UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS EXATAS E DA NATUREZA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

JOSÉ RAPHAEL TEIXEIRA MARQUES

SISTEMA DE ALTO DESEMPENHO PARA COMPRESSÃO SEM PERDAS DE IMAGENS MAMOGRÁFICAS

João Pessoa

2010

JOSÉ RAPHAEL TEIXEIRA MARQUES

SISTEMA DE ALTO DESEMPENHO PARA COMPRESSÃO SEM PERDAS DE IMAGENS MAMOGRÁFICAS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática, da Universidade Federal da Paraíba, como requisito parcial para obtenção do título de Mestre em Informática.

ORIENTADOR: Prof. Dr. Leonardo Vidal Batista

João Pessoa

2010

ATA DA SESSÃO PÚBLICA

DEDICATÓRIA

A minha mãe, Maria Vitória.

AGRADECIMENTOS

Agradeço,

A Deus, por tudo.

A minha família pelo amor e apoio em todos os momentos da minha vida.

A Luciane, pela cumplicidade, incentivo e amor insubstituíveis na minha vida.

Ao meu orientador, professor Leonardo Vidal Batista, por ser um exemplo de professor e companheiro.

Ao meu coorientador, JanKees, pela amizade e pelo exemplo de profissionalismo.

Aos amigos do PET.Com.

Aos amigos, Adriano, Amanda, Bruno, Thallys, Tiago Dias e Yuri.

A todos os colegas com quem pude trabalhar e aprender.

E aos professores que contribuíram para minha formação.

RESUMO

A utilização de bancos de dados de imagens mamográficas em formato digital e as práticas de telemedicina exigem que se armazene e transmita grandes quantidades de dados. A digitalização das quatro imagens de um único exame mamográfico com resolução adequada pode ocupar até 120MB de espaço em disco. Esta quantidade de dados leva a uma situação ainda mais crítica ao considerar-se o grande número de exames diários efetuados rotineiramente em uma clínica. Assim, técnicas eficientes de compressão de dados são necessárias para reduzir os custos relativos ao armazenamento e à transmissão destas imagens.

O presente trabalho descreve o desenvolvimento de um sistema de alto desempenho para compressão sem perdas de imagens mamográficas baseado no algoritmo *Prediction by Partial Matching* (PPM), em conjunto com módulos para segmentação, mapeamento, codificação com Código Gray, decomposição em planos de bits e transformada *move-to-front* (MTF). O sistema desenvolvido mostrou-se eficiente tanto no que tange à razão de compressão quanto ao tempo de processamento, comprimindo imagens de 27MB em aproximadamente 13 segundos com razão de compressão média de 5,39.

Palavras-chave: *Prediction by Partial Matching* (PPM), Compressão sem perdas, Mamografia.

ABSTRACT

The usage of mammographic image databases in digital form and the practice of telemedicine require to store and to transmit large amounts of data. The image digitization from a single mammographic exam with appropriate resolution can take up to 120MB of space in disk, which becomes even more critical when considering the large number of exams per day on a clinic. Thus, efficient data compression techniques are needed to reduce storage and transmission costs.

This document describes the development of a high-performance lossless compressor based on Prediction by Partial Matching (PPM) algorithm with modules for segmentation, mapping, gray code, bit planes decomposition and move-to-front transform, for mammographic image compression. The compressor developed was efficient in both compression ratio and processing time, and compresses 27MB images in about 13 seconds with an average compression ratio of 5,39.

Keywords: Prediction by Partial Matching (PPM), Lossless Compression, Mammography.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE EQUAÇÕES

LISTA DE SIGLAS

ACR	*American College of Radiology*
API	*Application Programming Interface*
BIRADS	*Breast Image Reporting and Data System*
BWT	*Burrows-Wheeler Transform*
CAD	*Computer-Aided Diagnosis*
CALIC	*Context-based Adaptive Lossless Image Compression*
CSPE	*Chronological Sifting of Prediction Errors*
DAC	*Dados com Compressão*
DDR2	*Double Data Rate 2*
DDSM	*Digital Database for Screening Mammography*
DPCM	*Differential Pulse Code Modulation*
EDP	*Edge Directed Prediction*
GHz	*Giga Hertz*
GB	*Giga Byte*
HD	*Hard Disk*
JBIG	*Joint Bi-level Image Experts Group*
JNA	*Java Native Access*
JPEG	*Joint Photographic Experts Group*
JVM	*Java Virtual Machine*
KB	*Kilo Byte*
LAPIMO	Lab. de Análise e Processamento de Imagens Médicas e Odontológicas
LOCO-I	Low Complexity Lossless Compression for Images
LZW	Lempel-Ziv-Welch
MB	*Mega Byte*
MTF	*Move-to-Front*
PNG	*Portable Network Graphics*
PPAM	*Prediction by Partial Approximate Matching*
PPM	*Prediction by Partial Matching*
RAM	*Random Access Memory*
RC	Razão de Compressão
RLE	*Run Length Encoding*

SATA	*Serial Advanced Technology Attachment*
SIADM	Sistema de Auxílio ao Diagnóstico por Mamografia
SPIHT	*Set Partitioning In Hierarchical Trees*
TIFF	*Tagged Image File Format*

SUMÁRIO

1 INTRODUÇÃO

O câncer de mama é considerado um problema de saúde publica em quase todos os países. È o segundo tipo de câncer mais frequente no mundo. No Brasil, o câncer de mama é a maior causa de morte por câncer entre as mulheres. O Instituto Nacional de Câncer estima 49.240 novos casos de câncer de mama no Brasil para o ano de 2010 (INCA, 2009).

A Organização Mundial de Saúde recomenda o rastreamento em massa de doenças como o câncer de mama que podem diminuir a mortalidade se diagnosticados precocemente (ANDERSON *et al*., 2003). O Ministério da Saúde considera como uma das principais estratégias de rastreamento um exame mamográfico, no mínimo a cada dois anos, para mulheres de 50 a 69 anos. No caso de mulheres consideradas como pertencentes a grupo de risco elevado para câncer de mama (parentes de primeiro grau com histórico de câncer de mama), a recomendação é que o exame mamográfico seja feito anualmente a partir dos 35 anos (INCA, 2009).

Um sistema de diagnóstico auxiliado por computador - CAD (*Computer-Aided Diagnosis*) (ALMEIDA *et al*., 2006; ASTLEY e GILBERT, 2004) pode auxiliar o radiologista emitindo uma segunda opinião que pode colaborar para um diagnóstico precoce. Sistemas CAD utilizam imagens médicas em formato digital, e no caso de imagens mamográficas precisam de muito espaço de armazenamento. Sistemas para o armazenamento e transmissão dessas imagens requerem compressão das imagens pra acelerar a velocidade de transmissão e economizar espaço em disco para reduzir custos (SUNG *et al*., 2002). No caso de imagens médicas, é necessário preservar a qualidade das imagens comprimidas para não comprometer o diagnóstico médico.

Para exemplificar, uma única imagem mamográfica digitalizada com resolução adequada ao diagnóstico pode ocupar aproximadamente 30MB em disco, 120MB no caso de um exame completo composto por quatro imagens. Essa situação se torna mais crítica quando se leva em consideração uma clínica que realiza um grande número de exames diários.

A compressão de imagens, muitas vezes, envolve a eliminação de alguma informação, preservando apenas as informações mais relevantes da imagem. Este processo é chamado de compressão com perdas, porque há perda de informação impossibilitando uma recuperação perfeita da imagem original.

A compressão sem perdas permite a recuperação exata da imagem original. É, portanto, uma escolha segura para aplicações que envolvem imagens médicas, já que não afeta a qualidade das imagens (SUNG *et al*., 2002).

O desenvolvimento de um compressor sem perdas específico para imagens mamográficas permite explorar as características peculiares das imagens visando aumentar as taxas de compressão, reduzindo os custos de armazenamento e transmissão dessas imagens.

O algoritmo PPM (*Prediction by Partial Matching*) é o estado da arte em compressão sem perdas. O PPM é um compressor baseado em modelagem estatística adaptativa contextual. Uma vantagem do PPM é que ele ”aprende” o modelo estatístico mais rapidamente que outros algoritmos de modelagem adaptativa voltados à compressão. Outra vantagem do PPM é que sua modelagem contextual se adapta bem à compressão de arquivos obtidos de fontes reais, tais como imagens mamográficas.

Previamente à aplicação de um algoritmo como o PPM, muitas vezes são realizados outros procedimentos como transformadas, segmentações e codificações diferentes que ajudam na etapa de compressão. Dentre as transformadas existentes, podem-se destacar as transformadas BWT (*Burrows–Wheeler Transform*) e MTF (*Move-To-Front*), muito utilizadas em esquemas de compressão sem perdas. A segmentação de uma imagem tem como objetivo, separar regiões distintas segundo algum critério. No caso de imagens mamográficas, duas regiões bem distintas são a região do *background* (região escura da imagem) e a região da mama (região mais clara).

O grupo de pesquisa em Processamento e Análise de Sinais, Imagens e Vídeo e Reconhecimento de Padrões da Universidade Federal da Paraíba vêm desenvolvendo um sistema avançado de auxílio ao diagnóstico médico baseado em imagens mamográficas (SIADM), incorporando ampla funcionalidade, soluções inovadoras e uma arquitetura de software aberta e reconfigurável.

Este trabalho descreve a pesquisa e o desenvolvimento de um sistema de alto desempenho para compressão sem perdas de imagens mamográficas baseado no algoritmo PPM, que será futuramente integrado ao SIADM.

1.1 Trabalhos Relacionados

Schiabel e Escarpinati (2002) fizeram uma avaliação de sete técnicas de compressão sem perdas encontradas na literatura aplicadas a imagens mamográficas digitais de mamas densas. Foram avaliadas as técnicas: decomposição em planos de bits com codificação RLE unidimensional, técnica de Huffman, técnica DAC, codificação LZW, codificação JPEG sem perdas, LOCO-I e CALIC. Ainda foi testado o *software* comercial WinZip 8.0. As melhores taxas de compressão foram obtidas com a técnica JPEG sem perdas, sendo 12% superior à segunda colocada, LOCO-I, 35% superior ao software comercial WinZip 8.0 e 84%.superior à ultima colocada, a técnica de decomposição em planos de bits. A avaliação mostra a ineficiência da decomposição em planos de bits, mas sugere a utilização de outra técnica de compressão no lugar do RLE.

Przelaskowski (2004) também faz um estudo entre compressores voltado a imagens mamográficas. Os compressores sem perdas avaliados foram: codificador aritmético contextual binário, CALIC, JPEG-LS e JPEG2000. Os melhores resultados foram obtidos com o codificador aritmético, seguido do CALIC, JPEG-LS e por último o JPEG2000. É interessante destacar que é utilizado um codificador aritmético binário baseado em contexto, técnica com algumas semelhanças à utilizada nesta dissertação.

Silva e Scharcanski (2005) propuseram um método de compressão de imagens mamográficas digitais baseado na triangulação de Delaunay. O método utiliza a triangulação de Delaunay na predição e comprime erro obtido em relação a essa predição utilizando várias técnicas como PNG, RLE (*Run-Length Encoding*), DPCM (*Differential Pulse Code Modulation*) e JPEG-LS. Em uma comparação com outros compressores sem perdas o método proposto mostrou melhores resultados que o JPEG2000, o JPEG-LS, o JPEG-lossless e o PNG. Uma limitação do método é que é que ele não foi projetado para execução rápida. Além de não ser eficiente na codificação de regiões com ruído.

Neekabadi *et. al.* (2007) propôs um método de compressão gradual de erro de predição de forma iterativa chamado de CSPE (*Chronological Sifting of Prediction Errors*). Utilizando imagens mamográficas de 8 bits, o método proposto mostrou os melhores resultados quando comparado com outros compressores sem perdas: JBIG, JPEG-LS, JPEG2000 e PNG.

Nenhum dos métodos citados acima indica quais imagens foram utilizadas nos testes. Como a compressão pode variar de imagem para imagem, os resultados de compressão obtidos por cada método são irrelevantes, sendo mais importante a sua comparação com outros métodos.

Zhang e Adjeroh (2008) apresentaram o PPAM (*Prediction by Partial Approximate Matching*), um método para compressão e modelagem contextual de imagens. Diferente da modelagem do PPM que usa contextos exatos, o PPAM usa a noção de contextos aproximados. Duas variações do algoritmo proposto foram comparadas com outros compressores sem perdas: EDP (*Edge Directed Prediction*), CALIC, JPEG-LS, SPIHT (*Set Partitioning In Hierarchical Trees*) e JPEG 2000 e mostraram taxas de compressão competitivas com as taxas dos outros compressores. O PPAM mostrou melhores resultados na compressão de imagens médicas, porém, só foram testadas imagens relativamente pequenas de no máximo 512x512. Além disso, os tempos de processamento chegaram a ser, em alguns casos, 30 vezes maiores que os tempos do JPEG-LS e JPEG 2000, se aproximando apenas dos tempos obtidos com o EDP. A ideia de contexto aproximado do PPAM melhora a compressão, mas aumenta ainda mais o custo computacional do PPM.

1.2 Objetivos

O objetivo deste trabalho é pesquisar e desenvolver um sistema de alto desempenho para compressão sem perdas de imagens mamográficas.

Os objetivos específicos deste trabalho são:

1. Pesquisar e desenvolver um compressor sem perdas de alto desempenho baseado no algoritmo PPM.
2. Pesquisar e desenvolver um compressor sem perdas utilizando Código Gray.
3. Pesquisar e desenvolver um compressor sem perdas utilizando a transformada MTF.
4. Pesquisar e desenvolver um compressor sem perdas utilizando segmentação de imagens mamográficas.

5. Desenvolver o compressor de forma modular para facilitar a alteração dos módulos existentes e a inserção de novos módulos.
6. Desenvolver o compressor de modo a permitir que os arquivos comprimidos com uma versão do compressor possam ser descomprimidos por versões futuras através da implementação de retrocompatibilidade de arquivo.
7. Desenvolver um módulo de descompressão progressiva *lossy-to-lossless* que permita visualizar parcialmente as informações contidas na imagem à medida que ela é descomprimida.
8. Desenvolver o compressor para que execute a compressão e descompressão em tempos aceitáveis para uma aplicação real.
9. Realizar testes para medir eficiência do compressor em relação à razão de compressão e tempo de processamento, comparando-o com outros compressores comerciais ou descritos na literatura.

O restante da dissertação está organizada nos seguintes capítulos:

Capítulo 2 - Fundamentação Teórica. Neste capítulo são apresentados os principais conceitos teóricos necessários para o desenvolvimento do trabalho.

Capítulo 3 - Materiais e Métodos. Descreve os recursos computacionais, os bancos de dados e os métodos utilizados para desenvolvimento e avaliação empírica do compressor proposto.

Capítulo 4 - Resultados e Discussão: Este capítulo apresenta os resultados obtidos nos testes com o compressor desenvolvido.

Capítulo 5 – São apresentadas as contribuições e conclusões das atividades realizadas e perspectivas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos teóricos necessários para o desenvolvimento do trabalho.

2.1 Conceitos Básicos de Compressão de Dados

As técnicas de compressão tentam explorar a redundância existente nos dados gerados por fontes usuais de informação. Essas técnicas se baseiam em reduzir essa redundância otimizando a codificação dos dados gerados pela fonte, reduzindo a representação da informação gerada pela fonte e procurando preservar a informação em si (BATISTA, 2002).

Para medir o grau de compressão obtido, um indicador muito utilizado é a razão de compressão (*RC*), dada por:

$$RC = \frac{To}{Tc} \tag{1}$$

onde, ***To*** e ***Tc*** são os tamanhos em bits da mensagem original e comprimida, respectivamente. Quanto maior o valor da *RC*, maior é a compressão obtida, sendo que valores menores ou igual a 1 (um) significa que não houve compressão.

Na compressão com perdas, alguma informação é eliminada ou alterada, possibilitando apenas a recuperação de uma aproximação da mensagem original. A diferença entre a mensagem descomprimida e a mensagem original pode ser chamada de distorção, e em geral, razões de compressão maiores são obtidas aumentando a distorção.

