FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
CENTRO UNIVERSITÁRIO "EURÍPIDES DE MARÍLIA" - UNIVEM
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ROGÉRIO MACHADO MENDONÇA

ARMAZENAMENTO E RECUPERAÇÃO DE IMAGENS PARA TESTES DE UM SISTEMA DE RECONHECIMENTO DE IMPRESSÃO DIGITAL

MARÍLIA
2007

ROGÉRIO MACHADO MENDONÇA

ARMAZENAMENTO E RECUPERAÇÃO DE IMAGENS PARA TESTES DE UM SISTEMA DE RECONHECIMENTO DE IMPRESSÃO DIGITAL

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação do Centro Universitário Eurípides de Marília (UNIVEM), mantido pela Fundação de Ensino Eurípides Soares da Rocha, para obtenção do Título de Bacharel em Ciência da Computação.

Orientador (a):
Profª. Drª. Fátima L. S. Nunes Marques

MARÍLIA
2007

MENDONÇA, Rogério Machado

Armazenamento e Recuperação de imagens para testes de um Sistema de Reconhecimento de Impressão Digital / Rogério Machado Mendonça; orientadora: Fátima L. S. Nunes. Marília, SP: [s.n.], 2007.

72 f.

Monografia (Graduação em Computação) – Centro Universitário Eurípides de Marília – UNIVEM – Fundação de Ensino Eurípides Soares da Rocha.

1.Banco de Dados 2. Impressão Digital 3.Biometria.

CDD: 005.74

ROGÉRIO MACHADO MENDONÇA

ARMAZENAMENTO E RECUPERAÇÃO DE IMAGENS PARA TESTES DE UM SISTEMA DE RECONHECIMENTO DE IMPRESSÃO DIGITAL

Banca examinadora da monografia apresentada ao Programa de Graduação do UNIVEM, /F.E.E.S.R., para obtenção do Bacharel em Graduação em Ciência da Computação. Área de Concentração: Banco de Dados e Biometria.

Resultado: _____

ORIENTADOR (A): Profª. Drª. _____

1º EXAMINADOR: _____

2º EXAMINADOR: _____

Marília, ___ de _____ de 2007.

A Deus por ter dado seu filho para morrer na cruz e Salvar a Humanidade permitindo que hoje eu esteja vivo para usufruir dos dons concebidos.

AGRADECIMENTOS

Primeiramente a Deus por ter me dado forças para realizar este trabalho e poder terminar mais uma etapa da minha vida.

À minha orientadora Fátima L. S. Nunes Marques por ter despertado em mim a vontade de pesquisar.

À minha namorada pelo apoio e estímulo nas horas mais difíceis, e a compreensão mesmo quando me dediquei mais a este trabalho.

Ao meu grupo de sala que desde o primeiro ano tem mostrado que a união faz a força e estão terminando mais esta etapa juntamente comigo.

Ao professor José Carlos Miguel de Mendonça por ter acreditado em mim e me apoiado desde o primeiro ano.

A todos os professores que passaram o seu conhecimento para nos fazer um pouco melhor.

E à faculdade como um todo por ter mudado a minha vida profissional e pelo apoio financeiro.

MENDONÇA, Rogério Machado. **Armazenamento e Recuperação de imagens para testes de um Sistema de Reconhecimento de Impressão Digital.** 2007 72f. Monografia (Graduação em Computação) - Centro Universitário Eurípides de Marília - Fundação de Ensino Eurípides Soares da Rocha, Marilia, 2007.

RESUMO

O foco deste trabalho é estudar e definir a melhor forma de armazenamento de imagens com atributos específicos que caracterizam uma impressão digital e desenvolver técnicas e aprimoramentos para recuperação da imagem com maior desempenho a fim de permitir testes em um sistema de reconhecimento da mesma. São apresentados conceitos de biometria e suas definições que são utilizadas dentro do contexto de impressão digital através de técnicas como a extração de minúcias, reconhecimento de padrões e armazenamento de imagens. Com a implementação realizada neste trabalho foi possível obter um sistema para reconhecimento de impressão digital com um índice de erro aceitável e um desempenho satisfatório.

Palavras-chave: Banco de Dados, Impressão Digital, Biometria.

MENDONÇA, Rogério Machado. **Armazenamento e Recuperação de imagens para testes de um Sistema de Reconhecimento de Impressão Digital.** 2007 72f. Monografia (Graduação em Computação) - Centro Universitário Eurípides de Marília - Fundação de Ensino Eurípides Soares da Rocha, Marilia, 2007.

ABSTRACT

The focus of this work is to study and to define the best form of storage of images with specific attributes that characterize a fingerprint and to develop techniques and improvements for recovery of the image with better performance in order to allow tests in a system of recognition of the same one. They are presented biometric concepts and your definitions that are used inside of the fingerprint context through techniques as the extraction of details, recognition of patterns and storage of images. With the implements accomplished in this work it was possible to obtain a system for fingerprint recognition with an index of acceptable mistake and a satisfactory acting.

Keywords: Database, Fingerprint, Biometric.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ID: Impressão Digital

WSQ: Wavelet Scalar Quantization

FBI: Federal Bureau of Investigation

AFIS: Automated Fingerprint Identification System

DPI: Dots Per Inch

SUMÁRIO

INTRODUÇÃO

A biometria tem se destacado no contexto atual com grande eficiência no reconhecimento de indivíduos nas mais diversas áreas, fazendo com que a tecnologia e os métodos utilizados na identificação pessoal evoluíssem cada vez mais, possibilitando o surgimento de muitos aplicativos que fazem uso da biometria para embutir segurança e praticidade em seus sistemas automatizados.

Estudando os vários métodos de identificação por biometria, o reconhecimento através de impressão digital apresenta uma boa relação custo benefício e vem sendo largamente utilizado. A coleta desse tipo de imagem é facilmente obtida por meio de leitores específicos que são muito comuns no mercado (COSTA, 2001).

Além da aquisição de impressões digitais, o armazenamento e recuperação destas se tornam pontos importantes para um sistema computacional. Por esses motivos, é interessante o estudo dos bancos de dados existentes e o formato que é utilizado para armazenamento das impressões, o que contribui consideravelmente no desempenho obtido pelo sistema.

É possível citar vários exemplos de banco de dados de impressão digital que têm o seu crescimento rápido e que, na maioria dos casos, usam padrões específicos de armazenamento para que seja possível manter um desempenho aceitável e preparado para aumento constante de registros.

Objetivo do Trabalho

O objetivo geral deste trabalho é definir uma forma de armazenamento de imagens com atributos específicos que caracterizam uma impressão digital e desenvolver técnicas e aprimoramentos para recuperação da imagem com desempenho adequado a fim de permitir testes em um sistema de reconhecimento da mesma.

Os objetivos específicos deste trabalho são:

- Estudar e implementar a aquisição de impressões digitais para um sistema de reconhecimento já existente.
- Pesquisar bancos de dados existentes de impressões digitais para obter experiência das melhores formas de armazenamento;
- Testar as técnicas de armazenamento e recuperação em um sistema de reconhecimento de impressão digital já existente

Organização do Trabalho

Além desta introdução, este trabalho apresenta os seguintes capítulos:

- O Capítulo 1 descreve os conceitos de biometria e as suas derivações, assim como as características que constituem uma impressão digital e suas particularidades.
- O Capítulo 2 apresenta os bancos de dados de impressão digital existentes e relaciona exemplos das principais bases e os formatos utilizados. São apresentados também conceitos de bancos de dados e sistema gerenciador de banco de dados.
- O Capítulo 3 diz respeito aos métodos de trabalho deste estudo sendo apresentadas as funcionalidades principais do sistema de verificação existente e o planejamento das etapas que foram implementadas para se obter o resultado esperado.
- O Capítulo 4 apresenta os resultados obtidos com a implementação, assim como os gráficos de desempenho e a porcentagem de erro que o sistema tem como característica.

CAPÍTULO 1 - RECONHECIMENTO DE IMPRESSÃO DIGITAL

1.1 - Conceitos de Biometria

O termo “Biometria” vem da composição bio (vida) + metria (medida), ou seja, é a medida de características físicas ou comportamentais das pessoas como forma de identificá-las unicamente (PACHECO, 2003). A biometria baseia-se no conceito de identificar uma pessoa através de características físicas (como impressão digital aqui estudada, íris, retina, formato do rosto, veias da palma da mão, geometria da palma da mão e outros) ou através de características comportamentais como a caligrafia e reconhecimento da digitação, por exemplo. Em um futuro muito próximo espera-se que seja possível a identificação de seres através do DNA de forma automatizada e instantânea o que não ocorre hoje devido à necessidade de intervenção humana e que consequentemente gera uma demora no processo, que leva atualmente algumas horas (COSTA, 2001).

Hoje a biometria está presente em diversas áreas sendo utilizada na identificação de indivíduos sobre vários seguimentos como controle de ponto, controle de acesso, identificação criminal, etc. A biometria traz vantagens no sentido de que os utilizadores não necessitam de memorizar qualquer tipo de senha ou carregar consigo objetos como cartão, chaves, e outros para identificação, ou seja, o reconhecimento é realizado simplesmente por características presentes no indivíduo (MATIAS, 2004).

Dentre os vários métodos de identificação através da biometria existem consideráveis vantagens e desvantagens destacando-se o método de reconhecimento por impressão digital que oferece precisão aceitável com uma velocidade alta de reconhecimento e baixo custo, sendo alvo do estudo aqui apresentado.

1.2 - Definição de Impressão Digital

Impressão Digital são desenhos naturais formados por elevações na pele chamadas papilas e que estão presentes nas extremidades dos dedos das mãos, são únicas, ou seja, todo ser humano possui uma impressão digital própria e exclusiva formada desde o feto e são diferentes até em gêmeos idênticos. A impressão digital acompanha o ser humano pela vida inteira apresentando pouquíssimas mudanças durante o envelhecimento natural (COSTA, 2001).

Baseado nisso é possível determinar uma classificação básica que diz respeito as formas apresentadas em um impressão digital, essa classificação divide a impressão em três regiões: região basilar, região nuclear e região marginal, como mostrado na Figura 1.

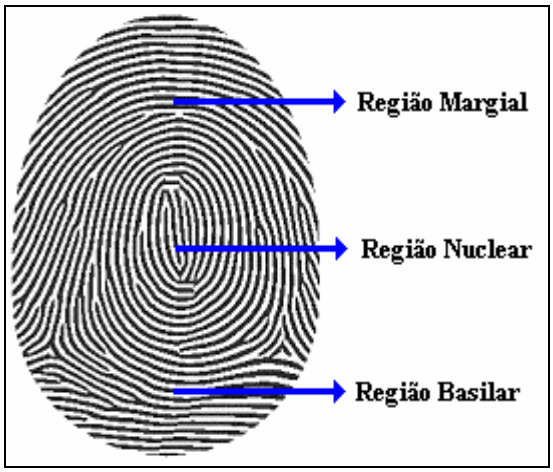


Figura 1 - Regiões dos sistemas de linhas

De acordo com Henry (1905), Francis Galton foi o primeiro a fazer um estudo dos aspectos de impressões digitais definindo que as linhas da impressão digital apresentam acidentes, por exemplo, linhas que terminam abruptamente ou se bifurcam, esses acidentes são chamados de Minúcias ou Detalhes de Galton e podem ser usados para o reconhecimento de impressões digitais.

Outra característica que pode ser analisada em uma impressão digital diz respeito ao núcleo que representa o “centro de gravidade” da impressão, e o Delta que é o ângulo ou triângulo formado pelas cristas papilares, pela brusca divergência das linhas paralelas ou pela bifurcação de uma linha simples, como mostrado na Figura 2.

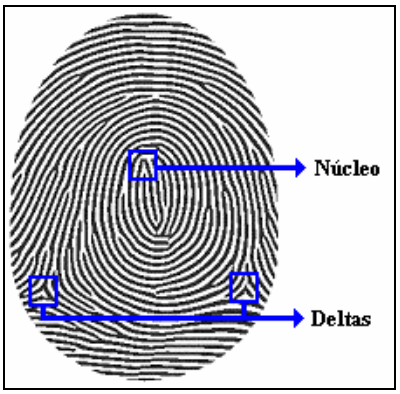


Figura 2 - Em destaque núcleo e deltas de uma ID

Segundo Kehdy (1968, p. 37), no sistema datiloscópico de Vucetich as IDs são classificadas nos tipos: arco, presilha interna, presilha externa e verticilo. O arco apresenta como características a ausência de delta. A presilha interna tem um delta à direita do observador e as linhas nucleares correm para a esquerda do observador. Já a presilha externa tem um delta à esquerda do observador e as linhas nucleares correm para a direita do observador. E o verticilo tem dois deltas, sendo um à direita e outro à esquerda do observador, onde as linhas nucleares ficam encerradas entre os dois deltas, assumindo configurações variadas, como mostrado na Figura 3.

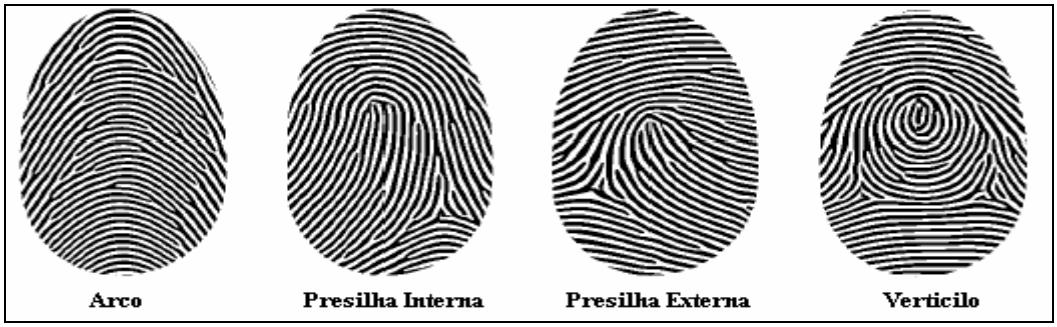


Figura 3 - Tipos fundamentais de IDs no sistema de Juan Vucetich

1.3 - Aquisição de Impressão Digital

Segundo Jain (1997), o leitor biométrico transforma aspectos fisicos extraídos da impressão digital, em um conjunto de características. Um leitor biométrico de impressão digital vem substituindo o antigo método de impressão tintada em papel (método *ink and paper*) que era ineficiente no sentido de acidentes durante a coleta, os leitores eletrônicos atuais extinguem este problema, através da leitura em uma área determinada, o que promove melhor qualidade da imagem capturada. Na Figura 4 são mostrados alguns modelos de leitores de impressão digital atuais.

Figura 4 - Leitores Biométricos atuais

Existem basicamente dois tipos de leitores de impressão digital do ponto de vista eletrônico, são eles: o leitor ótico e o leitor capacitivo. Os dois tipos resultam no mesmo tipo de imagem, mas chegam a ela de maneiras diferentes, sendo que o leitor ótico baseia-se no

princípio de fotosensor, ou seja, utiliza luz, enquanto o leitor capacitivo utiliza corrente elétrica para capturar a imagem dos sulcos e vales que formam as impressões digitais (HARRIS, 2007).

Algumas considerações importantes devem ser levadas em conta na escolha de leitores adequados à aplicação, como formato da imagem gerada, resolução da imagem, tamanho, quantidade de cores, tipo de conexão do leitor, e o custo benefício propiciado. Na Tabela 1 é mostrada a relação que existe entre os leitores mais relevantes do mercado atualmente.

Tabela 1 - Relação de Custo Benefício dos Leitores Atuais

Leitor	Conex.	Preço	Resol.	Imagem	Cores
Microsoft Fingerprint Reader	USB	R$ 139,00	512 dpi	355x390 pixels	256 tons de cinza
APC Biopod	USB	R$ 214,00	500 dpi	260x300 pixels	256 tons de cinza
Crossmatch V250/V300	USB	R$ 317,00	500 dpi	504x480 pixels	256 tons de cinza
Geomok (Testech) NetAcess	USB	R$ 384,00	500 dpi	355x390 pixels	256 tons de cinza
V Sony	USB	R$ 410,00	500 dpi	355x390 pixels	Armazena Pontos
Digital Persona U.are.U 4000	USB	R$ 410,00	512 dpi	355x390 pixels	256 tons de cinza
Hamster I Nitgen	USB	R$ 419,00	500 dpi	260x300 pixels	256 tons de cinza
Hamster II Nitgen	USB	R$ 550,00	500 dpi	260x300 pixels	256 tons de cinza

Este capítulo apresentou uma introdução sobre Biometria e sobre os conceitos relacionados à impressão digital, tais conceitos são importantes para enfatizar o objetivo principal de testar o armazenamento e recuperação de imagens com características particulares, essas características determinam diretamente o reconhecimento da impressão digital como foi descrito anteriormente e a aquisição dessas imagens será proporcionada pelo leitor com melhor custo benefício dentre os apresentados.

O armazenamento e recuperação das imagens adquiridas será estudado no próximo capítulo.

