Universidade Estadual de Campinas

Faculdade de Engenharia Elétrica e de Computação

Departamento de Comunicações

Uma Proposta de Estimação de Movimento para o Codificador de Vídeo Dirac

Autor: André Filgueiras de Araujo

Orientador: Yuzo Iano

Trabalho apresentado à Faculdade de Engenharia Elétrica e de Computação da UNICAMP como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Elétrica.

Comissão Examinadora

Prof. Dr. José Cândido Silveira Santos Filho (FEEC - UNICAMP)

Prof. Dr. Evaldo Gonçalves Pelaes (UFPA)

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

Araujo, André Filgueiras de
Ar15p Uma proposta de estimação de movimento para o codificador de vídeo Dirac / André Filgueiras de Araujo. --Campinas, SP: [s.n.], 2010.

Orientador: Yuzo Iano.
Dissertação de Mestrado - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Videodigital. 2. Telecomunicações. 3. Compressão de imagens - Codificação. 4. MPEG (Padrão de codificação de video). 5. Compressão de dados (Telecomunicações). I. Iano, Yuzo. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês: A proposal of motion estimation for Dirac video codec
Palavras-chave em Inglês: Digital video, Telecommunications, Image compression - Encoding, MPEG (Video coding standard), Data compression (Telecommunication)
Área de concentração: Telecomunicações e Telemática
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: José Cândido Silveira Santos Filho, Evaldo Gonçalves Pelaes
Data da defesa: 25/06/2010
Programa de Pós Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: André Filgueiras de Araujo

Data da Defesa: 25 de junho de 2010

Título da Tese: "Uma Proposta de Estimação de Movimento para o Codificador de Vídeo Dirac"

Prof. Dr. Yuzo Iano (Presidente): _____

Prof. Dr. Evaldo Gonçalves Pelaes: _____

Prof. Dr. José Cândido Silveira Santos Filho: _____

Resumo

Este trabalho tem como objetivo principal a elaboração de um novo algoritmo responsável por tornar mais eficiente a estimação de movimento do *codec* Dirac. A estimação de movimento é uma etapa crítica à codificação de vídeo, na qual se encontra a maior parte do seu processamento. O *codec* Dirac, recentemente lançado, tem como base técnicas diferentes das habitualmente utilizadas nos *codecs* mais comuns (como os da linha MPEG). O Dirac objetiva alcançar eficiência comparável aos melhores *codecs* da atualidade (notadamente o H.264/AVC). Desta forma, este trabalho apresenta inicialmente estudos comparativos visando à avaliação de métodos de estado da arte de estimação de movimento e do *codec* Dirac, estudos que fornecem a base de conhecimento para o algoritmo que é proposto na sequência. A proposta consiste no algoritmo *Modified Hierarchical Enhanced Adaptive Rood Pattern Search* (MHEARPS). Este apresenta desempenho superior aos outros algoritmos de relevância em todos os casos analisados, provendo em média complexidade 79% menor mantendo a qualidade de reconstrução.

Abstract

The main purpose of this work is to design a new algorithm which enhance motion estimation in Dirac video codec. Motion estimation is a critical stage in video coding, in which most of the processing lies. Dirac codec, recently released, is based on techniques different from the usually employed (as in MPEG-based codecs). Dirac video codec aims at achieving efficiency comparable to the best codecs (such as H.264/AVC). This work initially presents comparative studies of state-of-the-art motion estimation techniques and Dirac codec which support the conception of the algorithm which is proposed in the sequel. The proposal consists in the algorithm *Modified Hierarchical Enhaced Adaptive Rood Pattern Search* (MHEARPS). This presents superior performance when compared to other relevant algorithms in every analysed case, providing on average 79% less computations with similar video reconstruction quality.

A Adeline

Agradecimento à CAPES

Um agradecimento especial ao Programa CAPES de Formação de Recursos Humanos em Televisão Digital (RH-TVD) pelo apoio fundamental a todos que trabalham no Laboratório de Comunicações Visuais (LCV) do DECOM/FEEC/UNICAMP.

Agradecimentos

À família, sustentação que me permite toda esta aventura, exemplo de amor que carregarei sempre em mim. A meus pais e meu irmão, por serem tão especiais. A meus avós, tios e primos, pelo carinho que sempre tiveram comigo. Tenho muita sorte de ter cada um de vocês em minha vida.