Na compressão sem perdas toda a informação é preservada permitindo a recuperação perfeita da mensagem original. A compressão sem perdas é aplicada quando não é aceitável a eliminação ou alteração de nenhum bit da mensagem, como na

compressão de arquivos de texto. Para imagens médicas, a perda de informação pode comprometer o diagnóstico médico sendo, portanto, mais apropriada a utilização de compressores sem perdas.

2.1.1 Informação e Entropia

Dada uma fonte de informação que gera elementos (símbolos) pertencentes a um conjunto $A=\{a_0, a_1, ..., a_{M-1}\}$ (alfabeto da fonte, com M símbolos), um símbolo x gerado pela fonte pode ser tratado como uma variável aleatória com probabilidade $P(x=a_i)$. Associada à probabilidade de ocorrência desse símbolo, Shannon (1948) define sua auto-informação, $I(a_i)$, como:

$$I(a_i) = \log_2 \frac{1}{P(x = a_i)} \text{bits} \tag{2}$$

Considerando todos os símbolos gerados por uma fonte de informação, a entropia de ordem 1 da fonte é definida como:

$$H(S) = -\sum_{i=0}^{M-1} P(x = a_i) \log_2 P(x = a_i) \quad \text{bits/símbolo} \tag{3}$$

A entropia de ordem 1 pressupõe que os símbolos gerados pela fonte são independentes. Quando há dependência estatística entre os símbolos, podem-se utilizar as cadeias de Markov (SOLOMON, 2004) de ordem n, onde a auto-informação de um símbolo é calculada considerando a probabilidade de ocorrência de um símbolo condicionada à ocorrência dos n símbolos anteriores (contexto do símbolo).

A informação representa o menor número de bits necessários para representar um símbolo a_i dada a distribuição de probabilidade dos símbolos gerados pela fonte e a entropia representa o menor número médio de bits por símbolo com que a mensagem pode ser representada sem perda de informação (SHANNON, 1948). Assim, a entropia limita

inferiormente o número médio de bits por símbolo com que se pode codificar sem perdas uma mensagem produzida por uma fonte de informação qualquer. O problema de construir essa codificação foi resolvido com a codificação aritmética que será apresentada mais adiante.

Como a codificação ótima já é um problema resolvido, o que se pode fazer para obter melhores resultados de compressão é reduzir o valor da entropia de uma mensagem. Uma opção para isso é a utilização de cadeias de Markov que permitem reduzir a entropia da mensagem obtendo ganhos de compressão.

Como exemplo, considerando-se um texto qualquer na língua portuguesa, supõe-se que a letra ‘u’ ocorra com uma probabilidade próxima de 2%, o que equivale a uma auto-informação de aproximadamente 5,64 bits. Porém, quando uma letra ‘q’ é encontrada, a probabilidade de que a próxima letra seja um ‘u’ sobe para próximo a 100%, e a auto-informação aproxima-se de zero.

2.2 Técnicas de Compressão de Dados

Esta seção aborda as técnicas de compressão de dados utilizadas neste trabalho.

2.2.1 Modelagem

Muitas técnicas de compressão podem ser decompostas em duas etapas: modelagem e codificação. Isso permite a criação de vários compressores com características diversas a partir da junção de um número relativamente reduzido de métodos de modelagem e codificadores (BATISTA, 2002).

A modelagem é a extração das características estatísticas da fonte de informação. Um modelo é, então, a representação dessas características. A codificação é a etapa de representação dos símbolos da mensagem baseada nas suas probabilidades.

A entropia é calculada a partir das probabilidades condicionais dos símbolos da mensagem. Quanto melhor um modelo estatístico "aprender" sobre o comportamento da fonte, menor será sua entropia. Um modelo pode ser mais eficiente se levar em consideração a dependência estatística entre os símbolos de uma mensagem. Dessa forma, a entropia não depende apenas da mensagem, mas também do modelo estatístico.

A codificação ótima, dada uma distribuição de probabilidades, é um problema resolvido desde 1987, com a divulgação da codificação aritmética (WITTEN *et al*., 1987). Entretanto, pesquisas em modelagem estatística continuam ativas.

A criação de modelos mais precisos que capturem melhor o comportamento de uma fonte genérica permite reduzir sua entropia obtendo ganhos de compressão. Entretanto, o problema de definir um modelo que leve à menor entropia não possui solução definida (BELL *et al*., 1990).

2.2.2 Codificação de Entropia

Uma codificação prática consiste em representar os símbolos através de palavras-códigos formadas por sequências de bits. Atribuindo-se palavras-códigos menores à representação dos símbolos com maior probabilidade, é possível reduzir o número total de bits necessários para representar a mensagem inteira. Este é o principio básico dos codificadores de entropia (BATISTA, 2002).

2.2.3 Codificação Aritmética

É certo que a representação binária mais eficiente de uma mensagem ocorre quando o comprimento médio de bits por símbolo se iguala à entropia. Isso acontece quando cada símbolo é codificado com um número de bits igual à sua auto-informação. Entretanto, em codificadores como o algoritmo de Huffman (HUFFMAN, 1952) as palavras-códigos possuem comprimento inteiro impedindo, exceto em casos específicos, que o comprimento médio dos símbolos alcance a entropia (BATISTA, 2002).

O codificador aritmético resolve o problema de atribuir um código de comprimento inteiro para símbolos individuais atribuindo um único código para toda a mensagem (SALOMON, 2004), possibilitando praticamente igualar a entropia em todos os casos (BATISTA, 2002).

A codificação aritmética pode ser descrita nos seguintes passos (SALOMON, 2004):

- Comece definindo um intervalo corrente entre 0 e 1;
- Repita os passos seguintes para cada símbolo ***x*** a ser codificado:
  - Divida o intervalo corrente em subintervalos cujos tamanhos são proporcionais às probabilidades dos símbolos do alfabeto;
  - Selecione o subintervalo correspondente ao símbolo ***x*** e defina esse subintervalo como o novo intervalo corrente;
- Quando todos os símbolos forem processados, a saída do algoritmo pode ser qualquer número dentro do intervalo corrente, de preferência um número com a menor quantidade de dígitos possível.

A Figura 1 mostra uma representação do codificador aritmético codificando a sequência “00110000” onde o símbolo 0 possui probabilidade de 75% e o símbolo 1 probabilidade de 25%. O resultado final é o número 0,53.

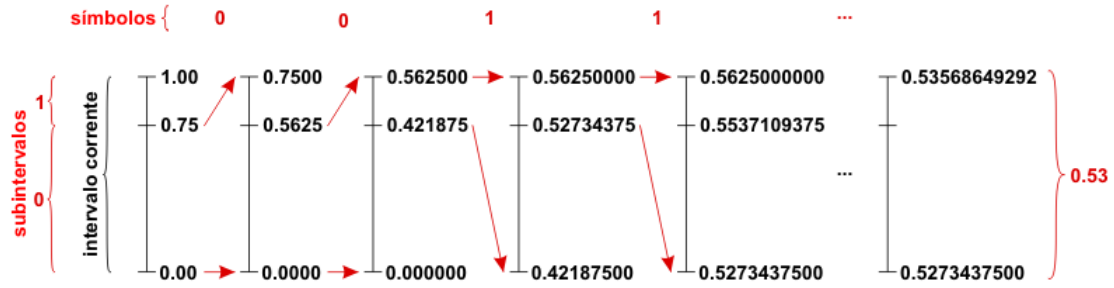

Figura 1: Codificador Aritmético.

A descrição básica do codificador aritmético é inviável para uma aplicação prática, pois exige aritmética de precisão ilimitada e só há alguma codificação quando todos os símbolos são processados. Porém, já existem mecanismos que possibilitam a construção de codificadores aritméticos executáveis em computadores atuais (SALOMON, 2004).

2.2.4 Compressor PPM

O PPM (*Prediction by Partial Matching*) (BELL *et al*., 1984; MOFFAT, 1990) é um algoritmo de compressão baseado na modelagem adaptativa com predição probabilística contextual e codificação de entropia. O modelo PPM utiliza um conjunto de no máximo ***K*** símbolos precedentes como contexto para estimar a distribuição de probabilidades condicionais para o próximo símbolo da mensagem.

O PPM mantém um modelo estatístico com as probabilidades condicionais dos símbolos associadas a contextos de tamanho máximo ***K*** e tamanho mínimo zero. Esse modelo pode ser visto como um conjunto de contadores onde cada contador armazena a frequência de ocorrência de um símbolo ***x*** em um contexto ***C***. No inicio do processamento, considera-se o modelo vazio, ou seja, sem nenhuma informação sobre a fonte e, portanto, sem nenhum contador. À medida que o modelo é atualizado pelo PPM, são criados novos contadores.

Dado um símbolo ***x*** e seu contexto ***C*** com os ***K*** símbolos precedentes, comprime-se o símbolo ***x*** baseado na sua probabilidade condicionada ao seu contexto. Na ocorrência de um novo símbolo no contexto ***C***, um símbolo especial chamado escape é codificado que

representa a ocorrência de um símbolo desconhecido nesse contexto. Após codificar um escape, tenta-se codificar o símbolo em um contexto de tamanho ***K*-1,** e o processo se repete até alcançar um contexto de tamanho zero. Se o símbolo não puder ser codificado em nenhum contexto, ele será codificado utilizando um modelo de "ignorância absoluta", onde se admite que todos os símbolos do alfabeto sejam equiprováveis.

A cada iteração do algoritmo, o modelo é atualizado incrementando os contadores específicos de cada símbolo nos contextos que ocorreram e adicionando novos símbolos e novos contextos.

A codificação do símbolo é feita geralmente utilizando um codificador aritmético.

2.2.5 Transformada *Move-to-front*

A transformada *move-to-front* (MTF) (BENTLEY *et al*., 1986) é uma codificação que auxilia na compressão por entropia.

Durante a codificação, é criada uma lista com os símbolos do alfabeto da mensagem. Cada símbolo é codificado com o valor referente à sua posição na lista (considerando a primeira posição como 0), em seguida, o símbolo na lista é movido para o início ocupando a posição 0. Com esse esquema, os símbolos mais frequentes estarão no início da tabela sendo codificados com valores menores. E no caso de sequências de símbolos repetidos, serão codificadas sequências de zeros.

Um exemplo do funcionamento da MTF é mostrado na Tabela 1 onde a mensagem "ABBBCCCAAAA" é codificada como "01002002000".

No caso de um alfabeto binário, a ocorrência de zeros na mensagem codificada é ainda maior. Por exemplo, a mensagem "0001111100011000" seria codificada como "0001000010010100".

Tabela 1: Transformada MTF

Lista com o alfabeto	Símbolo atual	Posição do símbolo na lista
A B C D	A	0
A B C D	B	1
B A C D	B	0
B A C D	B	0
B A C D	C	2
C B A D	C	0
C B A D	C	0
C B A D	A	2
A C B D	A	0
A C B D	A	0
A C B D	A	0

2.2.6 Transformada Burrows-Wheeler (BWT)

A transformada Burrows-Wheeler (BWT) (BURROWS e WHEELER, 1994) é muito utilizada em compressão de dados.

A BWT não altera os valores dos símbolos da mensagem, apenas altera suas posições. Se em uma mensagem ocorrer muitas repetições de alguma sequência de símbolos, a mensagem codificada conterá vários símbolos repetidos em sequência facilitando sua compressão.

A BWT é geralmente utilizada em conjunto com outras técnicas que se valem dos símbolos repetidos para obter uma maior compressão, como a transformada MTF ou algoritmos baseados em codificação por comprimento de sequência (*Run Length Encoding*, RLE).

A transformada BWT é feita ordenando todas as rotações da mensagem e codificando o último símbolo de cada rotação após ordenação. Adicionalmente, a BWT

precisa codificar um símbolo adicional que indica a posição da mensagem original na lista ordenada de mensagens rotacionadas, para viabilizar a decodificação.

A Tabela 2 mostra o funcionamento da BWT codificando a mensagem "ABRACADABRA" para "RDARCAAAABB".

Tabela 2: Funcionamento da BWT.

Entrada	Rotações	Rotações Ordenadas	Saída
	ABRACADABRA	AABRACADAB**R**	
	AABRACADABR	ABRAABRACA**D**	
	RAABRACADAB	ABRACADABR**A**	
	BRAABRACADA	ACADABRAAB**R**	
	ABRAABRACAD	ADABRAABRA**C**	
ABRACADABRA	DABRAABRACA	BRAABRACAD**A**	RDARCAAAABB
	ADABRAABRAC	BRACADABRA**A**	
	CADABRAABRA	CADABRAABR**A**	
	ACADABRAABR	DABRAABRAC**A**	
	RACADABRAAB	RAABRACADA**B**	
	BRACADABRAA	RACADABRAA**B**	

3 MATERIAIS E MÉTODOS

Este capítulo descreve os recursos computacionais, os bancos de dados e os métodos utilizados para desenvolvimento e avaliação empírica do compressor proposto.

3.1 Desenvolvimento

Foi definido que Java seria a linguagem de programação utilizada no desenvolvimento do projeto, devido à portabilidade, robustez, vasta API, suporte ao paradigma de programação orientada a objetos e melhor integração com o SIADM que está sendo desenvolvido em Java. O compressor foi desenvolvido utilizando a versão mais atual (6.0) da linguagem Java.

O compressor foi desenvolvido com atenção constante ao custo computacional. Procurou-se sempre a utilização de técnicas e algoritmos que minimizassem esse fator. Em alguns módulos foi utilizada programação concorrente para melhor aproveitamento dos processadores com múltiplos núcleos que estão se tornando mais populares.

Houve também a preocupação com escalabilidade para que o compressor suporte imagens ainda maiores. Por exemplo, evitou-se a utilização de funções recursivas que pudessem causar “estouro de pilha”, erro que pode ocorrer quando uma função recursiva chama a si mesma muitas vezes.

3.2 Codificador Aritmético Utilizado

Para desenvolvimento do compressor PPM, foi utilizado um codificador aritmético escrito em Java, extraído de outro compressor PPM de código aberto, disponível na

Internet (CARPENTER, 2009). Com isso foi possível aproveitar as várias otimizações de código do codificador aritmético, e ter as atenções voltadas apenas para o desenvolvimento do PPM para a compressão de imagens mamográficas.

Após seu entendimento, o codificador aritmético foi submetido a testes de funcionalidade a fim de confirmar seu funcionamento correto.

3.3 Banco de Dados de Imagens Mamográficas

No presente trabalho, foram utilizadas imagens mamográficas de duas bases de dados: o *Digital Database for Screening Mammography* (DDSM) (HEAT *et al*., 2000) e a Base de Imagens Mamográficas do Laboratório de Análise e Processamento de Imagens Médicas e Odontológicas da Escola de Engenharia de São Carlos / USP (LAPIMO, 2010). Esta base será identificada como Base LAPIMO no presente trabalho.

O DDSM é um banco de imagens que consiste em 2620 casos de exames mamográficos digitalizados com 16 bits por pixel, e tem sido amplamente utilizado como *benchmark* na área de mamografia porque é um banco livre e possui uma grande variedade de casos, compreendendo as imagens digitalizadas e a informação clínica e técnica correspondente, como a data do exame, idade do paciente, *scanner* utilizado na aquisição das imagens, resolução, tipo das lesões (de acordo com a classificação BIRADS® (*Breast Image Reporting and Data System*) (ACR, 2003)) e o contorno das lesões traçadas por um radiologista experiente.

A Base LAPIMO possui 1437 imagens com variação de várias características, tais como resolução espacial, classificação BIRADS®, densidade, tipos de achados mamográficos, aparelho de Raio-X e *Scanner*.

3.4 ImageJ

O ImageJ (2009) é um *software* de domínio público escrito em Java, inspirado no NIH Image para MacOS e voltado para o desenvolvimento de aplicações de processamento e análise de imagens. Por ser escrito em Java, é compatível com todos os sistemas operacionais e arquiteturas que possuam uma JVM (*Java Virtual Machine*).

A Figura 2 mostra a interface gráfica do ImageJ com as várias ferramentas disponíveis e uma imagem mamográfica sendo exibida.