CAPÍTULO 2 - BANCO DE DADOS PARA IMPRESSÃO DIGITAL

2.1 - Banco de Dados e Sistema Gerenciador de Banco de Dados

Segundo Elsmasri e Navathe (2005, p.4), Banco de Dados pode ser definido como uma coleção de dados relacionados, onde os dados são fatos que podem gravados e que possuem um significado. Isso implica em uma organização lógica e coerente, sendo assim uma coleção de dados randômica não pode ser considerada e interpretada como um Banco de Dados.

Um Banco de Dados pode assumir várias formas no contexto atual atrelado à complexidade e ao tamanho do mesmo, ou seja, os dados podem ser mantidos manualmente ou gerenciados por um sistema computadorizado seja através de aplicativos ou através de um software denominado Sistema Gerenciador de Banco de Dados (SGBD).

O SGBD consiste em uma coleção de programas que permite criar e manter um Banco de Dados. Tem por objetivo principal prover um ambiente que seja adequado e eficiente para armazenar e recuperar informações (KORTH, SILBERSCHATZ, 1995, p.1).

Na Figura 5 é mostrada a relação existente entre SGBD e o Banco de Dados:

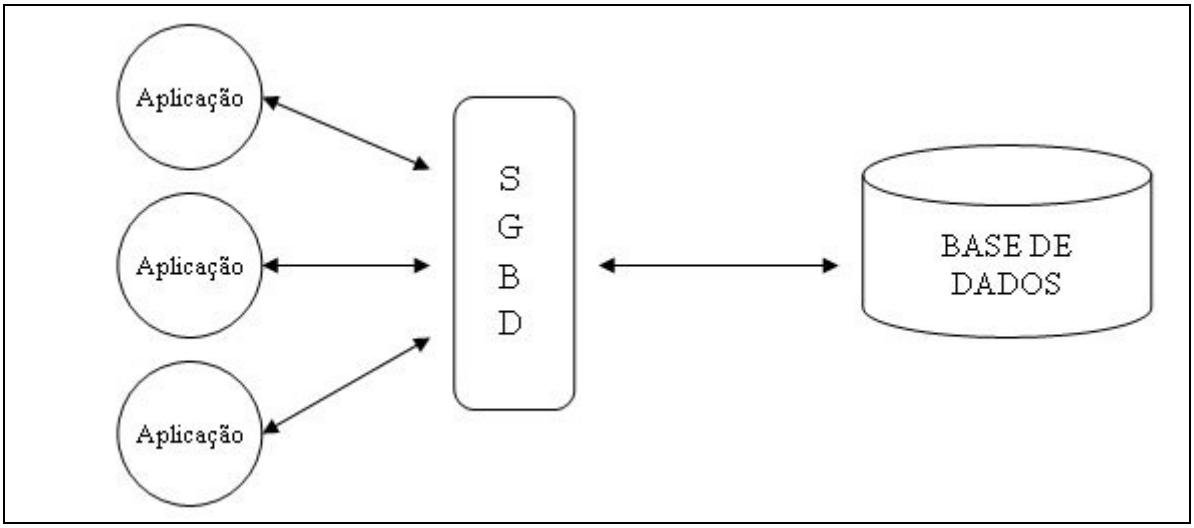


Figura 5 - Relação entre SGBD e Banco de Dados

Assumindo uma abordagem mais técnica, o banco de dados é uma coleção de registros armazenados em um computador de modo sistemático ou organizado, permitindo que um software específico possa consultá-lo para responder a questões que na prática são chamadas de Consultas. Além de consultas, o SGBD permite inserir, alterar e excluir dados levando em conta algumas restrições impostas por determinados bancos, restrições essas que compõem a segurança do banco e na prática são descritas pelo termo integridade referencial.

O termo Banco de Dados deve ser utilizado apenas aos dados em si, enquanto o termo SGBD deve ser aplicado ao software com a capacidade de manipular os dados de forma geral. Porém, é comum misturar os dois conceitos.

Segundo Sanches (2007) o principal objetivo do SGBD é prover ao usuário uma visão abstrata dos dados. Este conceito permite a divisão do sistema de banco em três níveis, como mostrado na Figura 6.

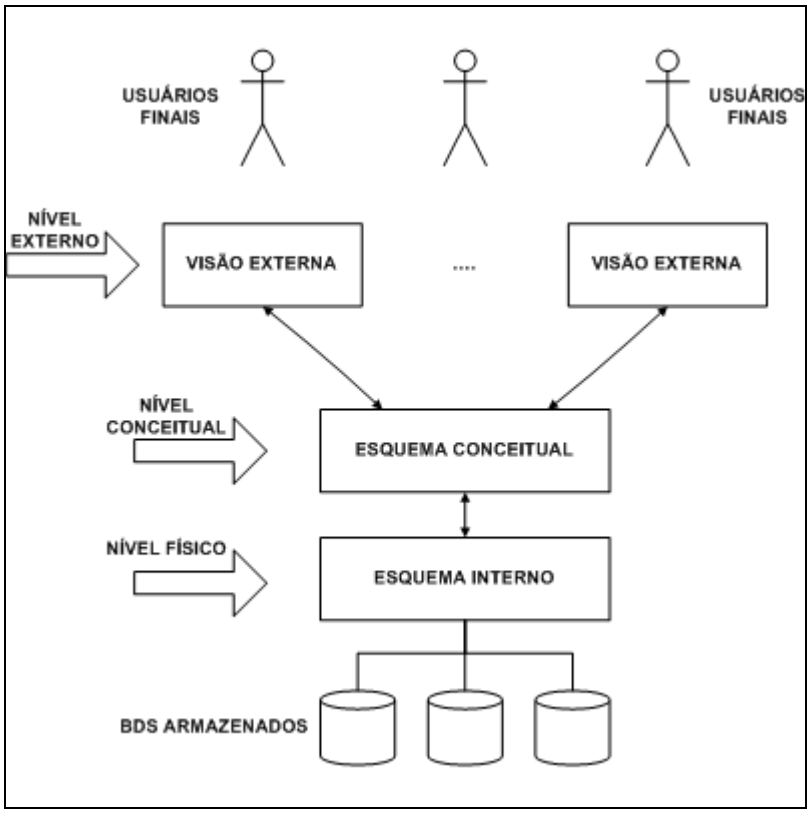


Figura 6 - Níveis de Abstração

2.2 - Bancos de Dados de Impressão Digital existentes

Existem vários bancos de dados alimentados com imagens de impressão digital e que seguem algum formato e tamanho específico, Alguns bancos armazenam milhares de identificações e em diversos casos são extraídas imagens de todos os dedos de cada indivíduo formando uma base de dados quantitativa.

De acordo com Gesellschaft (2006), o banco de dados do FBI em 2006 apresentava mais de 44 milhões de impressões digitais armazenadas em diversos formatos, e aproximadamente 25 milhões no formato WSQ, utilizado também por diversas agências para padronização de transporte e armazenamento de imagens. As imagens seguem um tamanho de 512 x 512 pixels, resolução de 500 dpi e escala de tons de cinza.

O Centro de Pesquisa da Costa do Leste nos Estados Unidos dispõe de uma base de 100.000 imagens de impressão digital em formato sem compressão de alta definição. A resolução das imagens é de 500 dpi e 256 tons de cinza. Como as imagens são utilizadas para testes em AFIS (Sistema automatizado de identificação de impressões digitais), estas foram selecionadas aleatoriamente para se conseguir uma quantidade representativa de impressões com diferente qualidade. A base de dados é comercializada e pode ser adquirida tanto para fins comerciais ou como alvo de estudo e pesquisas (EST, 2007).

Segundo Kovács (2000), o banco de dados “NIST Special Database 4”, tem armazenado 2000 pares de impressões digitais (4000 no total). As imagens são obtidas por digitalização de impressões digitais tintadas em papel com resolução de 500 dpi com 8 bits/pixel e tamanho de 512 x 512 pixels. O banco contém imagens de boa qualidade e imagens deformadas, sendo que 6% delas estão irreconhecíveis.

A FVC (2007) disponibiliza quatro bancos de impressões digitais totalizando 320 imagens que utilizam o formato TIFF e resolução de 500 dpi, as resoluções disponíveis são de 388 x 374 pixels, 296 x 560 pixels, 300 x 300 pixels e 288 x 384 pixels.

Existem diferentes formatos para o armazenamento de imagens, uma vez que se têm várias classes diferentes de representações de imagens. O armazenamento da imagem envolve basicamente três pontos principais: a forma como a imagem está representada, o tipo de compactação empregado e o cabeçalho contendo as informações acerca desta imagem (resolução, número de cores, classe da imagem, *palette*, compactação, etc). Um mesmo tipo de arquivo pode inclusive permitir o armazenamento de diferentes classes de imagens assim como a utilização de vários métodos de compactação (CASACURTA et al, 1998, p. 14).

Quanto maior o número de aplicações em que uma imagem digital pode ser tratada, diz-se que maior é a sua padronização. Normalmente programas gráficos podem ser classificados pela forma com a qual armazenam e apresentam as imagens. Para esta abordagem há duas categorias: formato de varredura e formato vetorial. Entre os formatos de imagens digitais mais utilizados estão: BMP, JPG, GIF, TIFF, PCX, CGM, ICO, RLE, TARGA, PostScript, entre outros (BROWN; SHEPHERD, 1998, p. 73).

Com o estudo dos bancos de dados de impressão digital existentes foi notório que em quase todos os casos a quantidade de informações armazenadas é grande e crescente. Isto justifica a necessidade da medida de desempenho nas consultas realizadas ao banco assim como as inserções e atualizações que podem ocorrer, baseado nisso é interessante estudar e definir as melhores formas de armazenamento e recuperação agregadas ao melhor desempenho obtido, conforme consta na proposta deste projeto, no próximo capítulo.

CAPÍTULO 3 - MATERIAIS E MÉTODOS

3.1 - Apresentação do sistema atual

Para que sejam realizados os testes propostos neste trabalho será utilizado um sistema já existente de reconhecimento de impressão digital construído em linguagem Java. O sistema faz o processamento da imagem através de técnicas conhecidas possibilitando a extração de informações para posterior reconhecimento. O diagrama de classes deste sistema é apresentado na Figura 7.

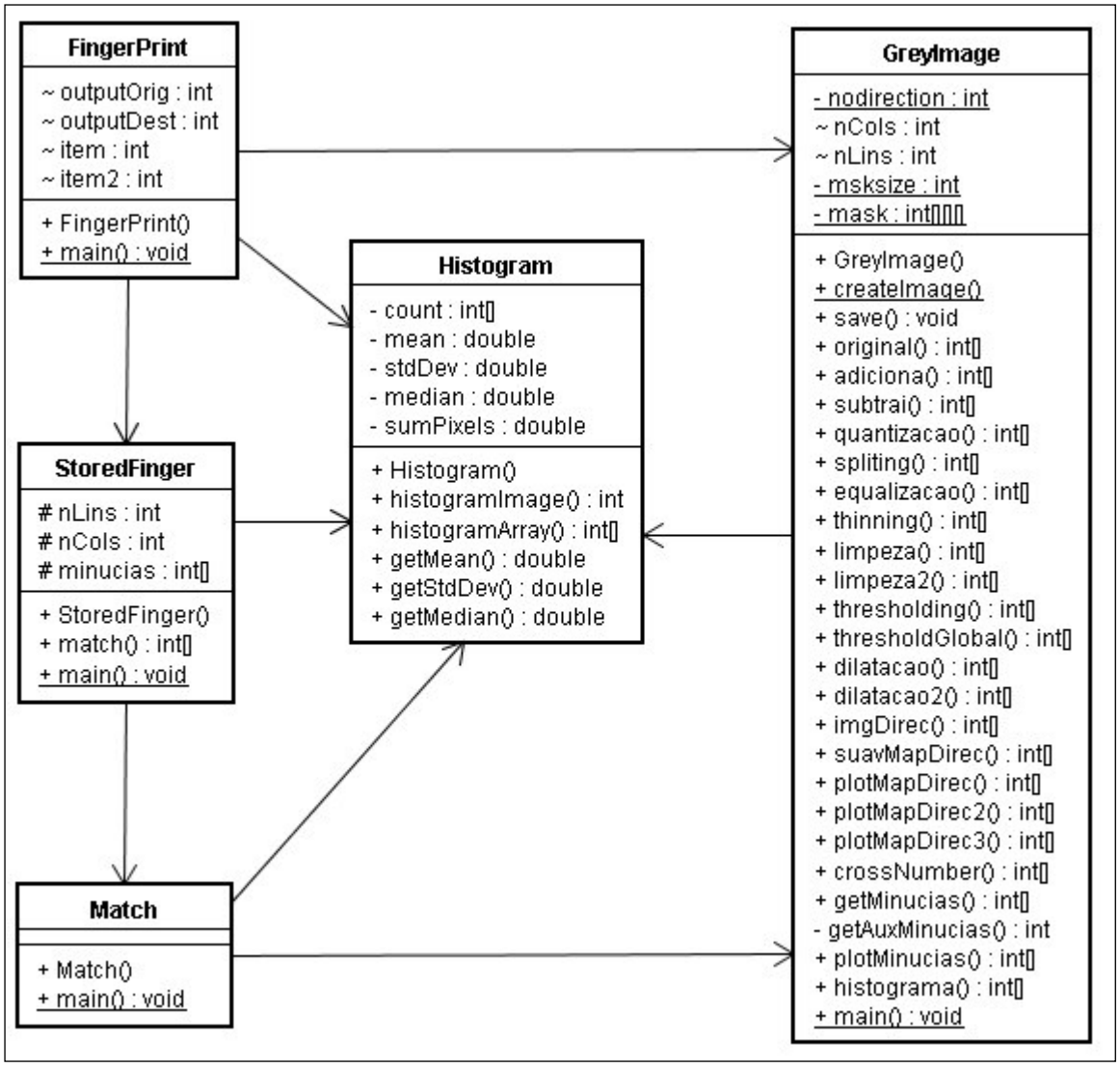


Figura 7 - Diagrama de Classes Fingerprint

Além de extrair informações sobre a imagem corrente o sistema aplica filtros como dilatação, limpeza e outros. As técnicas de processamento empregadas são: equalização, *thesholding e thinning*. O sistema também faz o cálculo da imagem direcional e a suavização do mapa direcional, além da extração e comparação de minúcias (PERENHA 2003).

A interface gráfica do sistema com as chamadas para as funções descritas e as informações extraídas da imagem são apresentadas na Figura 8.

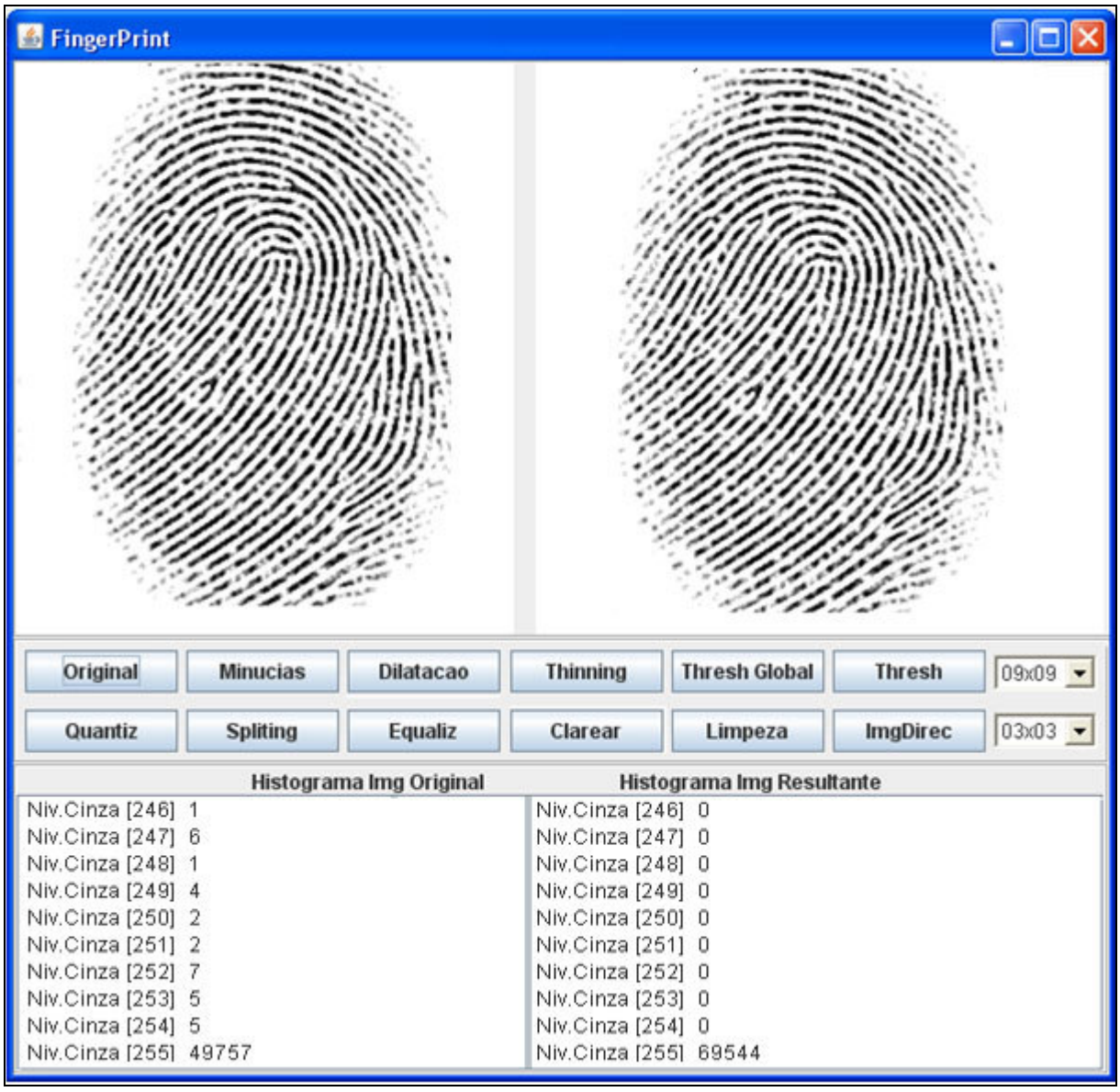


Figura 8 - Interface Gráfica Fingerprint

3.2 - Funcionalidades do sistema Fingerprint

O sistema Fingerprint possui algumas técnicas e métodos que possibilitam o reconhecimento da impressão digital. A seguir são mostradas algumas dessas funcionalidades para que seja possível o entendimento da etapa de armazenamento e recuperação das imagens (PERENHA 2003).

