VALDIR MARTINS

AVALIAÇÃO DA TAXA DE COMPRESSÃO SEM PERDAS EM PROJEÇÕES DE TOMOGRAFIA COMPUTADORIZADA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática, Curso de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Eduardo Parente Ribeiro

CURITIBA

2002

Ministério da Educação
Universidade Federal do Paraná
Mestrado em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno *Valdir Martins*, avaliamos o trabalho intitulado. *"Avaliação da taxa de compressão sem perdas em projeções de tomografia computadorizada"*, cuja defesa foi realizada no dia 30 de agosto de 2002, às dez horas. no anfiteatro B do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação. decidimos pela aprovação do candidato.

Curitiba, 30 de agosto de 2002.

Prof. Dr. Eduardo Parente Ribeiro
DELT/UFPR - Orientador

Prof. Dr. Humberto Gamba
CEFET/PR - Membro Externo

Hélio Pedrini
Prof. Dr. Hélio Pedrini
DINF/UFPR

AGRADECIMENTOS

Agradeço primeiramente a Deus pela oportunidade concedida e por nos ajudar a vencer as dificuldades e obstáculos.

Agradeço a minha esposa e filhos pelo amor, incentivo, paciência e compreensão pelas horas ausentes.

Ao meu orientador, professor Eduardo Parente Ribeiro, pela segurança como orientou e grande contribuição dada na realização deste trabalho.

Ao colegas mestres, Ionildo e Luciano, pelas contribuições e disponibilização de informações como programas e imagens, utilizadas em suas teses, as quais possibilitaram a realização dos experimentos e finalização deste trabalho.

Agradeço ainda aos demais colegas do curso de pós-graduação em informática, aos professores, amigos e todos aqueles que de alguma forma deram a sua contribuição.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho apresenta um estudo de caso na área de compressão sem perdas de projeções de tomografia computadorizada, chamadas senogramas. A principal vantagem em se comprimir o senograma ao invés da imagem reconstruída consiste em se preservar os dados originais, o que possibilita sua reutilização futura com outros algoritmos de reconstrução. Os métodos que apresentam perdas geralmente possuem melhores taxas de compressão, mas podem prejudicar o processo de reconstrução. São avaliados três métodos de compressão sem perda: transformada *wavelet* de inteiros reversível, LJPG (*Lossless* JPG - *Joint Photographic Experts Group*), e GZIP (Gnu ZIP), que é o padrão de compressão utilizado pelo projeto GNU.

ABSTRACT

This work presents a case study of lossless compression of computerized tomography projections, called sinograms. The main advantage of compressing sinogram in the place of reconstructed imaged lies in the preservation of the original data that allows its future reutilization with others reconstruction algorithms. Lossy image compression methods usually have better compression ratios, but they may alter the reconstrution of the original image. Three methods are considered: the reversible integer wavelet transform which allow lossless image compression; LJPG (Lossless JPG - Joint Photographic Experts Group), for lossless compression of still images; and GZIP (Gnu ZIP), that is the compression standard used by GNU project.

1. Introdução

A tomografia computadorizada (Computed Tomography - CT) é hoje largamente utilizada e tornou-se uma ferramenta indispensável em hospitais, clínicas e empresas de todo o mundo. As projeções de tomografia computadorizada são obtidas com medidas realizadas externamente, a partir de diferentes ângulos de um corpo. A partir destas projeções, os dados originais devem ser reconstruídos utilizando-se técnicas apropriadas. Várias técnicas e algoritmos estão disponíveis nos dias de hoje, permitindo a reconstrução da imagem a partir das projeções obtidas.

Dado ao grande volume de dados gerados nas imagens digitais, torna-se imprescindível a necessidade de se aplicar alguma técnica de compressão de dados, visando facilitar o armazenamento e a transmissão, mas garantindo a expansão e permitindo o processamento desses dados.

Através do processo de compressão de dados podemos reduzir a quantidade de *bits* necessários para representar uma determinada quantidade de informação. Vários métodos de compressão de dados sem perdas (reversíveis) e com perdas (irreversíveis) têm sido propostos na literatura. Cada um desses métodos procura explorar determinadas características da imagem. A escolha do método apropriado depende do conhecimento da teoria do método, bem como, um prévio conhecimento das características presentes na imagem. Desta maneira é possível alcançar maiores taxas de compressão.

O objetivo deste trabalho consiste, então, em estudar e avaliar alguns dos métodos de compressão de dados que deverão ser aplicados nas imagens e nas projeções de tomografia computadorizada, investigando o comportamento de cada um deles sobre estes tipos de imagens, assinalando suas vantagens e desvantagens.

Usaremos, portanto, nesta investigação, alguns dos métodos mais conhecidos em compressão de dados: GZIP (Gnu ZIP), LJPG (Lossless JPEG) e outros métodos recentemente implantados: Transformada S (*Wavelet* de Haar) e Transformada S+P (Said e Pearlman) [SAN01].

1.1 Motivação

A motivação para este trabalho veio da necessidade de pesquisas sobre investigação, implementação e avaliação de algoritmos para compressão sem perdas, de projeções de tomografia computadorizada e suas respectivas imagens, usando a transformada discreta *wavelet* (*discrete wavelet transform* - DWT). Como trata-se de um método relativamente novo, será realizada uma comparação com métodos já conhecidos e consagrados pelo uso, objetivando o estudo do comportamento destes métodos.

Alguns trabalhos foram desenvolvidos na área de compressão de imagens utilizando métodos baseados em *wavelets*. Entretanto, foram desenvolvidos poucos trabalhos utilizando a transformada *wavelet* sem perdas. Os trabalhos envolvendo compressão de imagens de tomografia computadorizada geralmente são realizados utilizando imagens já reconstruídas.

Neste trabalho, além da utilização de imagens já reconstruídas, serão utilizadas as projeções de tomografia computadorizada com objetivo de reduzir a quantidade de informação necessária e mantendo a integridade dos dados. Os dados originais também serão preservados, o que permitirá a aplicação de outras técnicas de reconstrução, as quais também são tópicos de constante pesquisa. Como estas informações não podem sofrer perdas no processo de compressão e descompressão, deverão ser adotados métodos de compressão que permitam que os dados possam ser reconstruídos para a sua forma original. Para tal, serão utilizadas a transformada *wavelet* de inteiros (transformada S e transformada S+P) , o método

GZIP e o Lossless JPEG. Métodos que resultam em perdas de informações não são aceitos para este tipo de dados, pois a perda de informações pode prejudicar a reconstrução e consequentemente a análise e o diagnóstico a partir da imagem final.

O interesse nesta investigação abordará apenas as técnicas de compressão de imagens médicas e seus respectivos senogramas.

1.2 Histórico

O processamento de imagens é uma área de interesse já faz algumas décadas, sendo ainda de interesse crescente, incluído aqui a área de compressão de dados. Conforme descrito em [MAR99], uma das primeiras aplicações em processamento de imagens remonta do começo do século, onde buscava-se formas de aprimorar a qualidade de impressão de imagens digitalizadas transmitidas através do sistema Bartlane de transmissão de imagens por cabo submarino entre Londres e Nova Iorque. Os primeiros sistemas Bartlane, no início da década de 20, codificavam uma imagem em cinco níveis de intensidade distintos. Esta capacidade seria expandida, já em 1929, para 15 níveis ao mesmo tempo em que era desenvolvido um método aprimorado de revelação de filmes através de feixes de luz modulados por uma fita que continha informações codificadas sobre a imagem.

Porém, o grande impulso para esta área viria cerca de três décadas mais tarde, com o advento dos primeiros computadores digitais de grande porte. Já o interesse pelas técnicas de compressão de imagens remonta a quase meio século atrás, na época, utilizando técnicas analógicas.

Em 1986, a necessidade de um padrão internacional para compressão de imagens estáticas resultou na formação do JPEG, hoje, formalmente conhecido como ISO/IEC JTC1/SC29/WG1.

Em relação às transformadas *wavelets*, [SAN01] descreve um histórico o qual reproduzimos a seguir.

A primeira menção de *wavelets* aparece no apêndice da tese de Alfred Haar em 1909 [GRA95]. Uma propriedade da *wavelet* de Haar é que ela possui um suporte compacto, ou seja, ela se anula fora de um intervalo finito. As *wavelets* de Haar não são diferenciáveis continuamente, o que limita de certa forma sua aplicação. A transformada *wavelet* é o resultado do trabalho de um grande número de pesquisadores. Um breve histórico sobre as *wavelets* é apresentado por JAWERTH e SWELDENS [JAW94] e GRAPS [GRA95], da qual segue um breve resumo.

Nos anos 30, vários grupos pesquisavam independentemente a representação de funções usando funções de base de escala variável, chamadas funções de base de Haar. O físico Paul Levy pesquisou movimentos Brownianos, um tipo de sinal randômico. Littlewood, Paley e Stein descobriram uma função que pode variar em escala e conservar energia quando computando a energia da função.

Ainda nos anos 30, a comunidade matemática percebeu que as técnicas desenvolvidas por Fourier não eram muito adequadas para a solução de muitos problemas e recorreram para às chamadas técnicas de Littlewood-Paley, freqüentemente substitutas eficientes.

Durante os anos 50 e 60, desenvolveram poderosas ferramentas para solucionarem equações diferenciais parciais e equações integrais, e perceberam que elas combinavam com a teoria de Calderón-Zygmund, uma área de análise harmônica.

No início dos anos 80, Strömberg descobriu as primeiras *wavelets* ortogonais. Independente dos desenvolvimentos em análise harmônica, o físico Alex Grossmann e o engenheiro Jean Morlet [GRO84], juntamente com seus colegas de trabalho estudaram a

transformada *wavelet* em sua forma contínua. Eles definiram *wavelets* em um contexto de física quântica. Surgia a teoria de "frames".

Em seguida, Yves Meyer, matemático Francês, e seus colaboradores perceberam que as ferramentas da teoria de Calderón-Zygmund, em particular as representações de Littlewood-Paley, poderiam levar a uma concepção unificada de muitos dos resultados em análise harmônica. Começaram também a compreender que aquelas técnicas poderiam substituir as séries de Fourier em aplicações numéricas.

Alex Grossmann e Jean Morlet sugeriram a palavra *wavelet* para os blocos construtivos, e o que antes se chamava teoria de Littlewood-Paley, passou a ser chamado teoria *wavelet*.

Pierre-Gilles Lemarié e Yves Meyer, independente de Strömberg, construíram novas expansões de *wavelet* ortogonal. Em 1985, Stephane G. Mallat [MAL89] introduziu o conceito de análise em multiresolução, dando um novo impulso a essa teoria. Ele descobriu algumas relações entre filtros de quadratura espelhada (*quadrature mirror filters* – QMF), algoritmos piramidais e bases de *wavelets* ortonormais. Inspirado em parte desses resultados, Yves Meyer construiu a primeira *wavelet* não trivial. Ao contrário da *wavelet* de Haar, as *wavelets* de Meyer são diferenciáveis continuamente; todavia elas não têm suporte compacto. A introdução da análise em multiresolução e a transformada *wavelet* rápida por Mallat e Meyer forneceu a conexão entre filtros sub-bandas e *wavelets*. Alguns anos depois, em 1988, usando o trabalho de Mallat, Ingrid Daubechies [DAU88] construiu uma família de *wavelets* com suporte compacto.

Desde então, muitos trabalhos têm sido desenvolvidos em diferentes áreas, sendo uma delas, as aplicações em compressão de imagens digitais.

1.3 Trabalhos Relacionados

Neste item, serão apresentados alguns dos trabalhos mais recentes na área de compressão de imagens baseados em diversos padrões. Os algoritmos de compressão de imagens propostos nestes trabalhos são com perdas e sem perdas.

No trabalho de [AGO00], Estudo de padrões de Compressão de Imagens para Aplicações VLSI (*Very Large Scale Integration*), realiza estudo entre vários métodos de compressão de imagens, tanto para abordagens sem perdas como abordagens com perdas. O objetivo, porém, não está no aspecto comparativo entre estes vários métodos, mas servir de introdução aos algoritmos discutidos no trabalho, assim como discussão de algumas arquiteturas possíveis para a implementação em hardware de alguns dos algoritmos apresentados.

Em [MEL94] realiza-se uma avaliação da utilização da transformada *wavelet* na compressão de imagens. Traz também uma comparação dos resultados obtidos no trabalho com o método de compressão JPEG para algumas imagens. Nos experimentos realizados, verifica uma disparidade grande entre as taxas de compressão do método JPEG, comercialmente disponível e o método da Transformada Wavelet. Conclui que é preciso avaliar cuidadosamente o processo proposto em seu trabalho.

No trabalho de [SAN01] está descrito que, recentemente, o algoritmo chamado SPIHT (*Set Partitioning in Hierarchical Trees*) foi proposto por Said e Pearlman [SAI96], o qual é uma implementação mais eficiente do esquema de codificação *zerotree* do Shapiro [SHA93]. Este algoritmo permite codificar uma imagem com perdas ou sem perdas. Eles usam um esquema de multiresolução em pirâmide o qual é melhorado via codificação preditiva através da transformada S+P (transformada Seqüencial + Predição). O algoritmo SPIHT difere de outros métodos no fato de que a predição é usada durante (em vez de

depois) a seqüência de transformações recursivas e, consequentemente, pode usar informações que não estão disponíveis depois que a imagem é transformada. O algoritmo SPIHT é baseado em três estágios: (*i*) aplicação da tranformada S+P para decompor a imagem, (*ii*) uma técnica de codificação baseada em *zerotree* e (*iii*) codificação aritmética adaptativa dos dados residuais.

Esta técnica produz significativamente melhor compressão do que os métodos de compressão *wavelet* tradicional com similar complexidade computacional, e representa o estado da arte em compressão de imagens de propósito geral.

A transformada S (Seqüencial) utilizada nesta técnica é uma adaptação da transformada de Haar para mapear os valores de inteiros para inteiros, permitindo assim a reconstrução exata das informações.

Calderbank [CAL97] apresenta duas abordagens para construir transformadas *wavelet* de inteiros para inteiros os quais podem ser usadas para compressão sem perdas. Na primeira parte do trabalho é apresentado uma adaptação do pré codificador desenvolvido por Laroia, Tretter e Farvardin. Na segunda abordagem é apresentado como conseguir uma implementação reversível usando o *lifting scheme* proposto por Sweldens [SWE97].

1.4 Estrutura da Dissertação

Este trabalho está estruturado em seis capítulos. No primeiro capítulo estão descritos a motivação, um histórico sobre compressões e transformadas *wavelets* e os trabalhos relacionados.

No segundo capítulo, inicialmente é apresentada uma teoria sobre tomografia computadorizada. Em seguida, comenta-se o padrão DICOM, utilizado para identificação, arquivamento e recuperação de imagens médicas, entre elas, as de tomografia

computadorizada e, finalizando o capítulo, as necessidades advindas da armazenagem dessas imagens.

O capítulo terceiro comenta a necessidade de compressão de imagens, dada ao grande volume existente nos dias atuais. Estão incluídos também comentários sobre o modelo genérico de compressão, formas de classificação e medidas de compressão. Finalizando o capítulo, estão descritos alguns métodos e algoritmos de compressão de dados: Codificação de Lempel-Ziv (LZ), Codificação de Huffmann, Codificação Aritimética, Codificação Preditiva, Codificação por Transformada e Lossless JPEG.

O quarto capítulo descreve a metodologia utilizada para a realização dos experimentos deste trabalho, dentro do objetivo, que é a realização de testes de compressão sobre imagens de tomografia computadorizada e imagens reconstruídas, utilizando vários métodos de compressão.

No capítulo cinco estão descritos os resultados obtidos com os experimentos, dentro das seguintes abordagens: taxas de compressão para senogramas reais; taxas de compressão para imagens e senogramas sem inserção de ruídos; taxas de compressão para senogramas reais com inserção de ruídos em escalas percentuais; taxas de compressão em uma imagem e um senograma, com inserção de ruídos em escalas percentuais; comparativo de compressão entre imagens e senogramas com e sem inserção de ruídos.

Por fim, no sexto capítulo, estão a conclusão e apresentação de propostas para trabalhos futuros.

Visando complementar as várias teorias estudadas, são apresentados nos apêndices, teorias sobre padrões de compressão de imagens, onde estão comentados os métodos GZIP, LJPG, transformadas *wavelets* e formatos de imagens utilizadas neste trabalho.

2. Tomografia Computadorizada (CT)

2.1 Tomografia de Raio-X

Em de [RIB96] foi descrito que a tomografia computadorizada é um método de radiografia que foi introduzido em 1972 para aplicações neurológicas, mas rapidamente com o avanço da tecnologia, permitiu examinar outras partes do corpo também. O primeiro tomógrafo computadorizado foi construído por Godfrey N. Hounsfield, feito que lhe conferiu, junto com Allan McLeod Comarck, o Prêmio Nobel de Fisiologia e Medicina em 1979. A tomografia de raio-X tem desde então se estendido também em aplicações industriais.

O termo tomografia é usado para se referir a qualquer método de obtenção de imagem do interior de um corpo ou objeto a partir de medidas realizadas externamente. O termo *"computerized tomography"*, ou somente CT, caiu no uso comum e até hoje se refere ao tomógrafo de raio-X. Existem outras técnicas de tomografia baseadas em outros princípios físicos como: tomografia de ressonância magnética, *Magnetic Ressonance Imaging* (MRI), a tomografia de emissão de pósitrons, *Positron Emission Tomography* (PET), a tomografia com ultra-som e tomografia de impedância elétrica, *Electrical Impedance Tomography* (EIT).

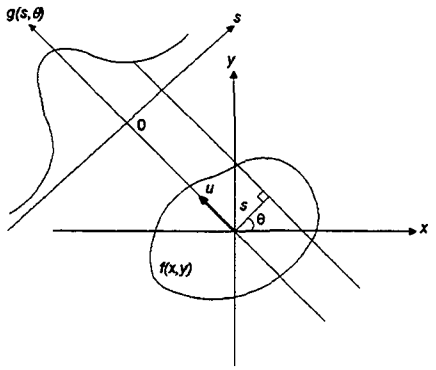


Figura 2.1 - Geometria da obtenção das projeções em CT.

A seguir apresenta-se um modelo simplificado para CT de raio-x [JAI89]. Seja $f(x,y)$ o coeficiente de absorção do objeto no ponto (x,y) numa faixa para algum valor fixo de z. Assumindo que os raios-x consistem de um feixe infinitamente fino e paralelo, a intensidade do feixe detectado do outro lado do objeto é dado por

$$I = I_0 \exp\left[-\int_L f(x,y)du\right] \tag{2.1}$$

onde I_0 é a intensidade do feixe incidente, L é o caminho do raio e u é a distância ao longo de L, conforme mostrado na figura 2.1.

Considerando projeções em muitas direções diferentes, é possível reconstruir as características de densidade do corpo para cada ponto ou pequena região no interior do corpo.

A imagem obtida com a transformada de Radon de uma distribuição, que é a imagem que se obtém ao empilhar as linhas correspondentes às projeções obtidas para os vários ângulos, é muitas vezes chamada de senograma.

A partir das projeções, obtidas com as medidas realizadas externamente ao corpo, deve-se reconstruir esses dados originais utilizando alguma técnica de reconstrução. Existem vários algoritmos e técnicas que permitem a reconstrução da imagem a partir das projeções obtidas. As técnicas que permitem a reconstrução da imagem também têm sido tópicos de constantes pesquisas [SAN01].

2.2 Padrão DICOM

Em 1985, duas organizações norte-americanas, uma da área médica (*American College of Radiology*) e outra da área de equipamentos médicos (*National Electrical Manufacturers Association*) desenvolveram conjuntamente um padrão para o intercâmbio eletrônico de imagens que não dependesse do tipo de computador onde residem os dados. Esse padrão recebeu o nome de ACR-NEMA e foi o primeiro a ser adotado pelos fabricantes

de aparelhos geradores de imagens radiológicas, permitindo assim uma conexão mais fácil a computadores de uso geral. Posteriormente, foi criado, a partir dele, o padrão DICOM (*Digital and Communications in Medicine*) [DIC01], o qual foi adotado muito mais amplamente e que governa também as informações de texto (nome, número de registro do paciente, laudo radiológico, etc.). Usando-se um único comando, o DICOM permite que o médico recupere o registro completo de imagem médica de um determinado paciente. O DICOM define não somente como a imagem é representada digitalmente dentro do computador (formato de imagem), bem como ele deve ser arquivado. São enormes os benefícios trazidos por sistemas desse tipo, pois os programadores podem simplesmente incorporar o padrão em seus programas, ao invés de ter que desenvolver um novo programa a partir do zero. Além disso, eles não precisam se preocupar com as diferenças entre os vários modelos de aparelhos de raios-X, tomografia, ultra-som, medicina nuclear, etc., ou com os computadores onde rodam os seus programas. Várias organizações, como o American College of Cardiology e o American College of Pathology estão adotando o DICOM como padrão, fazendo com que um largo espectro de imagens médicas seja padronizado e disponíveis através de alguns comandos simples [HOG98].

2.3 Necessidade de Armazenagem

No manejo de informação dentro de um hospital, por meio de uma rede de computadores, surgiu inicialmente o conceito de Sistemas de Informação Radiológica - RIS (*Radiology Information Systems*) demonstrando que é possível utilizar sistemas computadorizados para melhorar o gerenciamento dos pacientes, geração e distribuição de relatórios, facilidades de utilização dos recursos disponíveis, localização dos filmes, e rotinas de funcionamento do setor de radiologia. Freqüentemente, estes sistemas são integrados ao Sistema de Informação Hospitalar - HIS (*Hospital Information Systems*). Embora o RIS

resolva a parte de gerenciamento de dado, porém não trabalha com as próprias imagens, na década de 80, este conceito foi ampliado para incluir o que se chama de PACS (*Picture Archiving and Communication System*), ou Sistemas de Arquivamento e Comunicação de Imagens. É um sistema que permite, como o nome diz, armazenagem e recuperação das imagens em uma rede de computadores.

Atualmente, a maior parte das imagens são registradas e armazenadas em filme. Igualmente imagens como CT e MRI, as quais são inerentemente digitais, são transferidas para o filme depois que os técnicos a tenham otimizado para a visualização. Ocasionalmente como no caso de estudos ultrasonográficos, as imagens são transferidas para fitas de vídeo para posterior revisão e interpretação.

O armazenamento de filmes requer um grande espaço físico em departamentos de radiologia. Em geral os departamentos de radiologia têm a capacidade de armazenar filmes somente para pacientes que tenham sido estudados nos últimos 6 a 12 meses. Estudos antigos são retidos por no mínimo 7 anos e estocados em um porão por exemplo. Além do espaço físico, as imagens sofrem deterioração ao longo dos anos, uma vez que o filme químico perde a qualidade. Outro problema é o perigo do armazenamento, pois trata-se de material altamente imflamável. A aquisição digital de todas as imagens dentro de um hospital, além de resolver os problemas de deterioração de imagens e guarda de material inflamável, oferece um excitante panorama de redução do espaço físico requerido, custo de material, redução do trabalho manual tradicional de manuseio de filmes, rápida recuperação de imagens via pedido de informação à base de dados e alta velocidade de transmissão de imagens através de redes.

O desenvolvimento dos sistemas PACS é uma área ativa de pesquisa em informática médica. Um número de complexos problemas tiveram que ser resolvidos antes de por em prática, incluindo padronização de transmissão de imagens e formatos de

armazenagem. Armazenar todos os dados de imagens médicas digitais pode criar um grande problema de gerenciamento, que dificilmente pode ser resolvido por métodos que não envolvam computação [ALM98].

Com a expansão da imagem digital em termos de qualidade, tamanho e aplicabilidade, surge a necessidade de desenvolver mecanismos de compressão de forma a tornar viável a troca deste tipo de informação.

Todo o esforço para reduzir o espaço ocupado pela imagem tem de levar em consideração a sua qualidade. Dessa forma, a interoperacionalidade de sistemas relacionados com a transferência de imagem digital (tal como o DICOM) é conseguida através da melhor relação do binômio nível de compressão/qualidade.

3. Compressão de Imagens

Grandes hospitais e clínicas de diagnóstico por imagem realizam milhares de procedimentos de imagens médicas, podendo gerar centenas ou até milhares de *gigabytes* de dados de imagens por ano. Existem técnicas de computação que podem comprimir informações como a compressão de dados, que geralmente aproveita a redundância na informação. A compressão típica para dados é geralmente na faixa de 2 ou 3 para 1. Algumas técnicas de alta compressão de imagens, conhecidas como "com perda", conseguem proporções de até 50 para 1, mas podem distorcer e alterar a imagem e prejudicar o diagnóstico em caso de estruturas muito pequenas na imagem (bordas, microcalcificações) [ALM98].

A compressão de uma imagem é possível porque as imagens, em geral, apresentam um alto grau de coerência, que se traduz em uma redundância de informação quando codificada. Considere, por exemplo, um elemento qualquer de uma imagem no formato matricial. Provavelmente, a cor desse elemento será igual a dos elementos vizinhos, porque há uma grande probabilidade de todos eles pertencerem a um mesmo objeto da imagem. Caso isso não ocorra, certamente uma relação mais complexa existirá na imagem. Baseado neste fato, os métodos de compressão visam produzir, através da redução ou eliminação de redundância, um código mais compacto que preserve as informações contidas na imagem [SAN01].