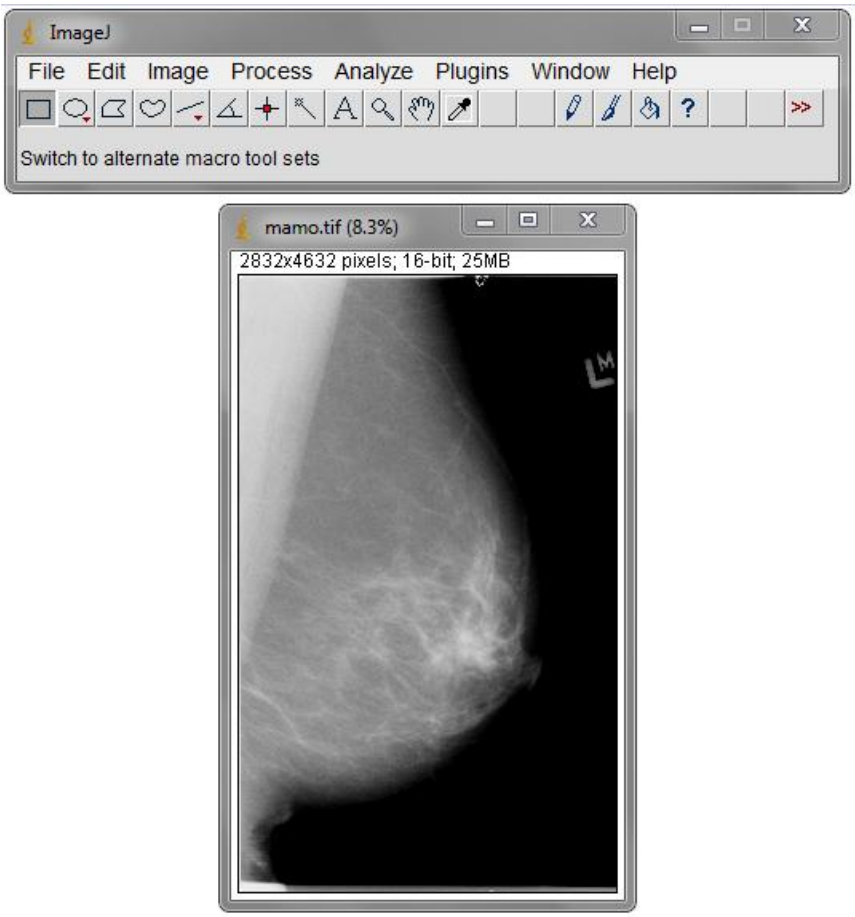

Figura 2: Interface gráfica do ImageJ

O ImageJ foi utilizado para a criação do módulo de descompressão progressiva que será descrito posteriormente e para leitura e gravação das imagens em disco devido ao suporte de vários formatos de imagem, incluindo o formato TIFF, utilizado para armazenagem das imagens do DDSM e da Base LAPIMO.

3.5 Segmentação de *background* e mama

Com o intuito de melhorar a compressão, o *background* (região escura da imagem, em que não aparece a imagem radiográfica da mama) foi separado da região da mama através de segmentação por limiarização. Seja $S(x,y)$ o valor do pixel na posição (x,y) e L um valor de limiar escolhido de acordo com as características do *background* da imagem mamográfica. Os pixels com valor menor ou igual a L são considerados como *background*, e os demais como mama.

O processo de segmentação baseia-se no fato de que o *background*, em geral, é a região mais escura da mamografia, cujos pixels normalmente possuem nível de cinza igual à zero. São criados dois modelos diferentes para o PPM, um para cada região presente na mamografia, o que tende a melhorar a compressão, pois os modelos serão mais específicos e precisos para cada região.

Uma imagem binária adicional, com bits de valor '1' indicando região de mama e de valor '0' indicando região de *background* é anexada ao arquivo comprimido, para permitir a descompressão com os modelos corretos para cada pixel.

No caso das imagens do DDSM, o *background* é completamente preto (nível de cinza zero). Por isso foi escolhido o limitar $L=0$ fazendo com que a região de *background* seja composta apenas pelo nível de cinza zero. Como essa região é composta apenas por zeros, ela não precisa ser comprimida, o que melhora a compressão.

3.6 Limpeza de *background*

Imagens mamográficas geralmente contêm informações relacionadas ao exame no *background* da imagem. Após a digitalização, esses dados podem ser armazenados de outras formas mais eficientes como em um arquivo texto.

A presença desses dados pode introduzir um viés na avaliação empírica de compressores de imagens mamográficas. Assim, essas informações foram removidas

manualmente de forma que a imagem resultante apresentasse um *background* uniforme, como mostra a imagem Figura 3.

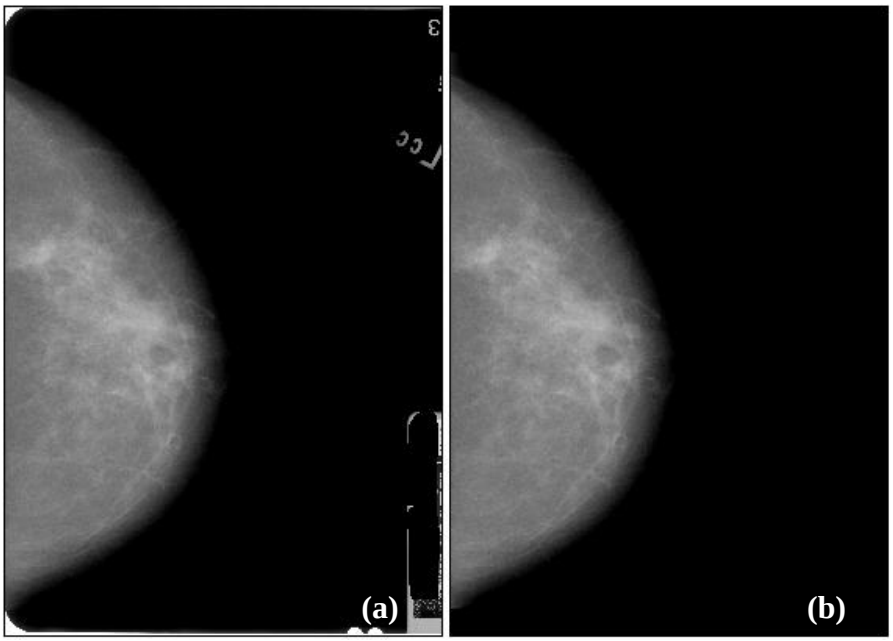

Figura 3: Limpeza de *background*: (a) Imagem original; (b) Imagem limpa

3.7 Decomposição em Planos de Bits

A decomposição em planos de bits decompõe uma imagem ***S*** de ***n*** bits por pixel (ou seja, 2^n níveis de cinza) em ***n*** imagens binárias, ou planos de bits, $S_0, S_1, ..., S_{n-1}$. Um bit $b(x,y)$ no plano S_i equivale ao *i*-ésimo bit do pixel $S(x,y)$ da imagem ***S***, considerando $i=0$ como sendo o bit menos significativo.

A Figura 4 ilustra a decomposição em planos de bits. Nesse exemplo, cada pixel na imagem original possui três bits; assim, a decomposição gera três imagens binárias. O primeiro plano de bits é formado pelos bits mais significativos de cada pixel, e assim sucessivamente.

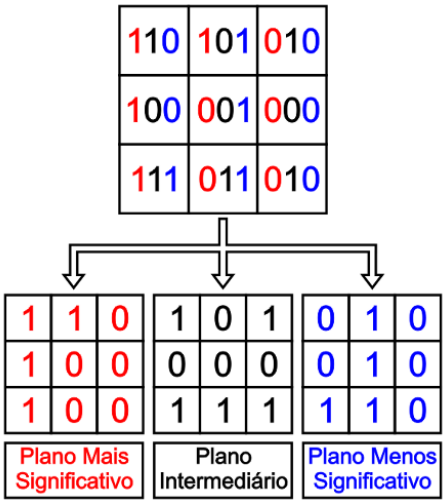

Figura 4: Decomposição em planos de bits

A Figura 5 mostra uma imagem mamográfica e alguns planos de bits.

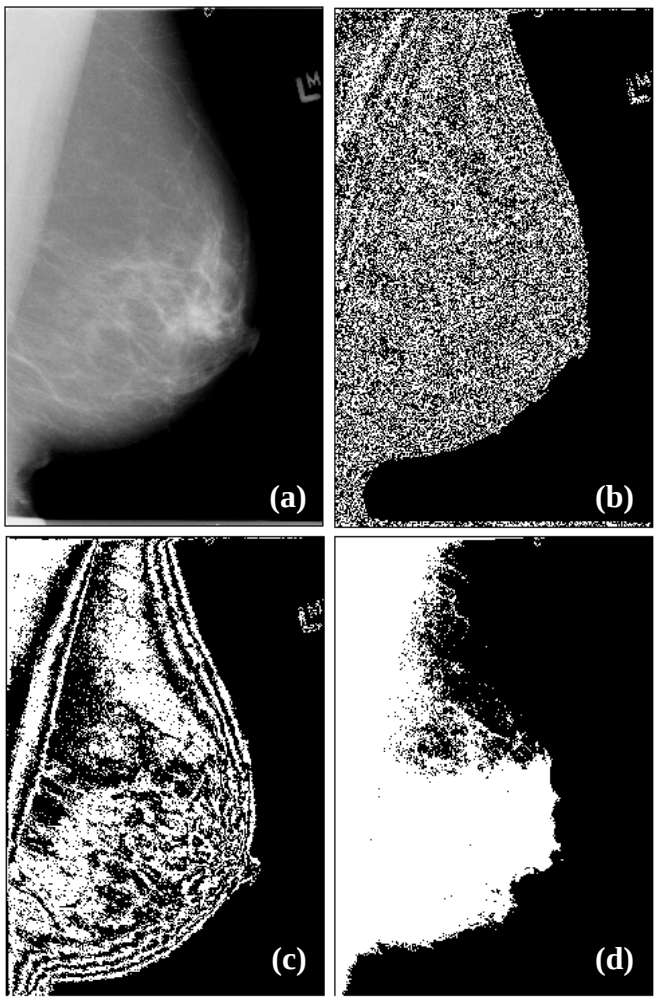

Figura 5: Planos de Bits: (a) Imagem original; (b) Plano S_0; (c) Plano S_8; (d) Plano S_{11}

3.8 Mapeamento

As imagens do DDSM possuem, em média, 2949 níveis de cinza diferentes, codificados em 16 bits de tal forma que assumem valores inteiros entre 0 e 65535. Uma vez que 12 bits seriam suficientes para codificar cada pixel, antes da decomposição em planos de bits os níveis são mapeados (MARQUES *et al*., 2007) para valores inteiros entre 0 e 2948, e representados em 12 bits, gerando 12 planos de bits. O grau de compressão é aferido considerando que as imagens originais têm 12 bits/pixel, e não 16 bits/pixel, uma vez que este último número não corresponde à informação efetivamente disponível nas imagens. Este processo é dinâmico, se adaptando aos níveis de cinza existentes em cada imagem processada.

A decomposição das imagens em planos de bits permite a utilização de um PPM com alfabeto binário, o que reduz drasticamente o custo computacional associado ao algoritmo.

3.9 Código Gray

Código Gray é um tipo de representação binária. A definição de Código Gray determina que a representação de dois números sucessivos deva se diferenciar por exatamente um bit.

A vantagem da utilização do Código Gray está diretamente ligada à utilização de planos de bits em compressão de sinais. Geralmente, em sinais digitais extraídos de sistemas físicos, como eletrocardiograma, imagens de raios-X e ultrassonografia, os valores de duas amostras vizinhas não variam muito devido à natureza contínua dos sinais originais.

Na compressão por plano de bits, quanto menor a variação dos bits em cada plano, melhor a compressão obtida (MARQUES *et al*., 2007). Com a utilização do Código Gray buscou-se reduzir consideravelmente essa variação dos bits. Como pode ser visto na

Tabela 3, na variação do valor 3 para o valor 4, com o Código Binário convencional, houve a alteração de todos os bits enquanto que no Código Gray apenas um bit foi alterado.

Tabela 3: Código decimal, binário e Gray

Decimal	Binário	Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

A Figura 6 mostra como a utilização de Código Gray reduz a variação dos bits em cada plano. Esse comportamento é mais evidente nos planos mais significativos. Nos planos menos significativos que possuem uma aparência muito ruidosa o Código Gray não surte muito efeito.

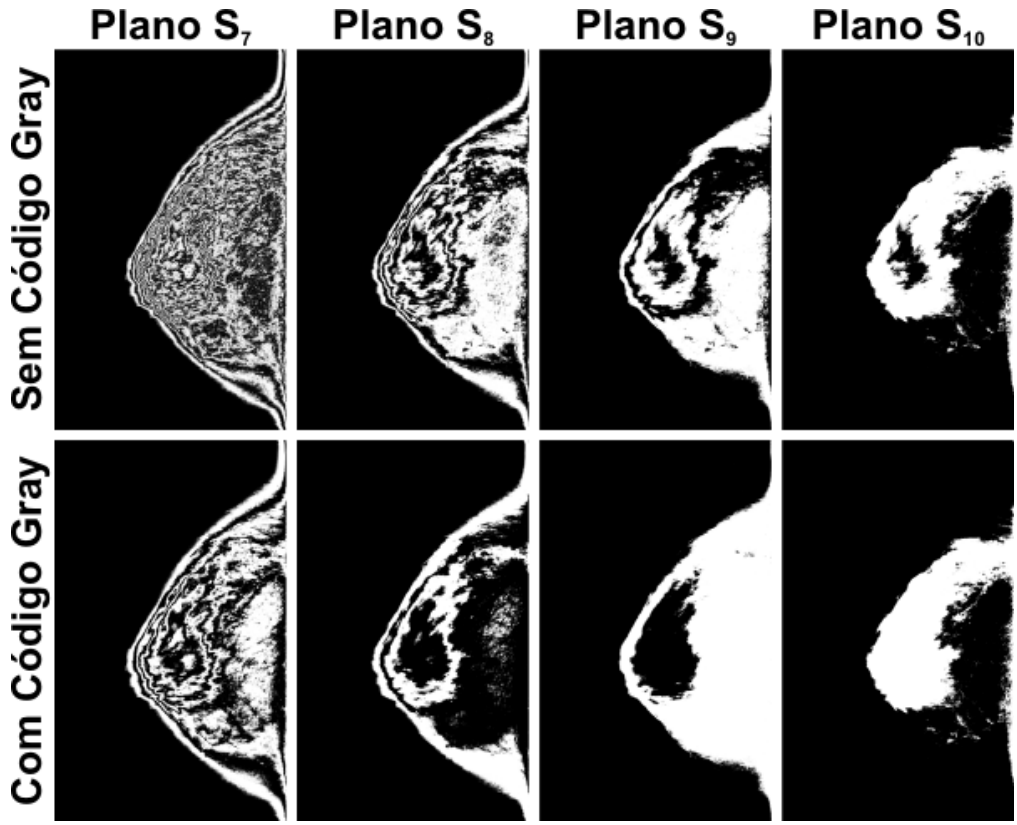

Figura 6: Planos de bits sem e com Código Gray

3.10 PPM Binário

Uma desvantagem do PPM com alfabetos grandes é o custo excessivo para armazenar os modelos em memória, como também o processo demorado de busca dos símbolos no modelo.

Como solução foi proposto um PPM Binário onde cada contexto indexa diretamente a posição de um vetor onde são armazenados os contadores do modelo. Para exemplificar, utilizando um alfabeto binário, as probabilidades condicionais estimadas para o contexto 0110_2 são armazenadas na posição $0110_2 = 6_{10}$ do vetor. Com a indexação direta, a busca por um modelo específico representa uma economia enorme de tempo de processamento (SOUZA *et al*., 2007).

Outra vantagem dessa abordagem é a economia de memória, não sendo mais necessário manter os contextos na memória, já que cada contexto é implicitamente identificado pelos índices do vetor de modelos.

Com a utilização de um alfabeto binário, foi eliminada a utilização de escape. O símbolo de escape indica um símbolo desconhecido em um determinado contexto, um símbolo que não ocorreu. No caso binário, como só há dois símbolos, se o símbolo existente é zero, o escape logicamente representa o símbolo um, e vice-versa. Por esse motivo a utilização do escape torna-se desnecessária com alfabetos binários.