Inicialmente o sistema solicita a imagem de entrada que será utilizada como base para aplicação de todos os filtros e técnicas. A imagem deve ser fornecida com base em seu nome e extensão.

As extensões que são contempladas pelo sistema são: bitmap, gif, jpeg e png, sendo que a imagem pode ser de qualquer tamanho. A interface de entrada de dados da imagem pode ser vista na Figura 9.

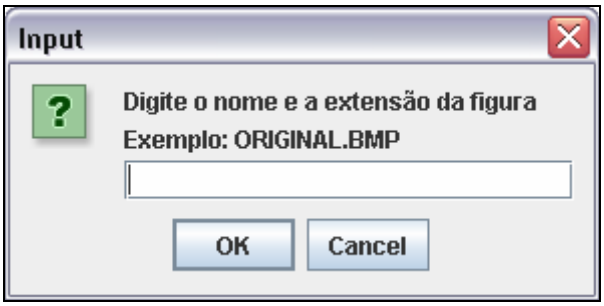


Figura 9 - Entrada dos Dados da Imagem

Após o carregamento da imagem é possível executar várias operações as quais serão descritas a seguir. Cada botão está associado a uma ação que é responsável pela chamada dos métodos criados na classe *GreyImage* bastando apenas a passagem dos parâmetros variáveis de alguns deles.

A quantização, também conhecida como agrupamento do histograma, faz com que cada pixel assuma um valor inteiro não negativo de intensidade luminosa (nível de cinza) reduzindo a quantidade de níveis de cinza diferentes da imagem e, consequentemente, removendo gradações indesejáveis.

Esta operação promove um efeito de aumento de contraste como pode ser observado na Figura 10.

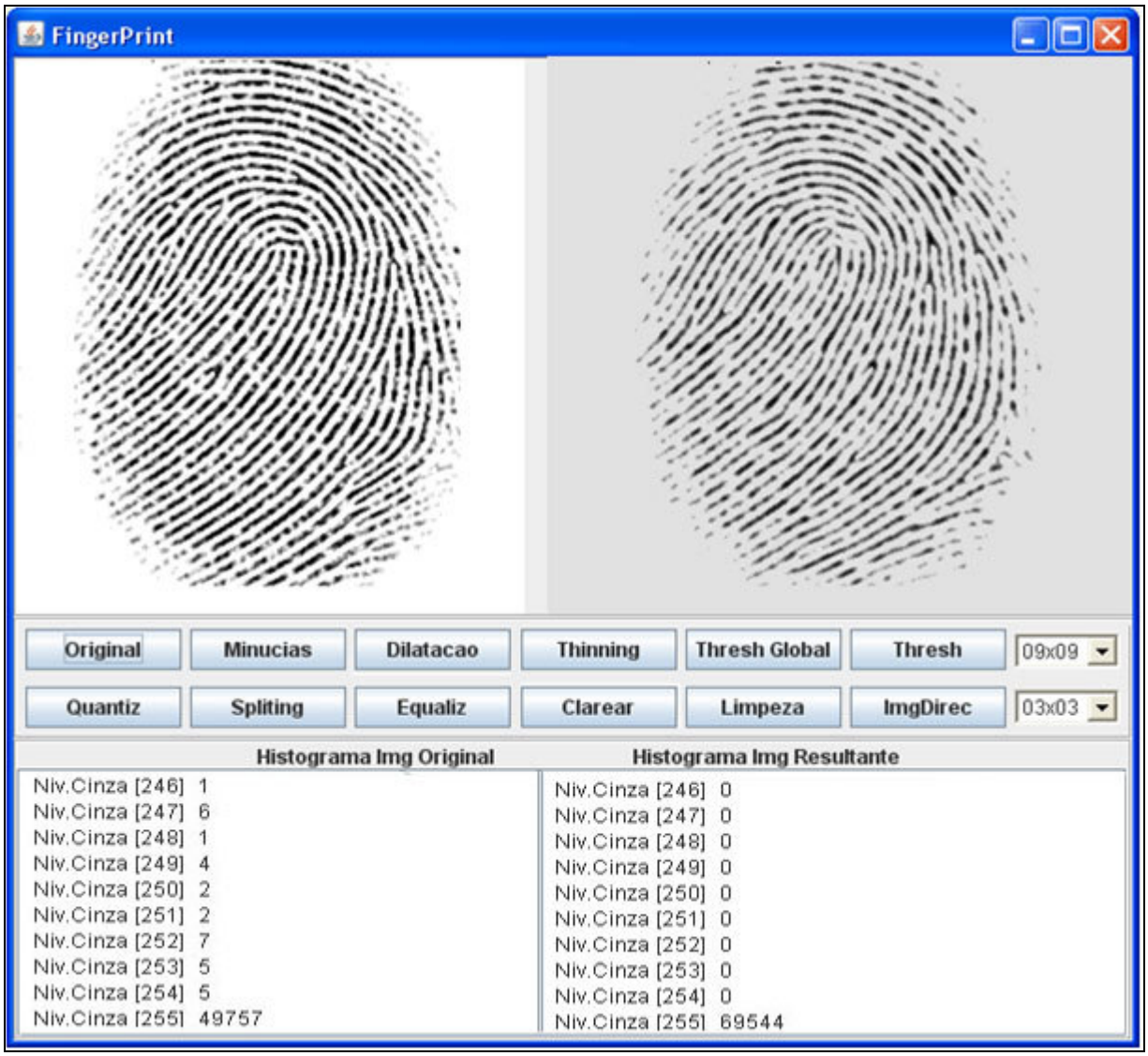


Figura 10 - Exemplo de aplicação da técnica de Quantização no sistema Fingerprint

A técnica de Thresholding faz com que a imagem assuma níveis de cinza restritos ao valor máximo ou mínimo, o que dá o efeito de realce bastante desejado neste caso em que a imagem trata-se de uma impressão digital.

O resultado desta técnica pode ser constatado na Figura 11:

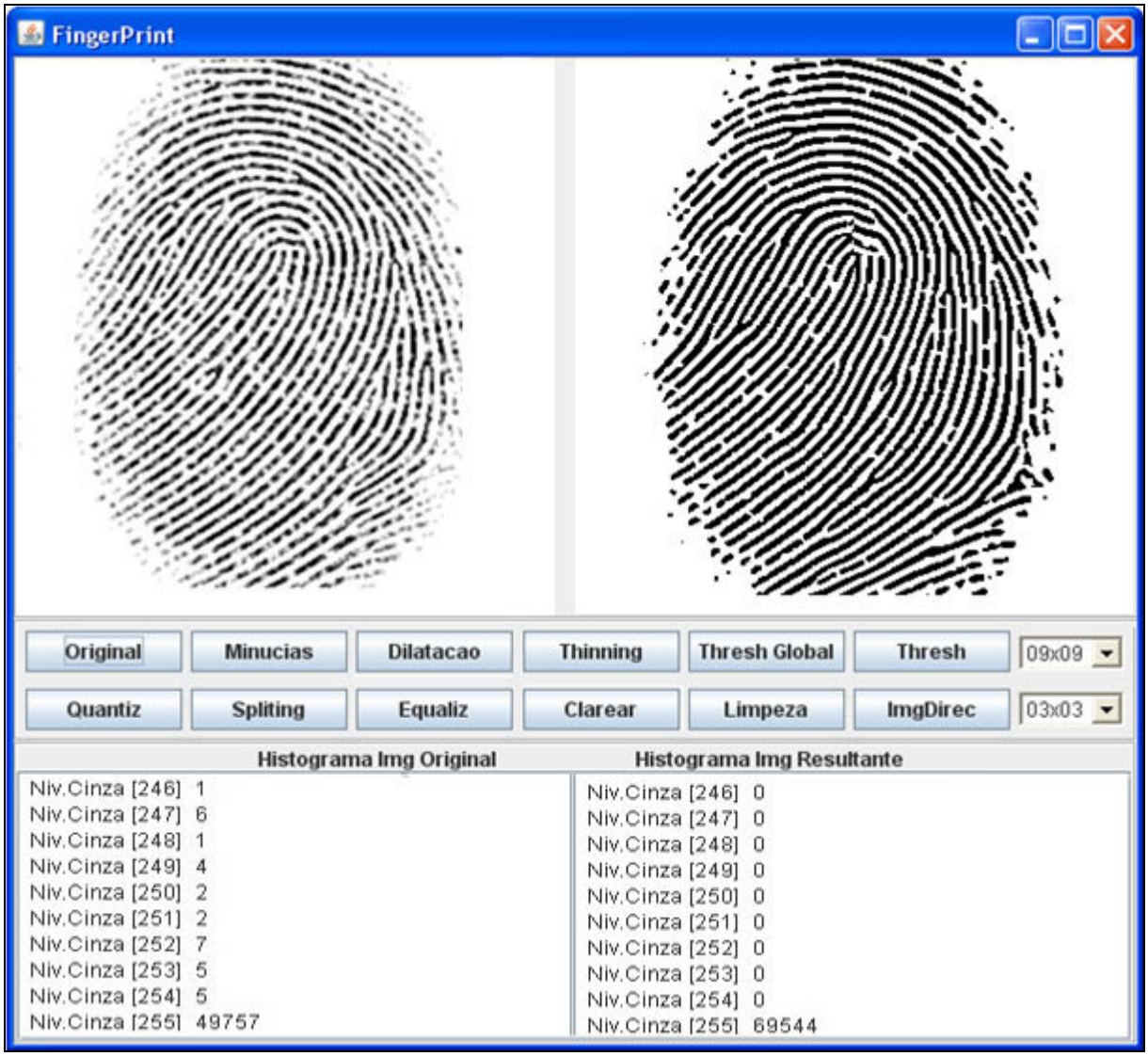


Figura 11 - Exemplo de aplicação da técnica de Thresholding no sistema Fingerprint

A equalização parte do princípio que o contraste de uma imagem seria otimizado se todos os 256 possíveis níveis de intensidade fossem igualmente utilizados ou, em outras palavras, a técnica de equalização também conhecida como linearização de histograma, explora a obtenção de uma imagem com histograma uniforme, através do espalhamento da distribuição dos níveis de cinza, isso nem sempre é possível devido à natureza discreta dos dados digitais de uma imagem. Esta operação é muito poderosa, conseguindo muitas vezes recuperar imagens consideradas perdidas, o resultado pode ser visto na Figura 12.

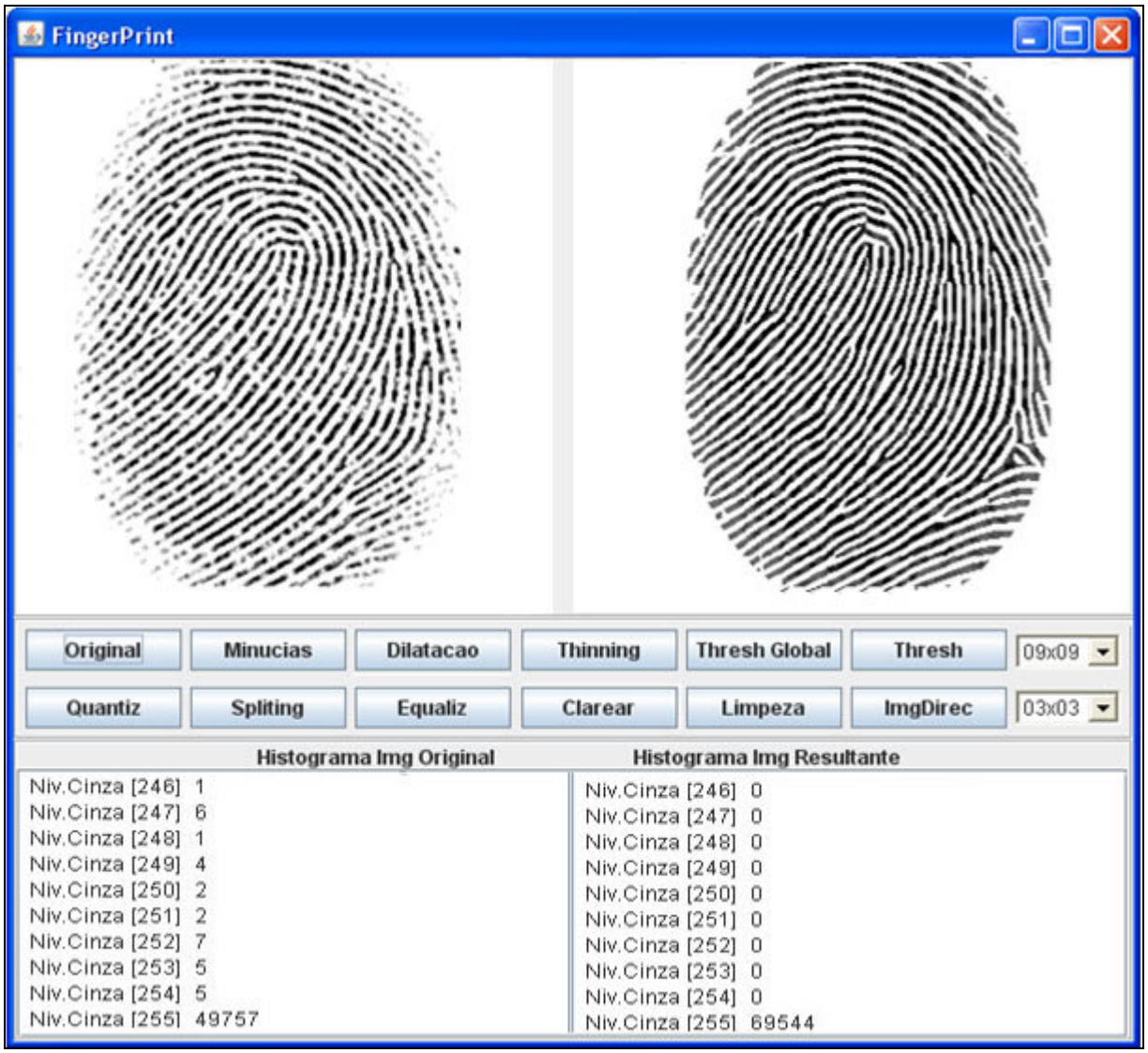


Figura 12 - Exemplo de aplicação da técnica de Equalização no sistema Fingerprint

A técnica de Thinning, também conhecida como esqueletização, consegue sintetizar a imagem em suas linhas principais, neste caso a técnica promove o afinamento das linhas da impressão digital, sendo de suma importância para a extração das minúcias.

A esqueletização é na maioria das vezes o primeiro passo quando se quer extrair características de um objeto em uma imagem, esta técnica deve representar o objeto com um número reduzido de pixels, porém sem perder as informações originais tais como orientação, tamanho e posição, o resultado pode ser visto na Figura 13.