Em uma imagem digital existem, basicamente, três tipos de redundâncias de dados que podem ser identificadas e exploradas: redundância de código, redundância interpixel e redundância psicovisual, já estudadas e relatadas em alguns trabalhos [SAN01, AGO00].

Na redundância de código, o processo de codificação determina um código binário com tamanho variável. Aos valores que ocorrem mais freqüentemente na imagem, é atribuído um código menor (em número de *bits*), enquanto que os valores que são menos freqüentes, um código maior é atribuído.

A redundância interpixel permite prever razoavelmente o valor de um pixel a partir do valor do pixel de seus vizinhos, com isso a informação dos pixels é relativamente pequena. Esta correlação espacial resulta da relação estrutural ou geométrica entre os objetos na imagem.

A redundância psicovisual ocorre pelo fato de algumas informações terem menos importância do que outras informações em um processamento visual normal. Essas redundâncias podem ser eliminadas sem comprometer a qualidade da imagem. Todavia, a eliminação de redundância psicovisual resultará numa perda de informação, o que leva a um processo de compressão com perdas.

Dependendo da área de aplicação, as informações que se deseja preservar podem ser de natureza objetiva ou subjetiva. No primeiro caso o método de compressão deve permitir a recuperação exata dos dados da imagem original. Diz-se, nesse caso, que o processo é reversível, ou que se tem uma compressão sem perda (*lossless*). Caso contrário, ele é chamado irreversível, então se tem uma compressão com perda (*lossy*), isto é, ele não permite a reconstrução exata dos dados originais.

Os recentes avanços das técnicas de compressão com perdas incluem diferentes métodos tais como a transformada discreta cosseno (*Discrete Cosine Transform*-DCT), quantização vetorial, codificação *wavelet*, redes neurais e codificação fractal. Apesar desses métodos poderem realizar altas taxas de compressão (tipicamente 50:1 ou maior), eles não permitem a reconstrução exata da versão original dos dados de entrada. Portanto, métodos

com perdas não são aceitos para certas aplicações como: imagens médicas digitais, imagens de satélite, dados sísmicos, imagens de alta fidelidade, arquivos executáveis e assim por diante. Técnicas de compressão sem perdas, todavia, permitem a reconstrução perfeita dos dados originais, mas as taxas de compressão são comparativamente baixas (de 2:1 a 4:1) [SAN01].

3.1 Modelo Genérico de Compressão de Imagens

A idéia geral utilizada para a compressão de imagens é reorganizar os *pixels* de maneira que a redundância possa ser mais eficientemente descartada. A figura 3.1 apresenta um modelo genérico de compressão de imagens. Nesta figura, a operação de mapeamento converte os dados de entrada para uma outra forma organizada, removendo as redundâncias. Isso significa que os pixels são mapeados em outro domínio, no qual menos *bits* são necessários para codificar o dado mapeado em comparação com o número de *bits* necessários para codificar o dado de entrada original.

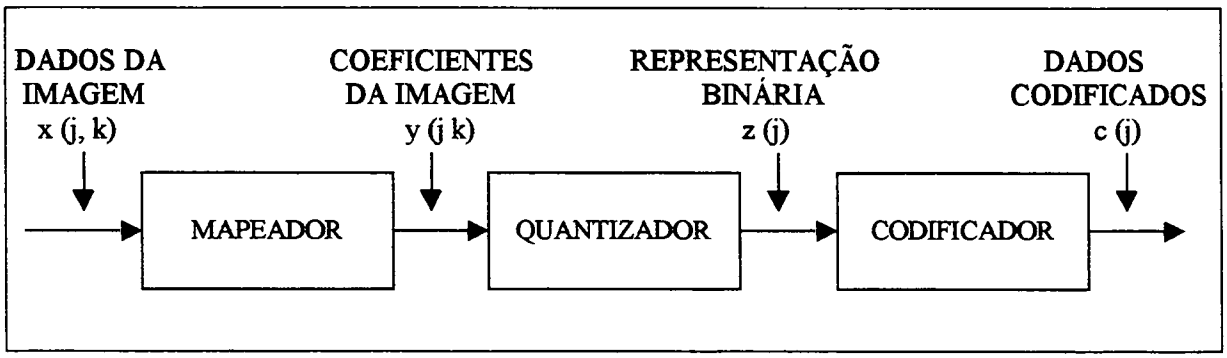


Figura 3.1 - Modelo genérico de compressão de imagens [MEL94].

A operação de quantização converte os coeficientes em um número menor de valores mais convenientes para a operação de codificação. O quantizador arredonda cada valor mapeado para dentro de um pequeno conjunto de possíveis valores. A operação de

codificação representa os dados fornecidos pelo quantizador em um conjunto mais conciso de símbolos.

O modelo representado na figura 3.1 é genérico e em muitos esquemas de compressão o mapeador e/ou o quantizador são suprimidos. Em algumas técnicas de compressão, a efetiva redução de informação se encontra no mapeador, porém, na maioria dos casos, a compressão é obtida nos processos de quantificação e/ou codificação.

A operação de mapeamento pode ou não ser reversível, isto é, permitir retornar aos dados originais. Já o quantizador sempre introduz erro, e após a quantização, os dados originais não podem mais ser recuperados. O codificador sempre implementa uma operação reversível [MEL94].

3.2 Classificação da Compressão de Imagens

Uma classificação rígida dos vários métodos de compressão de imagens é muito difícil, uma vez que existe uma grande variedade de algoritmos e técnicas de compressão que utilizam princípios diferentes e que muitas vezes são utilizados simultaneamente. Várias classificações já foram propostas, porém nenhuma delas pode ser considerada definitiva.

De um modo geral, é útil distinguir entre as técnicas de compressão que são reversíveis, isto é, permitem recuperar a imagem original sem perda de informação, e aquelas que são irreversíveis, onde a imagem recuperada possui menos informação que a imagem original. Esta separação é encontrada em quase todas as classificações propostas na literatura, porém as semelhanças terminam por aí.

As técnicas ditas reversíveis são também chamadas de redução de redundância, compressão sem perda ou sem ruído. As técnicas irreversíveis são conhecidas também por redução de entropia, compressão com perda ou com ruído [MEL94].

A Figura 3.2 apresenta uma taxionomia dos diversos métodos e algoritmos de compressão existentes [AGO00]. Destes métodos, são abordados neste trabalho os algoritmos Aritmético, em conjunto com a transformada *wavelet*, Lempel-Ziv (GZIP) e Huffman (LJPG), descritos mais adiante no item 3.5.

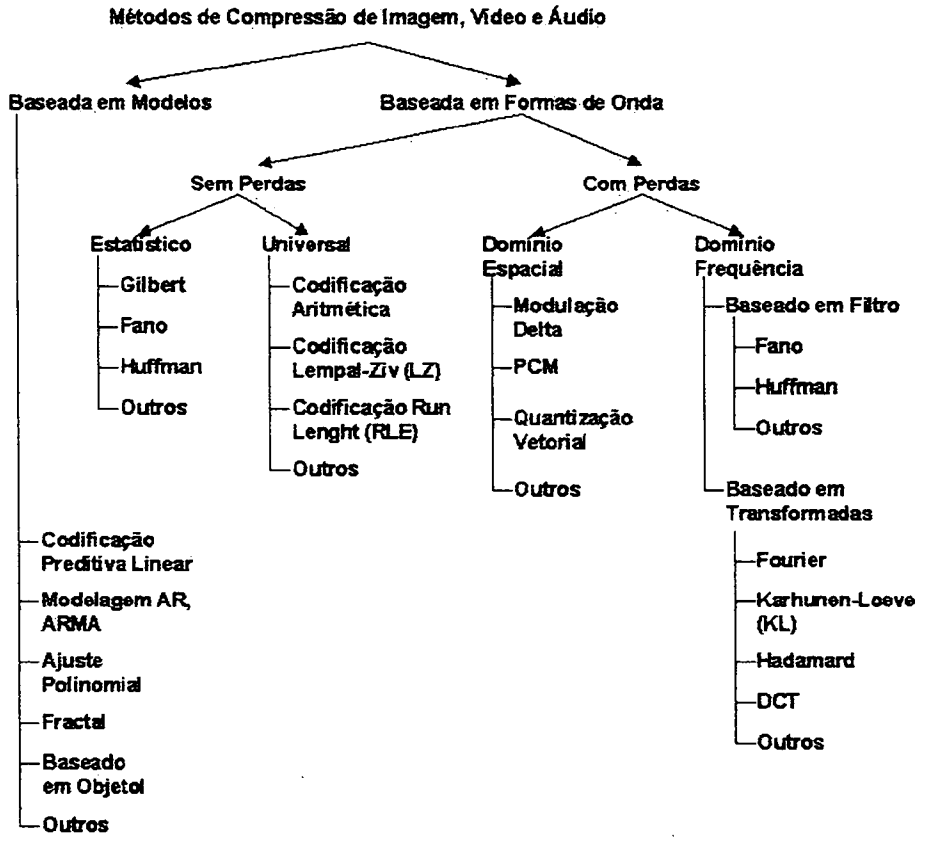


Figura 3.2 - Taxionomia dos métodos de compressão.

3.3 Taxa de Compressão

A taxa de compressão é a razão entre os tamanhos da imagem comprimida e da imagem original. Esta taxa fornece uma medida de quanto menor ficou o arquivo após a aplicação da compressão.

Em geral, é comum fornecer a taxa de compressão ou na forma de um vetor absoluto ou em percentual. Por exemplo, a taxa de compressão para uma imagem de 200

Kbytes que após ser comprimida ficou com 100 *Kbytes* de tamanho, é de 0,5 ou 50% [MEL94].

Diferentes fórmulas são utilizadas para o cálculo das taxas de compressão. Alguns cálculos são feitos em *bits/bytes*. Esta medida de compressão fornece o número de *bits* necessários por *pixel* depois da compressão. O valor *bits* por *pixel* (bpp) permite comparar os métodos de compressão, permite calcular o tamanho da imagem comprimida e a taxa de transmissão necessária para qualquer tamanho de imagem. Outros preferem utilizar razões, por exemplo, 2:1. Isto significa que a quantidade de dados foi reduzida pela metade. Será utilizada neste trabalho a fórmula (em porcentagem) definida por:

$$\left(1-\frac{dados\ comprimidos}{dados\ originais}\right)*100 \qquad (3.1)$$

Isto significa que um arquivo que não muda de tamanho após a compressão terá uma taxa de compressão de 0% (zero por cento). Um arquivo comprimido pela metade do seu tamanho original terá uma taxa de compressão de 50%. Na teoria, a taxa de compressão máxima é 100%. Isto só ocorrerá se o tamanho do arquivo comprimido for igual a zero, o qual somente ocorrerá se o tamanho dos dados originais também for zero. Se o tamanho do arquivo comprimido for maior do que o arquivo original teremos uma taxa de compressão negativa.

3.4 Compressão com Perdas e Compressão sem Perdas

O processo de compressão de dados pode ser separado em dois tipos: os que levam a perda de informação e os que mantêm a informação intacta. O primeiro conduz às melhores taxas de compressão, mas com o ônus de perder parte da informação contida na imagem original.

Muitas aplicações exigem imagens com a melhor qualidade possível (como em áreas médicas, por exemplo) e, portanto, não podem existir perdas de informações no processo de compressão. Nestas aplicações pode-se obter uma taxa de compressão típica variando de 2:1 até 10:1.

Todos os formatos de compressão que buscam eliminar apenas as redundâncias de codificação e interpixel são considerados sem perdas, uma vez que ambos os processos são reversíveis e, portanto, possível se obter na saída do decodificador uma imagem idêntica, *pixel* a *pixel*, quando comparada com a imagem colocada na entrada do codificador. Desta forma, pode-se perceber que o processo de compressão é efetuado apenas pelos estágios de mapeamento e codificação de símbolo, apresentados na Figura 3.1.

A compressão com perdas é utilizada em aplicações nas quais o menor tamanho possível da imagem é requerido (como na internet, por exemplo). Nestes casos, o efeito da perda de informação, que nem sempre é visual, é aceitável. Com isso, pode-se obter taxas de compressão maiores que 30:1, chegando a taxas de 10:1 até 20:1 com imagens virtualmente indistinguíveis das originais.

A perda de informação na compressão com perdas é gerada pelo estágio de quantização na Figura 3.1, que busca a redução da redundância psicovisual. Como este estágio é irreversível, a informação descartada não mais poderá ser recuperada. [AGO00].

3.5 Tipos de Codificação

Vários métodos e algoritmos de compressão de dados têm sido estudados e propostos, mas alguns deles mereceram notável destaque, uma vez que são usados em vários formatos de arquivos de imagem com compressão, além de outros usos na compressão de dados. Como os formatos de arquivo com compressão que serão abordados neste trabalho

compartilham o uso de alguns métodos de compressão, optou-se por tratar destes métodos antes de entrar no escopo dos formatos de arquivos propriamente ditos.

É importante não confundir os algoritmos de compressão de imagem com os formatos de arquivos que utilizam estes algoritmos [MUR96]. O formato do arquivo descreve apenas a organização dos dados, indicando cabeçalhos, tabelas, etc., enquanto que os algoritmos de compressão são os responsáveis pela redução na quantidade de dados necessária para representar a informação. Cada formato de arquivo define onde e como a informação comprimida será colocada no arquivo a ser gerado.

Existe uma grande variedade de métodos de compressão, mas apenas alguns mais usados para a compressão de imagem e os que são de uso livre fazem parte do escopo deste trabalho.

A seguir, comenta-se os métodos de codificação utilizados pelos padrões estudados neste trabalho.

3.5.1 Codificação de Lempel-Ziv (LZ)

O LZ é um dos algoritmos mais comuns usados para a compressão de imagens e dados em geral. Ele foi criado em 1977 por Abram Lempel e Jakob Ziv e, desde então, muitas variantes deste algoritmo têm sido utilizadas em vários formatos de arquivos. Este método de compressão visa reduzir a redundância de codificação a partir do uso de dicionários de dados e não gera perdas.

Uma das versões mais conhecidas do LZ é o chamado LZW, criado em 1984 por Terry Welch. O LZW é utilizado no formato de arquivo GIF. A versão do LZ abordada neste item é chamada de LZ77. Esta versão utiliza uma janela de dados móvel que será chamada de janela LZ77 para efetuar a compressão e a descompressão.

A seqüência dos dados comprimidos pode conter valores literais ou comandos para expansão. Os comandos para expansão são pares comprimento/distância que informam a localização, na janela LZ77, do primeiro dado de uma seqüência a ser expandida e quantos dados contém esta seqüência.

No caso de compressão de textos, a cada passo um novo caracter é comparado com os caracteres que estão na janela LZ77. Se não existe nenhum caracter igual a este novo caracter, então a janela se desloca para incluir o novo caracter e este caracter é colocado para a saída.

Se o caracter já existe na janela LZ77, um novo caracter é pego dos dados não comprimidos. Então é feita uma nova comparação, mas agora para verificar se existem na janela os dois caracteres. Este procedimento é repetido até que a seqüência de caracteres obtidos não exista na janela LZ77. Então o último caracter lido é desconsiderado e um par distância/comprimento é escrito na saída, informando a distância que a seqüência de caracteres está do início da janela LZ77 e, por quantos caracteres é composta esta seqüência. Então a janela é deslocada pelo número de caracteres da seqüência para incluí-los e o processo é reiniciado com o próximo caracter.

Como pode ser percebido, para a compressão, são necessárias duas estruturas, uma para varrer a janela LZ77 e outra para varrer os dados da entrada. Estas duas estruturas são complementares e os seus dados são deslocados em conjunto.

No algoritmo LZ77, quanto maior o tamanho da janela LZ77 maior será a taxa de compressão obtida, mas também maior será a complexidade do algoritmo, principalmente no que diz respeito às operações de procura nas janelas. Por isso, existe um compromisso entre estes dois fatores que deve ser levado em consideração. Uma alternativa para o uso de

tamanhos maiores de janelas é utilizar tabelas *hash*, que aumentam a velocidade da procura [MIA99].

Para uma melhor compreensão do método, apresenta-se a seguir um exemplo ilustrativo, extraído do trabalho de [AGO00]. O tamanho da janela LZ77 foi definida em 16 caracteres (aplicações normais usam até 32K bytes). A seguinte frase será comprimida no exemplo: "A BANCA DA BIANCA É BRANCA.". A tabela 3.1 apresenta todos os passos do algoritmo LZ77 para a frase e seu conteúdo a cada passo. Na tabela 3.2 apresenta-se o vetor saída para o mesmo exemplo.

Passo	Janela LZ77 (0 … 15)	Entrada
0		A BANCA DA BIANCA É BRANCA.
1	A	BANCA DA BIANCA É BRANCA.
2	A B	ANCA DA BIANCA É BRANCA.
3	A BA	NCA DA BIANCA É BRANCA.
4	A BAN	CA DA BIANCA É BRANCA.
5	A BANC	A DA BIANCA É BRANCA.
6	A BANCA	DA BIANCA É BRANCA.
7	A BANCA D	A BIANCA É BRANCA.
8	A BANCA DA	BIANCA É BRANCA.
9	A BANCA DA B	IANCA É BRANCA.
10	A BANCA DA BI	ANCA É BRANCA.
11	BANCA DA BIANCA	É BRANCA.
12	ANCA DA BIANCA É	BRANCA.
13	CA DA BIANCA É B	RANCA.
14	A DA BIANCA É BR	ANCA.
15	BIANCA É BRANCA	.
16	BIANCA É BRANCA.	

Tabela 3.1 - Conteúdo da janela LZ77 e da Entrada durante a compressão.

Passo	Saída
0	A
1	A
2	A B
3	A BA
4	A BAN
5	A BANC
6	A BANC<10:2>
7	A BANC<10:2>D
8	A BANC<10:2>D<13:2>
9	A BANC<10:2>D<13:2>B
10	A BANC<10:2>D<13:2>BI
11	A BANC<10:2>D<13:2>BI<6:5>
12	A BANC<10:2>D<13:2>BI<6:5>É
13	A BANC<10:2>D<13:2>BI<6:5>É<7:2>
14	A BANC<10:2>D<13:2>BI<6:5>É<7:2>R
15	A BANC<10:2>D<13:2>BI<6:5>É<7:2>R<7:4>
16	A BANC<10:2>D<13:2>BI<6:5>É<7:2>R<7:4>.

Tabela 3.2 - Conteúdo da saída a cada passo da compressão.

O processo de descompressão é substancialmente mais simples, porque apenas com a janela LZ77 e com os dados comprimidos é possível efetuar a descompressão dos dados. Caso o dado lido seja um valor literal, ele é simplesmente escrito na janela LZ77 e na saída. Caso o dado seja um par distância/comprimento, o algoritmo de descompressão copia da janela LZ77 o número de caracteres definido pelo campo comprimento, onde o primeiro caracter da seqüência está na distância do início da janela definida pelo campo distância. Então a janela é deslocada para agregar os caracteres copiados e estes dados são colocados na saída. Caso a janela tenha excedido a sua capacidade, os dados do início da janela são removidos. A tabela 3.3 apresenta o processo de descompressão para o exemplo [AGO00].

Passo	Saída (0 ... 15)	Entrada
0	A	BANC<10:2>D<13:2>BI<6:5>É<7:2>R<7:4>.
1	A	BANC<10:2>D<13:2>BI<6:5>É<7:2>R<7:4>.
2	A B	ANC<10:2>D<13:2>BI<6:5>É<7:2>R<7:4>.
3	A BA	NC<10:2>D<13:2>BI<6:5>É<7:2>R<7:4>.
4	A BANC	<10:2>D<13:2>BI<6:5>É<7:2>R<7:4>.
5	A BANC	<10:2>D<13:2>BI<6:5>É<7:2>R<7:4>.
6	A BANCA	D<13:2>BI<6:5>É<7:2>R<7:4>.
7	A BANCA D	<13:2>BI<6:5>É<7:2>R<7:4>.
8	A BANCA DA	BI<6:5>É<7:2>R<7:4>.
9	A BANCA DA BI	<6:5>É<7:2>R<7:4>.
10	A BANCA DA BI	<6:5>É<7:2>R<7:4>.
11	A BANCA DA BIANCA	É<7:2>R<7:4>.
12	A B ANCA DA BIANCA É	<7:2>R<7:4>.
13	A BAN CA DA BIANCA É BR	<7:4>.
14	A BANC A DA BIANCA É BR	<7:4>.
15	A BANCA DA BIANCA É BRANCA	.
16	A BANCA DA BIANCA É BRANCA.	

Tabela 3.3 - Descompressão dos dados do exemplo por LZ77.

3.5.2 Codificação de Huffman

A codificação de Huffman é a técnica de compressão de dados mais popular para a remoção de redundância de codificação e é o mais eficiente método para a codificação de comprimento fixo para comprimento variável [JAI89].

A idéia é atribuir códigos de comprimentos variáveis para cada símbolo. O menor código é atribuído ao símbolo de maior probabilidade de ocorrência na imagem e assim sucessivamente, até que o maior código seja atribuído ao símbolo de menor probabilidade de ocorrência. O problema é que, para tornar possível a decodificação sem a inserção de tabelas de identificação de símbolos, as seqüências de *bits* já atribuídas não podem ser repetidas no

início das novas seqüências. Com isso, o comprimento dos códigos tende a crescer rapidamente.

A codificação de Huffman torna-se interessante para aplicações onde existe um desequilíbrio grande entre as probabilidades de ocorrência dos símbolos. Para aplicações com probabilidades aproximadamente iguais, o tamanho do arquivo tende a ser maior que o original. Na prática, a maioria das aplicações se encaixa no primeiro grupo, com probabilidades de ocorrência bem diferentes.

O primeiro passo para implementar a codificação de Huffman é a criação de uma tabela com os símbolos presentes no código e suas probabilidades de ocorrência. A partir desta tabela gera-se uma estrutura de árvore, construída a partir de uma seqüência de operações onde as duas menores probabilidades são unidas em um nó para formar dois ramos da árvore. Cada nó já construído é tratado como um simples símbolo, com a probabilidade sendo a soma de todas as probabilidades dos símbolos combinados neste nó. Então, arbitrariamente, atribui-se 0 para um dos ramos e 1 para o outro [PEN92]. Considerando esta probabilidade combinada, novamente as duas menores probabilidades são unidas, formando um novo nó e uma nova probabilidade combinada. Este processo se repete até que todos os símbolos estejam contidos na árvore [AGO00]. Um exemplo de codificação utilizando Huffman pode ser visto no item 3.5.6 (Codificação Lossless JPEG com Huffman).

3.5.3 Codificação Aritmética

Conforme explanado em [SAN01], os métodos de codificação de tamanho variável (*Variable Length Coding* – VLC) mais utilizados na codificação dos coeficientes *wavelets* são a codificação de Huffman e a codificação aritmética. A codificação de Huffman é mais rápida enquanto que a codificação aritmética permite maior compressão. A primeira é preferível quando os recursos de hardware são limitados e o tempo de

codificação/decodificação é o objetivo principal. A codificação aritmética é um pouco mais lenta do que a codificação de Huffman, mas é muito mais versátil e eficiente.

No código de Huffman, o número de *bits* necessários para codificar uma informação tem que ser um número inteiro e isso algumas vezes pode ser um problema. Por exemplo, se a probabilidade de ocorrência de um caracter é $\frac{1}{3}$, o melhor número de *bits* para codificar este caracter é de aproximadamente 1,58 *bits*. Com isso, o código de Huffman precisaria atribuir dois *bits* para o código, conduzindo para uma mensagem comprimida maior do que é teoricamente possível de acordo com o cálculo da entropia. Este problema aumenta quando a probabilidade de um caracter é muito alta.

Como o código de Huffman usa um número inteiro (k) de *bits* para cada símbolo, isto implica que nunca k será menor que 1. Numa imagem com duas cores (1 *bit* por *pixel*), como as usadas em fax, a compressão se torna impossível. A solução seria agrupar os *bits* em pacotes e então aplicar Huffman, porém isso impede o código de Huffman de ser um compressor universal.

A codificação aritmética substitui os símbolos de entrada com um código específico. Um fluxo de *bits* de símbolos de entrada é codificado em um único número de ponto flutuante maior ou igual a zero e menor do que 1. Este número pode ser unicamente decodificado para extrair os símbolos originais.

Existem dois pontos fundamentais na codificação aritmética: a probabilidade de um símbolo e o limite de seu intervalo na codificação. As probabilidades dos símbolos de origem determinam a eficiência da compressão. Eles também determinam o limite do intervalo dos símbolos de origem para o processo de codificação. Estes limites de intervalo estão contidos dentro de um intervalo de 0 a 1. O limite do intervalo para o processo de codificação determina a saída comprimida.

