.getHeight()*image.getWidth()-1));
      desvioG =
Math.sqrt(sumDesvioG/(image.getHeight()*image.getWidth()-1));
      desvioB =
Math.sqrt(sumDesvioB/(image.getHeight()*image.getWidth()-1));


      for (int x = 0; x < image.getWidth(); x++) {
           for (int y = 0; y < image.getHeight(); y++) {
             sumInclR =sumInclR + Math.pow(Math.abs((cores[0]-
mediaR)),3);            ;
             sumInclG =sumInclG + Math.pow(Math.abs((cores[1]-
mediaG)),3);
             sumInclB =sumInclB + Math.pow(Math.abs((cores[2]-
mediaB)),3);
           }
      }
      if(sumInclR == 0){
         inclR = 0.0;
         inclG =
Math.cbrt(sumDesvioG/(image.getHeight()*image.getWidth()-1));
         inclB =
Math.cbrt(sumDesvioB/(image.getHeight()*image.getWidth()-1));
      }else if(sumInclG == 0){
         inclR =
Math.cbrt(sumInclR/(image.getHeight()*image.getWidth()-1));
         inclG = 0.0;
         inclB =
Math.cbrt(sumInclB/(image.getHeight()*image.getWidth()-1));
      }else if(sumInclB == 0){
         inclR =
Math.cbrt(sumInclR/(image.getHeight()*image.getWidth()-1));
         inclG =
Math.cbrt(sumInclG/(image.getHeight()*image.getWidth()-1));
         inclB = 0.0;
      }else{
         inclR =
Math.cbrt(sumInclR/(image.getHeight()*image.getWidth()-1));
         inclG =
Math.cbrt(sumInclG/(image.getHeight()*image.getWidth()-1));
         inclB =
Math.cbrt(sumInclB/(image.getHeight()*image.getWidth()-1));
      }
      vetor[0] = mediaR;
      vetor[1] = mediaG;
      vetor[2] = mediaB;
      vetor[3] = desvioR;
      vetor[4] = desvioG;
      vetor[5] = desvioB;

      return vetor;
      }
```

Média, Desvio e Obliquidade em *Pixels* em Nível de Cinza:

```
public double[] mediaDesvioCinza(String path){
      double sum = 0;
      double sumDesvio = 0;
      double sumIncl = 0;
      double mediaIn;

      PlanarImage image = JAI.create("fileload",path);
      double[] vetorCaracteristicas = new double [2];
      double[][] bandCombine = {{0.5f,0.71f,0.25f,0.0f}};
      PlanarImage bandObj = JAI.create("BandCombine",image,
bandCombine);

      Raster rs = bandObj.getData();
      for (int x = 0; x <bandObj.getWidth(); x++) {
              for (int y = 0; y < bandObj.getHeight(); y++) {
                  rs.getPixel(x, y, cores); // captura da combinação de
cor do pixel  "Cores é um VETOR"
                  sum = sum + cores[0];
              }
      }
      mediaIn = sum/(bandObj.getHeight()*bandObj.getWidth());
      media = normalizeCores(mediaIn);
      for (int x = 0; x < bandObj.getWidth(); x++) {
           for (int y = 0; y < bandObj.getHeight(); y++) {
             sumDesvio = sumDesvio+ Math.pow(Math.abs((cores[0]-
media)),2);
           }
      }

      desvio =
Math.sqrt(sumDesvio/(bandObj.getHeight()*bandObj.getWidth()-1));

      for (int x = 0; x < bandObj.getWidth(); x++) {
           for (int y = 0; y < bandObj.getHeight(); y++) {
             sumIncl =sumIncl + Math.pow((Math.abs(cores[0]-media)),3);
          }
      }
      incl = Math.cbrt(sumIncl/256);

      vetorCaracteristicas[0]=media;
      vetorCaracteristicas[1]=desvio;

      return vetorCaracteristicas;

   }
```

Contraste:

```
public double calculaContraste(int[][] matriz){
      double sumComponentes = 0.0;
      double sumSub = 0;

      double[][] normalizada = new double[4][4];
      double[][] normal = new double[4][4];
      for(int i=0; i<matriz.length; i++){
         for(int j=0; j<matriz.length; j++){
            sumComponentes += matriz[i][j];
         }
      }
      for(int i=0; i<matriz.length; i++){
         for(int j=0; j<matriz.length; j++){
            normalizada[i][j] = (matriz[i][j]/sumComponentes);
         }
      }
```

Entropia:

```
public double calculaEntropia(int[][] matriz){
      double sumComponentes = 0.0;
      double[][] normalizada = new double[4][4];
      double sum = 0.0;

      for(int i=0; i<matriz.length; i++){
         for(int j=0; j<matriz.length; j++){
            sumComponentes += matriz[i][j];
         }
      }
      for(int i=0; i<matriz.length; i++){
         for(int j=0; j<matriz.length; j++){
            normalizada[i][j] = (matriz[i][j]/sumComponentes);
         }
      }
      for(int i=0; i<matriz.length; i++){
         for(int j=0; j<matriz.length; j++){
               sum += (normalizada[i][j]/(1 + Math.abs(i-j)));
         }

      }

      return  sum;
    }
```

Momentos invariantes de Hu – Escala, Rotação e Translação:

```
double calculaCentral(int[][] matriz,int p,int q, int l, int alt){
      double xl, yl;
      double sum = 0.0;

      xl =
calculaMomento(matriz,1,0,l,alt)/calculaMomento(matriz,0,0,l,alt);
      yl =
calculaMomento(matriz,0,1,l,alt)/calculaMomento(matriz,0,0,l,alt);

       for(int i = 0; i< l; i++){
               for(int j = 0; j< alt; j++){
                 sum += ((Math.pow((matriz[i][j]- xl),p))*
(Math.pow((matriz[i][j]-yl),q))*matriz[i][j]);
               }
       }
      return sum;
   }

   double norma(int[][]matriz, int p, int q, int l, int alt){
      double central =0;
      central = (calculaCentral(matriz,p,q, l,
alt)/(Math.pow(calculaCentral(matriz,0,0,l,alt),(((p+q)/2)+1))));
      return central;
   }

   double escolheNorma(int[][]matriz,int opt, int l, int alt){
      double valor=0.0;

      switch(opt){
      case 1:
         valor=0;
         valor=norma(matriz,2,0,l,alt)+norma(matriz,0,2,l,alt);
         break;
      case 2:
         valor=0;
         valor=Math.pow((norma(matriz,2,0,l,alt)-
norma(matriz,0,2,l,alt)),2)+(4*Math.pow(norma(matriz,1,1,l,alt), 2));
         break;
      case 3:
         valor=0;
         valor=Math.pow((norma(matriz,3,0,l,alt)-
3*norma(matriz,1,2,l,alt)),2)+ Math.pow(3*norma(matriz,2,1,l,alt)-
(norma(matriz,0,3,l,alt)),2);
         valor = valor/10000.0;
         break;
      default:
         System.out.println("Valor inválido");
         break;
      }
      return valor;

   }
```