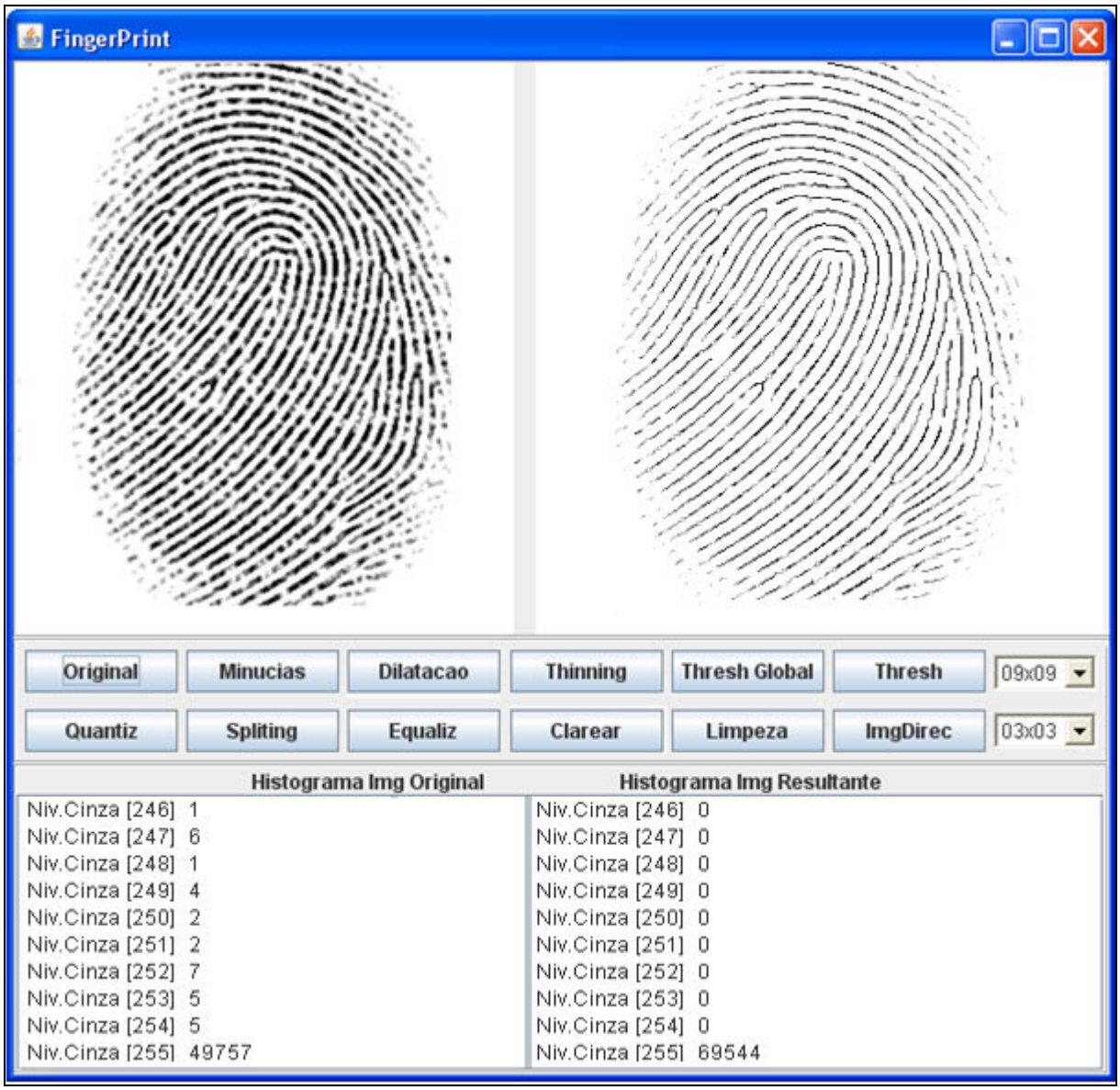


Figura 13 - Exemplo de aplicação da técnica de Thinning no sistema Fingerprint

A partir das técnicas apresentadas torna-se possível a extração de algumas características da imagem que constituem uma impressão digital.

Uma destas características é o mapa direcional da imagem que pode ser calculado por métodos pré-definidos. Neste caso o método proposto por Karu (KARU, 1996) define que o pixel pode assumir oito posições diferentes, de acordo com a máscara selecionada.

Após a aplicação do cálculo, os resultados são submetidos a um segundo cálculo estatístico para suavização e retorna uma imagem como descrita na Figura 14.

Figura 14 - Exemplo de aplicação da técnica de Mapa Direcional no sistema Fingerprint

Dentre as características que podem ser extraídas da imagem, a que se faz mais interessante no contexto deste estudo são as minúcias, as quais serão determinantes para a classificação das impressões digitais, e consequentemente, no armazenamento e recuperação das mesmas.

Na extração das minúcias as técnicas descritas são aplicadas e em seguida são calculados e identificados os tipos de minúcia, destacando as bifurcações e as terminações como segue na Figura 15.

Figura 15 - Extração de Minúcias

Além das técnicas aqui descritas o sistema oferece filtros como Dilatação, Clareamento, Escurecimento que são necessários de acordo com o estado em que a imagem foi capturada.

3.3 - Visão geral da implementação

As implementações realizadas neste sistema tiveram como objetivo obter as melhores formas de armazenamento e recuperação de impressões digitais baseado em medida de desempenho como demonstrado nos testes realizados.

A realização deste trabalho está dividida em três pontos fundamentais os quais foram estudados detalhadamente para que objetivos esperados fossem alcançados.

Em princípio foi estudado como deve ser feita a aquisição das impressões digitais definindo um leitor biométrico a ser usado que proporcione uma boa relação custo benefício e qualidade nas imagens capturadas. A definição do leitor adquirido foi baseada em fatores como resolução de imagem, quantidade de cores, tamanho da imagem gerada, além do custo.

Em seguida foi feita a análise das melhores formas de recuperar imagens de um banco de dados com desempenho adequado. Isso pôde ser alcançado definindo uma forma de armazenamento que favoreça a recuperação rápida e que seja eficiente no reconhecimento de impressões digitais mesmo em grandes bases de dados.

Outro ponto realizado é a adaptação do sistema atual para manipular o banco de dados escolhido, possibilitando a inserção das informações extraídas das imagens e a consulta quando solicitada a comparação de minúcias.

A partir da definição dos pontos citados foi possível realizar testes de desempenho na inserção e recuperação de dados, variando o padrão de imagens a serem comparadas.

Para cada variação de classe foi avaliado o desempenho do sistema, verificando a porcentagem de imagens reconhecidas corretamente e o desempenho obtido.

Admitindo uma ordem cronológica, as seguintes fases foram conduzidas, Figura 16.

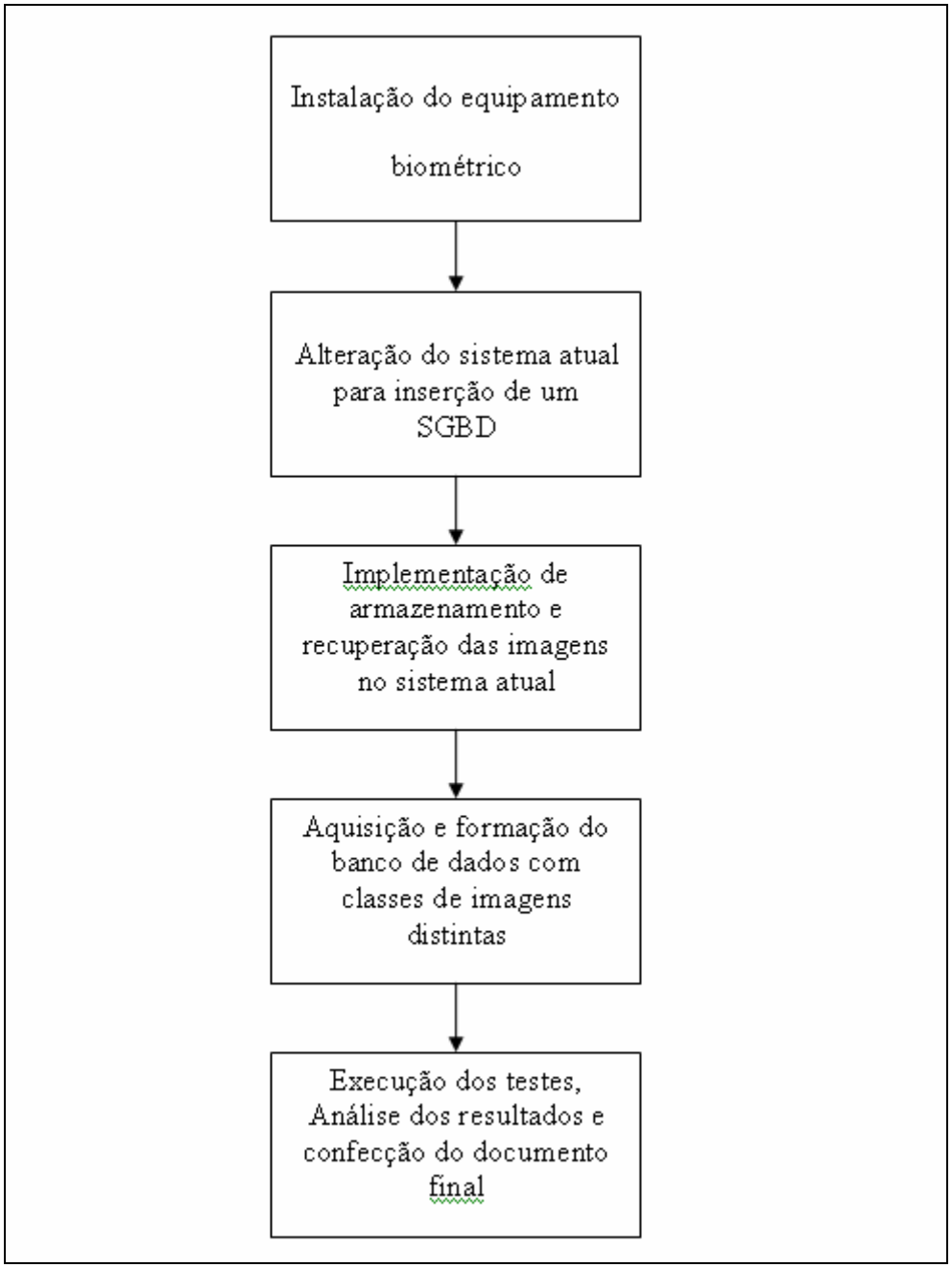


Figura 16 - Fluxo de implementação

A instalação do equipamento de aquisição foi realizada a partir da interface oferecida pelo fabricante, respeitando os requisitos mínimos e exigências do aparelho. A programação necessária para manipular o leitor de forma a se adequar às necessidades do software de reconhecimento foi realizada com sucesso.

A alteração do sistema atual de reconhecimento foi realizada utilizando-se do código disponível em Llinguagem Java agregando as melhorias necessárias para que seja possível a utilização das técnicas escolhidas, assim como, a integração com o banco de dados e com o equipamento biométrico.

A aquisição de imagens foi dada através das bases de dados disponíveis via Internet, estas bases possuem um grande número de impressões e com variações estratégicas para que seja feito o reconhecimento de vários tipos de impressão digital e com isso realizar testes mais efetivos no sistema.

3.4 - Instalação do Equipamento

Neste tópico serão descritas as etapas de instalação, configuração e manipulação de um equipamento leitor de impressão digital destacando os pontos de dificuldade e as particularidades deste tipo de aparelho.

Como já foi descrito anteriormente o leitor de impressão digital escolhido é aquele que apresenta melhor relação custo benefício dentre os existentes do mercado e que possibilite a captura de imagens com qualidade suficiente para a extração de minúcias e a classificação posterior.

Alguns dos leitores biométricos pesquisados continham software de apoio para o desenvolvimento de aplicaçãoes, como é o caso do SecuGen Hamster, neste tipo de leitor o

tratamento do hardware é facilitado pois as bibliotecas de manipulação do leitor são disponibilizadas com o software.

Geralmente os aparelhos que acompanham bibliotecas para o desenvolvimento tem custo maior e são desenvolvidos especificamente para o uso em sistemas de reconhecimento de impressão digital, muitos deles implementam soluções em hardware para melhorar a qualidade de captura da imagem, estas melhorias auxiliam no reconhecimento da impressão digital visto que o processamento necessário para retirada de ruídos ou acidentes na captura é minimizado.

Baseado nestes critérios o leitor comercializado pela Microsoft identificado como Fingerprint Reader atendeu às condições iniciais.

Com a aquisição do aparelho foi interessante observar que mesmo este leitor contendo a marca da Mircrosoft, este é originalmente fabricado pela Digital Persona, uma empresa de grande porte em tecnologia biométrica e que também fabrica leitores com marca própria.

O leitor veio acompanhado do manual de instruções e do CD de instalação. As especificações técnicas são descritas na Tabela 2.

Tabela 2 - Especificações Técnicas do Leitor

Leitor Microsoft Fingerprint Reader	Modelo: 1033	Fabricante: Digital Persona
Alimentação: 5V 500mA	Conexão:USB	Resolução: 512 dpi
Cores: 256 tons de cinza	Lote: 0724	Imagem: 355x390 pixels

O software que acompanha o aparelho é util apenas para gerenciamento de senhas dos sistemas operacionais da Microsoft o que mostra que este equipamento não é destinado especificamente para a captura de imagens de impressão digital em sistemas de reconhecimento.

Na Figura 17 é mostrado o cadastro de impressões digitais para gerenciamento de senhas no sistema operacional Windows.

Figura 17 - Software fornecido com o leitor

Neste ponto o comportamento do software se tornou uma dificuldade, visto que para o tipo de aplicação que este trabalho tem como objetivo, o ideal é que se tenha um controle total do equipamento de aquisição.

Através de pesquisa foi possível descobrir que existem drivers genéricos para este fim e que contemplam o equipamento em questão, A biblioteca *libusb-win32* é um software licenciado sob a GPL e que tem seu código fonte disponível para download em diversas linguagens incluindo a linguagem Java, utilizada neste projeto.

A biblioteca descrita pode ser encontrada no site http://libusb-win32.sourceforge.net/, onde também está disponível a documentação e especificações técnicas.

Na Figura 18 é mostrado o software de instalação da biblioteca libusb-win32 para leitores Microsoft, Digital Persona, SecuGen e outros.

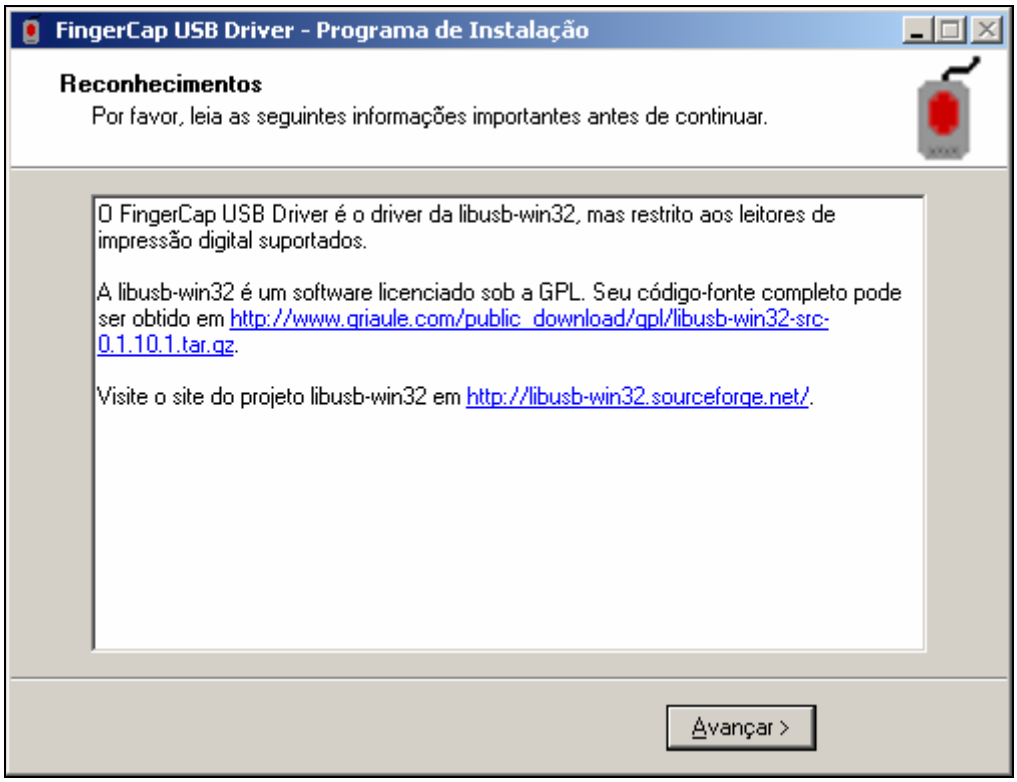


Figura 18 - Instalação da biblioteca libusb-win32

A mensagem que é exibida (Figura 19) comprova a limitação dos drivers originais oferecidos pela Microsoft, nesta etapa estes drivers são substituídos por outros que fornecem maior autonomia sobre o aparelho.

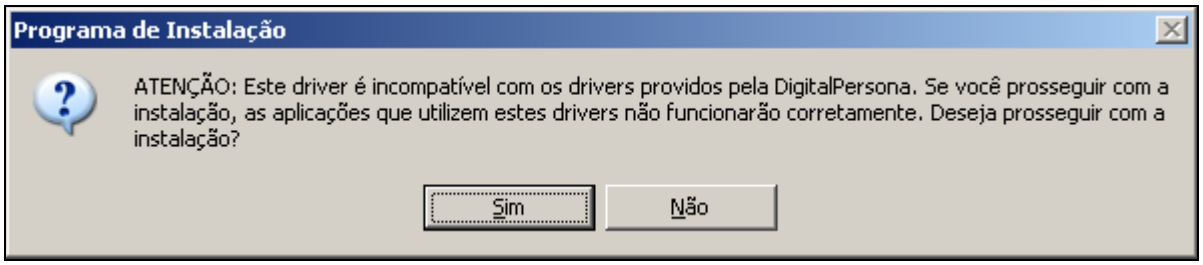


Figura 19 - Substituição dos drivers originais

Com a instalação dos novos drivers, são adicionadas algumas DLLs que promovem controles do leitor como ativação, desativação, detecção, captura, etc.