Em [SAN01], o processo de codificação de um codificador aritmético foi explicado através do seguinte exemplo: supondo que os símbolos de entrada são {*a*, *b*, *c*, *d*} e a probabilidade destes símbolos são {0.1, 0.4, 0.2, 0.3 }, respectivamente. Então baseado em suas probabilidades, o intervalo [0, 1) pode ser dividido como 4 subintervalos: [0, 0.1), [0.1, 0.5), [0.5, 0.7) e [0.7, 1), onde [*x*, *y*) denota um intervalo fechado a esquerda e aberto a direita, os quais inclui *x* mas exclui *y*. A informação acima pode ser resumida na tabela 3.4, abaixo:

Símbolos	*a*	*b*	*c*	*d*
Probabilidades	0.1	0.4	0.2	0.3
Intervalo Inicial	[0, 0.1)	[0.1, 0.5)	[0.5, 0.7)	[0.7, 1)

Tabela 3.4 – Representação dos símbolos e seus respectivos intervalos iniciais.

Imagine transmitindo a mensagem "*cadacdb*". Inicialmente, tanto o codificador quanto o decodificador conhecem que o intervalo é [0, 1). Após ler o primeiro símbolo, *c*, da mensagem sabemos que o seu intervalo de codificação é [0.5, 0.7). Após este símbolo ter sido processado, o tamanho do modelo é modificado para o intervalo [0.5, 0.7), como mostrado na figura 3.3, já que o limite do segundo símbolo, *a*, da mensagem é [0, 0.1), ele é codificado tomando o primeiro décimo do novo tamanho do modelo [0.5, 0.7) como um novo intervalo [0.5, 0.52). Similarmente, codificando o terceiro símbolo, *d*, nós temos um novo intervalo [0.514, 0.52). Depois codificando o quarto símbolo, *a*, o novo intervalo é [0.514, 0.5146). E assim por diante até que após a codificação do último símbolo, *b*, tem-se o intervalo [0.5143876, 0.514402). A saída desta mensagem pode ser qualquer número que esteja dentro deste último intervalo. Para um melhor entendimento, o processo descrito acima está resumido na tabela 3.5.

Novo caracter	Limite inferior	Limite superior
	0.0	1.0
c	0.5	0.7
a	0.5	0.52
d	0.514	0.52
a	0.514	0.5146
c	0.5143	0.51442
d	0.514384	0.51442
b	0.5143876	0.514402

Tabela 3.5– Processo de codificação do exemplo de codificação aritmética.

Visualmente, pode-se representar o processo acima através da figura abaixo:

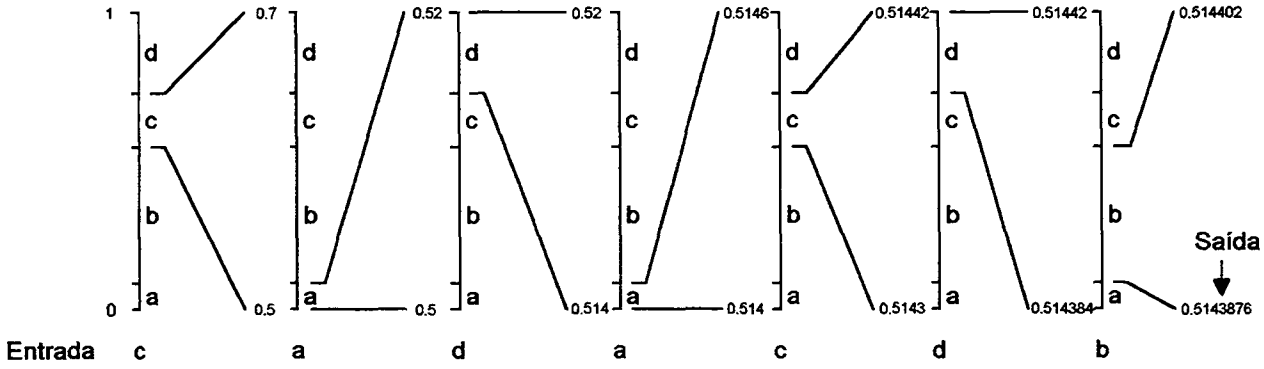


Figura 3.3 – Representação do processo de codificação aritmética.

Dado este esquema de codificação, é relativamente fácil ver como o processo de decodificação funciona. Inicialmente, o decodificador verifica em qual intervalo o primeiro símbolo da mensagem se encontra. Como 0.5143876 está dentro do intervalo [0.5, 0.7) o primeiro caracter da mensagem deverá ser o *c*. Como o limite inferior e superior de *c* é conhecido, que é [0.5, 0.7), reverte-se o processo que o gerou. Primeiro, subtrai-se pelo limite inferior de *c* que implicará em 0.0143876. Então se faz a divisão pelo tamanho da largura do intervalo de *c*, ou seja, 0.2. Isto resulta em 0,071938. Pode-se verificar que o próximo símbolo será o *a*, pois 0,071938 se encontra no intervalo [0, 0.1). Em seguida tem-se o valor

0,71938, que se encontra no intervalo [0.7, 1) resultando no símbolo *d*. E assim por diante, todos os símbolos da mensagem inicial serão recuperados a partir da saída 0,5143876 (tabela 3.6).

Número Codificado	Símbolo de Saída	Menor	Maior	Tam. Intervalo
0.5143876	*c*	0.5	0.7	0.2
0,071938	*a*	0.0	0.1	0.1
0,71938	*d*	0.7	1.0	0.3
0,0646	*a*	0.0	0.1	0.1
0,646	*c*	0.5	0.7	0.2
0,73	*d*	0.7	1.0	0.3
0,1	*b*	0.1	0.5	0.4
0.0				

Tabela 3.6 – Processo de decodificação do exemplo de codificação aritmética.

A ordem do modelo, refere-se ao número de símbolos anteriores que são usados para calcular a probabilidade de cada símbolo de entrada. Neste exemplo, foi utilizado o modelo de ordem-0 que calcula a probabilidade de cada símbolo independentemente de qualquer símbolo anterior.

3.5.4 Codificação Preditiva

As técnicas de compressão preditiva operam baseadas sobre um modelo estatístico da imagem. O modelo está embutido em algo conhecido como preditor. O preditor estima o próximo valor do pixel em uma determinada ordem, baseada nos pixels que antecederam aquele. O preditor pode também ser bidimensional para obter vantagem da correlação bidimensional inerente às imagens.

Por exemplo, um modelo muito simples prediz que o próximo *pixel* terá o mesmo valor do *pixel* prévio. Este modelo não é uma má suposição, uma vez que os valores dos *pixels* variam lentamente através de imagens em tom contínuo. A cada predição gerada, a diferença entre o *pixel* atual e o *pixel* predito é calculada. O desvio padrão do histograma das

diferenças é, em geral, menor que o da imagem original, refletindo que a correlação foi removida.

Assim, pode-se utilizar menos *bits* para codificar as diferenças do que o número de *bits* utilizado para representar um *pixel*. Quanto menor o número de *bits* utilizados nesta quantização, maior é o grau de perda de fidelidade da imagem, porém maior a taxa de compressão.

Os compressores preditivos podem ser adaptáveis, fazendo o quantizador e/ou o preditor adaptáveis. Esta capacidade de adaptação pode ser utilizada para conseguir maior compressão ou maior fidelidade [MEL94].

3.5.5 Codificação por Transformada

De acordo com [SAN01], a codificação por transformada envolve uma transformação linear do sinal de entrada e a quantização dos coeficientes, isto é, do resultado da transformação. O propósito da transformação é mudar a organização dos *pixels*.

Na codificação por transformada, geralmente divide-se uma dada imagem de tamanho NxN em um número de sub-imagens de tamanho $n \times n$, $n \leq N$, que são codificadas independentemente. A transformação é dita unidimensional quando a sub-imagem é de tamanho $1 \times n$ e bidimensional quando a imagem possui tamanho $n \times n$.

É importante ressaltar que a transformação em si não realiza a compressão. A compressão é conseguida pela quantização e pela posterior compressão sem perda (ver modelo genérico de compressão de imagens no item 3.1). A função da operação de transformação é a de obter coeficientes mais independentes e ordená-los (em termos de suas variâncias) de maneira que eles possam ser mais eficientemente quantizados.

A escolha da transformada não é determinada apenas pela correlação entre os coeficientes obtidos. Como a operação de transformação requer memória e tempo

computacional para sua implementação, esses parâmetros também são importantes para tal escolha.

Do ponto de vista da correlação entre os coeficientes, o problema é determinar a matriz de transformação descorrelacional ótima A, a matriz de transformação inversa (ou de reconstrução) B e o quantizador ótimo, de modo que a distorção media quadrática total seja minimizada. A solução do problema é tal que a matriz A é a transformada Karhunen-Loève (KL) da entrada, a matriz B é função da matriz inversa de A e o quantizador é o quantizador Lloyd-Max. Com tal quantizador, $B=A^{*T}$. A transformada KL, também chamada de Hotteling ou componente principal, apesar de ser ótima, é difícil de computar e não possui um algoritmo rápido para sua resolução. Ela é utilizada freqüentemente como parâmetro de referência para avaliar o desempenho de outras transformadas. Diversas transformadas são utilizadas na compressão de imagens e entre elas podemos ressaltar a transformada de Fourier, cosseno, seno, Walsh-Hadamard, Haar e Slant. As três primeiras são baseadas em senoides e as outras em ondas quadradas, que, em termos computacionais, são mais rápidas. Outros quantizadores mais simples, que não Lloyd-Max, também são utilizados na prática [JAI89, MEL94].

O processo de compressão para transformadas *wavelet* está ilustrado na figura 3.4. Nesta figura, foram acrescentadas duas novas etapas que não estão sendo consideradas neste trabalho e que têm a função de reconstruir as imagens de tomografia a partir dos senogramas, permitindo assim a sua análise.

As informações fornecidas pelo tomógrafo ao computador constitui de valores inteiros no formato binário. Estes valores variam de 0 a 65.535, ou seja, 16 *bits* não sinalizados. Dessa maneira, um senograma de 768 linhas por 90 colunas é armazenado no computador usando 138.240 *bytes*.

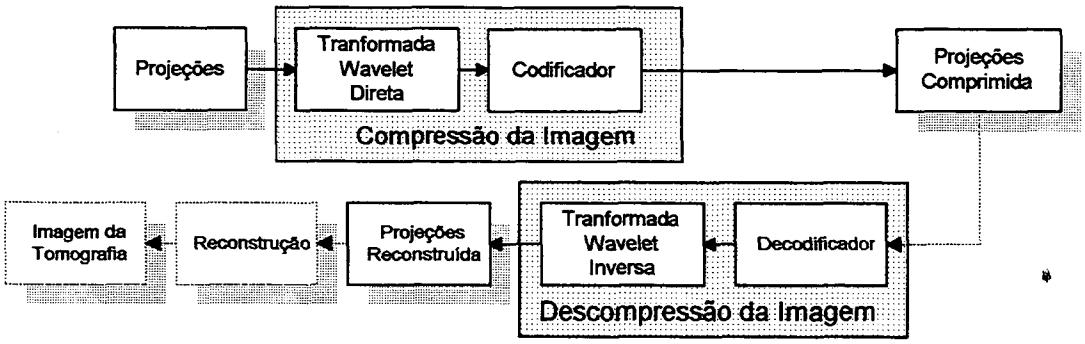


Figura 3.4 – Esquema de compressão usado nos experimentos.

Neste trabalho serão considerados apenas as transformadas *wavelet* de inteiros, isto é, a transformada S e S+P, as quais possuem a propriedade de preservação de precisão (PPP), comentada com maiores detalhes no item 1.3 do apêndice 1. Neste trabalho foram utilizadas as transformadas *wavelet* bidimensionais implementadas por [SAN01].

As imagens foram decompostas em 5 níveis de resolução, tanto nas linhas quanto nas colunas. Em cada nível de decomposição, o número de coeficientes são reduzidos pela metade, pelo fato de que apenas os coeficientes passa baixa serão usados na aplicação da transformada no próximo nível de resolução. Como as imagens possuem 90 linhas, 5 níveis de decomposição são suficientes para reduzir para apenas 3 *pixels*, quando a transformada bidimensional é aplicada nas colunas da imagem. Na fase de codificação, foi utilizada a codificação aritmética.

3.5.6 Codificação Lossless JPEG com Huffman

Alguns trabalhos descrevem a codificação Lossless JPEG [HUA94, NET98]. O padrão de compressão para imagens paradas ISO JPEG é popularmente conhecido como uma técnica de compressão baseada em transformada discreta cosseno (DCT - *Discrete Cosine Transform*). Entretanto, o modo de operação sem perdas não utiliza a DCT, mas sim uma codificação por predição. A técnica de compressão *Lossless* JPEG utiliza a forma de

modulação de código de pulso discreto (DPCM - *Discrete Pulse Code Modulation*) [PEN93]. Considere uma imagem fonte com duas dimensões. Cada amostra é um *pixel* contendo ou três *bytes* representando os níveis vermelho, verde ou azul (RGB) para uma imagem colorida, ou um único *byte* de uma imagem em escalas de cinza. Na técnica *Lossless* JPEG cada *pixel* é especificado como um inteiro variando de 2 a 16 *bits* de precisão. O parâmetro ponto de precisão, *Pt*, pode ser usado para reduzir o nível de quantização da imagem fonte. Se *Pt* não for zero, cada amostra de entrada é deslocada à direita por *Pt bits*.

O padrão JPEG define oito valores de seleção de predição (PSV - *Prediction Selection Values*), os quais são especificados na tabela 3.7. O PSV 0 (zero) é reservado para codificação diferencial no modo progressivo hierárquico (não utilizado neste codificador *lossless*). Para cada *pixel*, $P_{x,y}$, uma combinação linear dos *pixels* vizinhos esquerdo ($P_{x,y-1}$), superior ($P_{x-1,y}$) e superior esquerdo ($P_{x-1,y-1}$), são utilizados para calcular o preditor de $P_{x,y}$. Os PSVs 1, 2 e 3 são preditores de uma dimensão, ou seja, usam somente um *pixel* vizinho. Os PSVs 4, 5, 6 e 7 são bidimensionais, por utilizarem dois ou mais *pixels* vizinhos.

PSVs	**Preditores**
0	sem predição (codificação diferencial)
1	$P_{x,y-1}$
2	$P_{x-1,y}$
3	$P_{x-1,y-1}$
4	$P_{x,y-1} + P_{x-1,y} - P_{x-1,y-1}$
5	$P_{x,y-1} + (P_{x-1,y} - P_{x-1,y-1}) / 2$
6	$P_{x-1,y} + (P_{x,y-1} - P_{x-1,y-1}) / 2$
7	$(P_{x,y-1} + P_{x-1,y}) / 2$

Tabela 3.7 - Preditores para o *Lossless* JPEG

A primeira linha e coluna da imagem são tratadas de forma especial. O PSV 1 é utilizado para a primeira linha de amostras e o PSV 2 é utilizado para a primeira coluna. Para o *pixel* no canto superior esquerdo da imagem, o preditor 2^{P-Pt-1} é utilizado, onde P é o número de *bits* da amostra. O preditor calculado é subtraído do valor do *pixel* e a diferença é codificada por Huffman ou Codificação de entropia Aritmética. No caso deste codificador, é utilizado a codificação de Huffman. O modelo de entropia de Huffman particiona as diferenças em 16 diferentes categorias. A cada uma destas categorias é atribuído um código de Huffman usando ou a tabela padrão de Huffman definida na norma ISO/IEC DIS 10918-1, ou uma tabela customizada para a imagem. As diferentes categorias são mostradas na tabela 3.8, abaixo:

Categoria	Valores
0	0
1	-1, 1
2	-3, -2, 2-, 3
3	-7...-4, 4...7
4	-15...-8, 8...15
5	-31...-16, 16...31
6	-63...-32, 32...63
7	-127...-64, 64...127
8	-255...-128, 128...255
9	-511...-256, 256...511
10	-1023...-512, 512...1023
11	-2047...-1024, 1024...2047
12	-4095...2048, 2048...4095
13	-8191...-4096, 4096...8191
14	-16383...-8192, 8192...16383
15	-32767...-16384, 16384...32767

Tabela 3.8 - Tabela de categorias e valores de diferença

Para qualquer valor de diferença, uma sequência de *bits* adicionais são anexados no código de Huffman para identificar unicamente qual a diferença naquela categoria atualmente ocorrida. A quantidade de *bits* é dada pela própria categoria, com zero sendo um valor permitido. As regras de como os *bits* adicionais são especificados estão a seguir. Se a diferença é positiva, anexa-se a quantidade de *bits*, representada pela categoria, de mais baixa ordem da diferença. Se a diferença é negativa, subtrai-se um da diferença e anexa-se a quantidade de *bits*, representada pela categoria, de mais baixa ordem do valor resultante.

O exemplo a seguir ilustra o processo de codificação. A imagem possui 2 linhas por 3 colunas em escalas de cinza (8 *bits*). Foi utilizado o PSV 7 ($(P_{x,y-1} + P_{x-1,y}) / 2$) como preditor e a tabela de codificação Huffman, definida no padrão JPEG, para as diferentes categorias. *Pt* foi configurado como zero. O preditor para o *pixel* do canto superior esquerdo é igual a 128 (aplicando-se 2^{P-Pt-1}).

Valores dos *pixels* da imagem		
120	100	100
80	90	100

Diferença calculada (*pixel* - preditor)		
-8	-20	0
-40	0	5

Categoria e respectivos *bits* de Huffman		
4 (101)	5 (110)	0
6 (1110)	0	3 (100)

***Bits* adicionais**		
0111	01011	0
010111	0	101

Sequência de *bits* de saída (espaços não fazem parte)					
101 0111	110 01011	0	1110 010111	0	100 101

Para a decodificar a sequência de *bits* acima, primeiramente é extraído o código de Huffman 101, onde a decodificação de Huffman traz o símbolo de categoria 4. A seguir, os 4 *bits* adicionais, 0111, são extraídos da sequência. O *bit* inicial 0 significa que a diferença entre o *pixel* e o preditor é um número negativo, portanto o valor da diferença é 1 somado ao binário 0111. Adicionando ao valor reconstruído, -8, o preditor 128 (2^{P-Pt-1}), o valor do *pixel* 120 é recuperado. Este processo inverso de codificação é repetido até que a sequência inteira dos *bits* seja decodificada.

4. Metodologia

4.1 Introdução

Neste capítulo, será descrita a metodologia utilizada na execução das simulações realizadas neste trabalho. Foram estudados e avaliados os métodos de compressão de dados propostos, sendo aplicados em imagens e projeções de tomografia computadorizada por raios-X (chamados senogramas). O objetivo é investigar o comportamento de cada um desses métodos sobre estas imagens e senogramas, assinalando suas vantagens e desvantagens, em termos de taxa de compressão.

Foram comparados alguns dos métodos mais conhecidos em compressão: GZIP (Gnu ZIP), utilizado largamente para compressão de dados e LJPG (Lossless JPEG), utilizado especificamente para compressão de imagens; e dois outros métodos de recente utilização em compressão de imagens: Transformada S (*Wavelet* de Haar) e Transformada S+P (Said e Pearlman), abordados em [SAN01].

Além da utilização de senogramas reais, gerados por tomógrafo industrial, foram utilizados também senogramas construídos por um programa simulador de projeções, conforme ilustra a figura 4.1. Este programa simulador de projeções foi implementado por Medeiros [MED01]. Os senogramas foram gerados a partir de imagens médicas, com 256 colunas por 256 linhas e 8 bpp. O programa simulador de projeções efetua uma varredura de 180 graus na imagem com um passo de 2 graus, gerando, desta forma, senogramas compostos de 256 linhas por 90 colunas e 16 bpp. Tanto as informações fornecidas pelo tomógrafo quanto as gerados pelo programa simulador de projeções são constituídas de valores inteiros no formato binário. Estes valores variam de 0 a 65.535, ou seja, 16 *bits* não sinalizados.

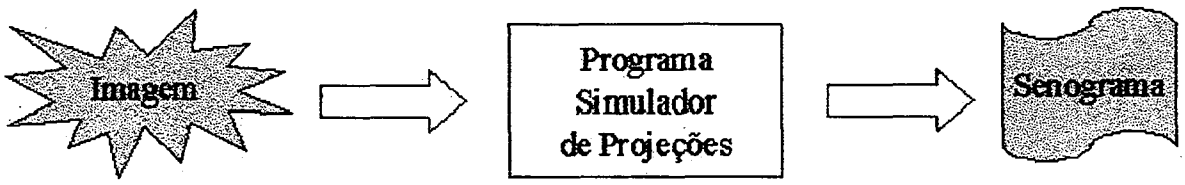


Figura 4.1 – Esquema de geração de senogramas pelo programa simulador de projeções.

As imagens e senogramas foram submetidos aos métodos de compressão, conforme ilustra a figura 4.2.

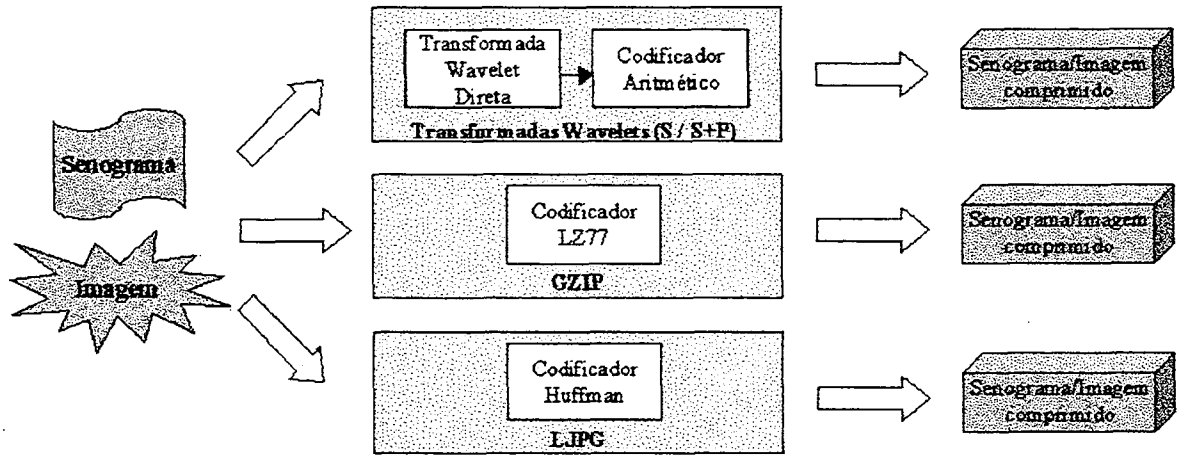


Figura 4.2 – Métodos de compressão utilizados para as imagens e senogramas.

Durante a realização dos experimentos, verificou-se que os senogramas reais apresentavam um elevado grau de variação nos valores dos *pixels*, quando comparados aos senogramas gerados pelo programa simulador. Através da análise destes valores, constatou-se que esta variação comporta-se como ruídos distribuídos de forma aleatória entre os *pixels*. Face a estas constatações, optou-se pela continuidade dos estudos considerando estas características dos senogramas reais. Foi utilizada a função *rand()*, em linguagem C, para gerar o ruído uniforme de média nula, a ser somado a cada *pixel* das imagens e senogramas, com o objetivo de se estudar o comportamento dos diversos métodos de compressão

utilizados nos experimentos. Os ruídos foram inseridos de forma percentual, conforme a fórmula a seguir:

$$\%ruído = \frac{AmplitudeRuido}{AmplitudeSinal} \tag{4.1}$$

No cálculo do *%ruído*, o valor do ruído, que pode resultar num valor positivo ou negativo, é somado ao *pixel*. Caso o valor resultante (valor do *pixel* + valor do ruído) seja maior que 65.535 ou menor que 0, o valor do ruído será rebatido, ou seja, terá o sinal invertido para ser somado ao *pixel*.

Além do percentual de ruído, foi incluída uma outra medida relacionada a ruído, que é a relação sinal / ruído (*SR*), calculada pela seguinte fórmula:

$$SR = \sqrt{\frac{\sum(sinal)^2}{\sum(ruído)^2}} \tag{4.2}$$

Para comparação entre os métodos de compressão, foi utilizada a taxa de compressão, que é a razão entre o tamanho da imagem comprimida e o tamanho da imagem original. Está comentada em mais detalhes no item 3.3, sendo a fórmula utilizada a seguinte:

$$\left(\frac{TamanhoOriginal - TamanhoFinal}{TamanhoOriginal}\right) * 100 \tag{4.3}$$

4.2 Imagens e Senogramas Utilizados

Foram utilizados 3 senogramas reais, gerados no tomógrafo industrial do laboratório da UFPR. Os senogramas são compostos de 768 colunas por 90 linhas e 16 bpp, totalizando 138.240 *bytes*. Estes senogramas foram designados por "dente01", dente02" e "dente03" sendo todos eles tomografia computadorizada de um dente humano. A partir desses senogramas, foram reconstruídas as imagens "dente01", composta de 256 colunas por

256 linhas e 8bpp; "dente02", composta de 176 colunas por 176 linhas e 8bpp e "dente03" composta de 112 colunas por 112 linhas e 8bpp. Os senogramas e suas respectivas imagens são apresentados no apêndice 4.

Utilizou-se uma imagem gerada artificialmente, denominada "Phantom", mostrada na figura 4.3. Essa imagem é semelhante a tomografia de uma cabeça [JAI89]. Esta imagem foi gerada por um programa, utilizado em [MED01]. Este programa também produz o respectivo senograma, mostrada na figura 4.4. Este programa também permite produzir senogramas de qualquer outra imagem, os quais serão produzidos e utilizados neste trabalho.