Aos amigos, perto ou longe, por serem estes companheiros excepcionais, para todas as horas. Sou muito grato por todos os momentos que passamos juntos.

A Adeline, por tornar minha vida muito mais feliz e completa todos os dias.

Ao Prof. Yuzo Iano, agradeço pelo suporte contínuo através da orientação, desde os primeiros trabalhos de Iniciação Científica, ainda em 2007, até a conclusão deste Mestrado. Obrigado imensamente por ter me ajudado de forma crucial durante toda esta caminhada.

Ao Prof. Rangel Arthur, pela grande dedicação, apoio e sugestões importantes para a realização desta dissertação. Ela teria ficado muito menos completa sem seu auxílio.

A toda a equipe do LCV, pelo apoio sempre correspondido quando solicitado.

À UNICAMP, à FEEC e a todos seus funcionários.

À CAPES, pelo apoio financeiro.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

AdaMSEA: *Adaptive Modified Successive Elimination Algorithm*

AM2BT: *Adaptive Modified Two-Bit Transform*

ASP: *Adaptive Search Pattern*

BBC: *British Broadcasting Corporation*

CIF: *Common Intermediate Format*

DCT: *Discrete Cosine Transform*

DWT: *Discrete Wavelet Transform*

EARPS: *Enhanced Adaptive Rood Pattern Search*

FS: *Full Search*

GOP: *Group of Pictures*

HDTV: *High-definition Television*

HEARPS: *Hierarchical Enhanced Adaptive Rood Pattern Search*

HVS: *Human Visual System*

ISC: *Initial Search Centre Prediction*

ITU: *International Telecommunications Union*

KLT: *Karhunen-Loève Transform*

MAE: *Mean Absolute Error*

MDB: *Minimum Distortion Block*

MHEARPS: *Modified Hierarchical Enhanced Adaptive Rood Pattern Search*

MPEG: *Motion Picture Experts Group*

MSE: *Mean Square Error*

MSSIM: *Mean Structural Similarity Index*

OBMC: *Overlapped Block Motion Compensation*

PBME: *Pattern-based Block Motion Estimation*

PSNR: *Peak Signal-to-Noise Ratio*

SSIM: *Structural Similarity Index*

SBTVD: Sistema Brasileiro de Televisão Digital

SAD: *Sum of Absolute Differences*

SDTV: *Standard-definition Television*

URP:*Unit-size Rood Pattern*

VQEG: *Video Quality Experts Group*

Trabalhos submetidos/publicados pelo autor

1. André Filgueiras de Araujo, Yuzo Iano, Rangel Arthur – "A Simple and Efficient Motion Estimation Algorithm for Dirac Video Codec" – submetida à revista IEEE Transactions on Circuits and Systems for Video Technology em Maio de 2010.

2. André Filgueiras de Araujo, Yuzo Iano, Rangel Arthur – "An Adequate Objective Performance Evaluation of Dirac Video Codec Compared to H.264/AVC" – aceito para apresentação no International Telecommunications Symposium, a ocorrer em Setembro de 2010.

3. André Filgueiras de Araujo, Yuzo Iano – "Comparative Analysis of State-of-the-art Block-based Motion Estimation Techniques" – apresentado no International Workshop on Telecommunications 2009, Fevereiro 2009.

4. André Filgueiras de Araujo, Yuzo Iano – "On the Performance of promising Dirac Video Codec" – Revista Ciência e Tecnologia, Unisal, vol. 11, nº 19, 2009.

Capítulo 1
Introdução

Atualmente, o tema de codificação de fonte de vídeo vem sendo bastante debatido pela comunidade científica nacional, em decorrência do advento do Sistema Brasileiro de Televisão Digital (SBTVD). A recente entrada em operação deste sistema nas principais metrópoles brasileiras, aliada à gradual queda nos preços dos mais recentes *set-top-boxes*, tem impactado este mercado de forma significativa.

A codificação para compressão de dados de imagem e vídeo é indispensável não só para aplicações relacionadas a televisão digital, mas também às mais diversas aplicações multimídia, como *streaming* de vídeo, fotografia digital e vídeo-conferência, dentre muitas outras, na medida em que possibilita uma redução extremamente significativa da quantidade de informação a ser armazenada e transmitida.