Após a instalação dos drivers foi necessário o uso de uma classe específica que interage com as DLLs da *libusb-win32* em baixo nível, esta classe pode ser obtida no site www.griaule.com/downloads.

Com esta classe foi possível elaborar um método de inicialização do leitor como segue na Figura 20.

```
01.  public void init(){
02.     try {
03.        grFinger = new GrFinger();
04.        //Inicializa as DLLs de leitores
05.        grFinger.initializeCapture(this);
06.     } catch (GrErrorException e) {
07.        //Escreve Log do Erro
08.        e.printStackTrace();
09.     }
10.  }
```

Figura 20 - Método de inicialização do leitor

Depois de compiladas as classes e executado o método *init* o leitor não foi inicializado corretamente, deste modo todos os outros métodos dependentes da ativação do leitor também não puderam ser testados. Esta situação foi mais uma dificuldade encontrada, pois neste caso a utilização do equipamento é totalmente dependente de drivers de terceiros.

Através de uma pesquisa um pouco mais aprofundada foi possível descobrir que o problema ocorria devido ao lote de fabricação específico do leitor adquirido que neste caso continha um *firmware* ainda não contemplado pela biblioteca *libusb-win32*.

Para resolver este problema foi necessário solicitar uma atualização específica da biblioteca, esta atualização não é disponibilizada para download, sendo necessário a requisição por e-mail para que seja enviado um novo instalador, a justificativa é que a versão que contempla o leitor PID 0x00CA lote 0724 (o mesmo descrito anteriormente) está em base de testes.

Executando o novo instalador, as DLLs necessárias foram substituídas e logo após a reconexão do aparelho o sistema operacional reconheceu o mesmo com sucesso e com seus devidos parâmetros.

Partindo desta etapa, a compilação das classes foi realizada novamente e o método de inicialização foi chamado para testar a ativação do leitor. Com as novas alterações o leitor inicializou corretamente e foi possível o progresso para um novo passo que é a criação dos métodos para detectar a presença do leitor.

O método *onPlug* é invocado toda vez que o leitor é conectado na porta USB. Nesta ocorrência o método emite uma mensagem ao usuário, e em seguida, coloca o sensor no status de espera, ficando sensível ao toque do dedo para ser iniciado a leitura.

O código fonte do método onPlug ser visto na Figura 21.

```
01.  public void onPlug(String idSensor) {
02.
03.     try {
04.        //Testa se o leitor foi plugado
05.        if(plug>0){
06.           writeLog("Leitor: "+idSensor+". Plugado");
07.        }
08.        //Inicializa o Leitor para captura
09.        grFinger.startCapture(idSensor,this,this);
10.        plug = plug+1;
11.     } catch (GrErrorException e) {
12.        //Escreve Log do Erro
13.        writeLog(e.getMessage());
14.     }
15.  }
```

Figura 21 - Método onPlug

O método *onUnPlug* é chamado toda vez que o leitor é desconectado da porta USB, nesta ocorrência o método emite uma mensagem ao usuário e em seguida interrompe a espera por imagem.

O código fonte do método onUnPlug pode ser visto na Figura 22

```
01.     public void onUnplug(String idSensor) {
02.        //Testa se o Leitor foi desplugado
03.        writeLog("Leitor: "+idSensor+". Desplugado.");
04.        try {
05.           //Desativa o leitor para captura
06.           grFinger.stopCapture(idSensor);
07.        } catch (GrErrorException e) {
08.           //Escreve Log de erro
09.           writeLog(e.getMessage());
10.        }
11.     }
12.
```

Figura 22 - Método onUnPlug

Baseado nestes métodos é possível realizar a captura da imagem em si e fazer com que a mesma seja mostrada na interface gráfica existente.

O método *onImage* detecta a presença de um corpo na área de leitura, captura a imagem e salva a mesma em disco.

Em seguida é emitida a mensagem ao usuário indicando que a imagem foi capturada e o método showImage é chamado para exibir a impressão digital na interface gráfica.

O código fonte do método *onImage* pode ser visto na Figura 23.

```
01. public void onImage(String idSensor, FingerprintImage fingerprint) {
02.    try {
03.       // Save image.
04.       fingerprint.saveToFile("cap.bmp",FingerprintImage.GRCAP_IMAGE_FORMAT_BMP);
05.    } catch (GrErrorException e) {
06.       //write error to log
07.       this.writeLog(e.getMessage());
08.    }
09.
10.    // Escreve Log de Imagem Capturada
11.    writeLog("Sensor: "+idSensor+": Imagem Capturada.");
12.    try {
13.       //Adiciona a Impressao no Painel
14.       showImage(fingerprint.newImageProducer());
15.
16.    } catch (GrErrorException e) {
17.       //Escreve Log do Erro
18.       writeLog(e.getMessage());
19.    }
20. }
```

Figura 23 - Método onImage

O método showImage é responsável por pegar a imagem salva e adiciona-la ao container de exibição destino.

O código fonte do método showImage pode ser visto na Figura 24.

```
01.  public void showImage(ImageProducer producer) {
02.        String ImgCapturada = "captured.bmp";
03.        try {
04.           GreyImage gi = GreyImage.createImage(ImgCapturada);
05.           gi = GreyImage.createImage(ImgCapturada);
06.           this.c2.setImage(gi);
07.        } catch (IOException e) {
08.           //Escreve Log do Erro
09.           e.printStackTrace();
10.        }
11.        c2.repaint();
12.     }
13.
```

Figura 24 - Método showImage

Com os métodos descritos foi possível capturar e analisar a qualidade da imagem obtida pelo leitor, e verificar que esta era satisfatória para extração das minúcias.

O resultado da imagem capturada pode ser visto na Figura 25.

Figura 25 - Captura da imagem com leitor

Os resultados obtidos com o leitor biométrico foram satisfatórios para a aplicação em questão, pois a imagem capturada obteve qualidade suficiente para a extração de minúcias, salvo alguns casos que foram necessários a utilização de mais filtros até se obter uma imagem útil.

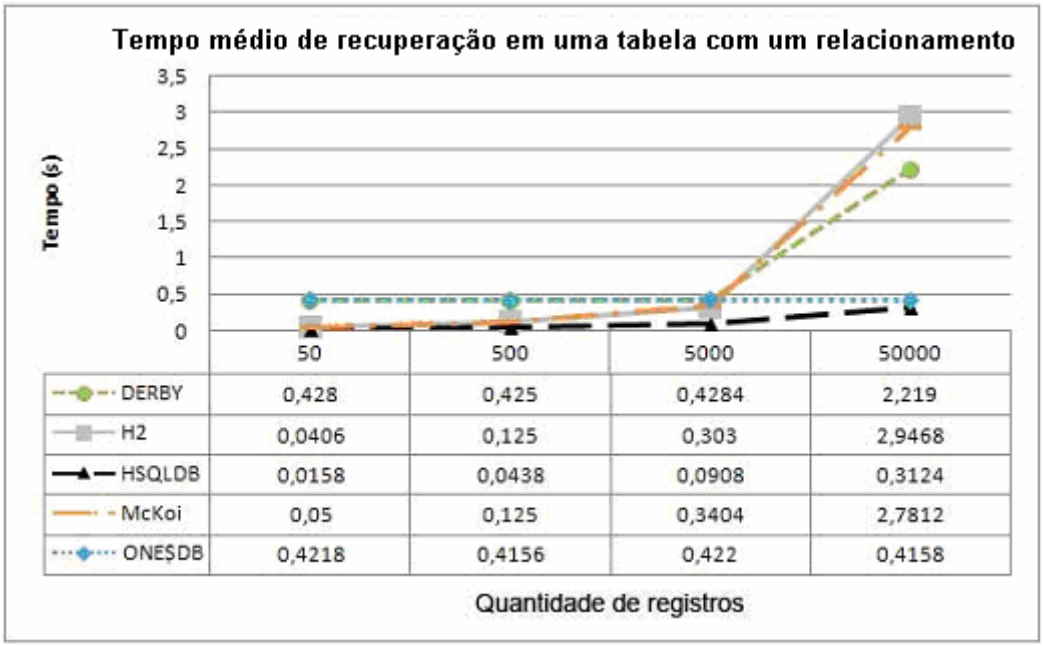
A combinação de filtros para melhorar a qualidade da imagem é vista na Figura 26.

Figura 26 - Aplicação de filtros para melhoria da qualidade

3.5 - Modelagem do Banco de Dados

O Sistema Gerenciador de Banco de Dados que foi utilizado para armazenamento das impressões digitais é o H2 Database Engine. A escolha está fundamentada em vários fatores, dentre eles o fato de ser um banco gratuíto, ter suporte nativo da linguagem Java, a robustez adequada ao sistema aqui estudado, a implementação de integridade referencial, e a performance.

De acordo com Sono (2007), o banco de dados H2 tem desempenho elevado se comparado a outros bancos que possuem as mesmas características, na Figura 27 é mostrado a comparação do tempo médio de recuperação em uma tabela com um relacionamento.



	50	500	5000	50000
DERBY	0,428	0,425	0,4284	2,219
H2	0,0406	0,125	0,303	2,9468
HSQLDB	0,0158	0,0438	0,0908	0,3124
McKoi	0,05	0,125	0,3404	2,7812
ONE$DB	0,4218	0,4156	0,422	0,4158

Figura 27 - Comparação de performance banco de dados

A performance do banco de dados é bastante importante para o objetivo buscado neste trabalho, visto que no sistema existente, cada impressão digital armazenada no banco de dados pode corresponder a uma quantidade alta de registros. Isso se complica ainda mais quando se faz à recuperação de informação, neste caso a quantidade de registros é multiplicada pela quantidade de impressões de uma mesma classe comparada.

A robustez é outro fator a ser levado em conta, pois em longo prazo a tendência de um sistema comum é aumentar a quantidade de impressões digitais armazenadas e em conseqüência, a quantidade de registros cresce muito, o que pode levar um banco de dados não preparado para grandes volumes à perda de informação ou estouro de índices.

O armazenamento das impressões digitais se dá através de seus pontos característicos, ou seja, as minúcias. Neste contexto as informações que devem ser inseridas no banco são: a posição da minúcia (linha e coluna) e o tipo desta minúcia (bifurcada, final, etc). Neste caso foi armazenada também a identificação do indivíduo proprietário da impressão, a fim de tornar mais didática à etapa de reconhecimento da impressão, na qual será mostrado o nome das pessoas com a impressão mais próxima da imagem dada.

O diagrama entidade relacionamento correspondente aos requisitos citados acima pode ser visto na Figura 28.

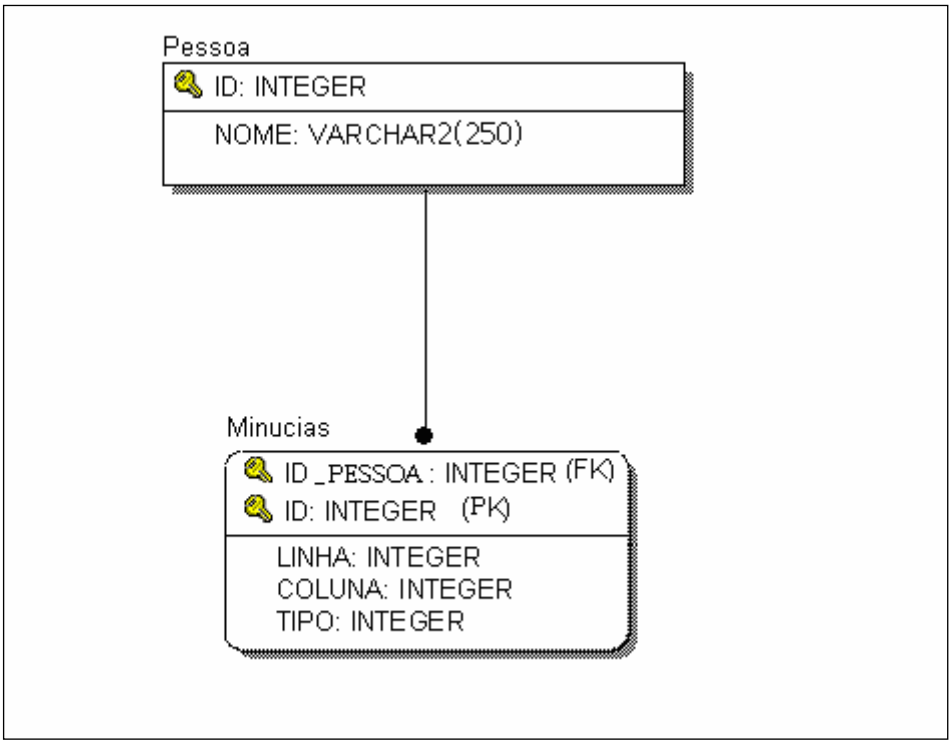


Figura 28 - Modelo entidade relacionamento

A entidade Minúcias é entidade fraca da entidade Pessoa. A cardinalidade do relacionamento neste caso é de 1 para N, visto que uma pessoa pode ter N minúcias a serem armazenadas.

O script de criação do modelo entidade relacionamento mostrado acima para o banco de dados H2 pode ser visto na Figura 29.

```
01. CREATE TABLE PESSOA(
02.    ID       INT             NOT NULL   AUTO INCREMENT    PRIMARY KEY ,
03.    NOME     VARCHAR2(255)   NOT NULL   UNIQUE
04. );
05. CREATE TABLE MINUCIAS(
06.    ID          INT NOT NULL   AUTO INCREMENT    PRIMARY KEY ,
07.    ID_PESSOA   INT NOT NULL
08.    LINHA       INT NOT NULL,
09.    COLUNA      INT NOT NULL,
10.    TIPO        INT NOT NULL
11. );
12.
13. ALTER TABLE MINUCIAS ADD FOREIGN KEY(ID_PESSOA) REFERENCES PESSOA(ID);
```

Figura 29 - Script para o banco H2

A implementação dos métodos de inserção de pessoa e das minúcias estão descritos no apêndice deste trabalho. Para a inserção de um registro, basicamente é recebido o nome a ser inserido como parâmetro. Em seguida é realizada a conexão com o Banco de Dados H2 e a inserção é executada, guardando o atributo *id* de pessoa para que sejam inseridos as minúcias com este *id* posteriormente.

Na inserção de minúcias, o *id* guardado é utilizado para inserir todas as minúcias da pessoa com seu respectivo código, o método recebe como parâmetro a linha, a coluna e o tipo da minúcia a ser inserida, neste estágio a conexão já foi estabelecida, portanto a inserção é realizada.

Este processo pode ser demorado dependendo da quantidade de minúcias que foram extraídas, para melhoria de desempenho as minúcias são consideradas em um determinado raio passado como parâmetro ao método que identifica tais características.

3.6 - Implementação do armazenamento e recuperação

Para implementar as funcionalidades de armazenamento e recuperação de impressões digitais foi necessário efetuar algumas modificações na interface gráfica, de forma a abrigar os novos botões responsáveis pela chamada dos métodos construídos.

Dentro deste contexto os botões incluídos na interface gráfica são: armazenar, reconhecer e limpar base de dados, cujas funcionalidades serão descritas a seguir.

Na Figura 30 pode-se observar a nova interface gráfica com a disposição dos componentes modificada.