Figura 4.3 - Imagem "Phantom" gerada por software.

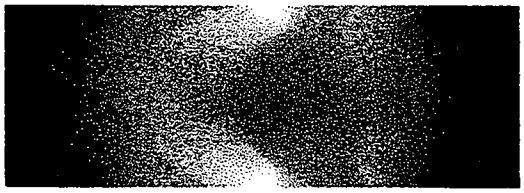


Figura 4.4 - Senograma do "Phantom" gerada por software.

Selecionadas ainda outras 5 imagens para realização dos testes, as quais foram obtidas de repositórios de dados de universidades, já no formato utilizado neste trabalho, ou

seja, com 256 colunas por 256 linhas e 8 bpp, regeradas a partir de uma projeção de TC (Tomografia Computadorizada), conforme relacionadas a seguir:

1. Imagem "Abdom", tomografia de um abdome;
2. Imagem "Brain", tomografia de um cérebro;
3. Imagem "Hernia", tomografia de uma hérnia da coluna;
4. Imagem "Head", tomografia da cabeça;
5. Imagem "Torax", tomografia de um tórax.

Utilizando o programa gerador de projeções, foi gerado um senograma para cada uma destas imagens, os quais também foram utilizados nos experimentos.

Foram geradas outras dezessete imagens da seguinte forma: sete imagens a partir da imagem "Abdom" com os percentuais de ruído (%ruído) 2%, 10%, 20%, 25%, 33%, 50% e 100%; e outras dez imagens a partir do senograma "Abdom" com os percentuais de ruído 0,1%, 0,2%, 1%, 2%, 10%, 20%, 25%, 33%, 50% e 100%. Portanto, foram utilizadas nos experimentos deste trabalho um total de trinta e duas imagens e/ou senogramas. Estas imagens e senogramas estavam inicialmente no formato BMP sem o uso da compactação RLE, padrão deste formato.

Para a utilização na compressão com os métodos transformada S, transformada S+P e GZIP foi necessário que as imagens e senogramas estivessem no formato binário (RAW), portanto, tiveram os cabeçalhos do formato BMP removidos, permanecendo somente os *bytes* de dados. As imagens permaneceram com 256 colunas (largura) por 256 linhas (altura) e 8 *bits* por *pixel*, totalizando 65.536 *bytes* e os senogramas com 256 colunas (largura) por 90 linhas (altura) e 16 *bits* por *pixel*, totalizando 46.080 *bytes*.

O método de compressão LJPG, utilizado nestes experimentos, trabalha somente com imagens no formato PGM. Portanto, para possibilitar a utilização do método, as imagens

do formato binário foram transformadas para este formato com a inclusão de um cabeçalho próprio com tamanho de 14 *bytes*.

As imagens e senogramas foram submetidos aos métodos de compressão propostos neste trabalho, sendo os resultados, em forma de tabelas, gráficos e comentários, apresentados no próximo capítulo. Todas as imagens e senogramas utilizados neste trabalho estão relacionadas no apêndice 4.

4.3 Métodos de Compressão

Foram utilizadas as transformadas *wavelet* bidimensionais [SAN01], escritas em linguagem C (Borland C++ versão 3.1) no sistema operacional DOS e, com pequenas modificações, também implementadas no sistema operacional LINUX COREL / OS. Constatou-se que os resultados obtidos com a compactação nos dois sistemas foram idênticos. O programa permite a entrada de diferentes tamanhos de imagens. Permite também optar pelo número de níveis de decomposição desejado. Foram desenvolvidos dois programas para serem utilizados conforme a organização da imagem ou senograma, ou seja, um programa para a unidade de informação de 1 *byte* (8 bpp) e outro para 2 *bytes* (16 bpp).

Para o cálculo da transformada S e transformada S+P, primeiramente a transformada é aplicada levando-se em consideração cada informação na sua forma original, ou seja, 8 *bits* para as imagens e 16 *bits* para os senogramas. Após a aplicação da transformada, o número de *bits* da informação se mantém o mesmo, em virtude da PPP, comentada no item 1.3 do apêndice 1. Após a aplicação da transformada, foi utilizada a codificação Aritmética, ao invés da codificação de Huffman, por ter apresentado melhor resultado em todas as imagens e senogramas utilizados, conforme constatado em [SAN01].

Os testes de compressão com o GZIP foram realizados no sistema operacional LINUX COREL/OS. As imagens e senogramas submetidos à compressão possuíam somente

os *bytes* de dados, ou seja, sem nenhum tipo de cabeçalho específico. O GZIP trabalha com o codificador LZ77, considerando a unidade de informação o *byte*, não importando se a imagem ou senograma sejam organizados em 8 ou 16 bpp.

Os testes de compressão com o LJPG foram realizadas no sistema operacional LINUX COREL / OS. O software utilizado foi o LJPG [HUA94], desenvolvido para uso experimental e de distribuição livre. Neste caso, as imagens e senogramas submetidos à compressão, por exigência do software, necessitavam possuir um cabeçalho no formato PGM (*Portable GrayMap*), tendo um acréscimo de 14 *bytes*. O software foi desenvolvido para comprimir arquivos com 8 bpp. Nos testes deste trabalho, utiliza-se senogramas com 256 colunas por 90 linhas e 16 bpp, totalizando 46.080 *bytes*. Para possibilitar a compressão destes senogramas com o método LJPG, foi utilizado o artifício de gerar o cabeçalho do senograma com as informações de 512 colunas por 90 linhas e 8 bpp, totalizando os mesmos 46.080 *bytes*, portanto, sem alterar o tamanho do senograma.

Ressalta-se que este fato do programa codificador interpretar um *pixel* de 16 *bits* como sendo 2 *pixels* de 8 *bits* deve afetar o desempenho deste método de compressão.

5. Resultados

5.1 Compressão de Senogramas

Os 3 senogramas relativos aos dentes, obtidos no tomógrafo industrial do LACTEC na UFPR, foram comprimidos com os métodos de compressão Transformada S, Transformada S+P, GZIP e LJPG. Os resultados, com as taxas de compressão, estão representadas na tabela 5.1. Estes resultados estão de acordo com os resultados apresentados no trabalho de [SAN01], porém tem-se que destacar a abordagem utilizada no levantamento destes dados. Os senogramas possuem 16 *bits* por *pixel* (bpp). Foi utilizada a técnica de separação do senograma em dois outros arquivos, o primeiro contendo a informação da parte alta do *pixel* original, ficando com 8 *bits* por *pixel* e o segundo contendo a informação da parte baixa, também ficando com 8 *bits* por *pixel*. Desta forma, melhores resultados foram obtidos com a transformada *wavelet* de inteiros nos casos com a transformada S+P computando com 16 *bits* e S computando com 8 *bits*, assim como os resultados obtidos com o LJPG (*lossless* JPG) e o GZIP.

A coluna Entropia, a qual faz parte da tabela 5.1, corresponde ao tamanho do arquivo comprimido que pode ser obtido levando em consideração apenas a redundância de código baseado no cálculo da entropia de primeira ordem [SAN01]. A coluna Aritmética corresponde somente a aplicação da codificação aritmética.

Senograma	Entropia	Aritmética	Transf. S	Transf. S+P	GZIP	LJPG
Dente01	8%	8%	31%	27%	9%	32%
Dente02	8%	8%	33%	29%	9%	33%
Dente03	7%	8%	33%	28%	9%	33%
Média	8%	8%	32%	28%	9%	33%
D. Padrão	1%	0%	1%	1%	0%	1%

Tabela 5.1 - Taxa de compressão de senogramas reais, separados parte alta e parte baixa.

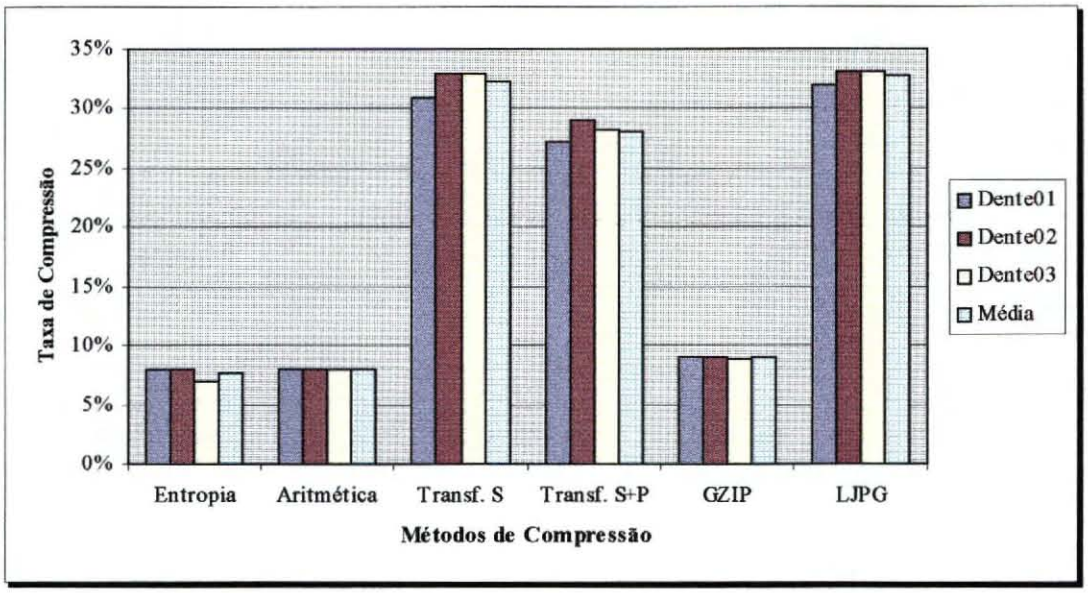


Figura 5.1 - Gráfico comparativo da compressão dos senogramas reais, separados em *8 bits*.

Pode-se observar, no gráfico da Figura 5.1, que as taxas de compressão das transformadas S são superiores às taxas geradas pelo método GZIP. O método LJPG, mesmo operando sobre um conjunto de 8 *bits*, neste caso, apresentou taxas compatíveis com as apresentadas pela transformada S.

Um novo levantamento foi realizado considerando outra abordagem, que é a utilizada neste trabalho. Os senogramas foram utilizados considerando o *pixel* com 16 *bits*. Portanto, os 3 senogramas relativos aos dentes, obtidos no tomógrafo industrial, foram comprimidos com os métodos de compressão Transformada S, Transformada S+P, GZIP e LJPG, sem a separação em arquivos com parte alta e parte baixa. A transformada foi aplicada levando em consideração cada informação na sua forma original, ou seja, com 16 *bits*.

Os resultados, com as taxas de compressão, estão representadas na tabela 5.2. A taxa de compressão relativa a entropia de 1ª ordem dos senogramas também foram calculadas para efeito de comparação.

Senograma	Tamanho	Entropia	Aritmética	Transf. S	Transf. S+P	GZIP	LJPG
Dente01	138240	8%	7%	25%	27%	9%	-1%
Dente02	138240	8%	7%	27%	29%	9%	-1%
Dente03	138240	7%	7%	27%	28%	9%	0%
Média	138240	8%	7%	27%	28%	9%	-1%
D. Padrão	0	0%	0%	1%	1%	0%	1%

Tabela 5.2 - Compressão de senogramas reais, com 16 bpp, entre diferentes métodos.

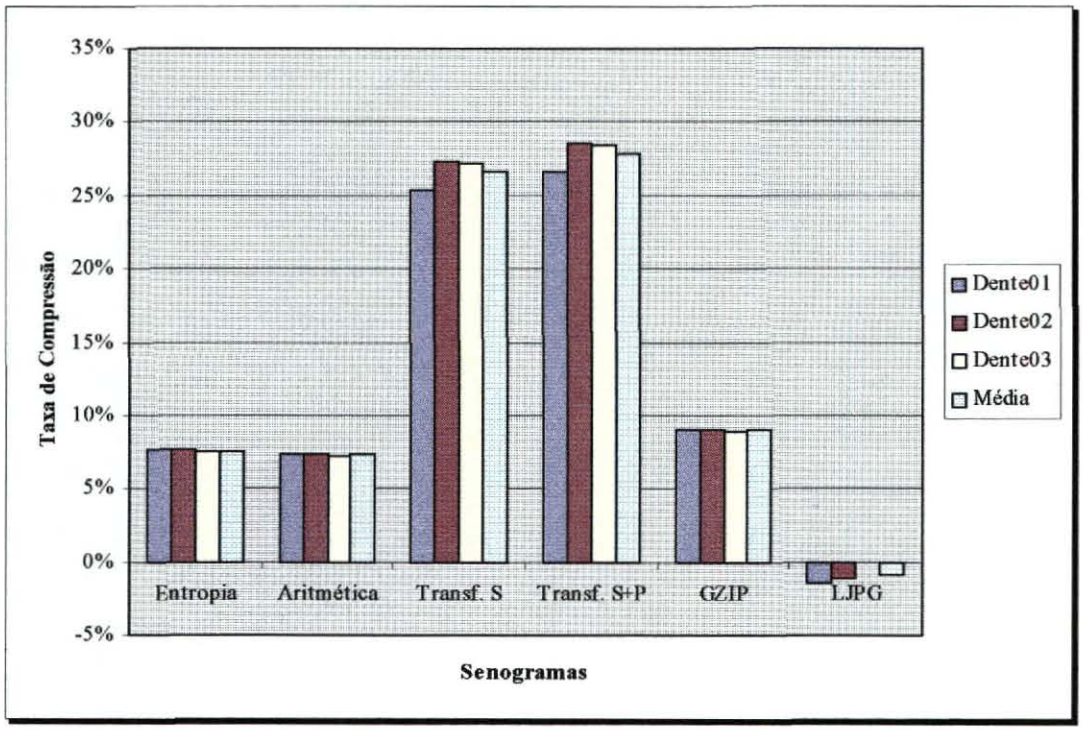


Figura 5.2 - Gráfico comparativo da compressão dos senogramas reais com 16 bpp.

Pode-se verificar que as taxas de compressão, nesta segunda abordagem, estão um pouco menores para as transformadas *wavelets* e bem inferiores para o método LJPG. Isto se explica pelo fato desta implementação do LJPG não ter trabalhado com os *pixels* de 16 *bits*, mas sim com os *bytes* altos e baixos alternadamente, o que compromete o bom desempenho do preditor usado nesta técnica.

Os experimentos com os demais senogramas foram realizados considerando a segunda abordagem, ou seja, os métodos de compressão foram aplicados levando em

consideração o tamanho original do pixel, com 16 *bits*, exceto para o LJPG que operou sempre em 8 *bits* sobre os *bytes* alto e baixo alternadamente.

Na Tabela 5.3, são apresentados os resultados entre os métodos para seis senogramas gerados pelo programa simulador de projeções. Os pontos do gráfico, entre os senogramas, foram ligados por retas para as comparações entre os métodos.

Senograma	Entropia	Aritmética	Transf. S	Transf. S+P	GZIP	LJPG
Abdom	37%	37%	46%	24%	89%	61%
Braim	47%	47%	46%	30%	87%	57%
Hernia	34%	34%	42%	20%	87%	58%
Head	47%	47%	51%	33%	88%	67%
Phanton	42%	42%	51%	28%	91%	72%
Torax	38%	38%	45%	23%	88%	59%
Média	41%	41%	47%	26%	88%	62%
D. Padrão	5%	5%	3%	5%	2%	6%

Tabela 5.3 - Compressão dos senogramas simulados.

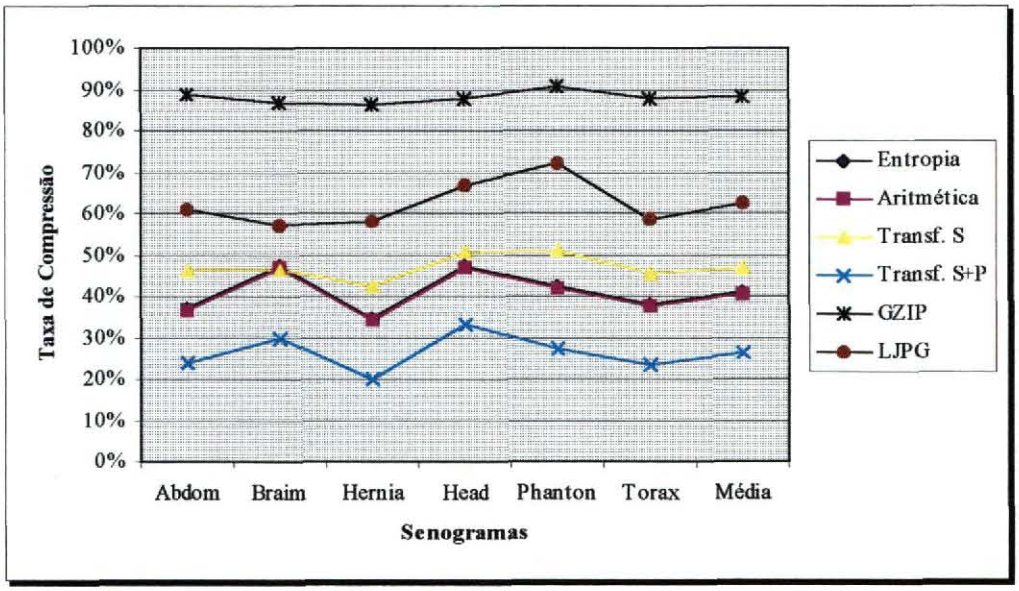


Figura 5.3 - Gráfico comparativo dos senogramas comprimidos

Pode-se observar que a compressão realizada pela codificação aritmética pura obteve um resultado diferente das compressões pelas transformadas *wavelets*, embora estas também utilizem a codificação aritmética. Pode-se observar também que as taxas de

compressão das transformadas S e S+P estão inferiores às taxas conseguidas pelos métodos de uso geral GZIP e LJPG. Esta característica é explicada pela organização dos *pixels* dos senogramas, onde possuem uma baixa variação de ruídos nas imagens.

Isto pode ser devido ao fato destes senogramas terem sido obtidos (simulados) a partir de imagens de 8 *bits*, ao contrário dos senogramas dos dentes, que foram obtidos diretamente de um tomógrafo industrial. Isto mostra que os senogramas simulados, da forma como estão, não representam bem a situação real. Numa situação real, o objeto tem uma variação contínua da radiopacidade e pode existir ainda um ruído presente durante a medida.

5.2 Compressão de Senograma com Escalas de Ruídos

Para que os senogramas simulados representassem mais fielmente a situação real, foi introduzida uma quantidade de ruído controlado em cada um de seus *pixels*. A tabela a seguir apresenta o senograma "Abdom" comprimido com inserção de ruídos. O percentual de ruído (*%ruído*) representa a relação entre o maior valor de *pixel* e o maior valor de ruído. A tabela 5.3 apresenta as taxas de compressão para o senograma "Abdom", com ruídos aleatórios somados aos *pixels* nos percentuais indicados. O item 4.1 mostra maiores detalhes sobre o processo de inserção aleatória de ruídos. O senograma "Abdom" possui 256 colunas por 90 linhas, com 16 bpp, totalizando 46.080 *bytes*.

% ruído	SR	Entropia	Aritmética	Transf. S	Transf. S+P	GZIP	LJPG
0%	-	37%	37%	46%	24%	89%	61%
0,1%	826,96	13%	12%	25%	21%	23%	29%
0,2%	406,58	12%	11%	21%	21%	17%	21%
1%	80,39	7%	6%	18%	19%	7%	12%
2%	40,31	5%	4%	17%	17%	5%	10%
10%	8,02	3%	3%	9%	8%	3%	2%
20%	4,02	3%	2%	5%	5%	2%	-2%
25%	3,23	3%	2%	4%	4%	2%	-3%
33%	2,43	3%	2%	3%	3%	2%	-4%
50%	1,61	2%	2%	2%	1%	2%	-6%
100%	0,80	1%	0%	0%	0%	0%	-7%

Tabela 5.4 - Compressão para o senograma "Abdom" com inserção de ruído

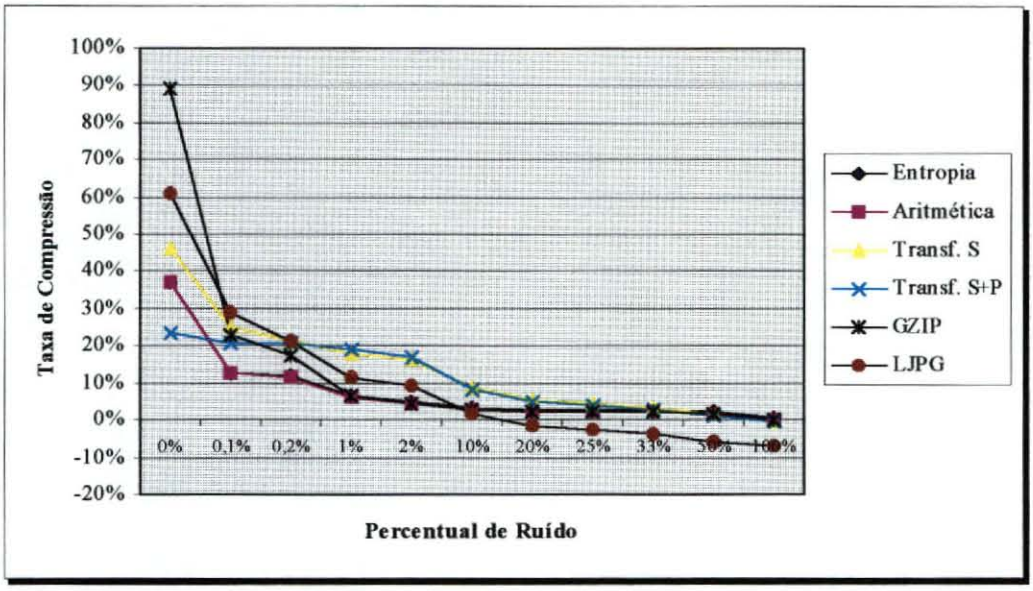


Figura 5.4 - Gráfico comparativo do senograma "Abdom" com escala percentual de ruído.

Ao se analisar os resultados, pode-se verificar que a partir do percentual de 0,2% de ruído, os métodos que utilizam a transformada *wavelet* passam a ter taxa de compressão superior em relação aos métodos de uso geral GZIP e LJPG. Isto mostra a importância de se considerar o ruído presente no senograma no desempenho da codificação.

5.3 Compressão dos Senogramas com Ruído Padrão

Nesta seção estão os resultados da compressão de senogramas com um percentual padrão de 1% de ruído, para se ter uma idéia do comportamento dos métodos de compressão com os outros senogramas. Estima-se que um senograma real possa ter um percentual de ruído proveniente do processo de detecção numa faixa de 0,01% a 10%, dependendo da qualidade do equipamento.

A tabela 5.5 mostra as taxas de compressão para os 6 senogramas simulados, com ruídos inseridos e os 3 senogramas reais. Para os senogramas reais não se têm os valores da

relação sinal/ruído (SR) porque não sofreram inserção de ruído controlado, visto que estão no estado original, já com ruídos inseridos pelo próprio processo de geração.

Senograma	SR	Entropia	Aritmética	Transf. S	Transf. S+P	GZIP	LJPG
Abdom	80,39	7%	6%	18%	19%	7%	12%
Brain	88,85	11%	11%	17%	18%	11%	11%
Hernia	89,65	5%	4%	17%	18%	5%	11%
Head	83,81	11%	11%	19%	20%	11%	13%
Phanton	88,55	9%	8%	19%	20%	9%	14%
Torax	81,57	7%	6%	18%	19%	7%	11%
Dente01	-	8%	7%	25%	27%	9%	-1%
Dente02	-	8%	7%	27%	29%	9%	-1%
Dente03	-	7%	7%	27%	28%	9%	0%
Média	85,47	8%	8%	21%	22%	8%	8%
D. Padrão	4,05	2%	2%	4%	4%	2%	7%

Tabela 5.5 - Compressão do senogramas com ruído padrão.

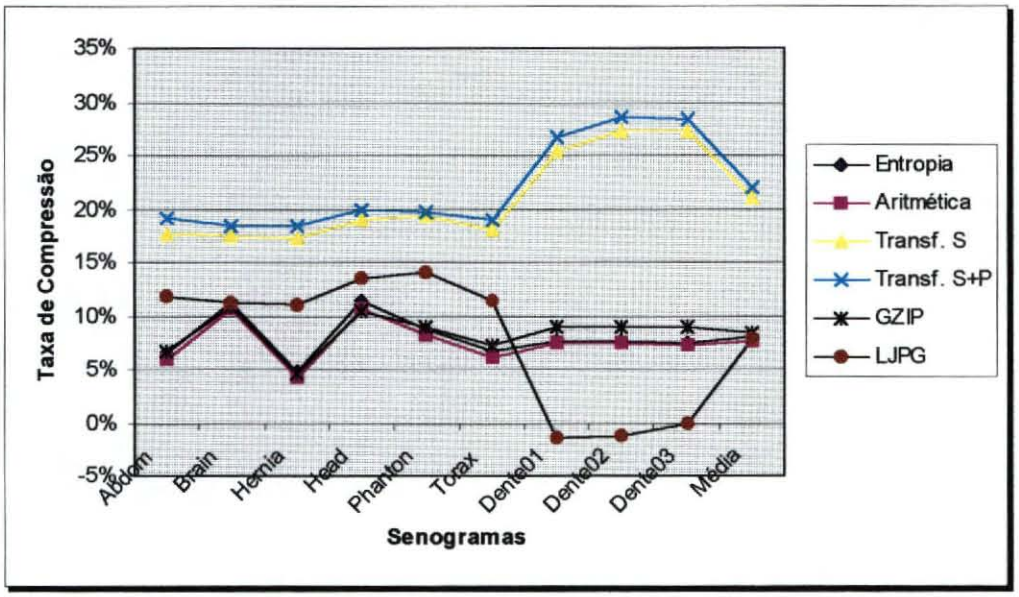


Figura 5.5 - Gráfico comparativo dos senogramas com ruído padrão.