Um sinal de um segundo de vídeo de qualidade de televisão requer cerca de 216Mbits sem compressão, fazendo necessária uma capacidade de armazenamento e de transmissão elevadíssimas, o que inviabilizaria na prática essas aplicações. Um filme de duas horas de duração requer, também sem compressão, 194 Gbytes de armazenamento, o que faria serem necessários 42 DVDs usuais de 4,7GB. Desta forma, a compressão se torna elemento chave das novas tecnologias de mídia digital (1).

É interessante constatar também que o desenvolvimento alcançado neste campo permitiu que diversas aplicações de comunicações visuais pudessem ser colocadas em prática muito rapidamente. Isto certamente não teria sido possível sem todo o esforço massivo de pesquisa e desenvolvimento nesta área nas últimas décadas (1).

Outro fator essencial para este rápido avanço foi certamente a padronização de métodos de compressão para imagem e vídeo, iniciada há cerca de vinte anos. A convergência de interesses de grandes organizações tornou prioridade o estabelecimento de um modo comum de codificação (2). Desde então, muitos padrões foram desenvolvidos, tais como: MPEG-1, MPEG-2, MPEG-4, H.261, H.262, H.263, H.263+, H.264.

A escolha de um método de codificação de fonte do sinal de vídeo para o SBTVD foi alvo de recentes e complexos estudos, culminando com a indicação do padrão H.264, conforme determina a norma brasileira ABNT NBR 15602-1 (3), válida desde dezembro de 2007. Quando comparado a outras técnicas de compressão, analisando-se vídeos e imagens estáticas, este padrão apresenta desempenho superior, conforme diversos estudos já publicados (4)(5)(6).

Uma das etapas essenciais para qualquer *codec* de vídeo (o termo *codec* é o acrônimo de "codificador-decodificador", sendo a forma mais comum de se referir a um método de codificação) como se conhece na atualidade é a de estimação de movimento. Crítica para o bom funcionamento do processo de codificação, esta visa a eliminar o máximo de redundância possível do vídeo a ser comprimido.

As técnicas mais utilizadas atualmente são baseadas em algoritmos que realizam comparações entre blocos. Essas são conhecidas como técnicas baseadas em bloco (*block-based techniques*), são eficientes e simples de implementar e são objeto de estudo deste trabalho.

Outro objeto de estudo deste trabalho é um novo *codec* lançado recentemente (sua versão 1.0.0 foi disponibilizada em setembro de 2008), que tem chamado a atenção da comunidade científica. Chamado de Dirac, em homenagem ao cientista britânico homônimo, o *codec* se destaca por utilizar conceitos distintos dos habitualmente empregados, sendo um método de codificação bastante promissor.

Este trabalho consiste no estudo de algoritmos de estimação de movimento de estado da arte e do *codec* Dirac, com o objetivo de avaliar as contribuições que têm sido dadas recentemente para o avanço destas linhas de pesquisa, além de, ulteriormente, propor um algoritmo mais eficiente para a realização de estimação de movimento no codificador Dirac.

1.1 Trabalhos relacionados

Os algoritmos de estimação de movimento para codificação de vídeo estão em constante evolução há mais de vinte anos. Abordagens distintas são utilizadas para tratar este desafio, inerente ao codificador. Três se destacam dentro do universo de possibilidades:

- Algoritmo baseado em padrão de procura

Esta linha de pesquisa compreende a maior parte dos métodos de estimação de movimento até então desenvolvidos. Iniciando com padrões de procura fixos (7)(8)(9)(10)(11), evoluiu para padrões adaptativos, com terminação rápida, utilizando-se de buscas largas seguidas de refinamento (12)(13)(14)(15)(16).

- Algoritmo baseado em construção de planos de *bits*

Métodos desenvolvidos com base nesta abordagem efetuam a construção de planos de *bits* a partir das imagens iniciais para, com comparações rápidas providas por implementação eficiente em *hardware*, tornar a procura mais rápida. Inicialmente, a utilização de um plano de *bit* foi proposta (17), evoluindo para implementações mais eficientes associando critérios adaptativos, terminação rápida e a construção de mais planos de *bit* (18)(19).

- Algoritmo ótimo baseado em redução eficiente de locais de busca

Nesta categoria, propriedades matemáticas são utilizadas com o propósito de eliminar blocos inúteis do processo de comparação, com a motivação de acelerar a comparação de blocos preservando a otimalidade de algoritmos exaustivos. Evoluções também ocorreram nesta linha, com a passagem de técnicas com análises rápidas mas pouco eficientes de blocos (20) a uma análise mais aprofundada de cada bloco (21).