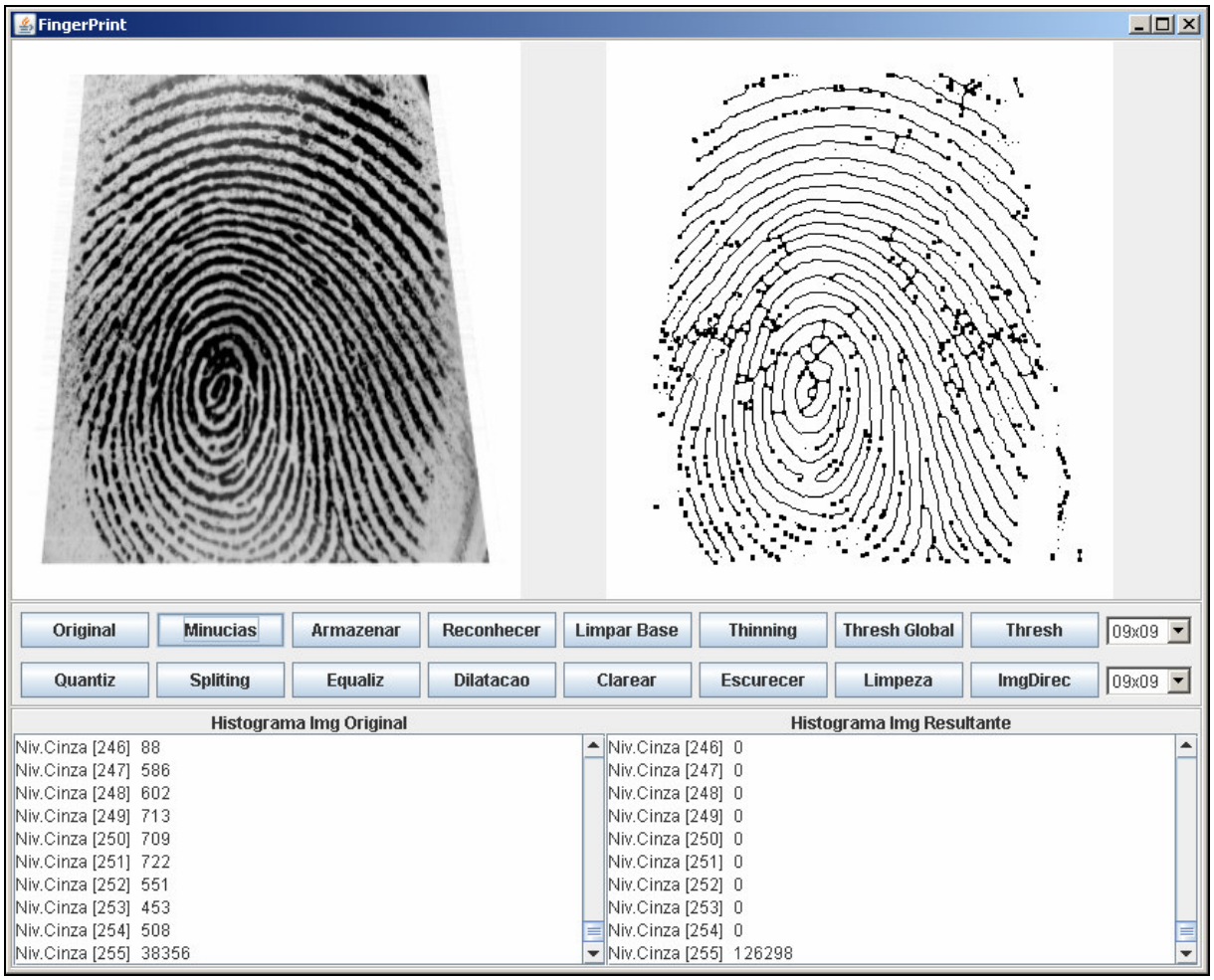


Figura 30 - Nova interface gráfica

Os métodos que são chamados no armazenamento e recuperação são definidos na classe *Database,* cujo diagrama pode ser visto na Figura 31.

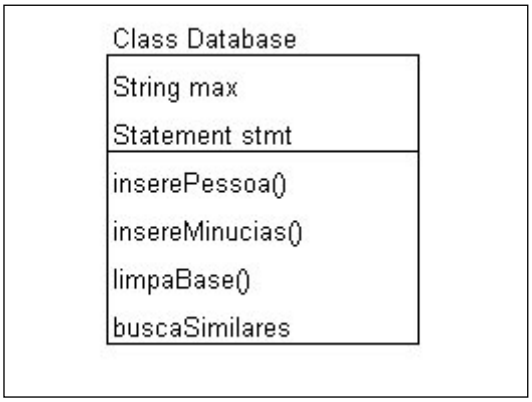


Figura 31 - Classe Database

A funcionalidade “Armazenar” dispara um evento que é capturado pelo método *actionPerformed*. Em um primeiro estágio são aplicados os filtros necessários à imagem capturada para que seja possível a extração das minúcias.

Em seguida o método de extração de minúcias é chamado e as características são armazenadas em um vetor de minúcias no formato linha, coluna e tipo da minúcia.

No próximo passo é solicitada a identificação da pessoa que será associada às minúcias extraídas, e o método de inserção das minúcias é chamado. Este método percorre o vetor gerado pela extração de minúcias, chamando em cada interação o método de inserção da classe *Database* o qual irá efetuar a conexão e inclusão do registro no banco de dados em si.

A funcionalidade “Limpar Base” apaga todas as impressões digitais armazenadas no banco. Para isto é chamado o método *limpaBase* da classe *Database* o qual irá efetuar a exclusão de todos os registros existentes.

A funcionalidade “Reconhecer” em um primeiro estágio aplica os filtros necessários à imagem capturada para que seja possível a extração das minúcias.

Em seguida o método de extração de minúcias é chamado e as características são armazenadas em um vetor, este vetor é passado por parâmetro para o método *buscaSimilares* da classe Database.

O método *buscaSimilares* seleciona no banco todas as pessoas existentes, e para cada pessoa é efetuado um outro *select* que retorna as minúcias da mesma, este resultado é armazenado em um vetor e em seguida é chamado o método *getMatchs* da classe *StoredFinger* passando como parâmetro a largura e a altura da imagem (recebidas como parâmetro neste método), a imagem de entrada (também recebida como parâmetro neste método no formato de vetor) e por fim os dados obtidos do banco também no formato de vetor.

O método *getMatchs* percorre o raio e a área da imagem passado por parâmetro. A partir daí é chamado o método *match* passando como parâmetro as duas imagens recebidas. O match devolve um vetor com as minúcias correspondentes nas duas imagens e o tamanho deste vetor é o retorno desejado que represente a quantidade de minúcias que foram consideradas como *match*.

Nesta etapa é armazenado o nome da pessoa e o retorno do método *getMatchs* em um vetor. Após o término da busca das imagens do banco, este vetor é ordenado e uma mensagem é exibida ao usuário mostrando as imagens que obtiveram maior quantidade de minúcias correspondentes, ou seja , as imagens com maior similaridade. O reconhecimento pode ser caracterizado pela imagem com maior similaridade encontrada.

CAPÍTULO 4 - RESULTADOS E DISCUSSÕES

Este capítulo apresenta e discute os testes e resultados obtidos, com a implementação do sistema proposto, assim como o desempenho do mesmo.

4.1 - Conjunto de testes criados

Para os testes deste sistema foram consideradas quatro classes de imagens distintas, no intuito de destacar em qual delas o sistema se comporta com maior precisão e desempenho.

A primeira classe de imagens foi obtida através de um software chamado *Sfinge Biolab* disponibilizado pela universidade de Bologna na Itália. O software é capaz de gerar imagens de impressões digitais computadorizadas baseado em alguns parâmetros de entrada, neste caso foram geradas impressões aleatórias de 8 bits para que estejam enquadradas nas mesma condições das outras classes. O resultado são impressões com ótima qualidade e que só poderiam ser obtidas com um leitor de precisão muito elevada.

A segunda classe de imagens é de origem do banco de impressões digitais NIST Special Database, citado anteriormente. Neste banco as imagens foram obtidas por digitalização de impressões digitais tintadas em papel o que torna a qualidade um fator crítico na imagem.

A terceira classe de imagens é de origem de um dos bancos de impressões da FVC, citado anteriormente, e dentre as categorias existentes foi escolhida a que contém imagens obtidas por leitores de impressão digital cujo a qualidade de captura é mais elevada do que a oferecida pelo leitor utilizado neste trabalho.

A quarta classe de imagens foi obtida através da captura de impressões digitais utilizando o leitor da Microsoft.

Na Figura 32 são mostradas as classes de impressão digital obtidas de diferentes formas, destacando a qualidade atingida em cada uma delas.

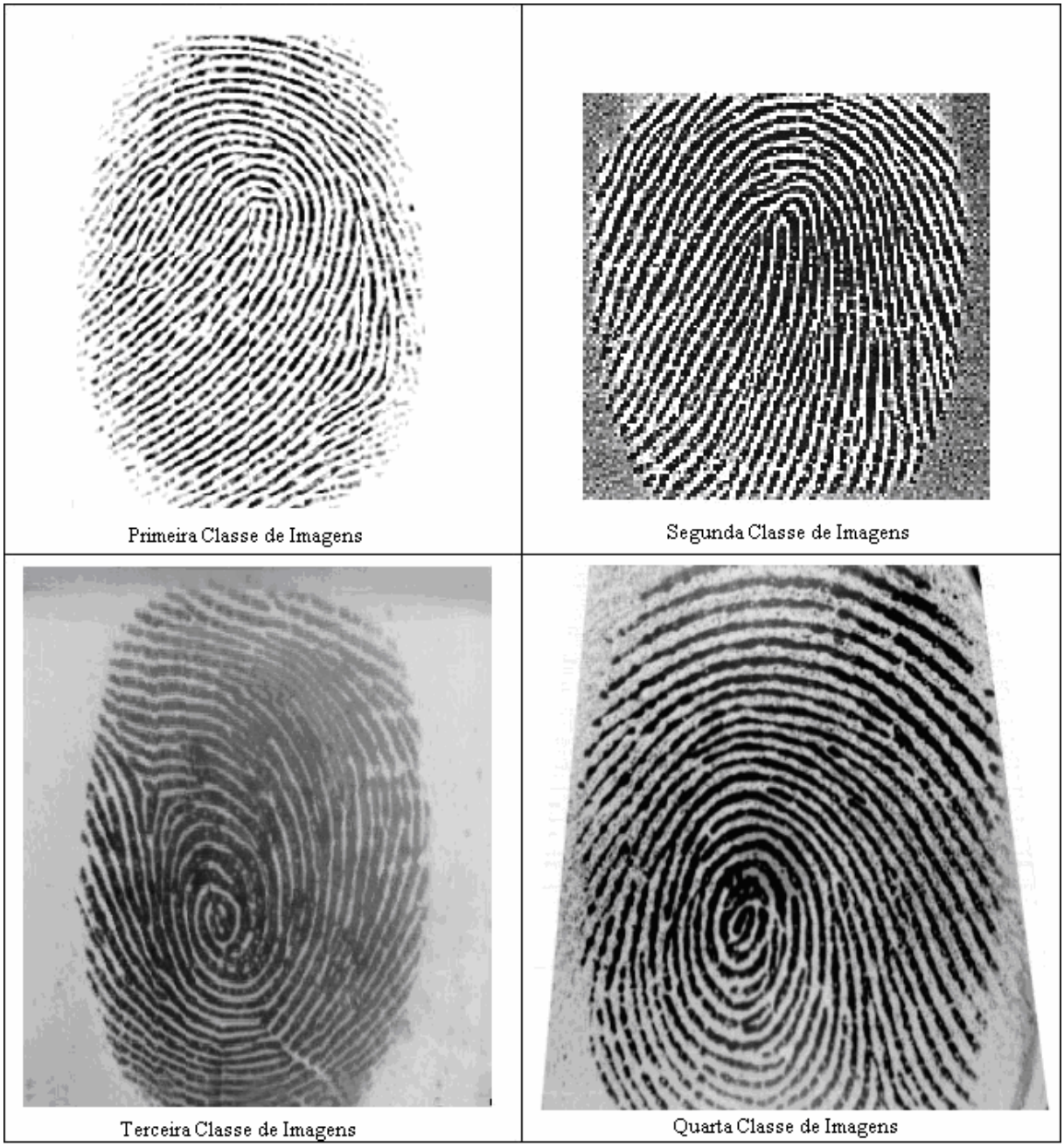


Figura 32 - Classes de imagens obtidas

A qualidade da imagem obtida influencia consideravelmente na precisão do processo de extração das minúcias, visto que quanto mais nítida a imagem se encontra, maior a possibilidade de se destacar os verdadeiros pontos característicos.

Neste contexto foi possível observar que as imagens com menor qualidade resultaram em um número maior de minúcias destacadas e, conseqüentemente, em uma quantidade alta de registros a ser inserida no banco de dados o que faz com que tempo de armazenamento da impressão se torne maioro.

Para testar o tempo de armazenamento foram inseridas vinte imagens de cada classe e obtido o tempo médio de inserção para cada impressão digital, o resultado pode ser visto na Figura 33.

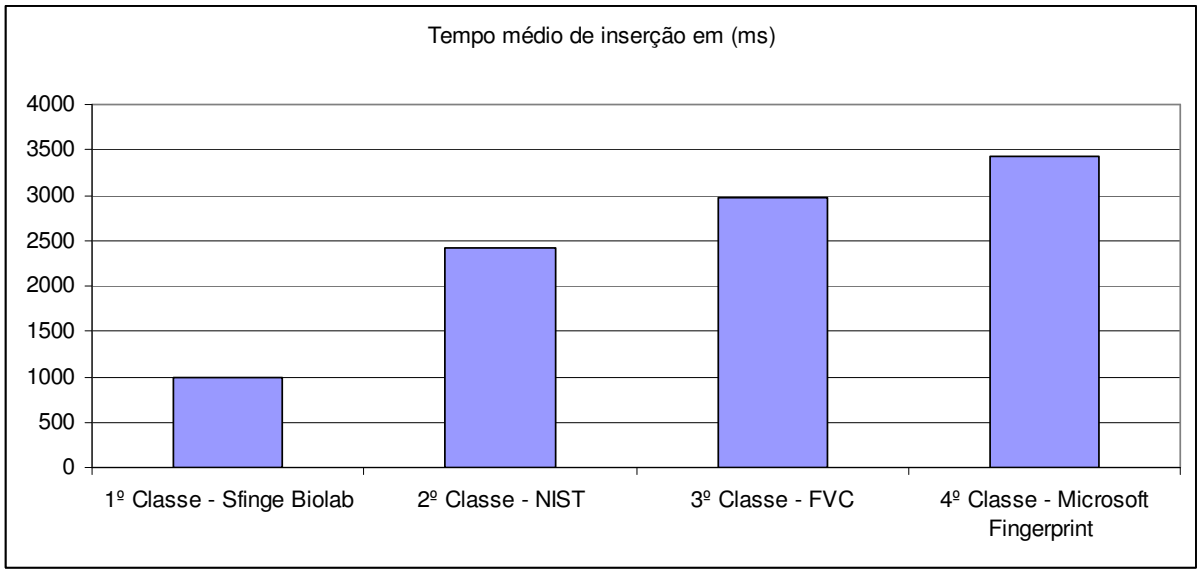


Figura 33 - Tempo médio de inserção da impressão digital

Nota-se que a primeira classe que continha imagens de ótima qualidade obteve um tempo de inserção muito baixo, porém em termos práticos o tempo de inserção em todas as classes pode ser considerado satisfatório para uma aplicação real, incluindo a classe que foi obtida através do leitor de impressão digital adquirido para este trabalho.

Os próximos testes a serem executados dizem respeito ao reconhecimento de uma impressão digital em cada classse de imagens, para isto foi testado o tempo de retorno da impressão correspondente aumentando gradativamente a quantidade de imagens a ser comparada.

Para realizar os testes de reconhecimento, foram submetidas dez impressões digitais de cada classe de imagens. A extração das minúcias foi executada em cada impressão digital e em seguida armazenada estas características no Banco de Dados.

Em um primeiro teste foram selecionados todos os registros existentes no Banco de Dados, este mesmo processo será executado sempre que é solicitado o reconhecimento de uma impressão digital.

O objetivo deste teste foi medir somente o tempo de retorno do banco de dados ao selecionar todos os registros existentes para um determinado padrão de imagem adquirida. Este teste foi executado separadamente para cada classe de imagens e foi repetido aumentando a quantidade de imagens submetidas ao processo de extração e armazenamento de minúcias, o resultado pode ser visto na Figura 34.

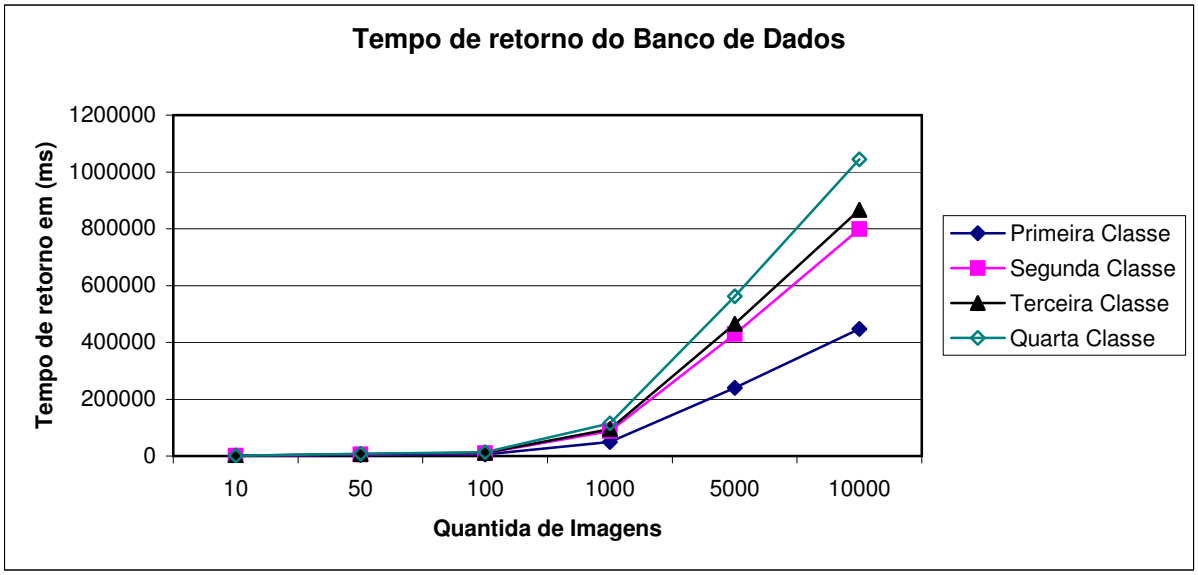


Figura 34 - Tempo de resposta do Banco de Dados

Para testar o tempo total de reconhecimento, foram submetidas as mesmas imagens, e os mesmos processos de extração e armazenamento foram executados.