Neste caso, pode-se notar que os métodos que utilizam a transformada *wavelet* apresentam uma taxa de compressão superior aos demais métodos. Os resultados são semelhantes para todas os senogramas aqui utilizados.

5.4 Compressão de Imagens

A Tabela 5.6 apresenta os resultados da compressão entre os métodos aqui utilizados, para seis imagens de TC, todas com 256 colunas por 256 linhas e com 8 *bits* por *pixel*, totalizando 65.536 *bytes*.

Imagem	Entropia	Aritmética	Transf. S	Transf. S+P	GZIP	LJPG
Abdom	63%	61%	55%	51%	80%	70%
Brain	74%	71%	66%	61%	87%	77%
Hernia	46%	45%	40%	38%	61%	50%
Head	75%	71%	66%	60%	86%	74%
Phanton	79%	78%	79%	65%	98%	86%
Torax	57%	55%	53%	49%	75%	66%
Dente01	36%	34%	33%	34%	34%	34%
Dente02	33%	32%	47%	48%	38%	50%
Dente03	28%	26%	40%	40%	28%	44%
Média	54%	53%	53%	50%	65%	61%
D. Padrão	19%	19%	15%	11%	26%	17%

Tabela 5.6 - Compressão das imagens de TC.

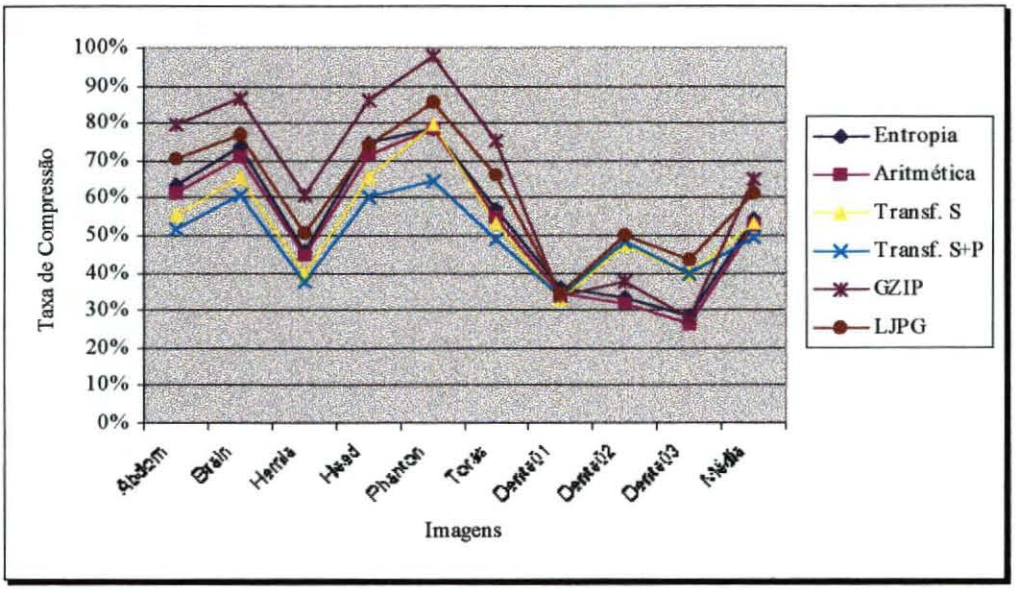


Figura 5.6 - Gráfico comparativo das imagens de TC comprimidas.

Neste caso, pode-se observar que as taxas de compressão das transformadas S e S+P estão inferiores às taxas conseguidas pelos métodos de uso geral.

A tabela 5.7, a seguir, mostra as imagens comprimidas com ruídos, adicionados antes da compressão. O item 4.1 mostra maiores detalhes sobre o processo de inserção aleatória de ruídos. A imagem "Abdom" possui 256 colunas por 256 linhas, com 8 bpp, totalizando 65.536 *bytes*.

% ruído	SR	Entropia	Aritmética	Transf. S	Transf. S+P	GZIP	LJPG
0%	-	63%	61%	55%	51%	80%	70%
2%	85,87	49%	47%	49%	48%	59%	57%
10%	9,31	27%	26%	35%	34%	28%	36%
20%	4,40	19%	18%	25%	23%	19%	25%
25%	3,47	16%	16%	22%	20%	16%	21%
33%	2,63	13%	13%	18%	15%	13%	17%
50%	1,71	11%	11%	11%	8%	10%	9%
100%	0,85	4%	4%	1%	1%	3%	-2%

Tabela 5.7 - Compressão para a imagem "Abdom" com inserção de ruído

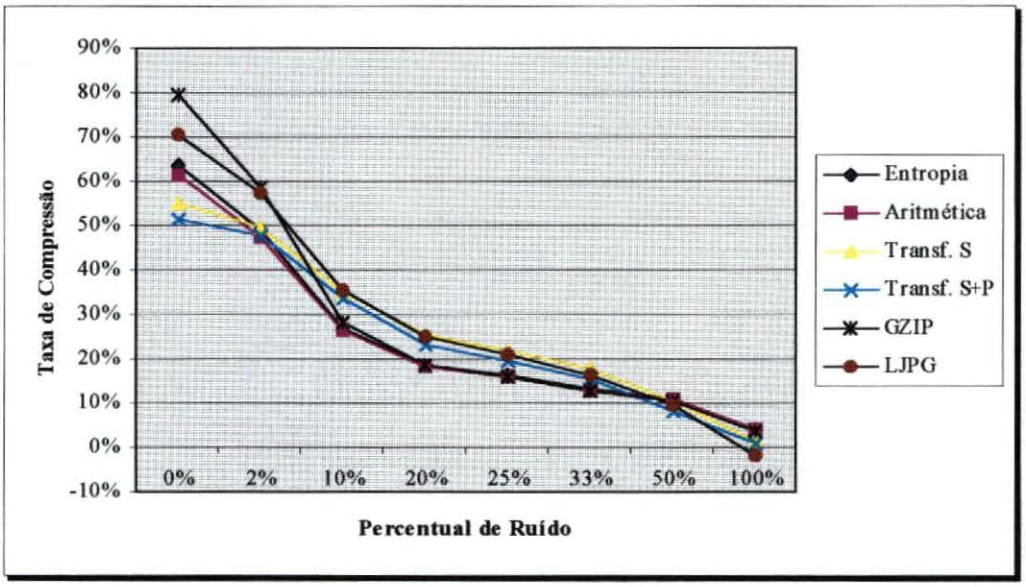


Figura 5.7 - Gráfico comparativo da imagem "Abdom" com escalas de ruído.

Neste caso, pode-se verificar que para um percentual de ruído em torno de 10%, os métodos que utilizam a transformada *wavelet* passam ter taxas de compressão maiores do que as dos métodos de uso geral GZIP e LJPG.

5.5 Comparativo de Compressão Imagens x Senogramas

Com o objetivo de avaliar o que é mais eficiente comprimir, apresenta-se um comparativo entre a compressão das imagens e senogramas. Vale lembrar que são dois domínios distintos, com dimensões e resolução de cores diferentes (8bpp e 16bpp), portanto é um cenário de difícil comparação quantitativa. Comparando-se o tamanho original, os senogramas possuem a vantagem de, em geral, serem menores, porém também podem ter tamanhos maiores. Neste trabalho, serão utilizados somente senogramas gerados com um passo de 2 graus, o que já fornece uma imagem adequada. Uma melhor qualidade pode ser conseguida diminuindo-se o passo. Se fosse utilizado um passo de 1 grau, seria gerado um senograma com 180 linhas, totalizando 92.160 *bytes*, portanto maior que os 65.536 *bytes* da imagem.

A tabela 5.8 mostra a taxa de compressão, com inserção de ruídos (1%) em senogramas. O cálculo desta taxa considera os tamanhos finais das imagens e senogramas, comparado com o tamanho original da imagem de 65.536 *bytes*. As imagens utilizadas estão sem ruídos inseridos e possuem 8 *bits* por *pixel* de informação. Os senogramas possuem 16 *bits* por *pixel* de informação.

	Normal	Entr	Aritm	S	S+P	GZIP	LJPG	Entr	Aritm	S	S+P	GZIP	LJPG
Imagem	Sen	Sen	Sen	Sen	Sen	Sen	Sen	Img	Img	Img	Img	Img	Img
Abdom	30%	34%	34%	42%	43%	34%	38%	63%	61%	55%	51%	80%	70%
Braim	30%	38%	37%	42%	43%	37%	38%	74%	71%	66%	61%	87%	77%
Hernia	30%	33%	33%	42%	43%	33%	37%	46%	45%	40%	38%	61%	50%
Head	30%	38%	37%	43%	44%	37%	39%	75%	71%	66%	60%	85%	74%
Phanton	30%	36%	35%	43%	44%	36%	40%	79%	78%	79%	65%	98%	86%
Torax	30%	34%	34%	42%	43%	35%	38%	57%	55%	53%	49%	75%	66%
Média	30%	35%	35%	42%	43%	35%	38%	65%	64%	60%	54%	81%	71%
D. Padrão	0%	2%	2%	1%	0%	2%	1%	13%	12%	14%	10%	12%	12%

Tabela 5.8 - Taxa de compressão considerando o tamanho original da imagem.

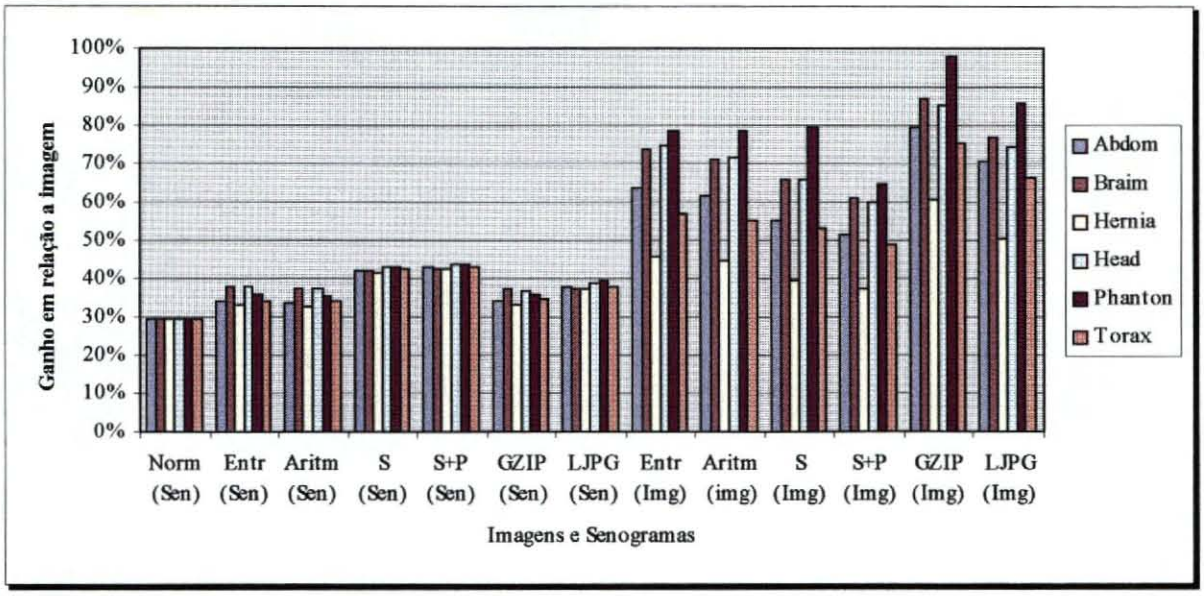


Figura 5.8 - Gráfico da taxa de compressão comparado ao tamanho original da imagem.

Pode-se notar que há uma tendência de maiores taxas de compressão nos métodos GZIP, em primeiro lugar e em segundo para o LJPG, considerando-se as imagens. O ganho destes métodos pode ser explicado pela organização das informações dos *pixels*, onde existe um desequilíbrio grande entre as probabilidades de ocorrência dos símbolos nessas imagens. Como pode ver verificado, com a inserção de ruídos, a taxa de compressão destes métodos decresce, portanto, pode-se concluir que a variação nos valores das informações exerce influência nos diversos métodos de compressão.

Ao se analisar os dados desta comparação, considerando os senogramas com ruído padrão de 1%, pode-se notar que a taxa de compressão para os senogramas são mais baixas quando comparadas com as taxas de compressão das imagens. Lembra-se novamente que são dois domínios diferentes e sua comparação direta deve ser apenas uma avaliação subjetiva.

De qualquer maneira, este resultado reforça a importância do número de *bits* por *pixel* no tamanho final do arquivo. Uma maneira de se melhorar a taxa de compressão dos senogramas seria reduzir o número de *bits* por *pixel*, porém esta redução no passo da

quantização poderia resultar em perda de informação, ou equivalente, na adição de um ruído de quantização mais elevado.

Na tabela 5.9 mostra-se as médias dos tamanhos resultantes, em *bytes*, das imagens comprimidas sem ruído e senogramas comprimidos sem ruído e com ruído padrão de 1%.

Imagem	Senograma	Entropia	Aritmética	Transf. S	Transf. S+P	GZIP	LJPG
Imagem	30%	65%	64%	60%	54%	81%	71%
Senograma sem ruído	30%	59%	58%	63%	48%	92%	73%
Senograma com ruído	30%	35%	35%	42%	43%	35%	38%

Tabela 5.9 - comparação entre as médias das compressões.

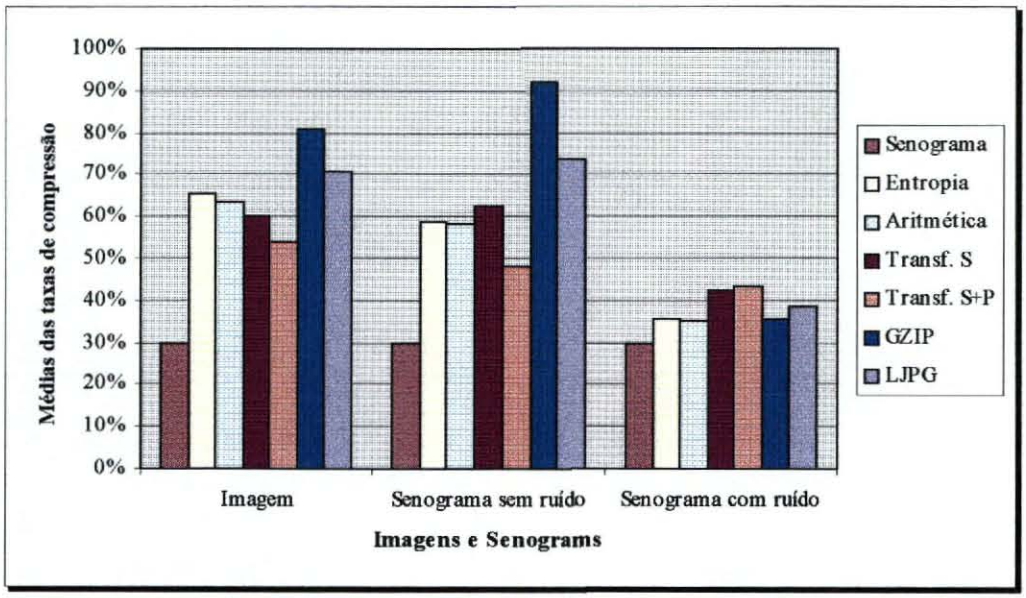


Figura 5.9 - Gráfico das médias das compressões com e sem ruído.

Foi verificado que a presença de ruídos determina maior ou menor ganho entre os métodos de compressão. Nos testes realizados com inserção de ruídos, foi constatado que ruídos em torno de 1% não alteram os resultados dos métodos de compressão para as imagens. Portanto, fica evidente que existem vantagens em se comprimir imagens, mesmo aquelas que possuem pequenos percentuais de ruído. No entanto, esta vantagem vai diminuindo à medida que o percentual de ruído aumenta, conforme verificado na figura 5.7.

As imagens aqui utilizadas na realização dos testes, com ruídos, não representam a imagem reconstruída a partir do seu senograma com ruído. Elas também tiveram ruídos inseridos, a partir da imagem original, nas mesmas proporções, a fim de se verificar a influência tanto no senograma quanto na imagem. A influência no senograma é muito maior devido a amplitude do seu sinal, que é representado por 2 *bytes*, podendo assumir valores entre 0 e 65.535. Na imagem, a influência do ruído é menor porque a amplitude do sinal é representado por 1 *byte*, podendo assumir valores entre 0 e 255.

6. Conclusão

As imagens digitais necessitam de grande espaço para armazenamento, a exemplo dos sistemas médicos que trabalham com diagnóstico por imagem. Além de gerar um grande volume de dados, podem ocasionar lentidão nos sistemas que utilizam catalogação, recuperação e transmissão destas imagens, o que torna evidente a necessidade de aplicar alguma técnica de compressão. No caso de imagens médicas, deve-se levar em conta que, em geral, não deve haver perda de informações, quando for aplicada alguma técnica de compressão. Existindo perdas, poderá haver alterações na imagem a ser analisada, podendo ocorrer erros no diagnóstico médico. Assim como as imagens, as projeções de tomografia computadorizada (senogramas) também não devem sofrer perdas na compressão, para permitir a reconstrução exata da imagem, após sua descompressão. A principal motivação para compressão e armazenamento de senogramas é permitir que os dados originais sejam preservados e as imagens possam ser regeradas com diferentes algoritmos de reconstrução.

Nos testes realizados neste trabalho, foram comparadas as taxas de compressão de alguns dos métodos de compressão sem perdas: GZIP (baseado no Lempel-Ziv e largamente utilizado para compressão de dados), LJPG (baseado na DPCM e utilizado especificamente para imagens) e *wavelets* de inteiros com codificação aritmética (método menos difundido). Os testes foram aplicados tanto sobre as projeções de tomografia computadorizada (senogramas) quanto para suas imagens reconstruídas.

No pequeno conjunto de senogramas utilizados (3 reais e 6 simulados com 1% de ruído adicionado), pode-se verificar que as transformadas *wavelets* apresentaram os maiores ganhos com cerca de 22% de taxa de compressão. Os métodos GZIP e LJPG ficaram abaixo com 8%. Ressalta-se que o LJPG operou sobre palavra de 8 *bits* e não 16 *bits*, o que pode ter

prejudicado sua capacidade de interpretação fiel da imagem. Porém esta é uma dificuldade presente na prática visto que a maioria das implementações não permite a configuração de 16bpp, embora pela definição do algoritmo não exista nenhuma restrição.

Pode ser observado o ganho introduzido pelo uso das transformadas *wavelets*. As transformadas S e S+P foram utilizadas neste trabalho em conjunto com a codificação aritmética, que após a reorganização dos pixels da imagem, utilizam esta codificação para efetuar a compressão propriamente dita. Os resultados obtidos pela compressão da codificação aritmética ficou muito próxima da entropia de 1ª ordem. Já a compressão pelas transformadas *wavelets* apresentaram as melhores taxas de compressão, comprovando a eficácia do método.

Foi realizada uma comparação entre a compressão dos senogramas e a imagens com o objetivo de avaliar a eficiência na compressão. O resultado obtido não indica ganho significativo em termos do tamanho final do arquivo.

Finalizando a análise entre os métodos aqui utilizados, podemos confirmar o potencial oferecido pela técnica da transformada *wavelet*, a qual capacita trabalhar com imagens de forma adequada na área de compressão de imagens.

A seguir, apresentamos algumas sugestões para trabalhos futuros, relacionadas à área de compressão de imagens e utilização das transformadas *wavelet*.

- realização de testes com os métodos de compressão, utilizando-se imagens reais, obtidas a partir de tomógrafos médicos;
- Realizar testes em imagens e senogramas com profundidade de *bits* diferente de 16. Utilizar imagens com 15, 14 , 13 e 12 *bits*, utilizando o menor número

necessário de *bits* em função do ruído do equipamento, objetivando descobrir a eficiência dos métodos de compressão;

- Aplicação de outros filtros *wavelet* que se adaptem melhor às características presentes nas projeções, usando o *lifting scheme* ou o método de correção;

Apêndice 1 - Padrões de Compressão de Imagem

1.1 Padrão GZIP

O padrão GZIP (Gnu ZIP) de compressão é um software desenvolvido sob a licença GNU e distribuído gratuitamente para compressão de dados de propósito geral. O algoritmo de compressão utilizado pelo gzip é o LZ77 (Lempel-Ziv 1977), comentado no item 3.5.1 [ZIV77].

O método encontra *strings* duplicadas nos dados de entrada. A segunda ocorrência da *string* é substituída por ponteiros para a *string* anterior, na forma de pares (distância, comprimento), onde o primeiro representa a distância para trás que a seqüência inicia e o segundo representa o número de caracteres na qual a seqüência é idêntica. Distâncias são limitadas por 32Kb e comprimentos são limitados por 258 *bytes*. Quando uma *string* não ocorre em nenhum lugar nos 32Kb anteriores, ela é emitida como uma seqüência literal de *bytes* (A *string* pode ser uma seqüência arbitrária de *bytes* e não é restrita a caracteres imprimíveis).

Literais ou comprimentos iguais são comprimidos com uma árvore de Huffman, e distâncias iguais são comprimidas com outra árvore. As árvores são armazenadas numa forma compacta no inicio de cada bloco. Os blocos podem ser de qualquer tamanho (exceto que os dados comprimidos para um bloco tem que encaixar dentro da memória disponível). Um bloco é terminado quando o algoritmo *deflate()* determina que deve ser conveniente começar outro bloco com árvores novas. (Isso é um tanto parecido com o comportamento do algoritmo de compressão LZW).

Strings duplicadas são encontradas utilizando uma tabela de *hash*. Todas as *strings* de entrada de tamanho 3 são inseridas na tabela de *hash*. O índice *hash* é calculado para os próximos 3 *bytes*. Se o encadeamento de *hash* (*hash chain*) para esse índice não é vazio, todas as *strings* nesse encadeamento (*chain*) são comparados com a *string* de entrada atual, e a maior combinação é selecionada.

Os encadeamentos de *hash* são procurados iniciando pelas *strings* mais recentes, para favorecer as distâncias menores e assim tirar vantagem da codificação de *Huffman*. Os encadeamentos de *hash* são unidos um por um. Não há remoções nos encadeamentos de *hash*, o algoritmo simplesmente descarta combinações que são velhas demais.

Para evitar uma situação de pior caso, encadeamentos de *hash* muito extensas são truncadas arbitrariamente num certo comprimento, determinado por um parâmetro do *deflateInit* (opção em tempo de execução). Assim, o algoritmo *deflate()* nem sempre acha a maior combinação possível, mas geralmente encontra uma que é comprida o bastante [AGO00].

1.2 Padrão JPEG

A necessidade de um padrão internacional para compressão de imagens estáticas resultou, em 1986, na formação do JPEG. Formalmente conhecido como ISO/IEC JTC1/SC29/WG1. O objetivo deste grupo foi desenvolver um método para compressão de imagens de tons contínuos que atendesse aos seguintes requisitos:

1. A qualidade da imagem reconstruída deve ser a mais perfeita possível quando comparada com a original. O codificador deve ser parametrizado permitindo a escolha da taxa de compressão e qualidade da imagem desejável.

2. Ser aplicável para praticamente qualquer tipo imagem digital de tons contínuos e não devem haver restrições com relação ao conteúdo da cena, tais como complexidade, número de cores ou propriedades estatísticas.
3. Ter uma complexidade computacional razoável, permitindo sua implementação tanto em software quanto em hardware.
4. Ter os seguintes modos de operação: (1) Codificação seqüencial, onde cada componente da imagem é codificado com uma simples varredura da esquerda para a direita e de cima para baixo; (2) Codificação progressiva, onde a imagem é codificada em múltiplas varreduras para aplicações no qual o tempo de transmissão é longo; (3) codificação sem perdas, onde a imagem é codificada garantindo a reconstrução exata da imagem original e (4) Codificação hierárquica. onde a imagem é codificada em múltiplas resoluções.

Uma expectativa comum com relação ao uso da transformada *wavelet* é que ela produz uma melhor qualidade subjetiva da imagem do que o codificador JPEG padrão. Em vista disso, o JPEG está propondo um novo padrão para compressão de imagens chamado JPEG-2000. Esse novo paradigma de compressão de imagens utilizará a transformada *wavelet* ao invés da DCT.

Em [SAN01] é comentado sobre o projeto JPEG 2000, onde foi motivado com a submissão do algoritmo CREW para a padronização de compressão sem perdas (*lossless*) e quase sem perdas (*near-lossless*) para imagens de tons contínuos, conhecido agora como JPEG-LS. Embora o LOCO-I tenha sido selecionado como a base para o JPEG-LS. Todavia, foi reconhecido que o CREW possui muitas características que merecia o desenvolvimento de um novo padrão. Em 1996, o JPEG-2000 foi aprovado como um novo item de trabalho.