Juntamente à eliminação da utilização do escape, quando um modelo é criado, é assumido que todos os símbolos já ocorreram uma vez. Essa modificação reduz o tempo de processamento já que um símbolo é sempre comprimido utilizando seu contexto completo, dispensando a utilização de contextos menores. O que ocorre no algoritmo tradicional do PPM quando é detectado que um símbolo nunca ocorreu em um determinado contexto.

Um vetor referente a um contexto com $\boldsymbol{K}$ símbolos binários possui $\boldsymbol{2^K}$ elementos. Como o PPM utiliza vários tamanhos de contexto, são necessários vetores de tamanhos diferentes, um para cada tamanho de contexto. Um vetor foi criado de forma a armazenar sequencialmente os vetores de todos os tamanhos de contexto necessários ao PPM. Cada posição desses vetores contem um inteiro que representa um contador do símbolo (zero ou um).

Apesar de não haver a mudança para contextos menores ao comprimir um símbolo, esses contextos menores ainda são necessários para comprimir os primeiros símbolos do

início de alguma região que não possuem símbolos precedentes suficientes para encher o contexto completamente.

A Figura 7 mostra um exemplo do modelo estatístico do PPM binário desenvolvido. Para a utilização de um contexto de tamanho K=2 é necessário criar três tabelas referentes aos diferentes tamanhos de contexto possíveis (K=2, K=1 e K=0). No caso da Figura 7, o contexto é 10_2 e o símbolo atual é 0_2, que juntos formam 100_2 correspondendo à posição 4_{10} da tabela K=2 (posição destacada em vermelho). Após a compressão do símbolo, o valor da tabela na posição 4_{10} é incrementada para o valor 3 significando que o símbolo 0_2 no contexto 10_2 ocorreu 3 vezes.

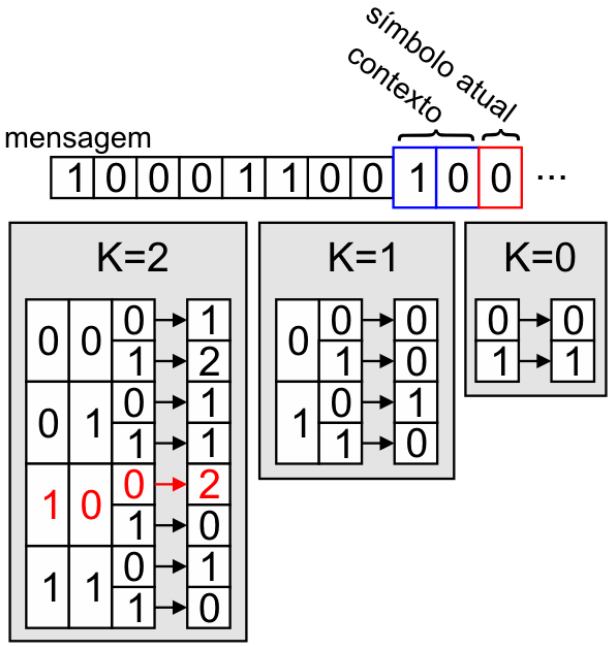

Figura 7: Modelo estatístico do PPM binário.

3.10.1 Pseudocódigo

Uma das vantagens do PPM binário desenvolvido, é a sua simplicidade, que se deve principalmente ao fato de utilizar um vetor de contadores no lugar de estruturas de dados complexas como árvores (comumente utilizadas na implementação de um PPM não binário).

Para implementar o modelo binário, é necessário armazenar quatro variáveis:

- ***C***: é um inteiro que armazenará os símbolos que já ocorreram em seus bits menos significativos. O número de bits desse inteiro define o tamanho máximo de contexto que pode ser utilizado. Por exemplo, com um inteiro de 32 bits é possível utilizar contextos de tamanho até ***K*=32**;
- ***N***: guarda o número de símbolos (bits) que estão atualmente contidos em ***C***;
- ***CONT***: é um vetor que contem os contadores de todos os símbolos de todos os contextos possíveis. No início da compressão, todos os elementos desse vetor possuem o valor 1;
- ***I***: é um vetor que contem em cada elemento na posição ***k***, o índice onde se inicia os contadores do contexto de tamanho ***K*=*k*** no vetor ***CONT***. Ou seja, guarda a posição inicial de cada tamanho de contexto de ***K*** até **0**.

A Figura 8 mostra o pseudocódigo da função **CRIAR_MODELO** que, a partir do valor do ***K***, prepara as variáveis necessárias do modelo.

```
CRIAR_MODELO ( )
        I ← vetor[ K+1 ]
        de i=0 até K faça
                I[ i ] ← (1 << (i+1)) – 2
        CONT ← vetor[ (1 << (K+2)) - 2 ]
        de i=0 até tamanho(CONT)-1 faça
                CONT[ i ] ← 1
```

Figura 8: Função CRIAR_MODELO

onde:

- ***vetor[x]*** cria um vetor de inteiros com ***x*** elementos;
- ***x* << *y*** é a operação de deslocamento de bits à esquerda. Quando ***x*=1** tem se **1 << *y***, que é equivalente à potencia $\mathbf{2^y}$. Quando ***y*=1** tem se ***x* << 1**, que é equivalente à multiplicação **2*x***. A operação de deslocamento de bits é utilizada porque é muito mais rápida que as operações tradicionais de potenciação na base 2 e multiplicação por 2;
- ***tamanho(x)*** retorna o número de elementos do vetor ***x***.

A Figura 9 mostra o pseudocódigo da função **CALCULAR_INDICE** que calcula a posição, no vetor ***CONT***, onde começam os contadores do contexto atual ***C***.

```
CALCULAR_INDICE ( )
        K_atual ← MÍNIMO(K, N)
        mascara ← (1 << K_atual) - 1
        índice ← I[ K_atual ] + ((C bitand mascara) << 1)
        retorna índice
```

Figura 9: Função CALCULAR_INDICE

onde:

- **MÍNIMO(*x*, *y*)** retorna o menor valor entre ***x*** e ***y***. Essa função é utilizada para garantir que não se use um ***K*** maior do que o número de símbolos que estão atualmente no contexto ***C***;
- ***x bitand y*** é a operação lógica ***and*** feita bit a bit entre os inteiros ***x*** e ***y***. Por exemplo: a operação $\mathbf{0011_2}$ ***bitand*** $\mathbf{0101_2}$ resulta em $\mathbf{0001_2}$.

A Figura 10 mostra o pseudocódigo da função **CONSULTAR_MODELO** que , a partir do contexto atual ***C***, retorna um vetor com três elementos com as informações necessárias para calcular o subintervalo (valores *low*, *high* e *total*) de um símbolo ***S*** nesse contexto.

```
CONSULTAR_MODELO ( )
        índice ← CALCULAR_INDICE ( )
        cont0 ← CONT[ índice ]
        cont1 ← CONT[ índice+1 ]
        total ← cont0 + cont1
        retorna {0, cont0, total}
```

Figura 10: Função CONSULTAR_MODELO

onde:

- ***índice*** é a posição, no vetor ***CONT***, onde começam os contadores do contexto ***C***. O primeiro é o contador do símbolo 0, e o seguinte (***índice*+1**) é o contador do símbolo 1;

- ***cont0*** e ***cont1*** são, respectivamente, os contadores do símbolo 0 e do símbolo 1 no contexto ***C***;
- ***total*** é o tamanho total do intervalo, portanto, a soma dos contadores dos dois símbolos.

A Figura 11 mostra a função **ATUALIZAR_MODELO** que atualiza as estatísticas do modelo incrementando o contador do símbolo ***S*** no seu contexto ***C***.

```
ATUALIZAR_MODELO ( S )
        índice ← CALCULAR_INDICE ( )
        CONT[ índice+S] ++
```

Figura 11: Função ATUALIZAR_MODELO

onde:

- ***índice+S*** é a posição do contador do símbolo ***S*** no vetor ***CONT***. Se o símbolo for 0, então é usado o próprio ***índice***, e caso o símbolo seja 1, é usado ***índice+1***. Esta operação poderia ser feita utilizando um comando condicional (se/senão), mas a operação de soma é mais rápida;
- **x++** é a operação de incremento que aumenta o valor de ***x*** em 1 unidade.

A Figura 12 mostra o pseudocódigo da função **ATUALIZAR_CONTEXTO** que adiciona o símbolo ***S*** ao contexto ***C***.

```
ATUALIZAR_CONTEXTO ( S )
        C ← (C << 1) bitor S
        N ← mínimo(N+1, K)
```

Figura 12: Função ATUALIZAR_CONTEXTO

onde, ***bitor*** é a operação lógica ***and*** feita bit a bit similar à operação ***bitand***.

A Figura 13 mostra o pseudocódigo da função **COMPRIMIR** que utiliza o modelo binário e o aritmético para comprimir o símbolo ***S***.

```
COMPRIMIR ( S )
        tabela ← CONSULTAR_MODELO ( )
        low ← tabela[ S ]
        high ← tabela[ S+1 ]
        total ← tabela[ 2 ]
        aritmético.CODIFICAR( low, high, total )
        ATUALIZAR_MODELO ( S )
        ATUALIZAR_CONTEXTO ( S )
```

Figura 13: Função COMPRIMIR

onde, a função **CODIFICAR** é uma função do codificador aritmético que utiliza o subintervalo do símbolo especificado pelo valores *low*, *high* e *total* para codificar o símbolo.

Na função **COMPRIMIR**, o cálculo dos valores de *low*, *high* e *total* é feito a partir da tabela para otimizar o código, sendo mais rápido que o uso de um comando condicional.

A Figura 14 mostra o pseudocódigo da função **DESCOMPRIMIR** que retorna o símbolo ***S*** descomprimido através do uso do modelo binário do aritmético.

```
DESCOMPRIMIR ( )
        tabela ← CONSULTAR_MODELO ( )
        valor ← aritmético.CONTADOR_ATUAL( tabela[ 2 ] )
        S ← 0
        se S valor >= tabela[ 1 ] então
                S = 1
        low ← tabela[ S ]
        high ← tabela[ S+1 ]
        total ← tabela[ 2 ]
        aritmético.DECODIFICAR( low, high, total )
        ATUALIZAR_MODELO ( S )
        ATUALIZAR_CONTEXTO ( S )
        retorna S
```

Figura 14: Função DESCOMPRIMIR

onde:

- **CONTADOR_ATUAL** é uma função do decodificador aritmético que, a partir do tamanho do intervalo atual (armazenado em ***tabela[2]***) passado como parâmetro,

retorna um valor qualquer dentro do intervalo referente ao subintervalo do símbolo que deve ser descomprimido. Esse valor deve ser comparado com o contador do símbolo 0 (armazenado em ***tabela[1]***) para saber qual o símbolo deve ser descomprimido (0 ou 1).

- **DECODIFICAR** é uma função do decodificador aritmético que utiliza o subintervalo (*low*, *high* e *total*) do símbolo para efetivamente decodificar o símbolo e passar para o próximo símbolo.

Todas essas funções foram desenvolvidas com o objetivo de minimizar o número de instruções executadas pelo processador a cada símbolo processado. Além de reduzir o número de instruções, procurou-se utilizar instruções mais rápidas, como o deslocamento de bits no lugar de potenciação na base dois.

Pequenas melhorias nessas funções podem significar grandes reduções no tempo de processamento do sistema completo, já que essas funções podem ser executadas milhões de vezes durante a compressão de uma única imagem.

Essa simplificação do algoritmo através da utilização de uma estrutura de dados simples como um vetor aliada à redução drástica dos requisitos computacionais, permitiu a implementação do PPM diretamente em *hardware* (SOUZA, 2007) (até então inviável devido à complexidade das estruturas de dados utilizadas e ao alto custo computacional). Essa implementação abre várias possibilidades para a utilização do compressor, como por exemplo, em sistemas embarcados.

3.11 Descarte de contexto

A utilização de contexto no PPM procura explorar a relação estatística que existe entre os símbolos anteriores e o símbolo atual. No caso de imagens vindas de sinais analógicos, esta relação existe porque o contexto está espacialmente próximo do símbolo atual. Contextos muito distantes não possuem relação com o símbolo atual e por isso não devem ser utilizados.

Em uma imagem (bidimensional), o processo de compressão ocorre, geralmente, percorrendo-se os pixels da esquerda para a direita em cada linha. Ao chegar ao final de uma linha, pula-se para o primeiro pixel da linha abaixo. Este processo de varredura permite que um pixel do início de uma linha tenha como contexto um pixel do final da linha superior, ou seja, o símbolo e seu contexto possuem uma distância espacial muito grande.

Para evitar a criação de contextos com símbolos distantes devido à varredura da imagem, o contexto atual é descartado ao mudar de uma linha para outra, assim, todos os pixels que são o início de uma linha não possuem contexto. O mesmo é feito nas mudanças de região de *background* para mama ou vice-versa, evitando que um pixel do inicio de uma região de mama tenha como contexto pixels do final de outra região de mama distante.

3.12 Detecção de mudança de região

Na codificação aritmética, o número de bits utilizados na codificação do símbolo depende da sua probabilidade de ocorrência. Quanto maior a probabilidade de ocorrência de um símbolo, menor o número de bits utilizados na sua codificação, e vice-versa. Por exemplo, pela Equação 2, um símbolo com probabilidade de 0,25 será codificado com 2 bits, já um símbolo com probabilidade de 0,75 será codificado com aproximadamente 0,42 bits.

No caso do PPM binário desenvolvido (sem a utilização de escape), só existem dois símbolos: “0” e “1”. Assim, ganhos de compressão ocorrem na codificação de um símbolo com probabilidade maior que 50%, caso contrário haverá perda de compressão ou nenhuma compressão se a probabilidade do símbolo for exatamente 50%. Por exemplo, se um símbolo tiver probabilidade de 0,1%, ele será codificado utilizando aproximadamente 10 bits, ou seja, um símbolo que era representado com 1 bit, será representado com 10 bits.

A codificação de símbolos pouco prováveis ocorre principalmente em regiões nos planos de bits onde há mudança de valor (de 0 para 1 ou vice-versa) após uma sequência longa de bits de mesmo valor.

A Figura 15(a) mostra uma área de um plano de bits de uma imagem mamográfica onde há mudança de região de 1 (branco) para 0 (preto). Essa mudança de região, após a transformada MTF, é exibida na Figura 15(b). O que ocorre nessa situação é que o bit 0 (preto) acaba ficando com uma probabilidade muito alta por ter ocorrido muitas vezes. Nos pixels da transição, ocorre um bit 1 (branco) que possui uma probabilidade muito baixa e, portanto, é codificado com um número alto de bits. Além de prejudicar a compressão, esse bit 1 na transição entre regiões também reduz a probabilidade de ocorrência do bit 0 prejudicando sua compressão. Este comportamento se repete por toda a imagem onde há transição entre regiões dessa forma.

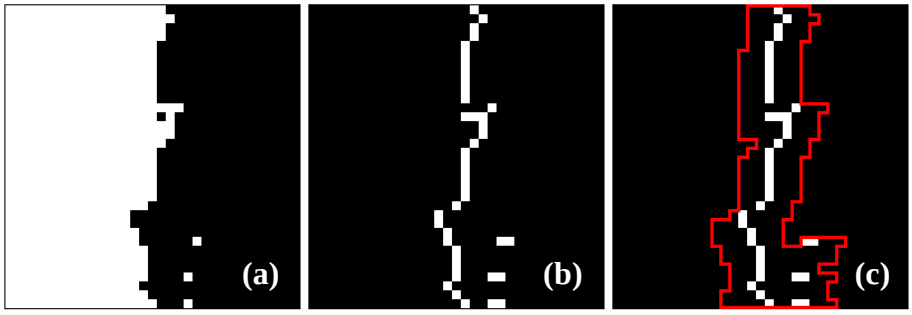

Figura 15: Detecção de mudança de região: (a) porção de um plano de bits com mudança de região; (b) mesma porção depois da MTF; (c) região marcada como mudança de região.

Para reduzir o efeito dos pixels de transição foi desenvolvido um método de detecção de mudança de região.