Este trabalho de Mestrado se inicia com uma análise de algoritmos de estimação de movimento que busca comparar os mais avançados de cada uma destas linhas, algo até então não existente na literatura. Esta etapa é fundamental para a extração de conclusões que norteiam a construção do método proposto ao final.

O codificador de vídeo Dirac é um método recente para a compressão de vídeos livre de *royalties* proposto pela BBC (*British Broadcasting Corporation*). Ele se baseia em técnicas distintas das usualmente utilizadas em outros codificadores modernos e ainda não teve uma avaliação relevante quanto ao seu desempenho na literatura. Pelo contrário: diversos trabalhos se habilitam a realizar esta tarefa, falhando, no entanto, nos parâmetros mínimos a serem estabelecidos para uma análise consistente das comparações realizadas (22)(23)(24)(25).

Um dos objetivos desta dissertação é, portanto, fornecer uma comparação de desempenho do Dirac frente ao H.264, um dos *codecs* com melhor desempenho atualmente, após uma análise detalhada dos diversos módulos constituintes do codificador do Dirac.

Finalmente, o último objetivo deste trabalho é o de propor uma melhoria na estimação de movimento do *codec* Dirac que, como constatado, ainda é bastante ineficiente (o que tem motivado também algumas outras contribuições para este fim , não conflitantes com a proposta deste trabalho (26)(27)). A construção deste novo algoritmo (MHEARPS, *Modified Hierarchical Enhanced Adaptive Rood Pattern Search*) se baseia fortemente nos resultados obtidos da avaliação de técnicas modernas de estimação de movimento, acrescentando-se funcionalidades que o deixam mais robusto e eficiente (notadamente a implementação de busca hierárquica).

1.2 Contribuições do trabalho

O trabalho de Mestrado aqui apresentado inicialmente contempla estudos comparativos, com o objetivo de adquirir conhecimento necessário para subsidiar, ulteriormente, a elaboração de um algoritmo de estimação de movimento para o *codec* Dirac.

Desta forma, suas principais contribuições são:

1. Análise e avaliação comparativa de algoritmos de estado da arte de estimação de movimento. Esta análise compreendeu métodos baseados em formas diversas de busca, de modo ainda não realizado na literatura.
   Constatou-se o desempenho superior de um algoritmo simples e eficaz, o *Enhanced Adaptive Rood Pattern Search* (EARPS) (28). Quando comparado a outros métodos de estado da arte (29)(30), baseados em outras formas de estimar o movimento, observou-se o melhor *trade-off* apresentado.

2. Análise e avaliação do *codec* Dirac, comparando-o com o H.264/AVC (o principal *codec* existente na atualidade). O estudo comparativo realizado é o primeiro disponível que leva em conta todos os parâmetros relevantes para a obtenção de resultados significativos.

O Dirac não alcança a performance do H.264 em termos da avaliação objetiva, com o uso do PSNR (*Peak Signal-to-Noise Ratio*) e do MSSIM (*Mean Structural Similarity Index*). É importante, porém, mencionar que foram constatados indícios de que uma avaliação subjetiva pode ser favorável ao Dirac. Ademais, este se apresenta em média 60% mais rápido que o H.264.

3. Proposta de algoritmo de estimação de movimento para o codificador Dirac, o MHEARPS. Este provê qualidade de resultados similar ao algoritmo Dirac usual, empregando em média 79% menos complexidade na estimação.

1.3 Organização do trabalho

Este trabalho está organizado da seguinte forma:

- Capítulo 2: Codificação de vídeo digital

Introdução sobre o tema de codificação de vídeo digital, ressaltando a base teórica que suporta todo o desenvolvimento deste trabalho.

- Capítulo 3: Estimação de movimento

Nesta seção, o foco é dado ao processo de estimação de movimento, entrando em mais detalhes e ressaltando suas peculiaridades. Métodos de estado da arte são analisados e comparados empiricamente.

- Capítulo 4: O *codec* Dirac

Este capítulo objetiva apresentar o *codec* Dirac, explicando em detalhes seu funcionamento. O Dirac é comparado, a partir de testes, com o *codec* H.264, um dos melhores codificadores existentes na atualidade.