JPEG 2000 refere-se para todas as partes do padrão. A proposta atual é constituída de 6 partes. As partes são:

- Parte 1, Sistema de codificação de imagens JPEG 2000 (o núcleo)
- Parte 2, Extensões (adicionar mais características e sofisticação para o núcleo)
- Parte 3, Motion JPEG 2000
- Parte 4, Adaptação
- Parte 5, Software de referência (atualmente implementações em Java e C)
- Parte 6, Formato de arquivo para imagens compostas.

O padrão JPEG-2000 terá como característica a eficiência da compressão melhorada (estimada em 30% dependendo do tamanho e da taxa de *bits* da imagem), permitirá compressão tanto com perdas quanto sem perdas, múltiplas resoluções, decodificação progressiva, codificação de região de interesse (*Region Of Interest* – ROI), o qual permite selecionar partes de uma imagem para ser codificada com mais alta qualidade, e ainda, tudo em um único fluxo de *bits* comprimido. Permitindo as aplicações manipularem ou transmitirem somente as informações essenciais para um dispositivo a partir da imagem fonte comprimida.

O JPEG 2000 tem muitas características novas em relação ao JPEG padrão, algumas delas são:

- A sua performance em compressão constitui o estado da arte para baixa taxa de *bits*;
- Transmissão progressiva por qualidade, resolução, componentes ou localidade espacial;
- Compressão com perdas e sem perdas unificado;
- Acesso aleatório para fluxo de *bits* (*bitstream*);

- Processamento no domínio comprimido (por exemplo, rotação e corte);
- Codificação da região de interesse por progressão (permite transmitir a região de interesse primeiro durante a transmissão progressiva).

A complexidade da transformada *wavelet* no JPEG 2000, depende do tamanho do filtro e dos filtros de ponto flutuante ((9,7), (10,18)) *versus* filtros inteiros ((13, 7), (3, 5), (5, 3), (2,10), etc.) utilizados. A Parte I exigirá uma *wavelet* de ponto flutuante (9, 7) e uma inteira (3, 5), enquanto a Parte II permitirá *wavelets* múltiplas incluindo as definidas pelo usuário.

Um anexo do padrão JPEG 2000 contém um formato de arquivo opcional para incluir informações tais como o espaço das cores dos pixels e informações de propriedade intelectual (direitos autorais) para a imagem. Este formato opcional de arquivo é extensível e a parte II definirá armazenamento de muitos tipos adicionais de metadados. A extensão utilizada para representar o formato do arquivo é ".JP2" [SAN01].

1.3 Transformada *Wavelet* Reversível

Um excelente estudo sobre transformada *wavelet* foi realizada por [SAN01], de onde, uma parte dela, será descrita nesta seção. A transformada *wavelet* tem sido amplamente utilizada como uma ferramenta poderosa para compressão de imagens. Todavia, até recentemente, seu uso era limitado para aplicações de compressão de imagens com perdas.

Transformadas *wavelets* de inteiros reversíveis são filtros lineares com arredondamento não linear os quais implementa sistemas de reconstrução exata utilizando aritmética de inteiros. As transformadas reversíveis, em geral, não são lineares. Por este motivo, a ordem na qual a transformada é aplicada torna-se significativa, ou seja, se a

decomposição foi aplicada primeiro nas linhas e em seguida nas colunas da imagem, a transformada inversa deve primeiro ser aplicada nas colunas e então nas linhas da imagem.

As transformadas *wavelet* de inteiros utilizadas na compressão dos senogramas e imagens são baseadas nas transformadas que possuem a propriedade de preservação de precisão (*property of precision preservation* – PPP) e permitem a compressão sem perdas de imagens com tamanho arbitrário. Estas transformadas são baseadas em técnicas de atualização e correção, tais como o *lifting scheme* e a transformada S+P. O método permite gerar uma série de transformações inteiras reversíveis que são muito próximas das transformadas *wavelet* biortogonais correspondentes e algumas transformadas *wavelet* não ortogonais, porém podem ser calculadas com somas de inteiros e operações de deslocamento de *bits*.

Após a aplicação da transformada direta, os coeficientes detalhe podem assumir valores negativos e o número de *bits* usados para representar a imagem podem não ser mais suficientes, havendo a necessidade de se usar um *bit* adicional para representar o sinal do número. A PPP permite manter o número de *bits* usados para representar cada *pixel* da imagem inalterado. No algoritmo de reconstrução o computador recuperará o sinal original através da mesma propriedade.

Alguns dos coeficientes obtidos da transformada PPP terão valores de saída diferentes dos obtidos com a transformada sem a aplicação desta propriedade. Os resultados obtidos com o uso desta propriedade possuem aproximadamente a mesma eficiência na compressão sem perdas, que significa que a entropia da imagem transformada dessas duas transformadas são quase a mesma. A transformada PPP não é aplicável para compressão com perdas.

Uma outra característica da PPP está na forma como o computador realiza aritmética com números inteiros. Se for considerado a computação da diferença de dois números inteiros dado como $c = b - a$ e a computação inversa de $a = b - c$. A computação será efetuado da seguinte maneira pelo computador:

$$c' = \begin{cases} b-a & se\ -2^{q-1} \le b-a < 2^{q-1} \\ b-a-2^q & se\ b-a \ge 2^{q-1} \\ b-a+2^q & se\ b-a < -2^{q-1} \end{cases} \quad \text{(A1.1)}$$

e a inversa será

$$a' = \begin{cases} b-c' & se\ -2^{q-1} \le b-c' < 2^{q-1} \\ b-c'-2^q & se\ b-c' \ge 2^{q-1} \\ b-c'+2^q & se\ b-c' < -2^{q-1} \end{cases} \quad \text{(A1.2)}$$

onde c' e a' indicam a representação interna no computador e o intervalo dos inteiros a, b e c é $[-2^{q-1}, 2^{q-1}-1]$. A representação interna de c' quando ocorrer um *overflow* ou *underflow* será um número com complemento de dois, então a representação interna pode não ser a mesma da representação externa de c. Porém, o mesmo código complementar para a' permitirá que a representação interna seja igual a representação externa de a.

A transformada *wavelet* de Haar na sua versão não normalizada envolve simplesmente pares de médias e diferenças:

$$s[n] = \frac{x[2n] + x[2n+1]}{2} \quad \text{(A1.3)}$$

$$d[n] = x[2n+1] - x[2n] \quad \text{(A1.4)}$$

Para o caso de uma imagem, deve-se aplicar a transformada acima nas linhas e nas colunas da imagem. A transformada inversa de Haar é dada por

$$x[2n] = s[n] - \frac{d[n]}{2} \quad \text{(A1.5)}$$

$$x[2n+1] = s[n] + \frac{d[n]}{2} \tag{A1.6}$$

Por causa da divisão por dois, as equações acima não são transformadas inteiras. Para evitar isso, poderíamos omitir a divisão por dois calculando a soma em vez da média. Porém, é preferível usar uma construção mais eficiente conhecida como transformada S (Sequencial) que é uma versão da transformada de Haar modificada para inteiros. A transformada S é a mais antiga transformada *wavelet* de inteiro para inteiro. Existe na literatura diferentes definições da transformada S, mas muitas diferem apenas na forma em que são implementadas. A transformada S será definida aqui como segue:

$$s[n] = \left\lfloor \frac{x[2n] + x[2n+1]}{2} \right\rfloor \tag{A1.7}$$

$$d[n] = x[2n+1] - x[2n] \tag{A1.8}$$

onde $\lfloor \cdot \rfloor$ corresponde a operação de truncamento para baixo. A primeira vista, o truncamento em $s[n]$ parece descartar alguma informação. Todavia, a soma e a diferença de dois números inteiros são também ambos pares ou ambos impares. Com base nessa informação, pode-se omitir o *bit* menos significativo da soma, pois ele é igual ao *bit* menos significativo da diferença ($d[n]$). A diferença poderá então ser usada para determinar se ocorreu o truncamento. Se este valor for impar indicará o truncamento para o menor inteiro.

Note que o índice de fator dois ($[2n]$), nos coeficientes da transformada, é o resultado da subamostragem por um fator de 2, enquanto que a operação de truncamento $\lfloor \cdot \rfloor$ é a fonte da não linearidade.

Como os *pixels* adjacentes em uma imagem são altamente correlacionados, a aplicação da transformada acima permite reduzir significativamente a entropia de primeira

ordem da imagem, que pode ser reduzida ainda mais se for usado um método de codificação preditiva.

A transformada S é inversível e sua inversa é dada por

$$x[2n] = s[n] - \left\lfloor \frac{d[n]}{2} \right\rfloor \tag{A1.9}$$

$$x[2n+1] = s[n] + \left\lfloor \frac{d[n]+1}{2} \right\rfloor \tag{A1.10}$$

Said e Pearlman propuseram a transformada S+P (Transformada S + Predição) na qual a predição linear é realizada nos coeficientes passa baixa para gerar um novo conjunto de coeficientes passa alta depois da aplicação da transformada S. A forma geral da transformada S+P é:

$$d^{(1)}[n] = x[2n+1] - x[2n] \tag{A1.11}$$

$$s[n] = x[2n] + \left\lfloor \frac{d^{(1)}[n]}{2} \right\rfloor \tag{A1.12}$$

$$d[n] = d^{(1)}[n] + \left\lfloor \begin{matrix} \alpha_{-1}(s[n-2] - s[n-1]) + \alpha_0(s[n-1] - s[n]) \\ + \alpha_1(s[n] - s[n+1]) - \beta_1 d^{(1)}[n+1] \end{matrix} \right\rfloor \tag{A1.13}$$

Said e Pearlman examinaram várias escolhas para α_n e β_1 e sugerem $\alpha_{-1} = 0$, $\alpha_0 = \frac{2}{8}$, $\alpha_1 = \frac{3}{8}$ e $\beta_1 = \frac{2}{8}$ para imagens naturais e $\alpha_{-1} = -\frac{1}{16}$, $\alpha_0 = \frac{4}{16}$, $\alpha_1 = \frac{8}{16}$ e $\beta_1 = \frac{6}{16}$ para imagens médicas muito suave.

Apêndice 2 - Formatos de Imagem

2.1 Formato BMP

No trabalho de [AGO00] tem-se um resumo sobre o padrão BMP. É o formato de imagem nativo do sistema operacional Microsoft Windows. Este padrão suporta imagens com 1, 4, 8, 16, 24 e 32 *bits* por *pixel*. Este formato pode se apresentar com compressão, utilizando o algoritmo RLE [MUR96], ou ainda sem nenhuma compressão.

A versão inicial do formato BMP era formada apenas por um cabeçalho do arquivo e pelos dados dos *pixels*, conforme a Figura A2.1.

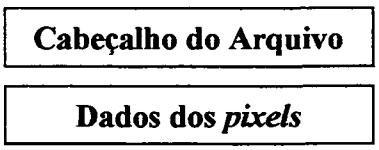


Figura A2.1 - Formato BMP na versão inicial.

A partir da segunda versão, o formato do arquivo BMP evoluiu, permanecendo desta forma até as versões atuais. Neste novo formato, além do cabeçalho do arquivo e dos dados de *pixels*, foi incluído um cabeçalho da imagem e uma paleta de cores na ordem apresentada na Figura A2.2. A paleta de cores só existe se o número de *bits* por *pixel* for 1, 4 ou 8.

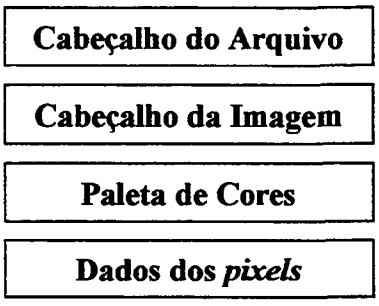


Figura A2.2 - Formato BMP a partir da segunda versão.

O cabeçalho do arquivo serve para identificar o formato do arquivo e está apresentado na Tabela A2.1.

Nome do Campo	Tamanho (bytes)	Descrição
Tipo	**2**	**Contém os caracteres "BM".**
Tamanho	**4**	**Tamanho do arquivo em bytes.**
Reservado	**2**	**Não utilizado (deve valer zero).**
Reservado	**2**	**Não utilizado (deve valer zero).**
Deslocamento	**4**	**Deslocamento para o início dos dados.**

Tabela A2.1 - Cabeçalho de arquivo do formato BMP

Imediatamente depois do cabeçalho do arquivo localiza-se o cabeçalho da imagem, podendo seguir duas estruturas diferentes, uma para Windows e outra para OS/2. Apenas o cabeçalho para Windows será detalhado neste trabalho. Na Tabela A2.2 está apresentado o formato de cabeçalho para Windows 3.x.

Na versão para Windows 95 mais campos foram adicionados a este cabeçalho, como forma de dar suporte, principalmente, aos novos números de *bits* por *pixel* suportados (16, 24 ou 32). Estes novos campos estão apresentados na Tabela A2.2 e são colocados exatamente depois dos campos da Tabela A2.1. Caso o número de *bits* por pixel seja 1, 4 ou 8, este complemento ao cabeçalho da imagem não existe, sendo substituído pela paleta de cores.

Na Tabela A2.2, o campo *Planos* deve ser 1 porque o padrão BMP usa apenas um plano de cores. O tipo de compressão, definido pelo campo *Compressão*, pode ser sem compressão (Compressão=0), comprimido por RLE8 (*Compressão*=1), comprimido por RLE4 (*Compressão*=2), ou utilizar máscaras de campo (*Compressão*=3), para significar que os dados não estão comprimidos e que não existe paleta de cores.

Nome do Campo	Tamanho (bytes)	Descrição
Tamanho	**4**	**Tamanho do cabeçalho (40 a 108 bytes).**
Largura	**4**	**Largura da imagem.**
Comprimento	**4**	**Comprimento da imagem.**
Planos	**2**	**Número de plano de cores (deve ser um).**
Bits por pixel	**2**	**Bits por pixel (1, 4, 8, 16, 24 ou 32).**
Compressão	**4**	**Tipo de compressão.**
Tamanho da Imagem	**4**	**Tamanho da imagem.**
Pixels por metro X	**4**	**Resolução horizontal em pixels por metro.**
Pixels por metro Y	**4**	**Resolução vertical em pixels por metro.**
Cores Usadas	**4**	**Entradas na paleta de cores usadas.**
Cores Importantes	**4**	**Número de cores significativas.**

Tabela A2.2 - Cabeçalho de imagem do formato BMP

Nome do Campo	Tamanho (bytes)	Descrição
Máscara Vermelha	**4**	**Identifica bits do componente Vermelho.**
Máscara Verde	**4**	**Identifica bits do componente Verde.**
Máscara Azul	**4**	**Identifica bits do componente azul.**
Máscara Alpha	**4**	**Identifica bits do componente alpha.**
Tipo de cor	**4**	**Tipo do espaça de cores.**
Vermelho X	**4**	**Coordenada X do ponto fim .**
Vermelho Y	**4**	**Coordenada Y do ponto fim .**
Vermelho Z	**4**	**Coordenada Z do ponto fim .**
Verde X	**4**	**Coordenada X do ponto fim .**
Verde Y	**4**	**Coordenada Y do ponto fim .**
Verde Z	**4**	**Coordenada Z do ponto fim .**
Azul X	**4**	**Coordenada X do ponto fim .**
Azul Y	**4**	**Coordenada Y do ponto fim .**
Azul Z	**4**	**Coordenada Z do ponto fim .**
Gamma	**4**	**Coord. do valor de escala do gamma.**
Gamma Verde	**4**	**Coord. do valor de escala do gamma verde.**
Gamma Azul	**4**	**Coord. do valor de escala do gamma azul.**

Tabela A2.3 - Continuação do cabeçalho de imagem do formato BMP

Na Tabela A2.3, os campos Máscara Vermelha, Máscara Verde, Máscara Azul e Máscara Alpha especificam, para BMPs de 16 ou 32 *bits*, quais *bits* em um valor de *pixel* correspondem a uma cor específica ou ao canal alpha, que armazena os dados de transparência do *pixel*, sendo que este valor pode variar de 0 (*pixel* completamente

transparente) até 255 (pixel completamente opaco) [MUR96]. O campo Tipo de Cor define o tipo de espaço de cores usado. As opções são: RGB calibrado (Tipo de Cor=0), RGB dependente de dispositivo (Tipo de Cor=1) e CMYK dependente de dispositivo (Tipo de Cor=2). Os campos Vermelho X, Vermelho Y, Vermelho Z, Verde X, Verde Y, Verde Z, Azul X, Azul Y e Azul Z são usados apenas para o espaço de cores RGB calibrado.

Após o cabeçalho de imagem, localiza-se a paleta de cores. Esta paleta é um vetor de estruturas que especificam os valores de intensidade das cores vermelho, verde e azul de cada cor em uma imagem. Cada pixel, nos dados de pixels, armazena um valor simples, que é usado como índice na paleta de cores. A paleta de cores só existe em arquivos BMP com 1, 4 ou 8 *bits* por *pixel*. Nos arquivos com 16, 24 ou 32 *bits* por *pixel*, existem máscaras para cada cor no lugar da paleta de cores. O valor da máscara é usado para fazer um AND com o valor do *pixel* e, com isso, capturar a intensidade de uma cor específica no *pixel*.

Após a paleta de cores, se ela existir, vêm os dados de pixel. Dependendo do número de *bits* por *pixel*, a interpretação dos dados é diferente. Se forem 1, 4 ou 8 *bits* por *pixel*, o dado armazenado é o índice do pixel na paleta de cores. Se forem 16, 24 ou 32 *bits*, os dados armazenados contêm toda a informação de cores do pixel, onde se usa as máscaras para interpretá-las separadamente.

A única compressão possível no formato BMP é a RLE, abordada no Apêndice 3 deste trabalho, portanto a compressão no BMP é sem perdas. Existem dois tipos diferentes de RLE utilizadas no formato *bitmap*. A primeira usa 4 *bits* como elemento atômico e a segunda usa 8 *bits*. Por isso, os pixels representados com 1, 16, 24 ou 32 *bits* não podem ser comprimidos no formato BMP [MUR96].

2.2 Formato PBM

A designação de formato de imagem PBM (*Portable Bitmap*) engloba três formatos de imagem para imagens a preto e branco, em escala de tons cinzentos e a cores, todos eles sem compressão e que apresentam uma estrutura comum. Os três tipos de formato de imagens são:

- PBM (*Portable BitMap*): para imagens de dois níveis (preto e branco)
- PGM (*Portable GrayMap*): para imagens em tons de cinza
- PPM (*Portable PixMap*): para imagens coloridas

A definição original destes formatos teve em vista permitir a transmissão de imagens por meio de correio eletrónico que, à data da definição, ainda não permitia a transmissão de arquivos anexados, binários ou não. Os formatos PBM, PGM e PPM representavam então os conteúdos das respectivas imagens por meio de caracteres ASCII representáveis. Esta característica permitia a inserção de uma imagem numa mensagem de correio eletrónico como se tratasse de texto, mas tinha como consequência que o tamanho dos arquivos fossem demasiadamente grande. A definição do formato mais tarde foi modificada para permitir a representação binária dos conteúdos das imagens.

Os formatos de imagem PBM são constituídos pelos seguintes campos:

- Identificador do tipo de formato (designado por "*magic value*"), de acordo com o tipo, definido para imagens gravadas no formato ASCII e formato binário, conforme tabela abaixo:

Tipo	Imagem em ASCII	Imagem em Binário
PBM	**P1**	**P4**
PGM	**P2**	**P5**
PPM	**P3**	**P6**

Tabela A2.4 - Identificador de tipo para os formatos PBM, PGM e PPM.

- Espaço em branco, podendo ser constituído por um qualquer número de caracteres em branco, TABs, CRs e LFs;
- Largura da imagem, em *pixels* e em valor decimal, formatada em caracteres ASCII;
- Espaço em branco;
- Altura da imagem, em *pixels* e em valor decimal, formatada em caracteres ASCII;
- Espaço em branco.

Se a imagem for dos tipos PGM ou PPM:

- Valor máximo em notação decimal dos tons de cinzento (PGM) ou das componentes de cor (PPM), formatado em caracteres ASCII;
- Espaço em branco.

Para os três tipos de formato:

- Valores dos *pixels* da imagem, em número igual à altura da imagem vezes a sua largura para os tipos PBM e PGM, e três vezes este número para o tipo PPM, uma vez que cada *pixel* é representado pelas três componentes RGB da respectiva cor.

A ordenação dos valores dos *pixels* corresponde a varredura das imagens linha a linha, de cima para baixo, e da esquerda para a direita em cada uma das linhas.

Quando o conteúdo da imagem for representado em ASCII, os valores correspondentes aos *pixels* serão apresentados em notação decimal e separados por espaços em branco, marcas de tabulação ou marcas de fim de linha.

As variantes binárias dos tipos PBM, PGM e PPM armazenam os valores correspondentes aos *pixels* das imagens em caracteres (*bytes*) contíguos, sem qualquer separador entre valores consecutivos. O tipo PBM combina os valores de cada 8 *pixels* consecutivos num único caracter (*byte*). Os tipos PGM e PPM, fazem corresponder a cada *pixel* um caracter para PGM e três caracteres para PPM.

Para as variantes ASCII dos três tipos de formato, aplicam-se ainda as seguintes regras:

- É permitida a inserção de comentários em qualquer parte do arquivo. O início de um comentário é assinalado por um caracter "#" e todo o texto desde este caracter até ao fim da respectiva linha é interpretado como texto do comentário.
- comprimento máximo de cada linha está limitado a 70 caracteres. Esta limitação deriva da especificação inicial visando a transmissão de imagens por correio eletrônico.

2.3 Formato RAW

O formato RAW é o mais simples dos formatos. Os valores dos *pixels* são salvos em duas ordenações, da esquerda para direita e de cima para baixo, um valor de cada vez. Para imagens em níveis de cinza com valores na faixa 0 a 255, 8 *bits* (ou 1 *byte*) por *pixel* é suficiente. Consequentemente, uma imagem NxM ocupa NxM *bytes*, com cada *byte* correspondendo ao valor de um *pixel*.

A desvantagem deste formato é que o tamanho original da imagem não pode ser detectado diretamente pela inspeção do arquivo. Por exemplo, uma imagem 256x512 não pode ser diferenciada de uma imagem 512x256, a não ser que a imagem seja visualizada na tela.

Todos os outros formatos são criados de forma que o arquivo contenha informações úteis sobre a imagem, em um cabeçalho, incluindo tamanho, mapeamento das cores (para imagens pseudo coloridas, ou seja, aquelas que usam uma tabela de cores), entre outras informações.

Apêndice 3 - Codificação Run Length (RLE)

Conforme abordado e descrito em [AGO00], o método Run-Length Encoding (RLE), também conhecido como RLC (Run-Length Coding), busca reduzir a redundância de codificação e, quando usada em conjunto com outros métodos, como a codificação de Huffman, torna a redundância de codificação aproximadamente igual a zero. A idéia básica é substituir uma seqüência de símbolos iguais pelo número de ocorrências do símbolo e o próprio símbolo [EGG99].

Por exemplo, a seguinte seqüência de símbolos:

0AAAAA7CCCCCCCCCC5555

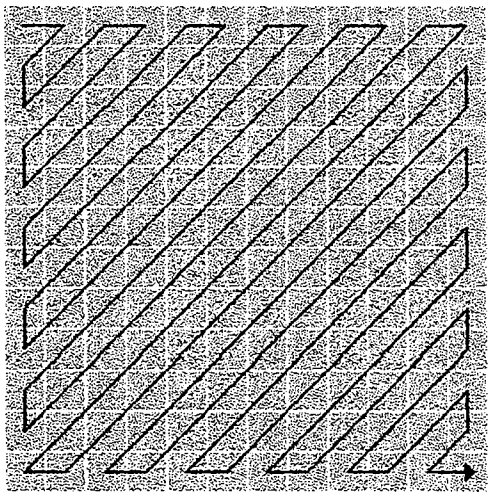
seria codificada por RLE como:

105A179C45

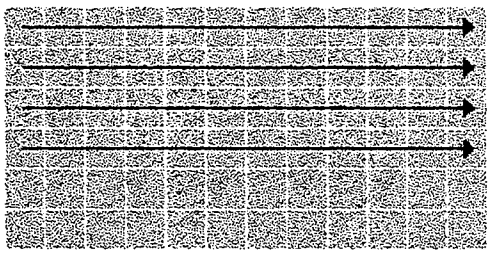
No exemplo, obteve-se uma taxa de compressão de 2:1, mas pode-se perceber que para seqüências de símbolos com elevado grau de variação, pode-se chegar num resultado extremamente indesejável, que é a compressão negativa [MIA99]. Em imagens, normalmente existem muitas repetições de pixels, por isso, a codificação RLE pode ser muito interessante.