Após este processo foi dada uma única impressão digital e o processo de reconhecimento foi iniciado, ou seja, a extração de minúcias atuou sobre esta impressão

digital e em seguida as características extraídas foram comparadas com todos os registros existentes no Banco de Dados.

O objetivo deste teste foi medir o tempo de retorno do banco de dados, acrescentando o tempo de comparação (*match*) das minúcias extraídas da impressão digital de entrada com todas as minúcias armazenadas no banco. O resultado pode ser visto na Figura 35.

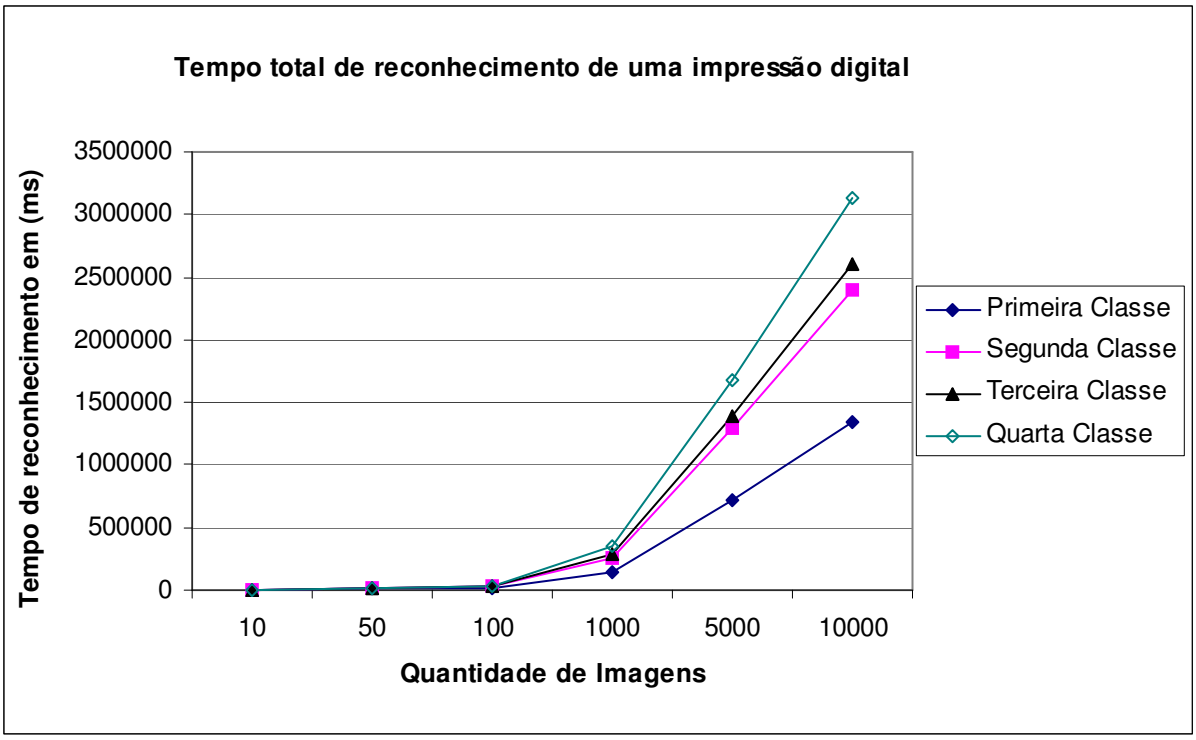


Figura 35 - Tempo total de reconhecimento do sistema

Como pode ser visto nos testes de performance no reconhecimento de impressão digital, o sistema tem um comportamento que pode ser considerado aceitável em todas as classes quando a quantidade de imagens não ultrapassa mil impressões digitais, porém a partir desta quantidade o tempo de espera pelo reconhecimento se torna um fator crítico na maioria dos sistemas.

Os próximos testes a serem executados dizem respeito à precisão no reconhecimento de impressões digitais. Para isto foi identificado a quantidade de erros que o sistema comete em relação à quantidade de imagens que está sendo comparada.

No primeiro caso a primeira classe de imagens foi submetida ao teste de reconhecimento e a percentagem de erros na identificação, o resultado de pode ser vista na Figura 36.

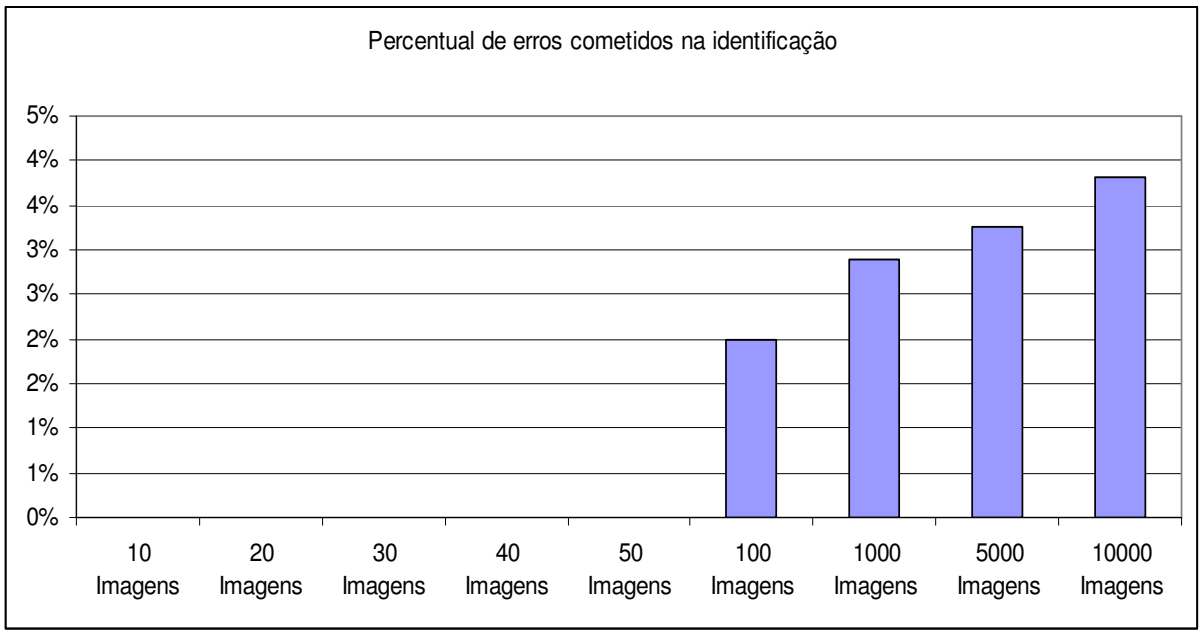


Figura 36 - Porcentagem de erros cometidos com a primeira classe de images

Neste caso a porcentagem de erro foi crescente e manteve-se em um percentual muito baixo, isto em decorrência das imagens geradas pelo software, estas nunca apresentam ruídos ou qualquer tipo de problema, portanto os erros cometidos pelo sistema são simplesmente pelo fato de existirem imagens muito parecidas, ou seja, quando se aumenta a quantidade de imagens a serem comparadas esta probabilidade existe naturalmente.

O mesmo teste foi executado com a segunda classe de imagens, e o resultado pode ser visto na Figura 37.

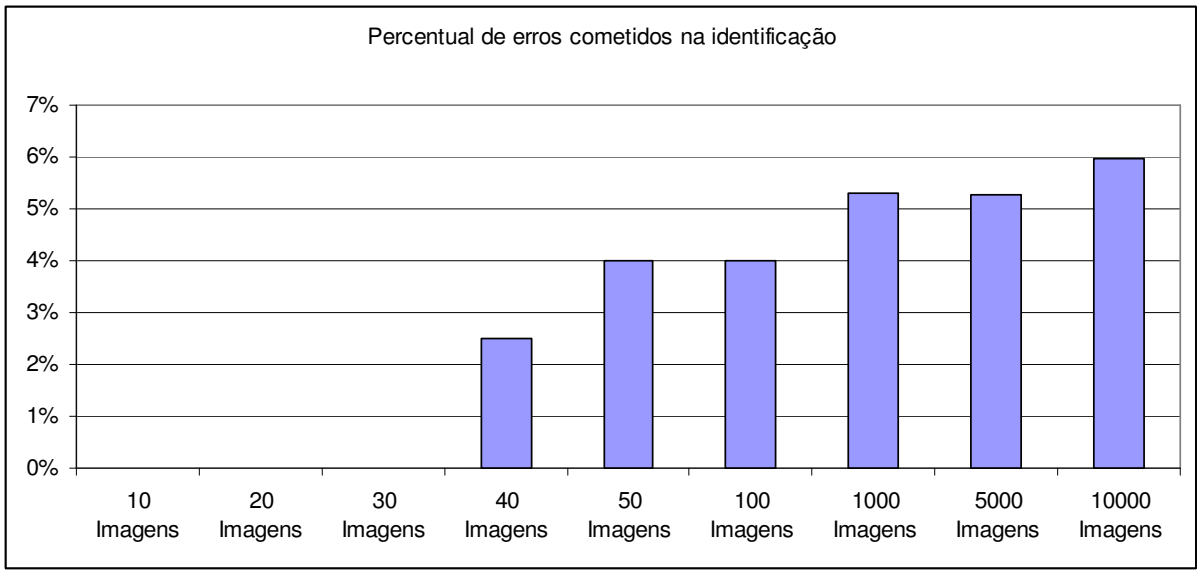


Figura 37 - Porcentagem de erros cometidos com a segunda classe de imagens

Nos resultados desta classe foi possível verificar a variação de percentual existente devido a algumas imagens irreconhecíveis desta classe, por ser tintada em papel diversas imagens estão borradas dificultando o reconhecimento.

O mesmo teste foi executado com a terceira classe de imagens, e o resultado pode ser visto na Figura 38.

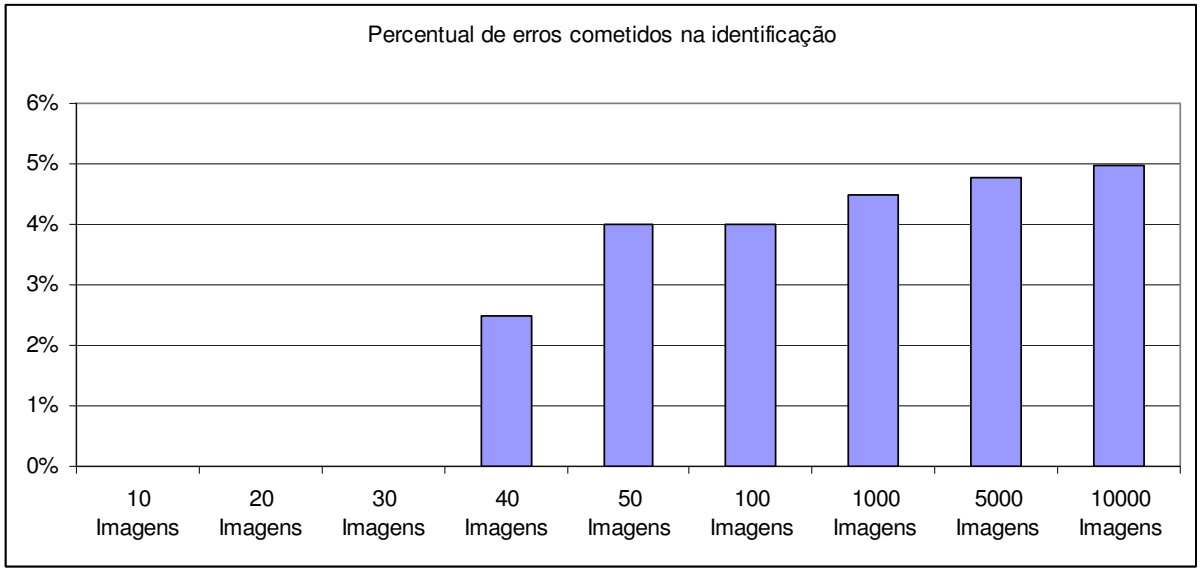


Figura 38 - Porcentagem de erros cometidos com a terceira classe de imagens

Nesta classe a quantidade de imagens irreconhecíveis é menor, porém a captura do leitor traz consigo diversos ruídos na imagem o que causa o não reconhecimento em algumas delas.

O mesmo teste foi executado com a terceira classe de imagens, e o resultado pode ser visto na Figura 39.

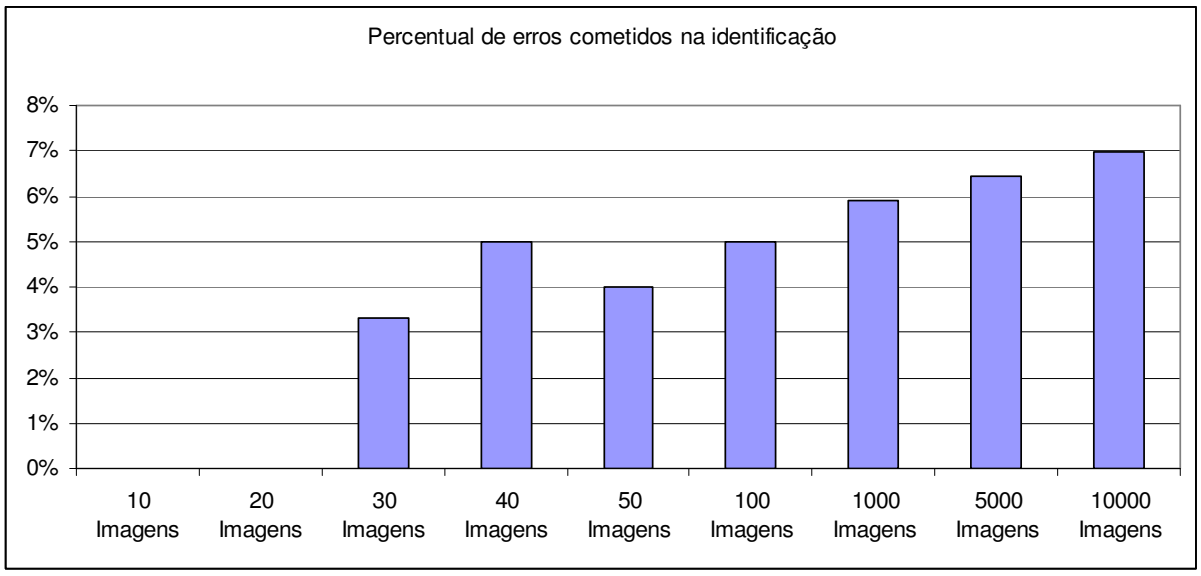


Figura 39 - Porcentagem de erros cometidos com a quarta classe de imagens

Os resultados da quarta classe são determinados pela junção dos dois problemas já descritos, o aparelho de captura da impressão digital, visto que são capturados vários ruídos da imagem inclusive os vestígios que se depositam na superfície do leitor óptico com o uso constante e as imagens irreconhecíveis que são geradas devido a estes problemas.

CONCLUSÕES E TRABALHOS FUTUROS

Com a implementação e os testes realizados neste trabalho foi possível à conclusão de um sistema de reconhecimento de impressão digital baseado no armazenamento e recuperação de informações em um banco de dados.

Com base nos testes é possível afirmar que o sistema obteve um desempenho satisfatório considerando uma base de dados com menos de mil imagens. Pode-se afirmar ainda que a porcentagem de erros no reconhecimento é aceitável diante das condições apresentadas e em relação aos sistemas existentes.

Podem ser citadas como contribuições deste trabalho: os métodos de manipulação do leitor de impressão digital que foram construídos, visto que estes métodos podem ser úteis para outros sistemas que façam uso da impressão digital como meio de reconhecimento ou autenticação. Outra contribuição a ser citada é o uso de banco de dados relacional para armazenamento de características físicas e a recuperação destes dados sendo utilizada para o reconhecimento de padrões.

Este trabalho pode ser complementado com a implementação de técnicas de alinhamento da imagem sendo possível uma otimização na extração das minúcias, e consequentemente aumentando o poder de reconhecimento do sistema. Além desta melhoria podem ser implementados filtros adequados a forma como a imagem foi capturada ou adquirida, isto faria com que os percentuais de erro descritos anteriormente diminuíssem consideravelmente.

Outro ponto a ser explorado é a implementação de outra técnica para determinar o casamento das minúcias, desta forma o desempenho do sistema pode ser aumentado consideravelmente dependendo da quantidade de imagens a ser comparada e do ganho obtido com a nova técnica.

REFERÊNCIAS

BROWN, Wayne C; SHEPHERD, Barry J. **Graphics file formats**: reference and guide. Greenwich: Manning Publications, 1995. 472 p.

CASACURTA, Alexandre;OSÓRIO, Fernando; FRANZ, Figueroa; MUSSE, Soraia Raupp. **Computação Gráfica:** introdução. 1998. 78 f. Disponível em <http://www.unipan.br/odair/CGSM/Apostilas/CS Unicinos.Pdf> . Acesso em: 20/06/2007 ás 15:34.