- Capítulo 5: Proposta de algoritmo de estimação de movimento para o codificador Dirac

Baseando-se em todo o conhecimento adquirido acerca do *codec* Dirac e de técnicas modernas de estimação de movimento, este capítulo apresenta um novo algoritmo proposto para tornar mais eficiente a estimação de movimento realizada por este *codec*.

- Capítulo 6: Conclusão

Finaliza-se este trabalho com a análise dos resultados obtidos nos tópicos anteriores, apresentando-se sugestões para trabalhos futuros.

Capítulo 2
Codificação de vídeo digital

Neste capítulo, tratar-se-á do tema de codificação de vídeo digital. Inicialmente, considerações serão feitas com relação à codificação de fonte, apresentando a teoria geral na qual o tema deste trabalho se encaixa. Introduz-se também uma teoria básica sobre vídeos digitais, para, em seguida, fornecer uma visão ampla da forma que são implementados os mais comuns esquemas de compressão de imagens e de vídeo, base sobre a qual se apoiará todo o trabalho desenvolvido.

2.1 Codificação de fonte

Um modelo básico de comunicação digital, conforme definição usual, conta com 7 blocos, conforme a Figura 2.1.

Dois estágios de codificação são definidos: codificação de fonte e codificação de canal. Após estes, ocorre a modulação do sinal a ser transmitido no canal, sob influência de ruído. No lado do receptor, as operações inversas ocorrem: demodulação, decodificação de canal e, por último, decodificação de fonte, quando se poderá obter a reconstrução da mensagem enviada.

O propósito da codificação de fonte é de eliminar redundâncias inerentes à fonte emissora de mensagens. Ou seja, o objetivo deste estágio é de reduzir ao máximo o tamanho das mensagens, de forma que o receptor consiga recuperá-las com exatidão (ou com uma boa precisão).

Já a codificação de canal, por outro lado, apresenta outro objetivo: o de adicionar redundância controlada à mensagem, com o objetivo de minimizar o erro ocorrido na transmissão. Incluem-se aí as simples técnicas de verificação de paridade, por exemplo. A ideia essencial a este tipo de codificação é de enviar informação extra para que seja explorada no decodificador e assim obter maior confiabilidade que a informação que se transmitiu é a recebida.

A modulação é um processo essencial para a transmissão, que realiza a passagem do sinal em tempo discreto para o contínuo (devido à natureza dos meios físicos utilizados). Desta forma, a modulação levará em conta características do canal – por exemplo, a frequência utilizada.

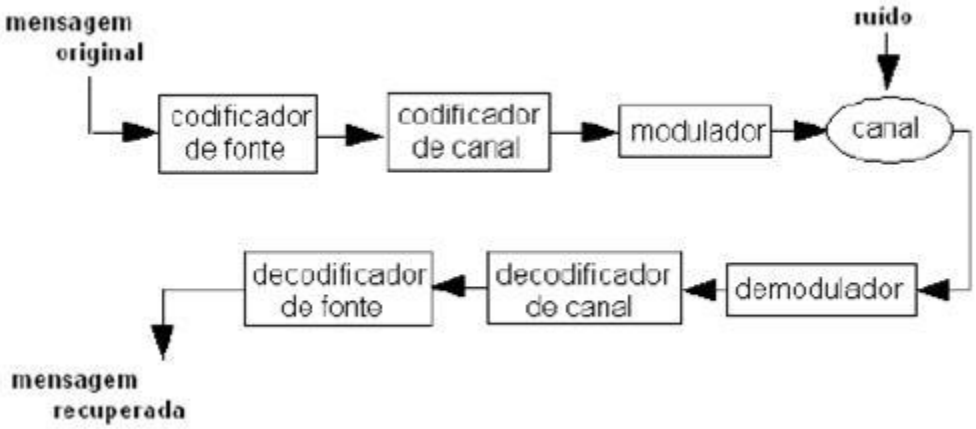


Figura 2.1: Blocos básicos de um sistema de comunicações (31).

Neste contexto, o tema desta dissertação se encaixa precisamente dentro do tema de codificação de fonte. A codificação de vídeo é uma forma de comprimir e compactar os dados provenientes deste tipo de fonte, de forma a minimizar a quantidade de informação necessária à decodificação do sinal captado pelo receptor, preservando um dado critério de fidelidade.