Na representação em duas dimensões, que é a forma como as imagens são armazenadas em computador, o processo a codificação RLE pode ser feito e várias formas. A seqüência de dados analisados pode determinar maiores ou menores taxas de compressão, por isso, o objetivo é determinar a seqüência com maior número e repetições possíveis. A seqüência mais natural e simples e implementar é a apresentada na Figura A3.1(b), onde é feita uma varredura que inicia no canto superior esquerdo da imagem,

vai até o final da linha, volta para o início da próxima linha e assim sucessivamente, até que a varredura chegue no canto inferior direito da imagem. A Figura A3.1(c) apresenta o mesmo processo seqüencial anterior, mas com as colunas sendo varridas ao invés das linhas. Uma solução muito interessante é a codificação em ziguezague, onde a varredura é feita em diagonal, começando no canto superior esquerdo e indo até o canto inferior direito a imagem. A codificação em ziguezague é muito utilizada em conjunto com a técnica DCT. A codificação RLE em ziguezague é apresenta na Figura A3.1(a).

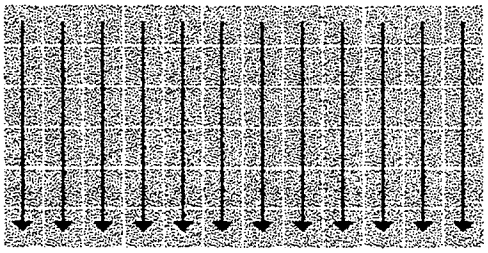


(a) Codificação em ziguezague.



(b) Codificação ao longo do eixo X.

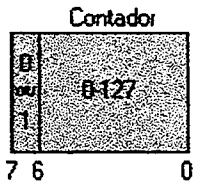
(c)



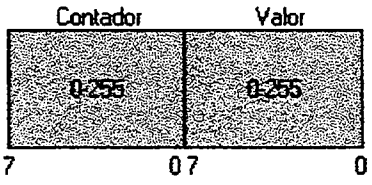
(c) Codificação ao longo do eixo Y.

Figura A3.1 - Codificação RLE (a) em ziguezague, (b) por linha e (c) por coluna.

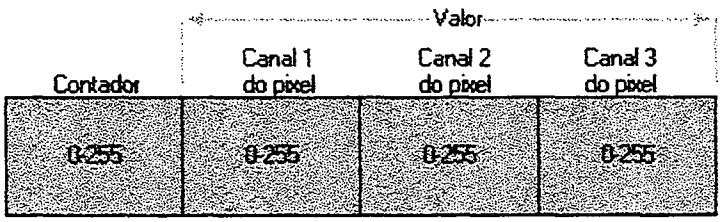
Existem três diferentes formas básicas de se efetuar a codificação RLE no que diz respeito a qual é o elemento atômico a ser codificado. A Figura A3.2(a) mostra a codificação em nível de *bit*, o que significa que o *bit* é a unidade básica a ser codificada, onde utiliza um *byte* para armazenar as informações sobre os *bits*. Utiliza um contador com 7 *bits* para armazenar o número de ocorrências e 1 bit para armazenar o *bit* que foi encontrado. A codificação também pode ser a nível de byte, mostrado na Figura A3.2(b), onde o byte é o elemento atômico e são necessários dois bytes para cada pacote, o primeiro para armazenar o número de ocorrências em seqüência do símbolo e o segundo é o próprio símbolo a ser codificado. Por fim, a Figura A3.2(c) mostra a codificação a nível de pixel, onde a unidade básica é um pixel, que ocupa três *bytes*.



(a) Pacote RLE a nível de *bit*.



(b) Pacote RLE a nível de *byte*.



(a) Pacote RLE a nível de *pixel*.

Figura A3.2 - Diferentes formas de codificação do RLE.

Apêndice 4 - Imagens e Senogramas Utilizados

As seguintes figuras correspondem as imagens reconstruídas e respectivos senogramas que foram utilizadas na realização dos testes neste trabalho. O termo reconstrução usado aqui não se trata da aplicação do processo inverso utilizado para comprimir os dados, mas sim o processo que permite reconstruir a imagem de tomografia a partir das projeções.

Nos senogramas, com 16 bits por *pixel*, o preto corresponde ao *pixel* de valor 0 e o branco corresponde ao *pixel* de valor 65.535. Nas imagens reconstruídas, com 8 bits por *pixel*, o preto corresponde ao *pixel* de valor 0 e o branco corresponde ao *pixel* de valor 255.

As figuras A4.1, A4.2 e A4.3, são senogramas reais gerados por um tomógrafo industrial e suas respectivas imagens reconstruídas. Embora as dimensões dos senogramas sejam 768 colunas por 90 linhas e 16 *bits* por *pixel*, totalizando um tamanho de 138.240 *bytes*, as imagens originais possuem tamanhos diferenciados, definidos pelo processo de reconstrução, conforme segue: imagem "Dente01" com 256 colunas por 256 linhas e 8 bpp, totalizando 65.536 *bytes*; imagem "Dente02" com 176 colunas por 176 linhas e 8 bpp, totalizando 30.976 *bytes*; e imagem "Dente03" com 112 colunas por 112 linhas e 8 bpp, totalizando 12.544 *bytes*.

A seguir são apresentadas as figuras A4.4 até A4.9, constituindo as demais imagens e com seus respectivos senogramas, também utilizados nos testes deste trabalho. Os Senogramas possuem 256 colunas por 90 linhas com 16 bits por pixel, totalizando 43.080 *bytes*. As imagens possuem 256 colunas por 256 linhas com 8 bits por pixel, totalizando 65.536 *bytes*.

Na sequência, as figuras A4.10 até A4.19, são os senogramas resultantes e respectivas imagens com a inserção de percentuais de ruídos, a partir do senograma "Abdom" e sua imagem .

Conforme comentado no item 5.5, as imagens aqui mostradas, com ruídos, não representam a imagem reconstruída a partir do seu senograma com ruído. Elas também tiveram ruídos inseridos, a partir da imagem original, nas mesmas porcentagens, a fim de se verificar a influência tanto no senograma quanto na imagem. A influência no senograma é muito maior devido a amplitude do seu sinal, que é representado por 2 *bytes*, podendo assumir valores entre 0 e 65.535. Na imagem, a influência do ruído é menor porque a amplitude do sinal é representado por 1 *byte*, podendo assumir valores entre 0 e 255.

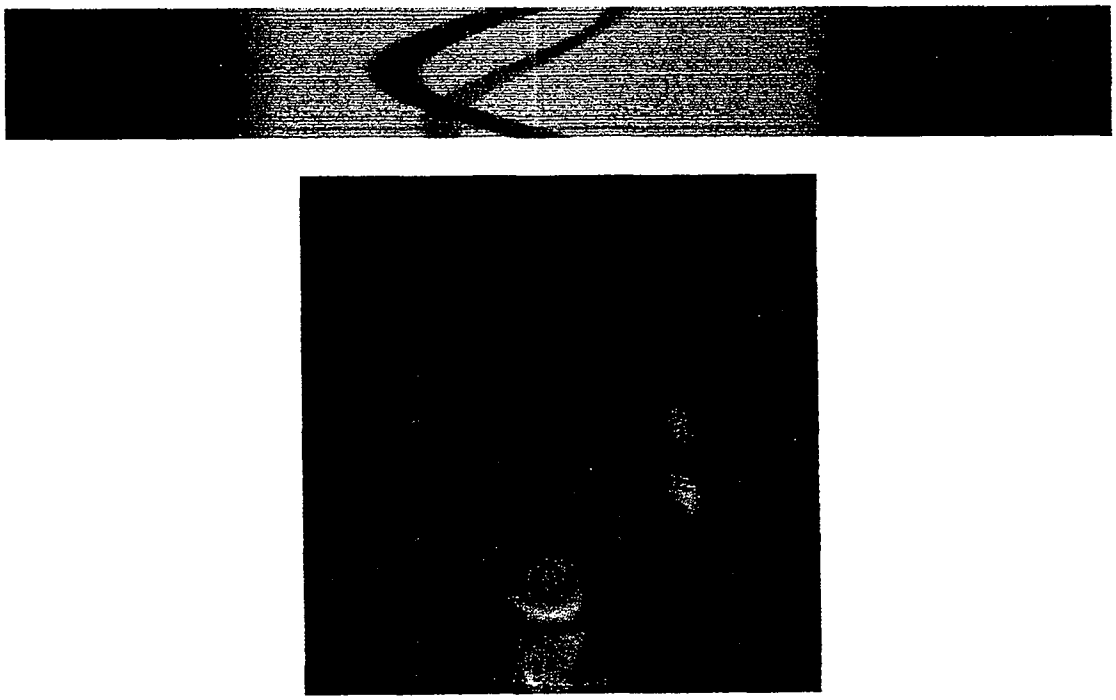


Imagem A4.1 – Senograma "Dente01" e imagem reconstruída de um dente.

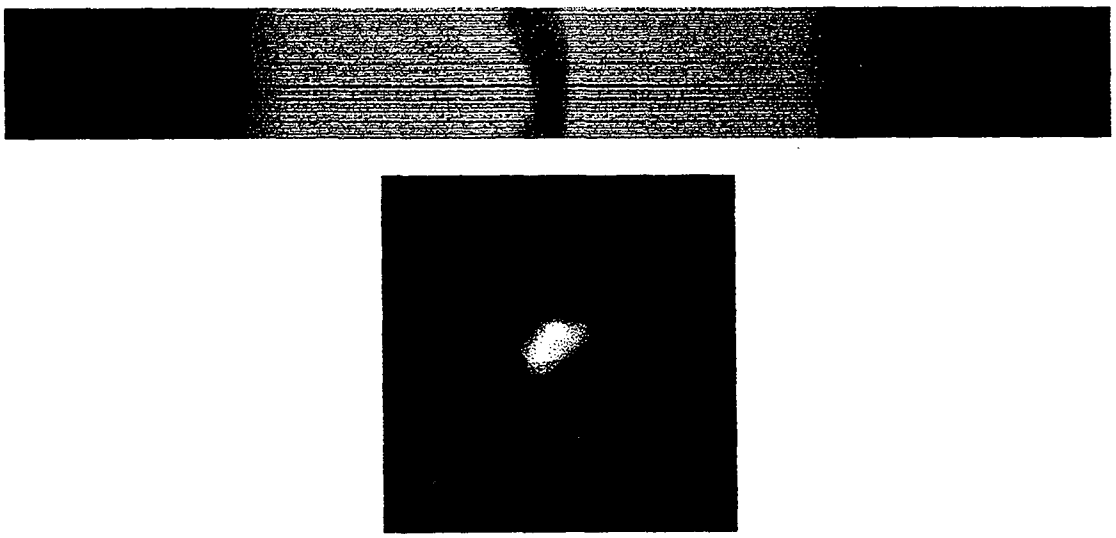


Imagem A4.2 – Senograma "Dente02" e imagem reconstruída de um dente.

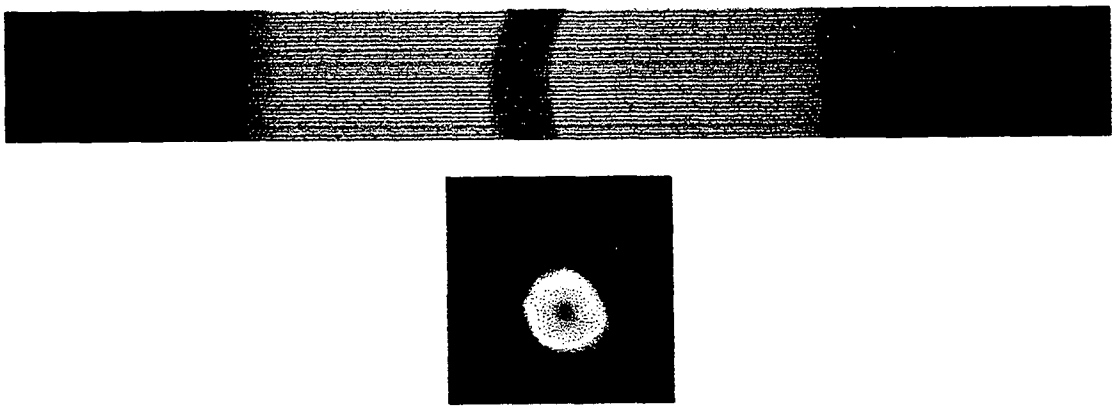


Imagem A4.3 – Senograma "Dente03" e imagem reconstruída de um dente.

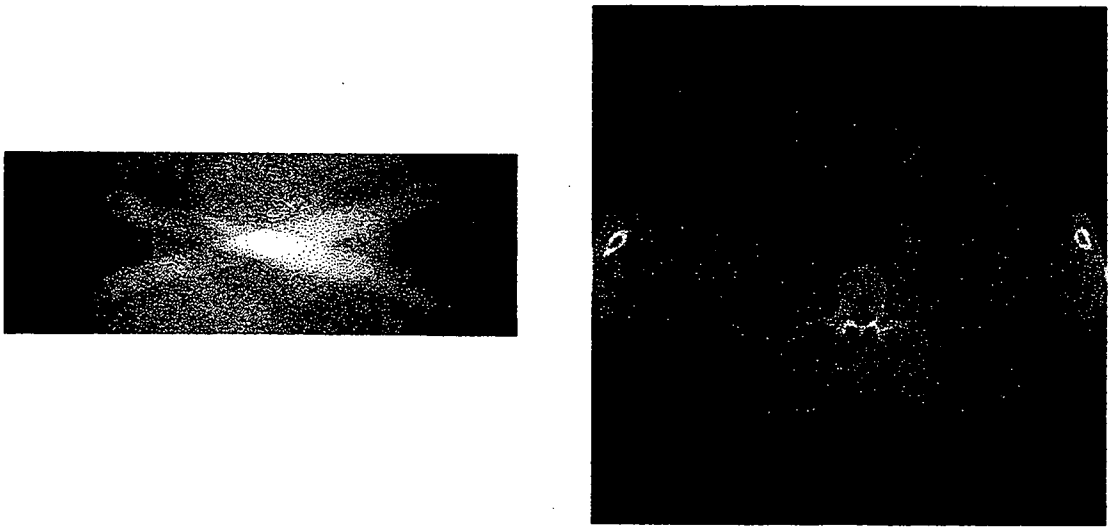


Imagem A4.4 - Senograma "Abdom" e imagem reconstruída de um abdomem.

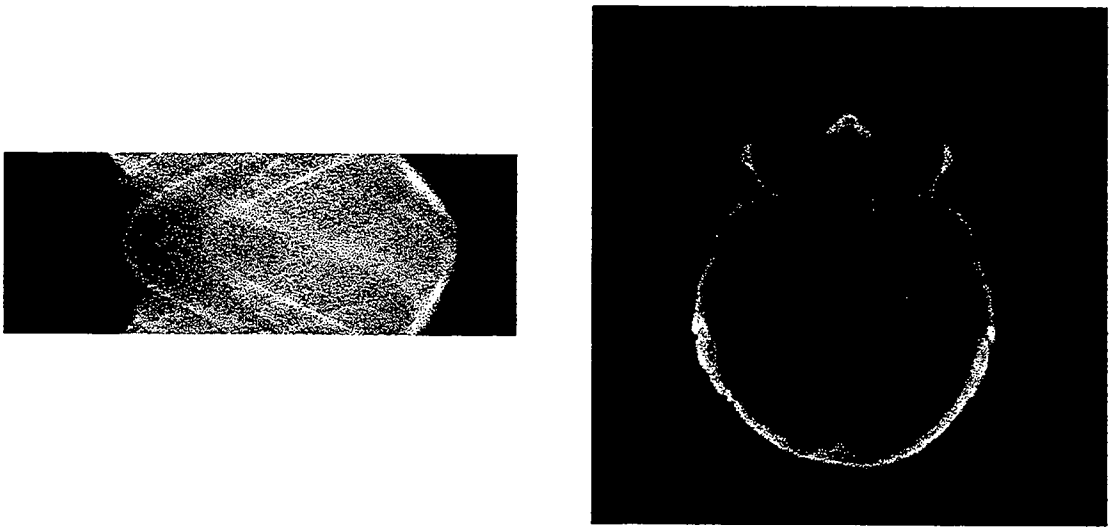


Imagem A4.5 - Senograma "Brain" e imagem reconstruída de um cérebro.

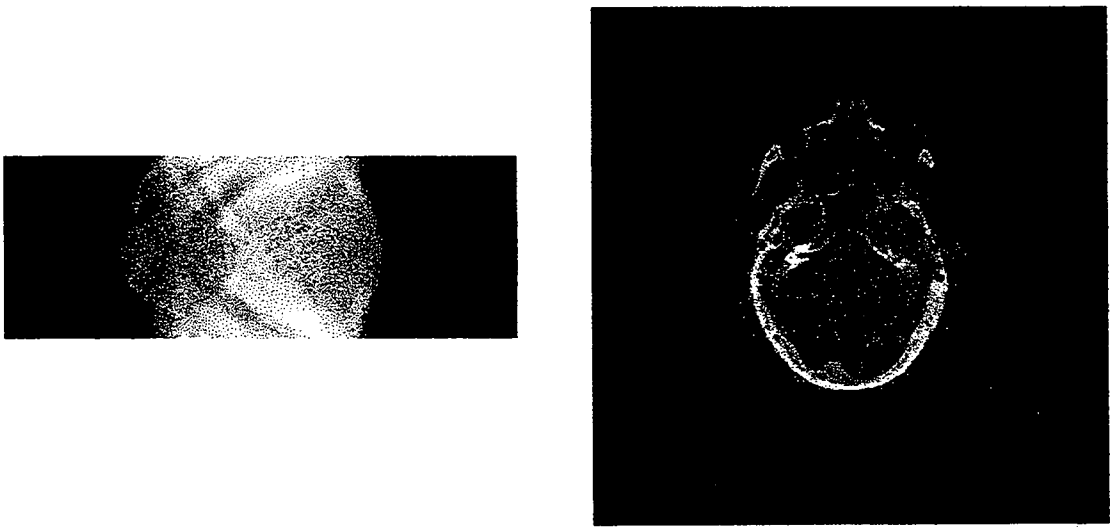


Imagem A4.6 - Senograma "Head" e imagem reconstruída de uma cabeça.

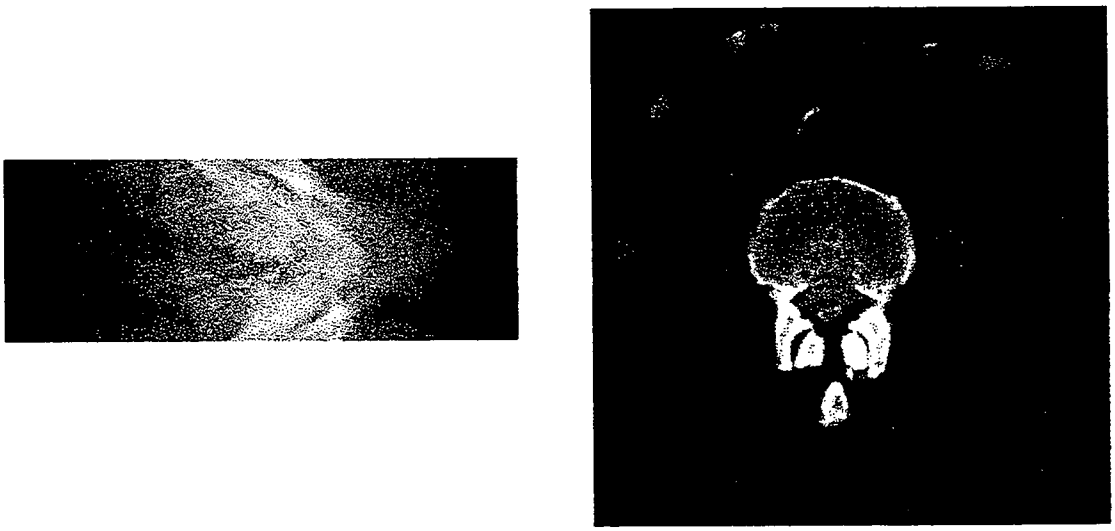


Imagem A4.7 - Senograma "Hernia" e imagem reconstruída de vértebra com hérnia.

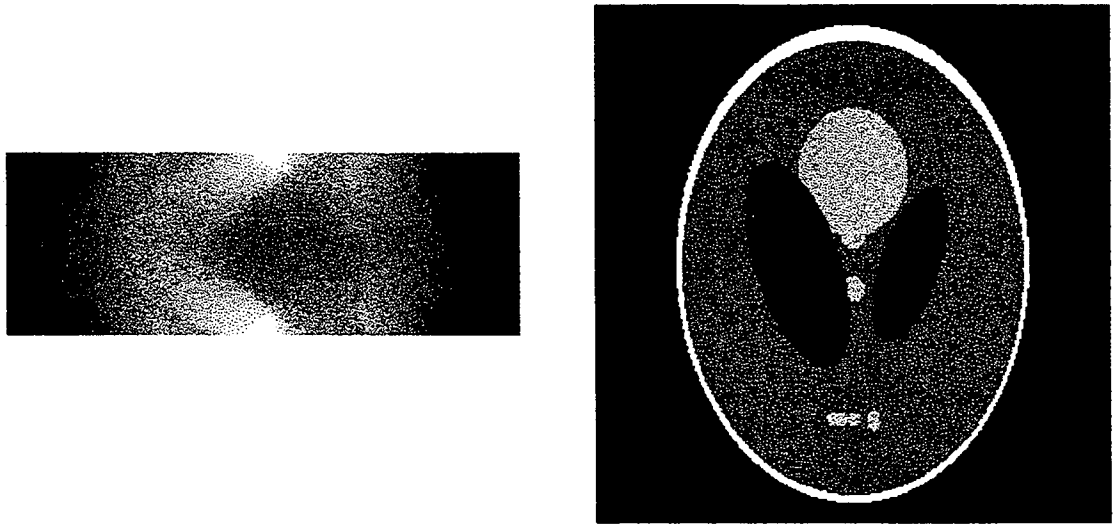


Imagem A4.8 - Senograma "Phantom" e imagem de testes representando uma cabeça.

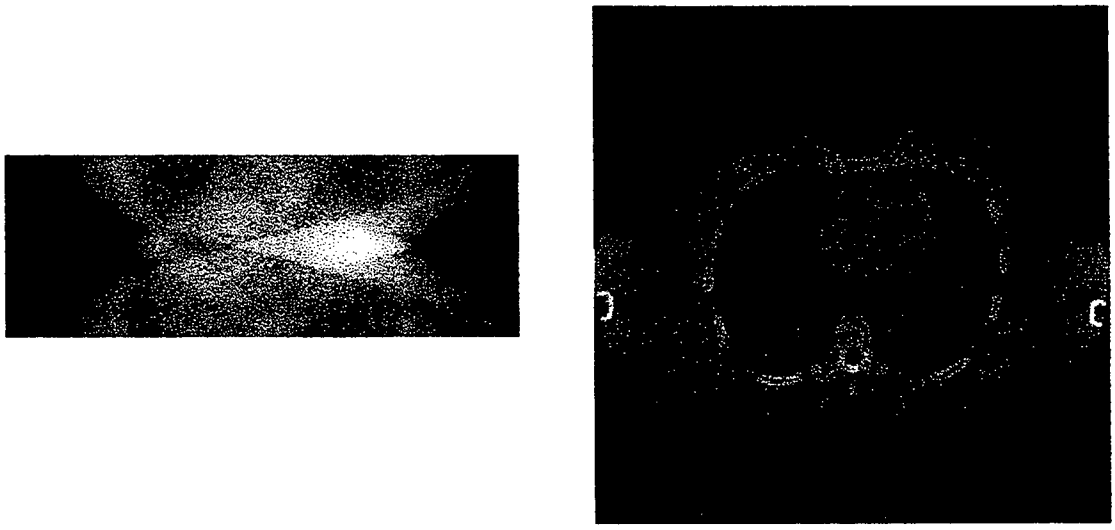


Imagem A4.9 - Senograma "Torax" e imagem reconstruída de um tórax.

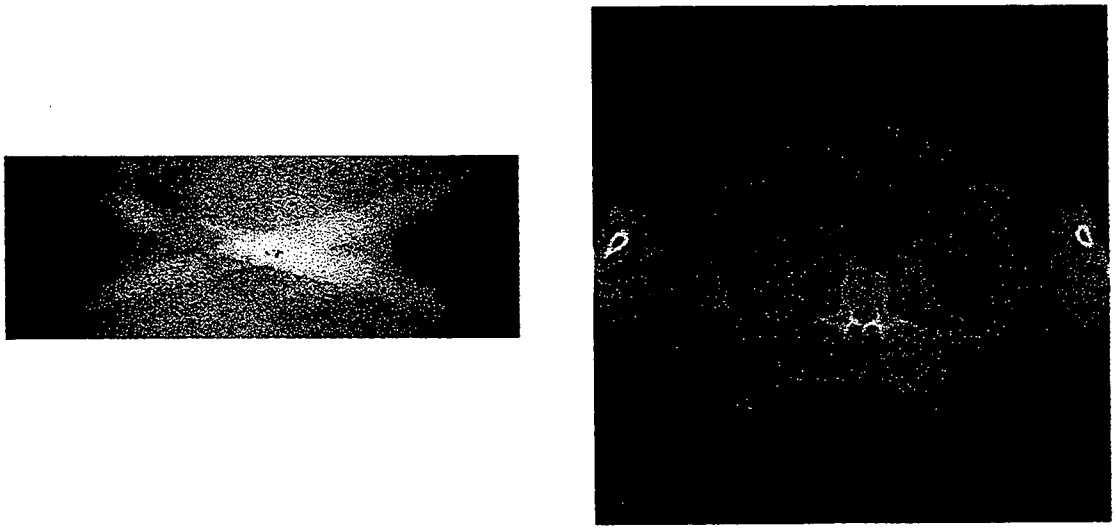


Imagem A4.10 - Senograma "Abdom" e imagem com percentual de 0,1% de ruído.