Quando ocorre um pixel de mudança de região, o método marca seus pixels vizinhos dentro de certo raio como sendo também pixels de mudança de região. São considerados pixels vizinhos apenas os pixels que ainda não foram comprimidos, ou seja, os pixels à direita na mesma linha, ou nas linhas abaixo. Esse método é baseado na disposição dos pixels de mudança de região que estão geralmente próximos uns dos outros formando uma fronteira entre duas regiões diferentes. Assim, quando se detecta um pixel de mudança de região, provavelmente haverá outro próximo a ele como pode ser visto na Figura 15(b). Dessa forma, vários pixels próximos aos pixels que realmente são de transição são marcados também. A Figura 15(c) mostra, dentro da região demarcada em vermelho, os pixels que foram marcados como pertencentes à mudança de região utilizando raio=3.

Para que os pixels marcados como pertencentes à mudança de região não interferissem no modelo estatístico dos demais pixels, foi criado um modelo estatístico separado para eles. Então, quando se usa o método de detecção de mudança de região, o compressor cria dois modelos estatísticos, um para os pixels de mudança de região e outro para os demais pixels.

O resultado desse método é um aumento da compressão tanto nos pixels de mudança de região quanto nos demais. As distribuições de probabilidades dos dois tipos de pixels não se misturam, gerando modelos estatísticos mais específicos para cada tipo de pixel.

A Figura 16(a) mostra em níveis de cinza a informação associada a cada pixel na mesma região da Figura 15(b). Neste caso, quanto mais claro um pixel, maior a informação a ele associada e, portanto, menor a compressão. A Figura 16(b), de forma similar, mostra em níveis de cinza a informação quando se utiliza a detecção de mudança de região com raio=3. Nota-se que muitos pixels da Figura 16(b) são mais escuros que os pixels correspondentes na Figura 16(a), o que indica uma quantidade menor de informação e, consequentemente, uma melhor compressão.

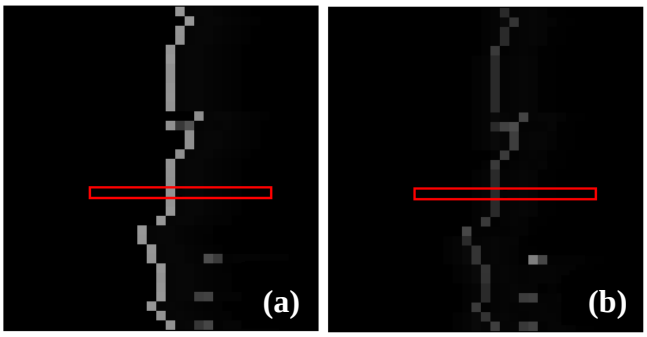

Figura 16: Efeito da detecção de mudança de região: área sem (a) e com (b) detecção de mudança de região.

Para exemplificar o efeito do método de detecção de mudança de região, foram selecionadas duas faixas de pixels (demarcadas em vermelho) na Figura 16(a) e Figura 16(b) e calculada a soma da informação presente em cada faixa. A faixa de pixels sem a utilização do método de detecção de mudança de região soma 8,354 bits de informação, enquanto que a faixa com a utilização do método de detecção de mudança de região soma apenas 3,419 bits. Este ganho de compressão ocorre por toda a imagem onde houver mudança de região.

3.13 Planos contaminados por ruído

Os planos de bits menos significativos das imagens mamográficas não podem ser comprimidos, independentemente do tamanho de contexto utilizado, o que indica, pela Teoria da Informação, que estão muito fortemente contaminados por ruído (MARQUES, 2008). De fato, pode-se afirmar que a ausência de compressão é um forte indício de que os planos contêm, praticamente, apenas ruído.

Baseado no fato de que os planos menos significativos não podem ser comprimidos, esses planos menos significativos são copiados diretamente para o arquivo comprimido sem passar pelo PPM. Isso não influi na compressão, mas diminui o tempo de processamento porque reduz o número de planos que serão processados pelo PPM.

É adicionada ao cabeçalho do arquivo comprimido a especificação de quais planos foram comprimidos e quais foram apenas copiados.

3.14 Compressor Desenvolvido

O compressor PPM desenvolvido é dividido em módulos independentes para possibilitar a inclusão, alteração e remoção de módulos, o que facilita a manutenção e evolução do compressor.

Tanto as etapas de pré-processamento (segmentação, mapeamento, conversão para Código Gray e transformada MTF) como as de compressão (PPM e Codificador Aritmético) foram tratadas como módulos.

A Figura 17 mostra a arquitetura básica do compressor com o fluxo de compressão à esquerda e o fluxo de descompressão à direita.

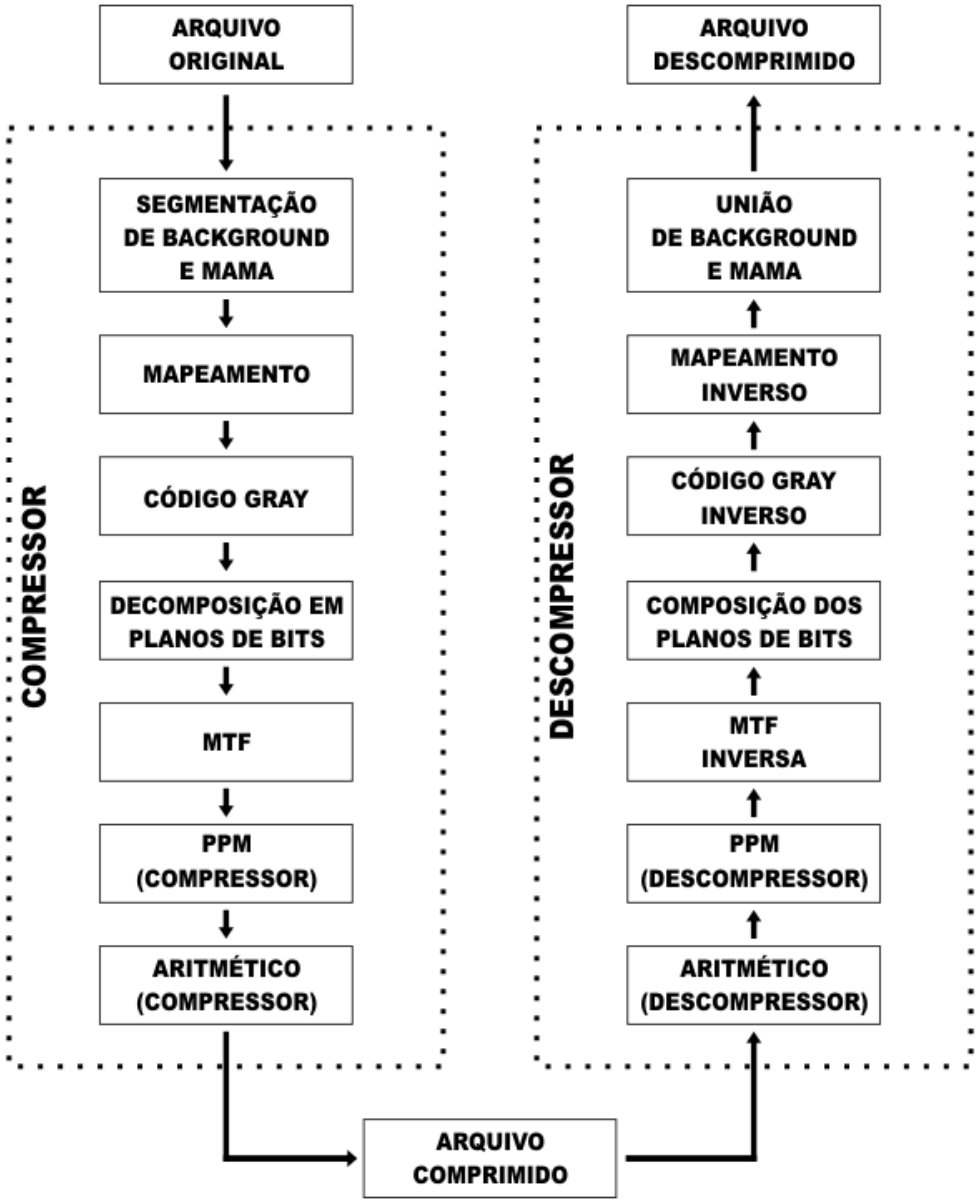

Figura 17: Arquitetura do compressor

Cada módulo se liga ao compressor através de uma interface bem definida. Dessa forma é possível a criar várias versões de um módulo e alternar entre eles de forma muito simples.

3.15 Retrocompatibilidade de Arquivo

Uma característica importante de um compressor comercial é a retrocompatibilidade de arquivo, que é a possibilidade de criar novas versões do compressor sem perder a capacidade de descomprimir arquivos que foram comprimidos por versões anteriores.

A retrocompatibilidade de arquivo foi implementada utilizando versionamento de arquivo. Os primeiros quatro bytes de um arquivo comprimido formam um número correspondente à versão do compressor utilizado. Durante a descompressão, esse número de versão é lido e utilizado para escolher a versão correta do descompressor.

3.16 Descompressão Progressiva *Lossy-to-Lossless*

Descompressão progressiva *lossy-to-lossless* é uma técnica que possibilita que parte da informação de uma imagem possa ser exibida à medida que ela vai sendo descomprimida, ou seja, parte da imagem pode ser visualizada antes da descompressão terminar. Essa técnica é importante para um sistema de recuperação de imagens mamográficas porque possibilita que o usuário possa ver parte da imagem que está recebendo antes do *download* terminar, podendo continuar o *download* ou pará-lo no meio da transmissão.

A descompressão progressiva é possível porque os planos de bits são comprimidos um de cada vez, do plano mais significativo para o menos significativo. Então, durante a descompressão, os planos mais significativos que são descomprimidos primeiro são exibidos para o usuário e à medida que novos planos são descomprimidos, eles são compostos para formar uma imagem cada vez mais próxima da original.

3.17 Processamento *multithread*

Um dos objetivos desse trabalho é que a compressão e descompressão sejam executadas em tempos aceitáveis para uma aplicação real.

Além da utilização de algoritmos otimizados, a utilização de multiprocessamento pode reduzir o tempo de processamento de uma aplicação, principalmente com os processadores multinúcleo (*multicore*) atuais que executam várias tarefas ao mesmo tempo.

No compressor desenvolvido há poucos lugares onde o multiprocessamento pode ser utilizado devido à natureza sequencial de execução dos módulos. É possível utilizar multiprocessamento dentro dos módulos de pré-processamento, mas não haveria um ganho significativo de desempenho porque cada um desses módulos já é executado em um tempo reduzido, em comparação com os módulos de compressão propriamente ditos.

Os módulos mais promissores para utilizar multiprocessamento são os de compressão (PPM e Codificador Aritmético) que são os que consomem mais tempo de processamento. Assim, aproveitando que cada plano de bits é comprimido independente dos demais, no caso de imagens de 12 bits por pixel, são utilizados 12 conjuntos de módulos de compressão (PPM + Codificador Aritmético) executando em paralelo. Os resultados da compressão de cada um dos 12 planos de bits são armazenados em memória para que, posteriormente, quando terminar a compressão de todos os planos, sejam concatenados no arquivo final. Um processo similar é utilizado na descompressão.

O mecanismo de multiprocessamento foi desenvolvido utilizando a classe Thread do Java da seguinte forma:

1. A *thread* principal cria as 12 *threads*, uma para cada plano de bits;
2. Todas as 12 *threads* são iniciadas através do método *Thread.start()*, cada *thread* instancia um módulo PPM e Codificador Aritmético para comprimir um plano de bits;
3. A *thread* principal espera que todas as 12 *threads* sejam finalizadas utilizando o método *Thread.join()*.
4. Após todas as 12 *threads* serem finalizadas, a *thread* principal assume e continua o processamento.

3.18 Módulos descartados

Nesta sessão serão descritos alguns dos módulos desenvolvidos para o compressor que foram descartados porque não resultaram em ganhos de compressão ou aumentaram muito o custo computacional.

3.18.1 PPM bidimensional

O PPM é basicamente um compressor voltado para mensagens unidimensionais como arquivos de texto ou qualquer outro sinal unidimensional. Ao ser usado para a compressão de imagens, em geral, o PPM codifica linha por linha da imagem descartando a relação bidimensional entre os pixels.

Foi desenvolvido um PPM bidimensional que leva em conta a relação bidimensional entre os pixels. Os pixels vizinhos são classificados pela distância em relação ao pixel que se quer codificar e ordenados do pixel mais próximo ao mais distante. Na ordenação pela distância, empates são resolvidos por qualquer regra sistemática.

Por exemplo, a ordem dos pixels do contexto bidimensional utilizando um tamanho de contexto K=14 é mostrada na Figura 18 onde os valores menores correspondem aos pixels mais próximos do pixel que se está comprimindo (posição 0) e os pixels marcados com um X não podem participar do contexto porque, pela ordem de varredura, ainda não foram comprimidos.

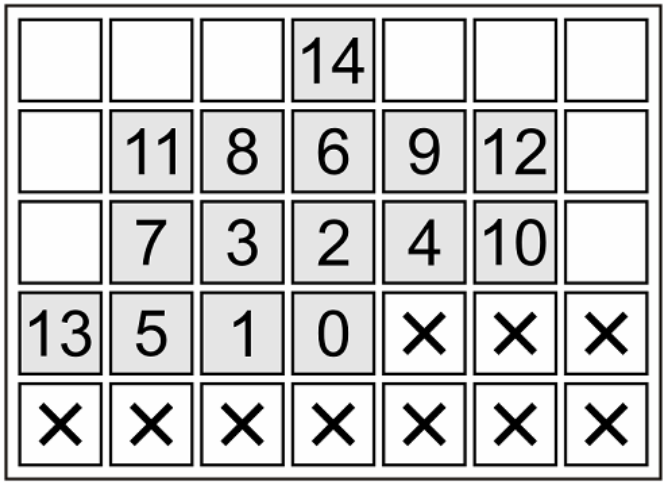

Figura 18: PPM bidimensional

O PPM bidimensional não obteve bons resultados se mostrando inferior ao PPM comum unidimensional. Isto se deve ao fato de os contextos bidimensionais raramente se repetirem, o que impede uma estimativa precisa das probabilidades.

3.18.2 PPM binário com contexto não binário

O PPM binário desenvolvido comprime um plano de bits por vez, e cada plano é comprimido de forma independente dos demais. Uma característica dessa abordagem é que na utilização de contextos maiores, pixels espacialmente distantes do pixel atual são utilizados na sua compressão.

Uma alternativa é a criação de um PPM binário que utilize informações de alguns planos para comprimir cada bit, ou seja, um PPM binário com contexto não binário. Esse novo compressor preserva até certo ponto o baixo custo computacional de um PPM binário comum. Assim, foi desenvolvido um PPM binário com contexto não binário que utiliza alguns dos planos mais significativos que o plano que está sendo comprimido como contexto do bit atual. O número de planos mais significativos utilizados será representado pela variável *M*, e o tamanho do contexto (*K*) representa o número de pixels anteriores ao pixel atual que serão utilizados como contexto.

A Figura 19 mostra o bit atual que está sendo comprimido (bit do plano S_1 do 4° pixel da imagem) marcado em azul, e o seu contexto marcado em vermelho. Neste caso foi utilizado K=2 e M=3.

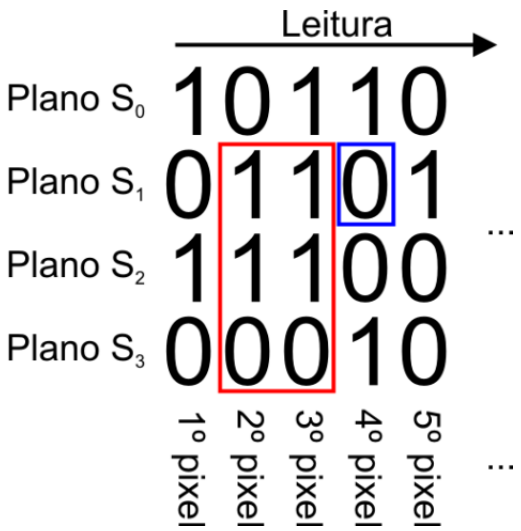

Figura 19: PPM binário com contexto não binário.

Cada vez que o compressor avança para comprimir o próximo bit de um plano, M novos bits são adicionados ao contexto e outros M bits são removidos.