Dentro do tema de codificação de fonte, dois procedimentos podem ser adotados: compactação e compressão. O primeiro visa a reduzir ao máximo a quantidade de dados necessária, tal que a reconstrução por parte do receptor seja exata – sem distorção. A compressão, por outro lado, admite a presença de distorção – em geral, pequena –, para que o tamanho dos dados possa ser diminuído ainda mais.

Como será visto mais adiante, no caso de codificação de vídeo, é o procedimento de compressão que se faz mais importante, visto que este tipo de sinal apresenta muita informação que pode ser desprezada, sem perda de qualidade significativa. Isto explica o fato de o termo "compressão de vídeo" (*video compression*, em inglês) ser usado indistintamente de "codificação de vídeo" em grande parte da bibliografia.

2.2 Vídeo digital

Antes de começar a tratar da codificação de vídeo digital, faz-se indispensável a caracterização precisa de vídeos digitais, que será brevemente realizada nesta seção.

Uma imagem estática convencional é a projeção de uma cena de três dimensões em um plano de apenas duas, perdendo-se informação referente a profundidade. Um vídeo representa a cena durante um determinado período de tempo.

Uma cena "real" é contínua espacial e temporalmente – para que ela se torne digital, é necessário amostrá-la nestas duas dimensões, conforme a Figura 2.2. Na grande maioria dos casos, a amostragem espacial é realizada numa grade retangular (com a largura freqüentemente maior que a altura), com certo número de "elementos de figura" – pixels (*picture elements*). A amostragem temporal é realizada pela captura a intervalos regulares de tempo (de acordo com uma certa taxa de quadros) de imagens estáticas. Logo, um vídeo digital pode ser gerado pela simples amostragem de sua versão analógica – isto é, de um sinal elétrico que representa as imagens do vídeo.

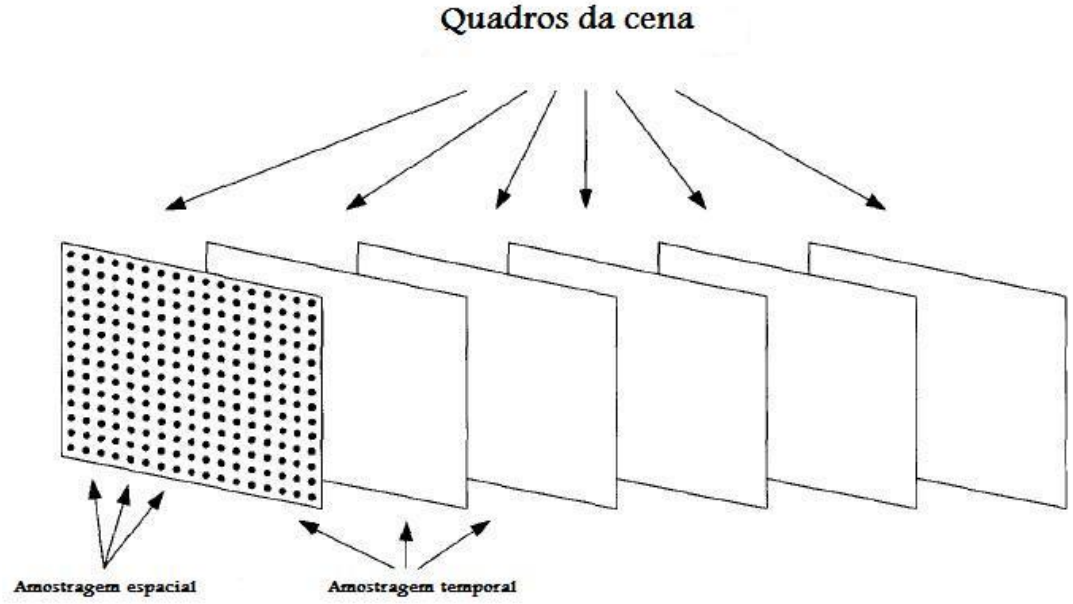


Figura 2.2: Amostragens espacial e temporal (1).