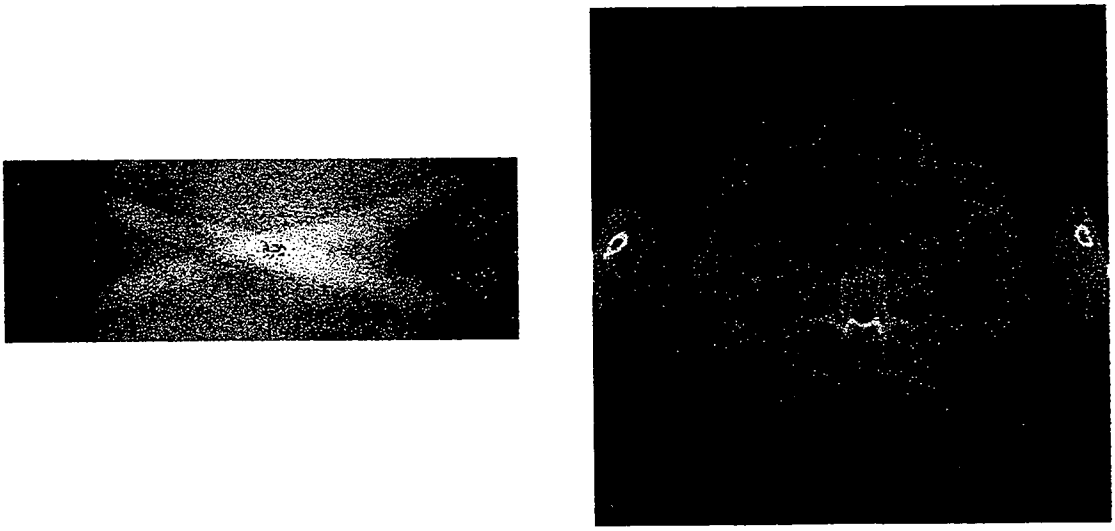


Imagem A4.11 - Senograma "Abdom" e imagem com percentual de 0,2% de ruído.

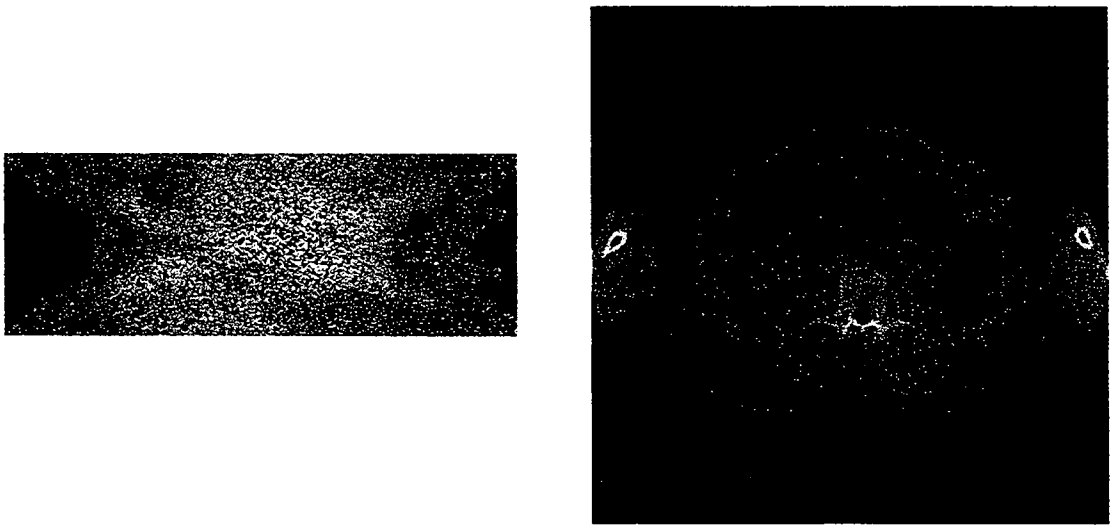


Imagem A4.12 - Senograma "Abdom" e imagem com percentual de 1% de ruído.

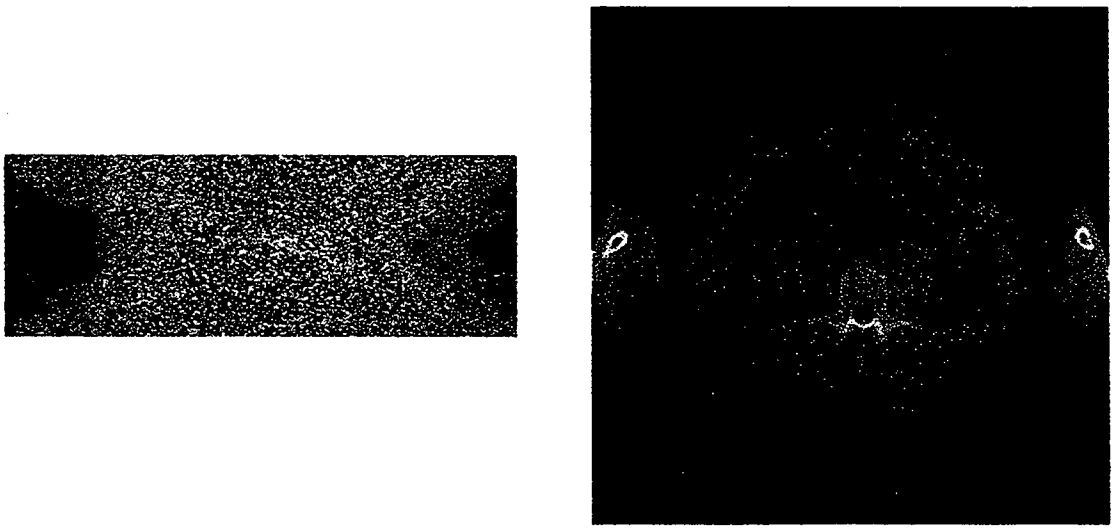


Imagem A4.13 - Senograma "Abdom" e imagem com percentual de 2% de ruído.

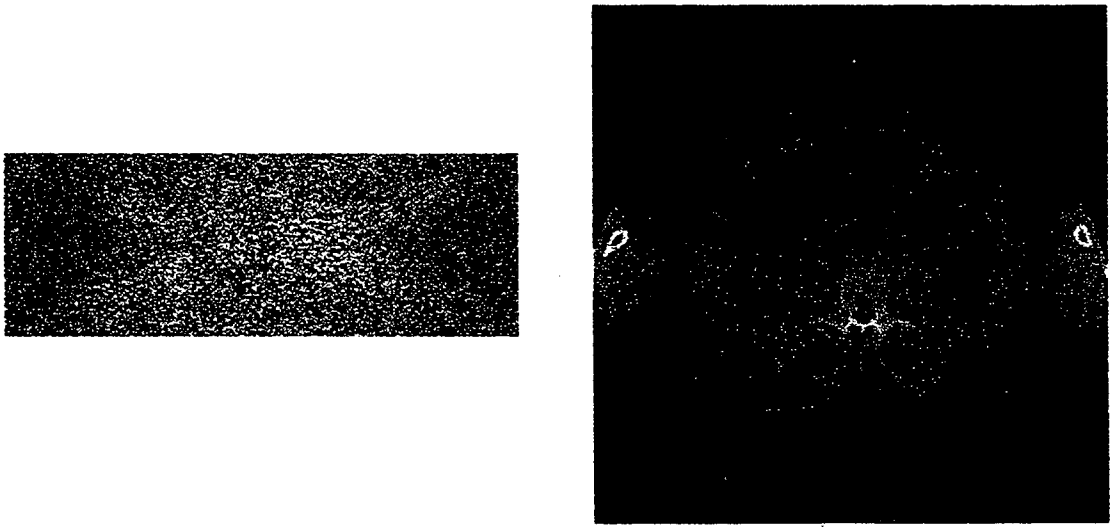


Imagem A4.14 - Senograma "Abdom" e imagem com percentual de 10% de ruído.

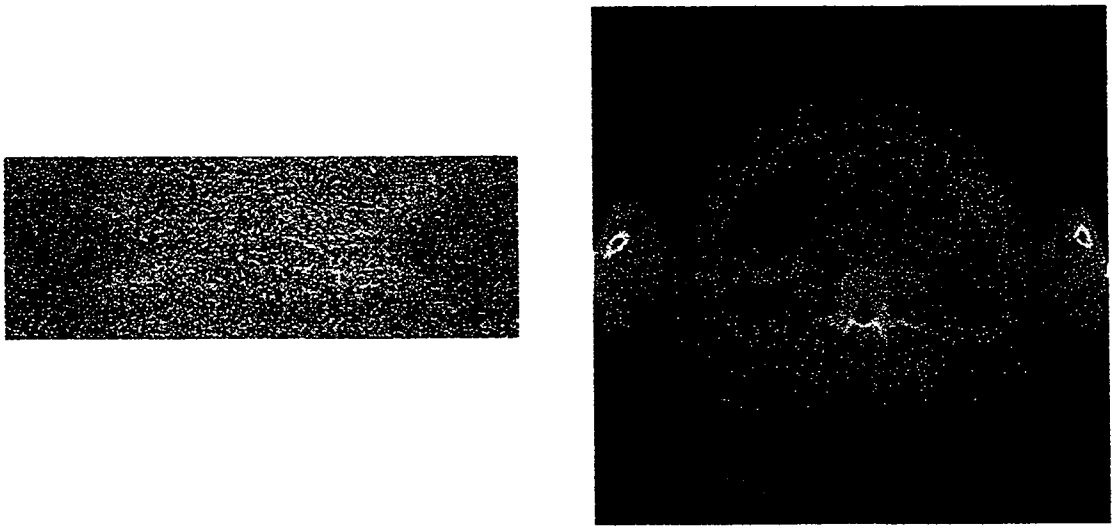


Imagem A4.15 - Senograma "Abdom" e imagem com percentual de 20% de ruído.

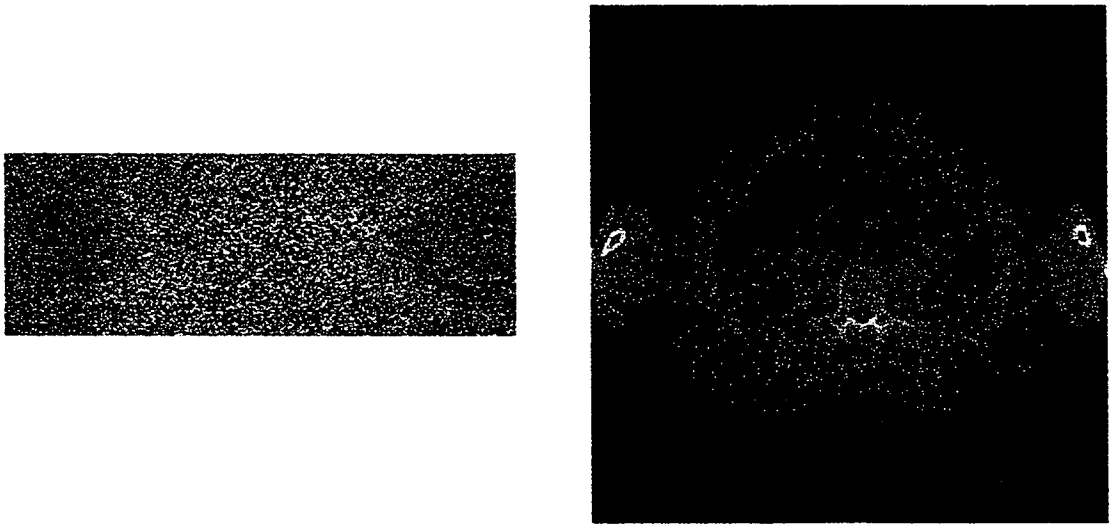


Imagem A4.16 - Senograma "Abdom" e imagem com percentual de 25% de ruído.

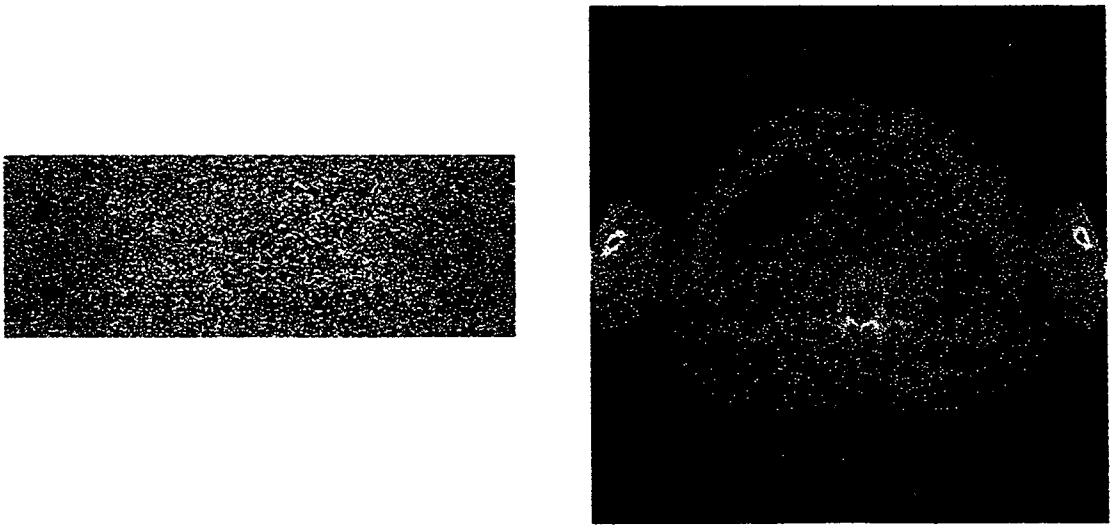


Imagem A4.17 - Senograma "Abdom" e imagem com percentual de 33% de ruído.

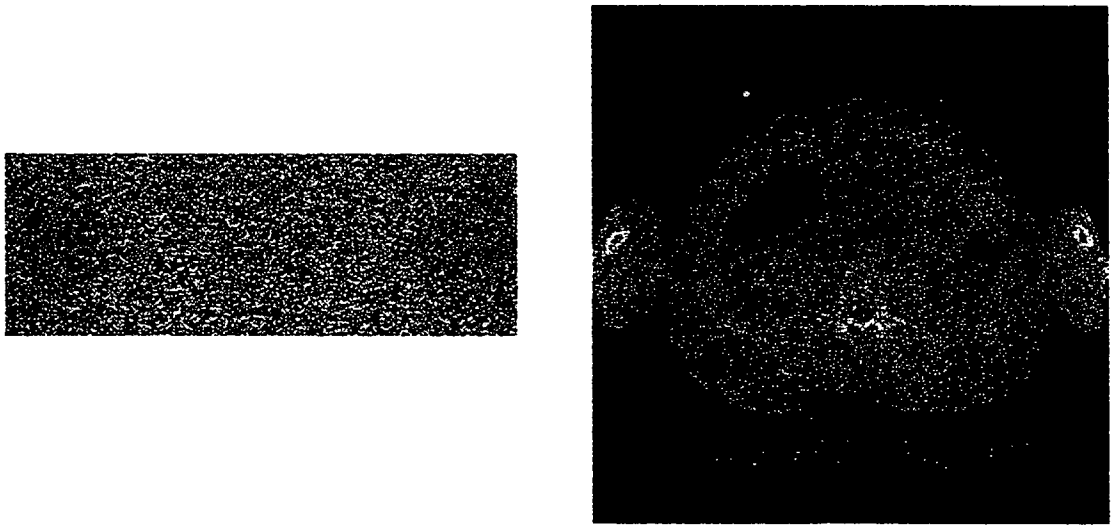


Imagem A4.18 - Senograma "Abdom" e imagem com percentual de 50% de ruído.

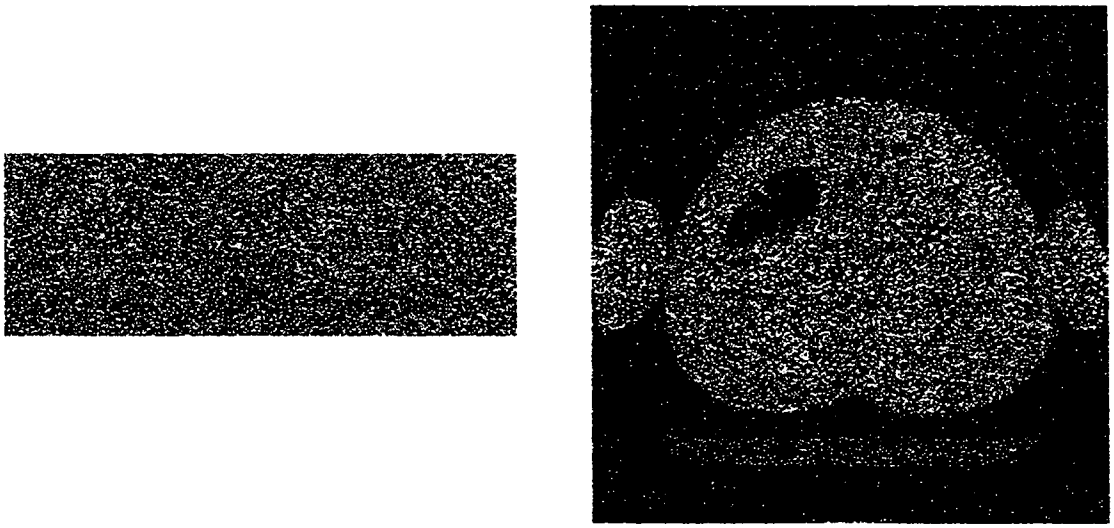
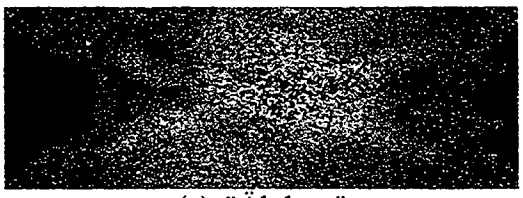
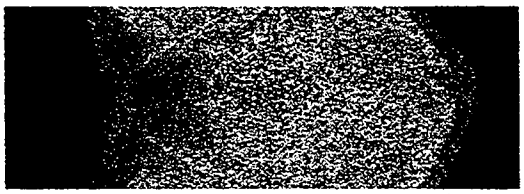


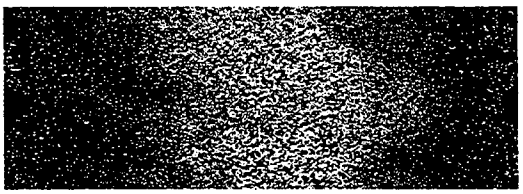
Imagem A4.19 - Senograma "Abdom" e imagem com percentual de 100% de ruído.



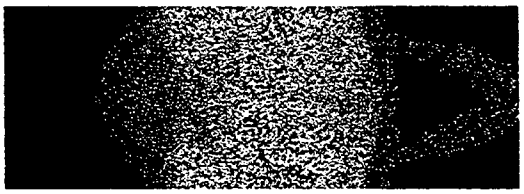
(a) "Äbdom"



(b) "Brain"

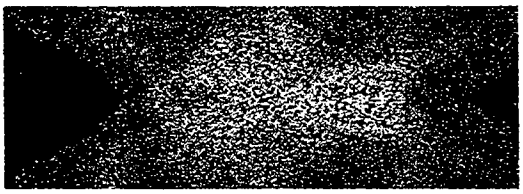


(c) "Hernia"



(d) "Head"

(d) "Phantom"



(f) "Torax"

Imagem A4.20 - Senogramas com inserçao de percentual padrão de 1% de ruído.

Referências Bibliográficas

[AGO00] AGOSTINI, L.V., "Estudo de Padrões de Compressão de Imagens para Aplicações VLSI", Trabalho Individual de Pos-Graduação, UFRGS, Porto Alegre, 2000.

[ALM98] ALMEIDA, A.B., "Usando o Computador para Processamento de Imagens Médicas", Revista Informática Médica, Vol. 1, N.6, Dezembro, 1998.

[CAL97] CALDERBANK, R. C., DAUBECHIES, I., SWELDENS, W., e YEO, B., "Lossless Image Compression using Integer to Integer Wavelet Transforms", International Conference on Image Processing (ICIP), v. 1, p. 596-599, 1997.

[DAU88] DAUBECHIES, I., "Orthonormal Bases of Compactly Supported Wavelets", Communications on Pure and Applied Mathematics, v. 41, pp. 909-996, November, 1988.

[DIC01] DICOM-Digital Imaging and Communications in Medicine, "The DICOM Standard", Published by National Electrical Manufacturers Association, Copyritht 2001. "http://medical.nema.org".

[EGG99] EGGER O., FLEURY P., EBRAHIMI T., KUNT M., . "High-Performance Compression of Visual Information - A Tutorial Review - Part I: Still Pictures" Proceedings off the IEEE, v.87, n.6, pp. 976-1011, June, 1999.

[GRA95] GRAPS, A., "An Introduction to Wavelets", IEEE Computational Science and Engineering, v. 2, n. 2, 1995.

[GRO84] GROSSMAN, A., MORLET, J., "Decomposition of Hardy Functions into Square-Integrable Wavelets of Constant Shape", SIAM Journal of Math. Anal., v. 15, n. 4, pp. 723-736, July, 1984.

[HOG98] HOGARTH, M.E., SABBATINI, R.M.E., "Informática e a Medicina do Século 21", Revista Informática Médica, Vol. 1, N.2, Abril, 1998.

[HUA94] HUANG, K., SMITH, B., "Experiments with a Lossless JPEG Codec", June, 1994. "http://www.cs.cornell.edu/Info/Projects/zeno/Projects/LJPG.html".

[JAI89] JAIN, A. K., "Fundamentals of Digital Image Processing", Prentice Hall, 1989.

[JAW94] JAWERTH, B., SWELDENS, W., "An Overview of Wavelet Based Multiresolution Analyses", SIAM Rev., v. 36, n. 3, pp. 377-412, 1994.

[MAL89] MALLAT, S. G., "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", IEEE Transactions on Pattern Analysis and Machine Intelligence, v. 11, n. 7, July, pp. 674-693, 1989.

[MAR00] MARCELLIN, M. W., GORMISH, M. J., BILGIN, A., BOLIEK, M. P., "An Overview of JPEG-2000", Proceedings of 2000 Data Compression Conference, Snowbird, March, 2000.

[MAR99] MARQUES, F. O., NETO, H. V., "Processamento Digital de Imagens", Rio de Janeiro, Brasport, ISBN 85-7452-009-8, 1999.

[MED01] MEDEIROS, L. F., "Reconstrução de Imagens Tomográficas com Redes Neurais Parcialmente Conectadas", Dissertação de Mestrado, DINF/UFPR, Curitiba, 2001.

[MEL94] MELO, A. C. W. P., "Compressão de Imagens em Escala de Cinza Utilizando a Transformada Wavelet", Dissertação de Mestrado, CEFET-PR, Curitiba, 1994.

[MIA99] MIANO, J., "Compressed Image File Formats - JPEG, PNG, GIF, XBM, BMP", Addison Wesley Longman Inc., USA, 1999.

[NET98] NETO, J.F., ALCOCER, P.R.C., "Compressão de Imagens Médicas Utilizando a Técnica JPEG-DPCM", IV Fórum Nacional de Ciência e Tecnologia em Saúde, p. 411-412, Curitiba, 1998.

[MUR96] MURRAY J. D., VANRYPER W., "Encyclopedia of Graphics File Formats, Second Edition", O'Reilly & Associats Inc, USA, 1996.

[PEN93] PENNEBAKER, W. B., MITCHELL, J. L.: "JPEG Still Image Data Compression Standard", Ed. Van Nostrand Reinhold, 1993.

[RIB96] RIBEIRO, E. P., "Tomografia de Susceptibilidade Magnética com Magnetômetro Supercondutor SQUID", Tese de Doutorado, PUC-RJ, Rio de Janeiro, dezembro, 1996.

[SAI96] SAID, A., PEARLMAN, W. A., "A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees", IEEE Transactions on Circuits and Systems for Video Technology, v. 6, n. 3, pp. 243-249, June, 1996.

[SAN01] SANCHES, I., "Compressão sem Perdas de Projeções de Tomografia Computadorizada usando a Transformada Wavelet", Dissertação de Mestrado, Departamento de Informática, UFPR, Curitiba, 2001.

[SHA93] SHAPIRO, J. M., "Embedded Image Coding using Zerotrees of Wavelet Coefficients", IEEE Transactions on Signal Processing, v. 41, n. 12, pp. 3445-3462, December, 1993.

[SWE97] SWELDENS, W., "The Lifting Scheme: A Construction of Second Generation Wavelets", SIAM J. Math. Anal, v. 29, n. 2, pp. 511-546, 1997.

[ZIV77] ZIV, J., LEMPEL, A., "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, v. IT-23, n. 3, p. 337-343, May, 1977.