Para comparação desse compressor com um PPM binário comum deve-se levar em conta quantos bits são utilizados como contexto. Assim, um PPM binário comum que utiliza contexto K=6 deve ser comparado com um PPM binário com contexto não binário que utilize alguma combinação possível de K e M que satisfaça a condição $K \times M = 6$ (exemplos: K=2 e M=3 ou K=1 e M=6).

Deve-se salientar que a utilização de M=1 é equivalente à utilização de um PPM binário comum, pois apenas os bits do plano atual serão utilizados como contexto do bit atual.

Esta abordagem obteve resultados próximos aos do PPM binário comum, mas foi descartada porque possuía um tempo de processamento um pouco maior e impossibilitava a descompressão progressiva e a utilização de multiprocessamento durante a descompressão.

3.18.3 Transformada Burrows–Wheeler (BWT)

O problema da BWT está na ordenação que é muito custosa quando aplicada a mensagens muito grandes como imagens mamográficas.

Foram testadas algumas variações da BWT e suas combinações: aplicar a transformada na imagem completa ou apenas em cada plano de bits isoladamente e quebrar a imagem em blocos menores ou considerar a imagem como um bloco só.

Os melhores resultados de compressão eram obtidos aplicando a BWT na imagem completa ordenando como um único bloco, mas essa abordagem era muito custosa, algumas vezes demorando até horas para processar uma única imagem. A solução foi fazer a ordenação em blocos menores, o que reduziu o custo computacional, mas também reduziu a compressão.

Os testes mostraram que a utilização da BWT em blocos, a única variação que apresentava um tempo de processamento aceitável, diminuía levemente a compressão alem de aumentar em alguns segundos o tempo de processamento. Por esse motivo a BWT foi descartada.

3.18.4 Utilização de modelo estatístico previamente criado

O PPM é um algoritmo de modelagem estatística adaptativa, ou seja, o modelo estatístico é criado à medida que a mensagem é comprimida.

Foram feitos testes utilizando modelos previamente criados a partir dos próprios arquivos, ou seja, cada arquivo era comprimido uma vez para gerar seu modelo estatístico, e outra utilizando este modelo criado. O ganho de compressão obtido foi mínimo, mas o processo era custoso porque precisava comprimir cada arquivo duas vezes e por isso foi descartado.

Esses testes mostraram que o PPM “aprende” muito rapidamente o comportamento da mensagem e que, portanto, não precisa utilizar um modelo previamente criado.

RESULTADOS E DISCUSSÃO

Este capítulo descreve os resultados obtidos nos testes.

Os testes foram realizados em um computador com a seguinte configuração: processador Pentium® Dual-Core T4200 2GHz, 3GB de memória RAM DDR2, disco SATA 5400rpm, sistema operacional Microsoft Windows 7 Professional com Máquina Virtual Java versão 1.6 Update 17.

Não foi possível a comparação direta com outros trabalhos da literatura, pois para realizar testes consistentes é necessária a utilização das mesmas imagens e não foram encontrados trabalhos que utilizassem imagens do DDSM ou da Base LAPIMO e especificassem quais.

Testes preliminares mostraram melhores resultados de compressão com a utilização de contexto de tamanho K=7 e raio=3 na detecção de mudança de região. Nos testes seguintes, quando não estiver especificado, foram utilizados esses valores como parâmetros, nenhum plano considerado como ruidoso, os módulos de pré-processamento (segmentação, mapeamento, Código Gray e transformada MTF) e multiprocessamento. Apesar de fixados para os testes seguintes, esses parâmetros podem ser escolhidos a cada imagem comprimida.

4.1 PPM binário por Planos de Bits

Uma compressão baseada em PPM é, geralmente, um processo com alto custo computacional envolvido. Porém, o sistema desenvolvido de armazenagem dos modelos binários em um vetor estático apresentou um baixo custo computacional comprimindo imagens mamográficas de 27MB em aproximadamente 20 segundos (13 segundos se considerar os quatro planos menos significativos como ruidosos) e consumindo em torno de 250MB de memória RAM. A maior parte da memória utilizada é destinada ao armazenamento do arquivo inteiro em memória e às informações necessárias aos módulos

de pré-processamento. Os módulos de compressão ocupam pouca memória. Dado um tamanho de contexto *K*, a quantidade de memória necessária para armazenar os modelos estatísticos é dada pela Equação 4. Assim, para um contexto de tamanho *K*=9 serão necessários apenas 8KB de memória RAM para cada módulo PPM instanciado.

$$M = \left(2^{K+2} - 2\right) \times 4 \quad \text{Bytes} \tag{4}$$

4.2 Imagens de teste

Os testes descritos nas seções seguintes foram realizados utilizando 30 imagens do DDSM e 20 imagens da Base LAPIMO, todas manualmente processadas para limpar as informações desnecessárias contidas no *background*. Os resultados mostrados são a média dos resultados obtidos com cada uma das imagens.

A Tabela 4 e a Tabela 5 mostram, respectivamente, as imagens utilizadas da Base LAPIMO e do DDSM.

Tabela 4: Imagens da Base LAPIMO utilizadas.

Imagens da Base LAPIMO utilizadas	
105358_314291000CCD1.tiff	105358_314291000CCE3.tiff
310001_256311000CCD1.tiff	310002_257251000CCD1.tiff
310002_257251000CCE3.tiff	310003_258301000CCD1.tiff
310003_258301000CCE2.tiff	02980170101CCD1.tiff
02980170101CCE3.tiff	04101240101CCD1.tiff
04101240101CCE3.tiff	04168240101CCD1.tiff
04168240101CCE3.tiff	04346250101CCD1.tiff
04346250101CCE3.tiff	3011100CCE2.tiff
3240400CCD1.tiff	3240400CCE2.tiff
3300100CCD1.tiff	3300100CCE2.tiff

Tabela 5: Imagens do DDSM utilizadas.

Imagens do DDSM utilizadas	
A_1122_1.RIGHT_CC.LJPEG.1.tiff	A_1207_1.RIGHT_CC.LJPEG.1.tiff
A_1212_1.RIGHT_CC.LJPEG.1.tiff	A_1222_1.RIGHT_CC.LJPEG.1.tiff
A_1237_1.RIGHT_CC.LJPEG.1.tiff	A_1262_1.LEFT_CC.LJPEG.1.tiff
A_1510_1.LEFT_CC.LJPEG.1.tiff	A_1510_1.RIGHT_CC.LJPEG.1.tiff
A_1592_1.LEFT_CC.LJPEG.1.tiff	A_1592_1.RIGHT_CC.LJPEG.1.tiff
A_1790_1.LEFT_CC.LJPEG.1.tiff	A_1827_1.LEFT_CC.LJPEG.1.tiff
A_1985_1.RIGHT_CC.LJPEG.1.tiff	B_3046_1.LEFT_CC.LJPEG.1.tiff
B_3046_1.RIGHT_CC.LJPEG.1.tiff	B_3055_1.LEFT_CC.LJPEG.1.tiff
B_3055_1.RIGHT_CC.LJPEG.1.tiff	B_3055_1.RIGHT_MLO.LJPEG.1.tiff
B_3058_1.LEFT_CC.LJPEG.1.tiff	B_3058_1.RIGHT_CC.LJPEG.1.tiff
B_3062_1.LEFT_MLO.LJPEG.1.tiff	B_3062_1.RIGHT_MLO.LJPEG.1.tiff
B_3072_1.LEFT_CC.LJPEG.1.tiff	B_3072_1.RIGHT_CC.LJPEG.1.tiff
B_3073_1.RIGHT_CC.LJPEG.1.tiff	B_3076_1.LEFT_MLO.LJPEG.1.tiff
B_3076_1.RIGHT_MLO.LJPEG.1.tiff	B_3079_1.LEFT_CC.LJPEG.1.tiff
B_3079_1.RIGHT_CC.LJPEG.1.tiff	B_3080_1.RIGHT_CC.LJPEG.1.tiff

4.3 Multiprocessamento

A utilização de multiprocessamento ajudou a diminuir o tempo médio de processamento de 30s para 20s, uma redução de 33%.

Os testes foram feitos em um computador com processador de dois núcleos. Provavelmente a utilização de processadores com mais núcleos (que estão se popularizando) irá reduzir ainda mais o tempo de processamento.

4.4 Tamanho do contexto

A Figura 20 mostra as compressões obtidas com as imagens do DDSM em função do tamanho do contexto.

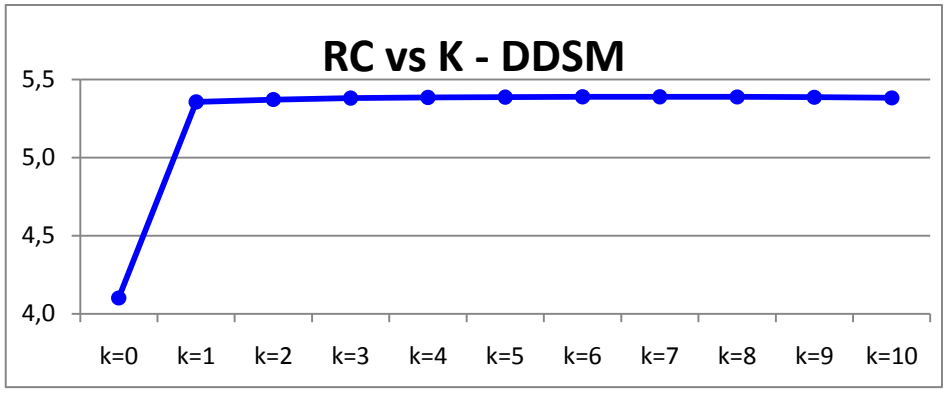

Figura 20: RC para imagens do DDSM em função do tamanho do contexto.

Os melhores resultados com as imagens do DDSM foram obtidos utilizando tamanho de contexto *K*=7 resultando em uma *RC* de 5,39.

A Figura 21 mostra as compressões obtidas com as imagens da Base LAPIMO em função do tamanho do contexto.

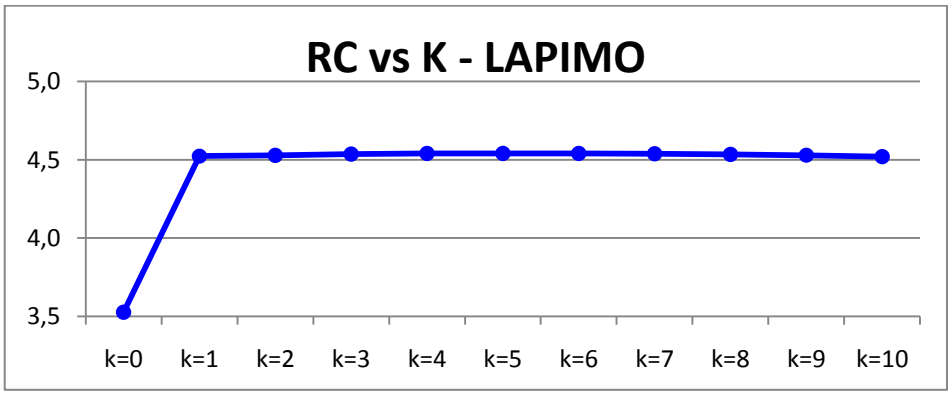

Figura 21: RC para imagens do LAPIMO em função do tamanho do contexto.

Os melhores resultados com as imagens da Base LAPIMO foram obtidos utilizando tamanho de contexto *K*=5 resultando em uma *RC* de 4,540. Já utilizando *K*=7 (tamanho do contexto fixado no restante dos testes) a *RC* resultante é 4,538, uma diferença insignificante em comparação com o melhor caso.

É importante ressaltar que com a utilização de contextos de tamanho *K*=1 a *RC* obtida é bem próxima da *RC* máxima que pode ser obtida, permitindo a utilização de contextos menores em sistemas com recursos computacionais reduzidos sem prejudicar muito a compressão.

A partir de certo tamanho, *K*=7 para as imagens do DDSM e *K*=5 para as da Base LAPIMO, o aumento do contexto apenas piora a compressão porque os símbolos do contexto passam a não ter muita relação com o pixel que está sendo comprimido, prejudicando a compressão.

A Figura 22 mostra os tempos de processamento necessários para compressão das imagens do DDSM, em função do tamanho do contexto.

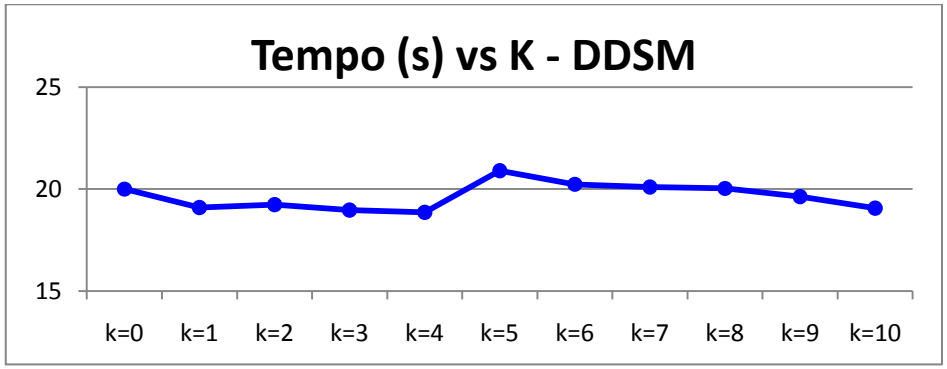

Figura 22: Tempo de processamento para imagens do DDSM em função do tamanho do contexto.

A Figura 23 mostra os tempos de processamento com as imagens da Base LAPIMO, em função do tamanho do contexto.

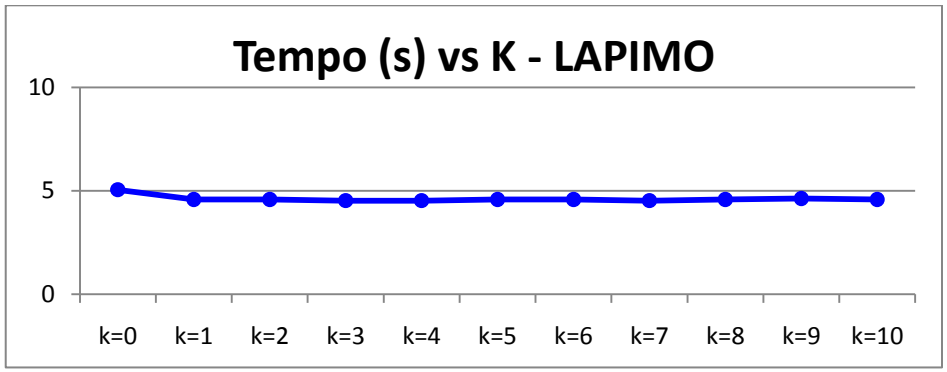

Figura 23: Tempo de processamento para imagens da Base LAPIMO em função do tamanho do contexto.

Os tempos de processamento das imagens do DDSM e da Base LAPIMO são bem diferentes. Isso se deve principalmente ao tamanho das imagens escolhidas dos dois bancos. O tamanho médio das imagens escolhidas do DDSM é 27MB, e do LAPIMO é 6MB. Essa diferença no tamanho das imagens pode explicar a diferença de compressão das imagens do DDSM e da Base LAPIMO. Como as imagens do DDSM possuem maior resolução, a redundância é maior, aumentando as chances de compressão.

O tempo de processamento manteve-se praticamente constante com a variação do tamanho do contexto. Pequenas variações de tempo devem ser desprezadas porque podem ser causadas pela execução de outros processos pelo Sistema Operacional.

4.5 Planos de bits

A Tabela 6 mostra a compressão obtida em cada plano de bits das imagens do DDSM em função do tamanho do contexto. A melhor compressão foi alcançada utilizando K=7 resultando em uma RC de 5,389. Os valores iguais na tabela são resultado do arredondamento para reduzir o número de casas decimais.

Nota-se a ausência de compressão (RC=1) nos cinco planos de bits menos significativos para todos os valores de K utilizados, o que demonstra, segundo a Teoria da Informação, que esses planos estão fortemente contaminados por ruído (MARQUES *et al*., 2007) e, consequentemente, podem ser removidos (MARQUES *et al*., 2008) podendo até melhorar o diagnóstico médico (KALLERGI *et al*., 2006).