A representação de cores de uma imagem requer o armazenamento de múltiplas informações por pixel – em contraste com a representação de uma imagem em escala de cinza, quando se faz necessário apenas o armazenamento da intensidade de cinza para cada pixel. Dois espaços de cores são comumente utilizados: RGB (vermelho, verde e azul) e YCrCb (luminância, crominância vermelha e crominância azul) para tal. No primeiro caso, cada pixel é representado

por três números, que indicam sua proporção de vermelho, verde e azul. Para o espaço de cor YCrCb, os três números representarão a luminância, a crominância vermelha e a azul. Este último é bastante mais eficiente que aquele, pelo fato de o sistema visual humano ser mais sensitivo ao brilho que às cores: o sistema RGB não provê uma forma simples de tirar vantagens disto, já que as três cores são igualmente importantes (1).

Toda a teoria e prática de comunicações por imagem e vídeo têm como base a forma como o ser humano recebe e processa a informação visual. Não faz sentido se utilizar de informações absolutamente imperceptíveis ao olho humano, que podem ser descartadas. Um dos objetivos do design de *codecs* é minimizar a distorção percebida da imagem/vídeo.

O Sistema Visual Humano (ao qual se refere comumente como HVS, do inglês *Human Visual System*) é o sistema pelo qual o observador humano vê, interpreta e responde ao estímulo visual. Seus componentes principais são o olho, a retina, o nervo ótico e o cérebro.

Alguns exemplos de características do HVS que influenciam diretamente os sistemas de vídeo digital são: maior sensibilidade a baixas frequências espaciais, maior sensibilidade a detalhes de luminância que de cores, a ilusão de movimento suave alcançada pela apresentação de quadros a taxas superiores a 20-30Hz (1).

Outra das observações constatadas com relação ao HVS é que as respostas às observações variam de indivíduo para indivíduo. Este fato é um complicador do estudo da qualidade das imagens – essencial para o design de *codecs* eficientes. Os métodos existentes para realizar este estudo são classificados em subjetivos e objetivos.

Os primeiros dependem da avaliação de pessoas, submetidas a uma sequência de imagens pré-determinada, seguindo-se de uma análise estatística dos resultados obtidos, existindo diversos procedimentos padronizados para tal.

Os métodos objetivos são muito mais simples e, por esta razão, são os mais utilizados. Estes realizam uma comparação entre as duas imagens em questão, baseando-se diretamente nos valores de seus pixels. Certamente a medida mais utilizada é o PSNR (*Peak Signal-to-Noise Ratio*), calculado segundo a Equação 2.1.

$$PSNR_{dB} = 10log_{10}\frac{(2^n - 1)^2}{MSE} \tag{2.1}$$

onde n é o número de bits utilizados para o armazenamento de um pixel e MSE é o erro quadrático médio (*Mean Square Error*) entre a imagem a comparar e a referência, dado por:

$$MSE = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(C_{ij} - R_{ij}\right)^2 \tag{2.2}$$

O PSNR, dado em decibéis, é calculado para cada componente (em geral, utiliza-se a luminância das imagens), de forma muito fácil e é frequentemente usado para comparar a qualidade de imagens codificadas e não codificadas. Entretanto, este sofre de diversas limitações, a principal sendo que ele não é bem correlacionado com medidas subjetivas (1). Imagens com erros muito distintos, alguns muito mais visíveis que outros, podem apresentar o mesmo PSNR, conforme mostra a Figura 2.3.

Figura 2.3: Comparação de imagens com diferentes tipos de distorção, todas com MSE = 210 (mesmo PSNR). (a) original. (b) com aumeto de contraste, MSSIM = 0,9168. (c) com média deslocada, MSSIM = 0,9900. (d) comprimida com JPEG, MSSIM = 0,6949. (e) imagem borrada, MSSIM = 0,7052. (f) com ruído impulsivo "salt-pepper", MSSIM = 0,7748 (32).

Por este motivo, foram elaboradas outras diversas técnicas objetivas de medição, sendo mais utilizado atualmente pela comunidade científica o MSSIM (*Mean Structural Similarity Index*) (32). Este indicador é baseado em informação estrutural, muito importante para o HVS, e é baseado na expressão:

$$SSIM\ (x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x{}^2 + \mu_y{}^2 + C_1)(\sigma_x{}^2 + \sigma_y{}^2 + C_2)} \tag{2.3}$$

onde *x* e *y* são as duas imagens comparadas, μ_x e μ_y são médias locais para cada pixel (uso de uma janela de análise que percorre todos os *pixels*), calculadas numa região no seu entorno, σ_x e σ_y são os desvios-padrão (calculados por estimadores não-polarizados) nesta mesma região, σ_{xy} é a correlação cruzada, e C_1 e C_2 são constantes usadas para evitar resultados instáveis quando $\mu_x{}^2 + \mu_y{}^2$ ou $\sigma_x{}^2 + \sigma_y{}^2$ estão muito próximos de zero.