COSTA, Silvia Maria Farani. **Classificação e Verificação de Impressões Digitais.** 2001. 123 f. Grau: Tese de Mestrado (Mestre em Engenharia Elétrica) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2001.

ELMASRI, Ramez; NAVATHE, Shamkant. **Sistema de Banco de Dados.** ed. 4. São Paulo: Pearson Education do Brasil, 2005. 723p.

EST, East Shore Technologies. **Fingerprint Image Database:** Flat Impressions**.** Disponível em < http://www.east-shore.com/data.html>.Acesso em 15/06/2007 ás 17:24.

GESELLSCHAFT, Volker Roth Fraunhofer. **Content-Based Retrieval from Digital Video.** Issue on Content-based Image Indexing and Retrieval, Darmstadt, 2006.

HARRIS, Tom. **Como funcionam os leitores de impressão digital:** Artigo de Segurança da Informação Disponível em <http://informatica.hsw.com.br/leitores-de-impressoes-digitais.htm>.Acesso em 12/04/2007 ás 22:10.

HENRY, E.R..**Classification and Uses of Fingerprints**. ed. Wyman and Sons Ltda, 1905.

JAIN, Anil; HONG, Lin; BOLLE, Ruud. **On-Line Fingerprint Verification**. IEEE Transactions on Pattern Analysis and machine Intelligence, vol.21,nº4,pp.348-359,1999.

KARU, Kalle; JAIN, Anil J. .Fingerprint Classification. Pattern Recognition, vol. 29, nº 3, pp. 389-404, 1996.

KEHDY, Carlos. **Exercícios de Dactiloscopia**. ed. Sugestões Literárias, 1968.

KORTH, Henry F; SILBERSCHANTZ, Abraham. **Sistema de Banco de Dados.** ed. 2. São Paulo: Makron Books, 1995. 753p.

KOVÁCS, Z. M. **A Fingerprint Verification System Based on Triangular Matching and Dynamic Time Warping**. IEEE Transactions On Pattern Analysis And Machine Intelligence, 2000.1276p.

MATIAS, Caio Rafael Silva Matias. **Protótipo de um Sistema de Identificação dos Delta(s) e Núcleo em Impressões Digitais utilizando redes neurais artificiais.** 2004. 83 f. Grau: Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2004.

MAIO, A.K; D. MALTONI; JAIN, S. Prabhakar. **Handbook of Fingerprint Recognition.** New York: Springer, 2003. 753p.

PACHECO, César Rodrigues dos Anjos. **Autenticação com Impressão Digital.** 2003. 59 f. Grau: Relatório submetido como requisito parcial para obtenção do grau de licenciado (Engenharia de Sistemas de Telecomunicações e Eletrônica) - Departamento de Engenharia de Eletrônica e Telecomunicações e de Computadores, Instituto Superior de Engenharia de Lisboa, Lisboa, 2003.

PERENHA, Rodrigo Afonso. **Sistema de Processamento de Imagens para reconhecimento de impressão digital.** 2003. 62 f. Grau: Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Centro Universitário Eurípides Soares da Rocha, Marília, 2003.

SANCHES, André Rodrigo. **Fundamentos de Armazenamento e Manipulação de Dados:** Arquitetura de Banco de Dados. 2007. Disponível em < http://www.ime.usp.br/~andrers/aulas/bd2005-1/aula4.html>. Acesso em 19/06/2007 ás 17:43.

SONO, Wellington Naoki. **Estudo Cumparativo de sistemas gerenciadores de banco de dados implementados como APIs Java.** 2007. 138 f. Grau: Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Centro Universitário Eurípides Soares da Rocha, Marília, 2007.

APENDICE A - CLASSE DATABASE

```
//=========================================================================
//=================== MANIPULACAO DO BANCO DE DADOS =======================
//=========================================================================


package gui;
import org.h2.*;
import java.sql.*;
import java.util.Arrays;
import java.util.Vector;

import javax.swing.JOptionPane;

public class Database {
	String max = null;
	Statement stmt = null;

	public Database(){
		try {
			Class.forName("org.h2.Driver");
			java.sql.Connection conexao =
DriverManager.getConnection("jdbc:h2:~/test", "sa", "");
			stmt = conexao.createStatement();
		} catch (ClassNotFoundException e) {
			e.printStackTrace();
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}

	public void inserePessoa(String nome) throws ClassNotFoundException,
SQLException{
		stmt.execute("INSERT INTO PESSOA VALUES(null,'"+nome+"')");

		ResultSet rs = stmt.executeQuery("SELECT MAX(ID) FROM PESSOA");

		while(rs.next()){
			max = rs.getString(1);
		}
		rs.close();
	}

	public void insereMinucias(int linha,int coluna,int tipo) throws
ClassNotFoundException, SQLException{
		stmt.execute("INSERT INTO MINUCIAS
VALUES("+max+","+linha+","+coluna+","+tipo+")");
	}

	public void limpaBase() throws ClassNotFoundException, SQLException{

		stmt.execute("DROP TABLE MINUCIAS");
		stmt.execute("CREATE TABLE MINUCIAS("+
				"ID_PESSOA  INT NOT NULL,"+
				"LINHA      INT NOT NULL,"+
				"COLUNA     INT NOT NULL,"+
				"TIPO       INT NOT NULL)");
		stmt.execute("ALTER TABLE MINUCIAS ADD FOREIGN KEY(ID_PESSOA)
REFERENCES PESSOA(ID)");
		stmt.execute("DELETE FROM PESSOA");
```

```
      }

       private Vector<ResultadoMatch> shellSort ( Vector<ResultadoMatch> v
)
     {
        int i , j , h = 1, value ;
        String nome;
        do { h = 3 * h + 1; } while ( h < v.size() );
        do {
           h /= 3;
           for ( i = h; i < v.size(); i++) {
              value = v.elementAt(i).getResultado();
              nome = v.elementAt(i).getNome();
              j = i - h;
              while (j >= 0 && value > v.elementAt(j).getResultado())
              {

v.elementAt(j+h).setResultado(v.elementAt(j).getResultado());
                    v.elementAt(j+h).setNome(v.elementAt(j).getNome());
                    j -= h;
              }
              v.elementAt(j+h).setResultado(value);
              v.elementAt(j+h).setNome(nome);
           }
     } while ( h > 1 );
        return v;
}

      public void buscaSimilares(int largura,int altura,int[] cr_imagem1){

            StoredFinger SF = new StoredFinger();
            String id,nome = null;

            Vector<ResultadoMatch> Resultados = new Vector();

            int resultado = 0;
            try {
                  try {
                        Class.forName("org.h2.Driver");

                        java.sql.Connection            conexao            =
DriverManager.getConnection("jdbc:h2:~/test", "sa", "");
                        stmt = conexao.createStatement();

                        int[]cr_imagem2 = new int[largura * altura];
                        int i = 0;

                        ResultSet   results   =   stmt.executeQuery("SELECT
ID,NOME from PESSOA");
                        int cont = 0;
                        while(results.next()){
                              i=0;
                              id = results.getString(1);

                              Statement stmt2 = conexao.createStatement();
                              ResultSet  rss  =  stmt2.executeQuery("SELECT  *
from MINUCIAS where ID_PESSOA = "+id);

                              while(rss.next()){
                                    cr_imagem2[i]                          =
Integer.parseInt(rss.getString(4));
```

```
                                    i++;
                                }
                                rss.close();

                                nome = results.getString(2);
                                resultado                                    =
SF.getMatchs(largura,altura,cr_imagem1,cr_imagem2);
                                Resultados.add(new
ResultadoMatch(nome,resultado));
                                cont++;

                          }
                          String qtde = JOptionPane.showInputDialog("Digite a
quantidade de resultados a retornar:");
                          String msg = "As pessoas com impressão mais próxima
são:\n\n";

                          for(i=0;i<=Integer.parseInt(qtde)-1;i++){
                                msg       =       msg+"Minucias       Casadas:
"+shellSort(Resultados).elementAt(i).getResultado()+"                        -
"+shellSort(Resultados).elementAt(i).getNome()+"\n";

                          }

                          JOptionPane.showMessageDialog(null,msg);



                          results.close();
                    } catch (ClassNotFoundException e) {
                          // TODO Auto-generated catch block
                          e.printStackTrace();
                    }
              } catch (SQLException e) {
                    e.printStackTrace();
              }
        }

}
```

APENDICE B - MÉTODOS DE MANIPULAÇÃO DO LEITOR

```
//=========================================================================
//==================== MANIPULACAO DO LEUTOR BIOMÉTRICO ====================
//=========================================================================

	public void init(){
		try {
			grFinger = new GrFinger();
			//Inicializa as DLLs de leitores
			grFinger.initializeCapture(this);
			writeLog("Carregado todos os drivers de leitores");
		} catch (GrErrorException e) {
			//Escreve Log do Erro
			e.printStackTrace();
		}
	}


	public void writeLog(String text) {
		//Escreve no JTextArea os Logs
		textAreaOrig.append("\n" + text + "\n");
	}

	public void onPlug(String idSensor) {

		try {
			//Testa se o leitor foi plugado
			if(plug>0){
				writeLog("Leitor: "+idSensor+". Plugado");
			}
			//Inicializa o Leitor para captura
			grFinger.startCapture(idSensor,this,this);
			plug = plug+1;
		} catch (GrErrorException e) {
			//Escreve Log do Erro
			writeLog(e.getMessage());
		}
	}

	public void onUnplug(String idSensor) {
		//Testa se o Leitor foi desplugado
		writeLog("Leitor: "+idSensor+". Desplugado.");
		try {
			//Desativa o leitor para captura
			grFinger.stopCapture(idSensor);
		} catch (GrErrorException e) {
			//Escreve Log de erro
			writeLog(e.getMessage());
		}

	}

	public void showImage(ImageProducer producer) {
		String ImgCapturada = "captured.bmp";
		try {
			GreyImage gi = GreyImage.createImage(ImgCapturada);
			gi = GreyImage.createImage(ImgCapturada);
			this.c2.setImage(gi);
```

```
            } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
            c2.repaint();
      }

      public void onImage(String idSensor, FingerprintImage fingerprint) {
            try {
                  // Save image.

      fingerprint.saveToFile("captured.bmp",FingerprintImage.GRCAP_IMAGE_FO
RMAT_BMP);
            } catch (GrErrorException e) {
                  //write error to log
                  this.writeLog(e.getMessage());
            }

            // Escreve Log de Imagem Capturada
            writeLog("Sensor: "+idSensor+": Imagem Capturada.");
            try {
                  //Adiciona a Impressao no Painel
                  showImage(fingerprint.newImageProducer());

            } catch (GrErrorException e) {
                  //Escreve Log do Erro
                  writeLog(e.getMessage());
            }
      }

      //=====================================================================
      //=====================================================================
      //=====================================================================
```

APENDICE C - MÉTODOS DE INSERÇÃO DAS MINUCIAS

```
//=========================================================================
//======================== INSERCAO DAS MINUCIAS ==========================
//=========================================================================

//Este método está na classe greyImage

public void insereMinucias(String Nome)
      {
        //INSERINDO PESSOA
            System.out.println("Inserindo Pessoa...");
        Database DB = new Database();
        try {
            DB.inserePessoa(Nome);
        } catch (ClassNotFoundException ex) {
            // TODO Auto-generated catch block
            ex.printStackTrace();
        } catch (SQLException ex) {
            // TODO Auto-generated catch block
            ex.printStackTrace();
        }

          System.out.println("Pessoa Inserida com Sucesso !");


            System.out.println("Inserindo  Minucias  desta  Pessoa...  !");


            for ( int l = 0; l < nLins; l++ )
                  for ( int c = 0; c < nCols; c++ )
                  {
                        //if(arrayMinucias[l][c]==6                        ||
arrayMinucias[l][c]==2){
                              //INSERINDO MINUCIAS
                              try {

      DB.insereMinucias(l,c,arrayMinucias[l][c]);
                              } catch (ClassNotFoundException e) {
                                    // TODO Auto-generated catch block
                                    e.printStackTrace();
                              } catch (SQLException e) {
                                    // TODO Auto-generated catch block
                                    e.printStackTrace();
                              }
                        //}
                  }

                  //System.out.println("  Minucias  Inseridas  com  sucesso
!");
               JOptionPane.showMessageDialog(null,"Impressão    Armazenada
com Sucesso !!!");
}

//=========================================================================
//=========================================================================
//=========================================================================
```

APENDICE D - MÉTODO MATCH COM IMAGEM DO BANCO

```
//=========================================================================
//==================== MATCH COM IMAGEM DO BANCO ==========================
//=========================================================================


public int getMatchs(int largura,int altura,int[] crossNumberg1,int[]
crossNumberg2){
            int matchs = 0;

            for (int i = 100; i >= 30; i-=10)
            {
                  GreyImage GI = new GreyImage(largura,altura);

                  int [] minuc1 = GI.getMinucias(crossNumberg1, i);
                  int[] minuc2 =  GI.getMinucias(crossNumberg2, i);

                  StoredFinger sf1 = new StoredFinger(altura, largura,
minuc1);
                  StoredFinger sf2 = new StoredFinger(altura, largura,
minuc2);
                  for (int j = 5; j <= 20; j+=5)
                  {
                        System.out.println("================ Area: " + i +
"% " +
                                    "Raio:      "      +      j      +      "
=========================");
                        int[] m = sf1.match(sf2, j);
                        System.out.println("Minucias    de    1:    "    +
minuc1.length);
                        System.out.println("Minucias    de    2:    "    +
minuc2.length);
                        System.out.println("Matches: " + m.length);
                        matchs = m.length;
                  }
            }
            return matchs;
      }

//=========================================================================
//=========================================================================
//=========================================================================
```

APENDICE E - CHAMADAS DOS MÉTODOS

```
//=========================================================================
//=================== CHAMADA DO MÉTODO BUSCA SIMILARES ===================
//=========================================================================

wr = dest.getRaster();
                  wr.setPixels(0,    0,    wr.getWidth(),    wr.getHeight(),
dest.quantizacao());
                  wr.setPixels(0,    0,    wr.getWidth(),    wr.getHeight(),
dest.spliting());
                  wr.setPixels(0,    0,    wr.getWidth(),    wr.getHeight(),
dest.thresholdGlobal());
                  wr.setPixels(0,    0,    wr.getWidth(),    wr.getHeight(),
dest.thinning(255));
                  wr.setPixels(0,    0,    wr.getWidth(),    wr.getHeight(),
dest.limpeza2(0));

//Obtendo o cross number da imagem destino
int[] crVet1 = dest.crossNumber(0);

Database DB = new Database();
DB.buscaSimilares(wr.getWidth(),wr.getHeight(),crVet1);



//=========================================================================
//=================== CHAMADA DO MÉTODO INSERE MINUCIAS ===================
//=========================================================================

//Aplica os Filtros
                   wr = dest.getRaster();
                   wr.setPixels(0,    0,    wr.getWidth(),    wr.getHeight(),
dest.quantizacao());
                   wr.setPixels(0,    0,    wr.getWidth(),    wr.getHeight(),
dest.spliting());
                   wr.setPixels(0,    0,    wr.getWidth(),    wr.getHeight(),
dest.thresholdGlobal());
                   wr.setPixels(0,    0,    wr.getWidth(),    wr.getHeight(),
dest.thinning(255));
                   wr.setPixels(0,    0,    wr.getWidth(),    wr.getHeight(),
dest.limpeza2(0));
                   //Encontra as Minucias
                   int cor = 0;
                   int[] minuc = dest.crossNumber(cor);

                   minuc = dest.getMinucias(minuc, 70);

                   //Pinta as Minucias na imagem
                   wr = dest.getRaster();
                   wr.setPixels(0, 0, wr.getWidth(), wr.getHeight(),
                                              dest.plotMinucias(minuc));

                   //Insere as Minucias
                   String   Nome   =   JOptionPane.showInputDialog("Digite   o
Nome");
                 dest.insereMinucias(Nome);
```

APENDICE F - CLASSE RESULTADOMATCH

```
//============================================================================
//========================== CLASSE RESULTADO MATCH ==========================
//============================================================================


package gui;

public class ResultadoMatch {

	private String nome;
	private int casadas;

	public ResultadoMatch (String nome,int casadas) {
		this.nome = nome;
		this.casadas = casadas;
	}
	public String getNome(){
	return nome;
	}

	public int getResultado(){
		return casadas;
	}

	public void  setNome(String nome){
		this.nome = nome;
		}

	public void  setResultado(int casadas){
		this.casadas = casadas;
		}

}
```

APENDICE G - SCRIPT BANCO DE DADOS

```
CREATE TABLE PESSOA(
      ID          INT         NOT NULL    AUTO_INCREMENT    PRIMARY KEY ,
      NOME        VARCHAR2(255)     NOT NULL    UNIQUE
);

CREATE TABLE MINUCIAS(
      ID          INT         NOT NULL    AUTO_INCREMENT    PRIMARY KEY ,
      ID_PESSOA   INT NOT NULL,
      LINHA       INT NOT NULL,
      COLUNA            INT NOT NULL,
      TIPO        INT NOT NULL
);

ALTER TABLE MINUCIAS ADD FOREIGN KEY(ID_PESSOA) REFERENCES PESSOA(ID);
```