A Tabela 7 mostra a compressão obtida em cada plano de bits das imagens da Base LAPIMO em função do tamanho do contexto. A melhor compressão foi alcançada utilizando K=5 resultando em uma RC de 4,54. Neste caso, apenas os quatro planos de bits menos significativos não puderam ser comprimidos (RC=1), o que indica que as imagens escolhidas da Base LAPIMO possuem menos ruído que as do DDSM.

Tabela 6: RC das imagens do DDSM em cada plano de bits em função do tamanho do contexto.

	***K*=0**	***K*=1**	***K*=2**	***K*=3**	***K*=4**	***K*=5**	***K*=6**	***K*=7**	***K*=8**	***K*=9**	***K*=10**
Plano 0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
Plano 1	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
Plano 2	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
Plano 3	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
Plano 4	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
Plano 5	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2	1,2
Plano 6	1,8	1,8	1,8	1,8	1,8	1,8	1,8	1,8	1,8	1,8	1,8
Plano 7	3,3	3,3	3,3	3,4	3,4	3,4	3,4	3,4	3,4	3,4	3,4
Plano 8	6,4	6,5	6,5	6,6	6,6	6,6	6,6	6,6	6,6	6,6	6,6
Plano 9	12,7	13,0	13,1	13,2	13,2	13,2	13,2	13,2	13,2	13,2	13,1
Plano 10	30,7	33,0	33,3	33,6	33,8	33,9	33,9	33,9	33,8	33,7	33,5
Plano 11	101,0	106,7	107,7	109,1	110,2	110,5	110,4	110,2	109,6	108,8	108,0
Total	**4,101**	**5,356**	**5,371**	**5,380**	**5,385**	**5,387**	**5,388**	**5,389**	**5,388**	**5,386**	**5,382**

Tabela 7: RC das imagens da Base LAPIMO em cada plano de bits em função do tamanho do contexto.

	***K*=0**	***K*=1**	***K*=2**	***K*=3**	***K*=4**	***K*=5**	***K*=6**	***K*=7**	***K*=8**	***K*=9**	***K*=10**
Plano 0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
Plano 1	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
Plano 2	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
Plano 3	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
Plano 4	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1	1,1
Plano 5	1,4	1,4	1,4	1,4	1,4	1,4	1,4	1,4	1,4	1,4	1,4
Plano 6	2,2	2,2	2,2	2,3	2,3	2,3	2,3	2,3	2,3	2,3	2,2
Plano 7	4,1	4,1	4,1	4,1	4,2	4,2	4,2	4,2	4,2	4,2	4,1
Plano 8	7,8	7,9	8,0	8,1	8,1	8,2	8,2	8,1	8,1	8,1	8,1
Plano 9	15,6	15,9	16,1	16,3	16,5	16,5	16,5	16,5	16,4	16,4	16,3
Plano 10	47,4	50,8	51,3	53,0	53,9	53,8	53,7	53,5	53,1	52,7	52,2
Plano 11	155,0	162,7	159,6	160,1	159,4	158,6	156,1	152,2	149,9	146,9	144,8
Total	**3,527**	**4,523**	**4,528**	**4,535**	**4,540**	**4,540**	**4,540**	**4,538**	**4,534**	**4,528**	**4,519**

4.6 Limpeza de *background*

A Figura 24 mostra a razão de compressão para as imagens do DDSM em função do tamanho do contexto antes e depois da limpeza do *background*.

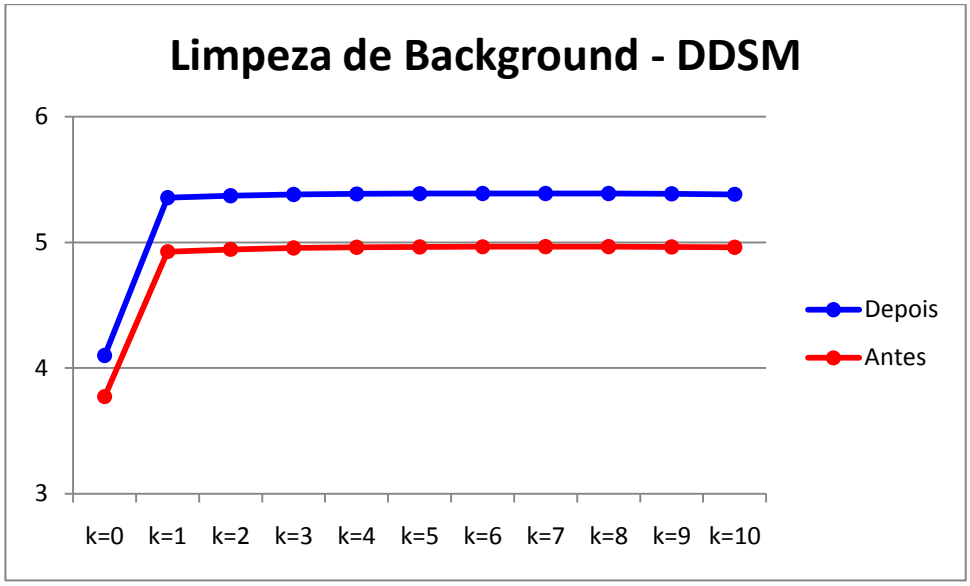

Figura 24: Limpeza de background - DDSM

A limpeza de *background* melhora a compressão para todos os valores de *K* utilizados. Utilizando *K*=7, a *RC* aumenta de 4,97 para 5,39 com a limpeza de *background*, o que representa um ganho de 8,5%.

4.7 Detecção de mudança de região

A Figura 25 mostra as compressões obtidas com as imagens do DDSM em função do raio da detecção de mudança de região.

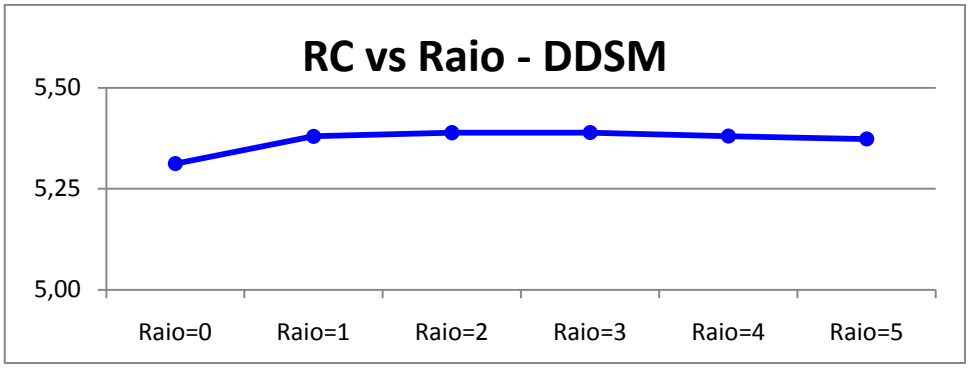

Figura 25: RC para imagens do DDSM em função do raio da detecção de mudança de região.

A Figura 26 mostra as compressões obtidas com as imagens do LAPIMO em função do raio da detecção de mudança de região.

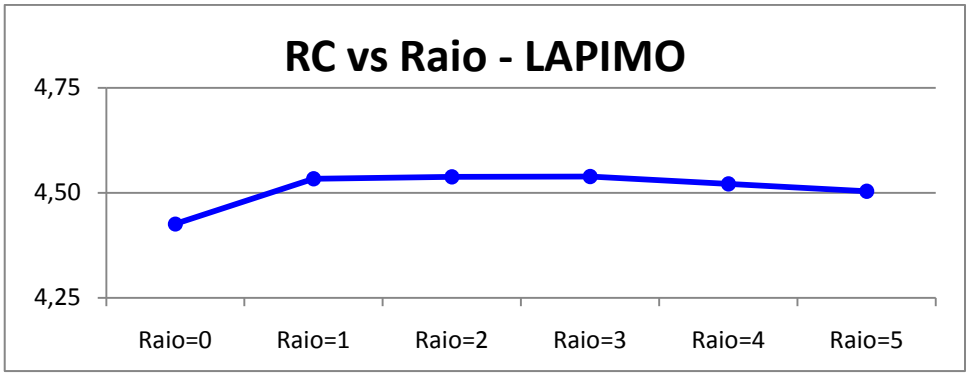

Figura 26: RC para imagens da Base LAPIMO em função do raio da detecção de mudança de região.

Os melhores resultados, tanto para as imagens do DDSM quanto para as do LAPIMO, foram obtidos com a utilização de raio=3. A utilização da detecção de mudança de região com raio=3 nas imagens do DDSM aumentou a *RC* de 5,31 para 5,39, um ganho de 1,5%. Já nas imagens do LAPIMO, a *RC* aumentou de 4,43 para 4,54, um ganho de 2,5%.

4.8 Planos ruidosos

A Figura 27 mostra as compressões obtidas com as imagens do DDSM em função do número de planos considerados ruidosos. Como citado anteriormente, os planos menos significativos considerados ruidosos são copiados diretamente para o arquivo comprimido sem compressão.

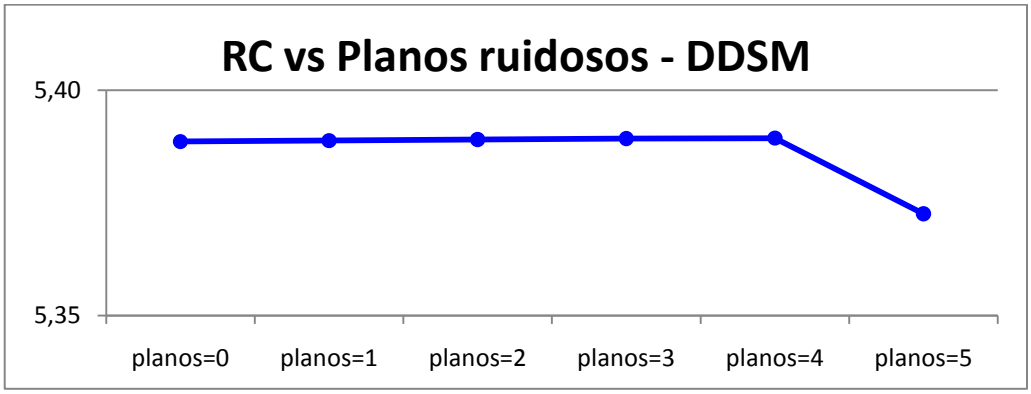

Figura 27: RC para imagens do DDSM em função do número de planos considerados ruidosos.

A Figura 28 mostra as compressões obtidas com as imagens do LAPIMO variando o número de planos considerados ruidosos.

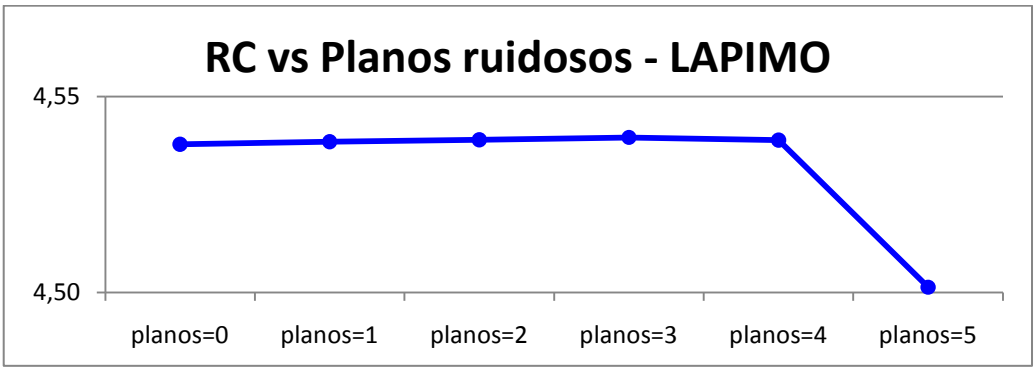

Figura 28: RC para imagens da Base LAPIMO em função do número de planos considerados ruidosos.

Tanto as imagens do DDSM quanto as do LAPIMO apresentam o mesmo comportamento em relação ao número de planos que são considerados ruidosos. Mesmo sem comprimir os quatro planos menos significativos, apenas os copiando para o arquivo

comprimido, a *RC* se mantém praticamente constante, só diminui quando o quinto plano não é comprimido.

A Figura 29 mostra os tempos de processamento obtidos com as imagens do DDSM em função do número de planos considerados ruidosos.

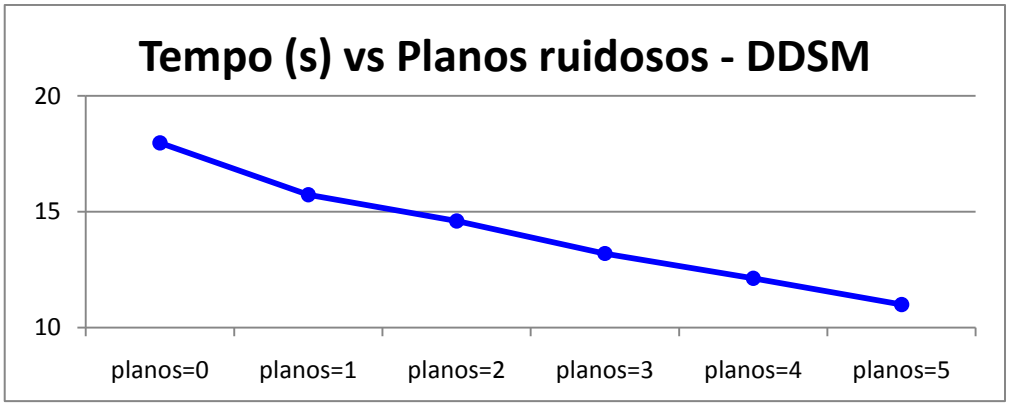

Figura 29: Tempo de processamento para imagens do DDSM em função do número de planos considerados ruidosos.

A Figura 30 mostra os tempos de processamento obtidos com as imagens da Base LAPIMO em função do número de planos considerados ruidosos.

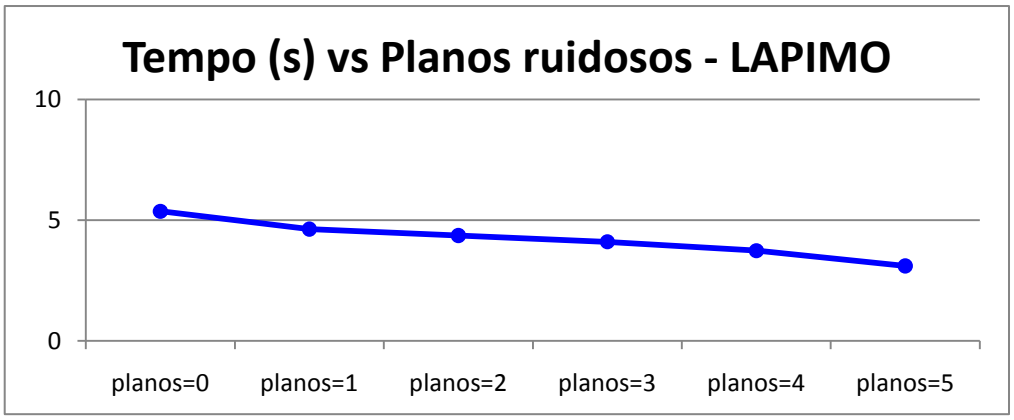

Figura 30: Tempo de processamento para imagens da Base LAPIMO variando o número de planos com ruído

Tanto nas imagens do DDSM quanto nas da Base LAPIMO houve redução no tempo de processamento por fazer com que alguns planos considerados ruidosos não passem pelo PPM.

4.9 Módulos de pré-processamento

A Tabela 8 e a Tabela 9 mostram as compressões e tempos de processamento variando os módulos de pré-processamento (segmentação, mapeamento, Código Gray e transformada MTF) utilizados para as imagens do DDSM e da Base LAPIMO, respectivamente.

Tabela 8: Módulos de pré-processamento para imagens do DDSM

Módulos	*RC*	Tempo (s)
Nenhum	3,012	26
Segmentação	3,043	32
Segmentação + Mapeamento	4,714	15
Segmentação + Mapeamento + Gray	5,285	15
Segmentação + Mapeamento + Gray + MTF	5,389	18

Tabela 9: Módulos de pré-processamento para imagens da Base LAPIMO

Módulos	*RC*	Tempo (s)
Nenhum	3,865	6
Segmentação	3,895	7
Segmentação + Mapeamento	3,899	4
Segmentação + Mapeamento + Gray	4,412	4
Segmentação + Mapeamento + Gray + MTF	4,537	4

A segmentação aumentou a *RC* em 1,1% para as imagens do DDSM e em 0,8% para as imagens da Base LAPIMO.

O mapeamento aumentou a *RC* em 54,9% para as imagens do DDSM e apenas 0,1% para as imagens da Base LAPIMO. O ganho de compressão é pequeno com as imagens da Base LAPIMO porque elas já ocupam apenas 12 planos de bits.

A codificação com Código Gray aumentou a *RC* em 12,1% para as imagens do DDSM e 13,1% para as imagens da Base LAPIMO.

A transformada MTF aumentou a *RC* em 2.0% para as imagens do DDSM e 2,8% para as imagens da Base LAPIMO.

No total, a junção de todos os módulos de pré-processamento aumentou a *RC* em 78,9% reduzindo 31,2% do tempo de processamento para as imagens do DDSM e aumentou a *RC* em 17,4% reduzindo 27,4% do tempo de processamento para as imagens da Base LAPIMO.

Os módulos de pré-processamento proporcionaram ganhos de compressão maiores para as imagens do DDSM, provavelmente porque, como já explicado, essas imagens possuem maior redundância de informação a ser explorada.

4.10 Descompressão Progressiva *Lossy-to-Lossless*

O módulo visual de descompressão progressiva *lossy-to-lossless* foi desenvolvido como um *plugin* do ImageJ. O usuário escolhe um arquivo a ser descomprimido, então é mostrada ao usuário uma barra de progresso e uma janela com a imagem que está sendo descomprimida. À medida que o arquivo é descomprimido, a barra de progresso avança e a imagem vai se revelando progressivamente. Quanto mais a descompressão avança, mais detalhes a imagem descomprimida possui, possibilitando ao usuário ter uma ideia da imagem que está descomprimindo antes mesmo da descompressão terminar.

A Figura 31 mostra a evolução no tempo da barra de progresso e da visualização da imagem durante a descompressão progressiva.

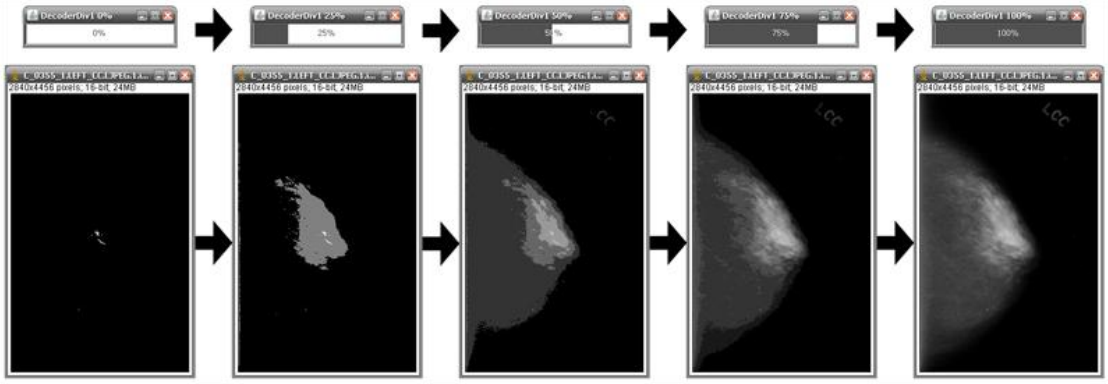

Figura 31: Descompressão progressiva *lossy-to-lossless*

É interessante ressaltar que, em uma aplicação que envolva transmissão das imagens por enlaces de telecomunicações, os planos mais significativos são recebidos muito mais rapidamente que os outros, porque esses planos são mais comprimidos. Como pode ser visto na Tabela 7, o plano mais significativo das imagens da Base LAPIMO são comprimidos com uma *RC* média de 162,7, para *K*=1.

4.11 Comparação com outros compressores sem perdas

Foram realizados testes para verificar a eficiência do compressor desenvolvido em relação a outros compressores sem perdas. Os testes foram feitos utilizando as opções de máxima compressão de cada compressor quando havia opção de configuração.

A Tabela 10 mostra que o compressor desenvolvido neste trabalho superou os outros compressores se mostrando bastante eficiente na compressão de mamografias. É importante destacar que o compressor que mais se aproximou do compressor desenvolvido foi o WinZip 14 onde uma das configurações utiliza uma versão do PPM, o que demonstra a qualidade do algoritmo utilizado nesse trabalho.

É importante destacar que os compressores comerciais utilizados são de propósito genérico e o compressor desenvolvido é voltado apenas para imagens mamográficas.

Tabela 10: Comparação com outros compressores comerciais.

Compressor	Configuração	*RC*	Tempo (s)
Compressor desenvolvido	...	5,389	13
WinZIP 14	Melhor Método	5,375	7
WinZIP 14	Máximo (PPMd)	5,265	7
7Zip 4.65	Ultra	3,948	8
JPEG 2000	Lossless	3,310	6
WinRAR 3.7 (RAR)	Ótimo	3,179	7
PNG 16 bits	-	3,023	4
WinRAR 3.7 (ZIP)	Ótimo	2,637	2

A configuração do compressor para comparação com os compressores comerciais foi: tamanho de contexto *K*=7, detecção de mudança de região com raio=3, 4 planos ruidosos, segmentação, mapeamento, Código Gray e MTF.

O compressor desenvolvido apresentou uma *RC* pouco superior à do segundo colocado, mas com quase o dobro do tempo de processamento. Apesar do tempo de processamento mais alto que os demais, o compressor desenvolvido possui a vantagem de permitir a descompressão progressiva e a análise do ruído presente na mamografia (características inexistentes no WinZip 14 e no 7Zip), ser multiplataforma, permitindo sua utilização em qualquer dispositivo que possua uma máquina virtual Java, além de boa parte do compressor poder ser implementado em hardware, o que é inviável para o PPM padrão.

Um dos prováveis motivos para a diferença de tempo de processamento entre o compressor desenvolvido e o WinZip é que o compressor foi desenvolvido em Java, rodando, portanto, sobre uma máquina virtual, enquanto que o WinZip roda nativamente e possui módulos muito otimizados escritos diretamente em linguagem de máquina (Assembly).

4 CONCLUSÕES

Neste trabalho foram apresentados a pesquisa e o desenvolvimento de um sistema de alto desempenho para compressão sem perdas de imagens mamográficas baseado no algoritmo PPM, juntamente com etapas de segmentação, mapeamento, codificação com Código Gray e transformada MTF.

O PPM Binário desenvolvido permitiu a redução drástica dos requisitos computacionais do PPM que é conhecido como um algoritmo de custo elevado.

A segmentação das imagens em mama e *background* permitiu uma compressão melhor da região de mama e resultou em um pequeno ganho na compressão total. O módulo de mapeamento permitiu um grande ganho de compressão com as imagens do DDSM reduzindo o número de planos de bits de 16 para 12.

A codificação em Código Gray e a aplicação da transformada MTF também permitiram ganhos de compressão ao reduzir a variação dos bits em cada plano de bits. No caso da MTF, além de reduzir a variação dos bits, também aumenta a ocorrência de bits 0 e, consequentemente, aumentando sua compressão.

A divisão do compressor em módulos independentes simplificou o processo de remoção e inserção de novos módulos, além da alteração dos existentes. Essa organização, juntamente com a retrocompatibilidade de arquivo, facilita a evolução do sistema sem perder a compatibilidade com arquivos comprimidos por versões anteriores.

A descompressão progressiva permite a visualização parcial das imagens enquanto são descomprimidas, característica muito útil em telemedicina.

A cópia direta dos quatro planos menos significativos para o arquivo comprimido possibilitou uma grande redução do tempo de processamento sem afetar a *RC*.

O sistema desenvolvido mostrou um alto desempenho, tanto em compressão quanto em tempo de processamento, comprimindo imagens mamográficas de 27MB em aproximadamente 13 segundos consumindo em torno de 250MB de memória RAM com razão de compressão média de 5,39.

Na comparação com outros compressores sem perdas, o sistema desenvolvido se mostrou como a melhor opção em razão de compressão, pouco superior ao segundo colocado, mas com um tempo de processamento bem maior, ainda que aceitável para uma

aplicação real. O tempo de processamento maior do compressor desenvolvido se deve, provavelmente, ao fato de ter sido desenvolvido em Java, com o programa sendo executado sobre uma máquina virtual, enquanto que os outros compressores são compilados para executar diretamente sobre o Sistema Operacional, podendo até ter módulos muito otimizados, escritos diretamente em linguagem de máquina.

Apesar da desvantagem do tempo de processamento, o compressor desenvolvido tem as vantagens de ser multiplataforma, permitir a descompressão progressiva e possibilitar a análise do ruído presente nas imagens.

Uma extensão direta do compressor desenvolvido seria a compressão com perdas possivelmente através da eliminação de alguns planos de bits menos significativos, ou com a utilização de técnicas de introdução de perdas que não comprometam o diagnóstico médico.

Pode ser feito um estudo comparando com outras técnicas a estimativa de ruído dada pelo compressor desenvolvido.

O PPM também pode ser utilizado para classificação de texturas (HONÓRIO *et al*., 2009). No caso de mamografias, o PPM pode ser utilizado para classificação de densidade mamária, uma característica útil no diagnóstico médico.

Para tentar reduzir o tempo de processamento, alguns módulos podem ser convertidos para linguagem C/C++ sendo, então, compilados e acessados pelo Java através da biblioteca JNA (*Java Native Access*) que permite o acesso a programas compilados em outras linguagens. A desvantagem dessa conversão para código compilado é que diminuiria a portabilidade do compressor já que seria necessário compilar o código em C/C++ para cada sistema operacional onde se pretende utilizar o compressor.

REFERÊNCIAS

ACR. American College of Radiology. Breast Imaging Reporting and Data System® (BI-RADS®). Atlas. 5th Edition. Reston, VA, 2003.

ALMEIDA, C. W. D.; SILVA, F. P. R.; BATISTA, L. V.; BRASIL, L. M. Desenvolvimento de um Sistema de Auxílio ao Diagnóstico para Mamografia Digital. Proceedings of the International Federation for Medical and Biological Engineering (IFMBE Proceedings), p. 1419-1422, 2004.

ANDERSON, B.O. et al. Overview of Breast Health Care Guidelines for Countries with Limited Resources. Breast J. , 9 (Suppl. 2):S42-S50, 2003.

ASTLEY, S.; GILBERT, F. Computer-aided Detection in Mammography. Clinical Radiology. 59:390-399, 2004.

BATISTA, L. V. Compressão de Sinais Eletrocardiográficos Baseada na Transformada Cosseno Discreta. Tese de doutorado, Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal de Campina Grande, 2002.

BELL, T. C.; CLEARY, J. G.; and WITTEN, I. H. Data compression using adaptive coding and partial string matching. IEEE Transactions on Communications, v. 32, n. 4, pp. 396-402, 1984.

BELL, T. C.; CLEARY, J. G.; and WITTEN, I. H. Text Compression, Englewood Cliffs: Prentice Hall, 1990.

BENTLEY, J. L.; SLEATOR, D. D; TARJAN, R. E.; WEI, V. K. A Locally Adaptative Data Compression Scheme. Communications of the ACM – Vol. 29, No. 4, 1986.

BURROWS, M.; WHEELER, D. J.; A Block-Sorting Lossless Data Compression Algorithm. Digital SRC Research Report, 1994.

CARPENTER, B. Compression via Arithmetic Coding in Java *http://www.colloquial.com/ArithmeticCoding/ - Último acesso em março de 2009.*

ESCARPINATI, M. C.; SCHIABEL, H. Avaliação de técnicas de compressão sem perdas aplicadas a imagens mamográficas digitais de mamas densas. XVIII Congresso Brasileiro de Engenharia Biomédica, São José dos Campos (SP). Anais do XVIII Congresso Brasileiro de Engenharia Biomédica. São José dos Campos (SP), v. 5. p. 195-198, 2002.

HEAT, M., BOWYER, K., KOPANS, D., MOORE, R., KEGELMEYER, P. The digital database for screening mammography. Digital Mammography – Proceedings of the 5th International Workshop on Digital Mammography (IWDM2000), Yaffe, M. J. (Ed.), Toronto, pp. 212-218, 2000.

HONÓRIO, T. C. S. ; BATISTA, L. V. ; DUARTE, R. C. M. Texture Classification Using Prediction by Partial Matching Models. Workshop de Visão Computacional (WVC), 2009, São Paulo. 5º Workshop de Visão Computacional (WVC 2009), 2009.

HUFFMAN, D. A. A Method for the Construction of Minimum Redundancy Codes. Proceedings of the Institute of Radio Engineers, n. 40, p. 1098-1101, 1952.

IMAGEJ - http://rsbweb.nih.gov/ij/ - Último acesso em março de 2009.

INCA. Estimativa 2010: incidência de câncer no Brasil. Instituto Nacional do Câncer. INCA, 2009.

KALLERGI, M.; LUCIER, B. J.; BERMAN, C. G.; HERSH, M. R.; KIM, J. J.; SZABUNIO, M. S.; CLARK, R. A. High-Performance Wavelet Compression for Mammography: Localization Response Operating Characteristic Evaluation. Radiology, v. 238, pp. 62-73, 2006.

LAPIMO. Base de Imagens Mamográficas LAPIMO EESC/USP. http://lapimo.sel.eesc.usp.br/bancoweb/ - Último acesso em março de 2010.

MARQUES, J. R. T.; HONORIO, T. C. S.; POEL, J. V. D.; SOUZA, A. R. C. ; BATISTA, L. V. . Compressão sem Perdas de Imagens Mamográficas Utilizando Código Gray e Algoritmo PPM. IFMBE Proceedings, v. 1, p. 1216-1220, 2007.

MARQUES, J. R. T.; POEL, J.K. van der; BATISTA, L. V.; PEREIRA, G. C. G. Mammographic Image Noisy Bit Planes Identification and Removal Using a Binary PPM

Algorithm and an Open Source Architecture. XXI Congresso Brasileiro de Engenharia Biomédica - CBEB 2008.

MOFFAT, A. Implementing the PPM data compression scheme. IEEE Transactions on Communications, v.38, n. 11, pp. 1917-1921, 1990.

NEEKABADI, A.; SAMAVI, S.; KARIMI, N.; NASR-ESFAHANI, E.; RAZAVI, S.A.; SHIRANI, S. Lossless Compression of Mammographic Images by Chronological Sifting of Prediction Errors. Computers and Signal Processing, 2007. PacRim 2007. IEEE Pacific Rim Conference on Communications, p. 58-61, 2007.

PRZELASKOWSKI, A. Compression of mammograms for medical practice. Symposium on Applied Computing. Proceedings of the 2004 ACM symposium on Applied computing, p. 249-253, 2004.

SALOMON, D. Data Compression – The Complete Reference. 3ª ed., 2004.

SHANNON, C. E. A Mathematical Theory of Communication. Bell Syst. Tech. J., v. 27, p. 379-423, 1948.

SILVA, L. S.; SCHARCANSKI, J. A lossless compression approach for mammographic digital images based on the Delaunay triangulation. IEEE International Conference on Image Processing, 2005. p. II 758-761, 2005.

SOUZA, A. R. C. ; MARQUES, J. R. T. ; LIMA, J. A. G. de ; BATISTA, L. V. Compressão Contextual sem Perdas de Eletrocardiogramas utilizando Recursos Reduzidos de Memória. IFMBE Proceedings, v. 1, p. 30-33, 2007.

SUNG, M. M.; KIM, H. J.; KIM, E. K.; KWAK, J. Y.; YOO, J. K.; YOO, H. S. Clinical Evaluation of JPEG2000 Compression for Digital Mammography. IEEE Transactions on Nuclear Science, v. 49, no. 3, 2002.

WITTEN, I. H.; NEAL, R. M.; CLEARY, J. G. Arithmetic Coding for Data Compression. Communications of the ACM, n. 30, p. 520-540, 1987.

ZHANG, Y,; ADJEROH, D. A. Prediction by Partial Approximate Matching for Lossless Image Compression. IEEE Transactions on Image Processing, vol. 17, nº 6, pp. 924-935, 2008.