A elaboração da expressão para cálculo do SSIM é baseada na influência de três componentes essenciais: luminância (baseado na comparação das médias locais), contraste (baseado na comparação dos desvios-padrão locais) e estrutura (baseado na comparação da correlação cruzada com o produto dos desvios-padrão). O produto destes três fatores, associado a uma escolha adequada de constantes, resulta na Equação 2.3.

O resultado do cálculo desta expressão fornece, na realidade, um mapa de índices, calculados em cada região (de fato, será obtido um valor para cada *pixel*). Desta forma, obtém-se o MSSIM pela média destes índices, já que, na prática, requer-se o uso de um único valor por imagem (isto permite inclusive alocar maior importância a uma região mais importante da imagem, pela utilização de uma média ponderada neste caso). A Figura 2.3 ilustra a eficácia de tal indicador, para diversos tipos de ruído afetando a imagem. Outra vantagem deste é sua independência com relação à resolução da imagem a ser avaliada, como também acontece com o PSNR.

É importante citar a iniciativa do *Video Quality Experts Group* (VQEG), grupo de especialistas em vídeo, ligados à *International Telecommunications Union* (ITU), cujo objetivo é propor e recomendar métodos objetivos de avaliação de qualidade de imagens reconstruídas. Até o momento, alguns indicadores foram propostos para os formatos 525/60 (formato com 525

pixels de altura e 60 quadros por segundo) e 650/50 (formato com 650 pixels de altura e 50 quadros por segundo) – as métricas para HDTV estão em curso de avaliação.

2.3 Compressão de imagens

A compressão de imagens é a base para a compressão de vídeo. Diversos princípios aqui utilizados são também empregados nos *codecs* de vídeo, de forma que se mostra interessante apresentar alguns métodos empregados na codificação de imagens.

Na compressão de imagens, assim como na de vídeo, se explora uma série de propriedades. Pixels vizinhos são, em geral, bastante correlacionados – ou seja, a redundância espacial em imagens e vídeos é bastante significativa. Outra característica relevante a ser explorada é a limitação do HVS – por exemplo, a sensibilidade humana a frequências baixas é muito maior que com relação a frequências mais altas, o que possibilita o descarte de certas faixas de frequências, com perdas muito pequenas.

Codificação por transformada

A forma atualmente mais utilizada para tirar proveito destas características é a utilização de uma codificação por transformada. Neste caso, aplicando-se este tipo de operação, obtém-se os dados em outro domínio, o da frequência, onde as redundâncias são bastante reduzidas, resultando em poucos coeficientes com valores importantes e, outros (a maioria), insignificantes, que podem ser desprezados sem afetar muito a qualidade da imagem resultante do processo de transformação inversa.

Duas opções se colocam: aplicar a transformação a blocos da imagem ou à imagem inteira. No primeiro caso, tipicamente se aplica a transformada a blocos de tamanho 4x4, 8x8, ou 16x16. A transformada baseada em blocos que obtém resultados ótimos em termos de descorrelação e de empacotamento de energia é conhecida como *Karhunen-Loève Transform* (KLT). Entretanto, sua aplicação prática é bastante restrita, já que seus coeficientes são dependentes dos dados de entrada, o que faz sua utilização se tornar bastante dispendiosa (33).

A transformada baseada em blocos mais utilizada é a *Discrete Cosine Transform* (DCT), que apresenta resultado bastante próximo ao da KLT, sendo também muito mais eficiente computacionalmente. Utilizada em muitos dos padrões internacionais, como o AVC/H.264 – um

dos melhores existentes atualmente -, sua aplicação em imagens é interessante por não trabalhar com números complexos e por ter a propriedade de separabilidade em duas dimensões.

A Figura 2.4 apresenta um exemplo de aplicação da DCT em algumas imagens simples. Nota-se a grande concentração de energia em poucos coeficientes (quanto mais claro, maior o valor do coeficiente), no ângulo superior esquerdo, correspondente às baixas frequências horizontal e vertical.

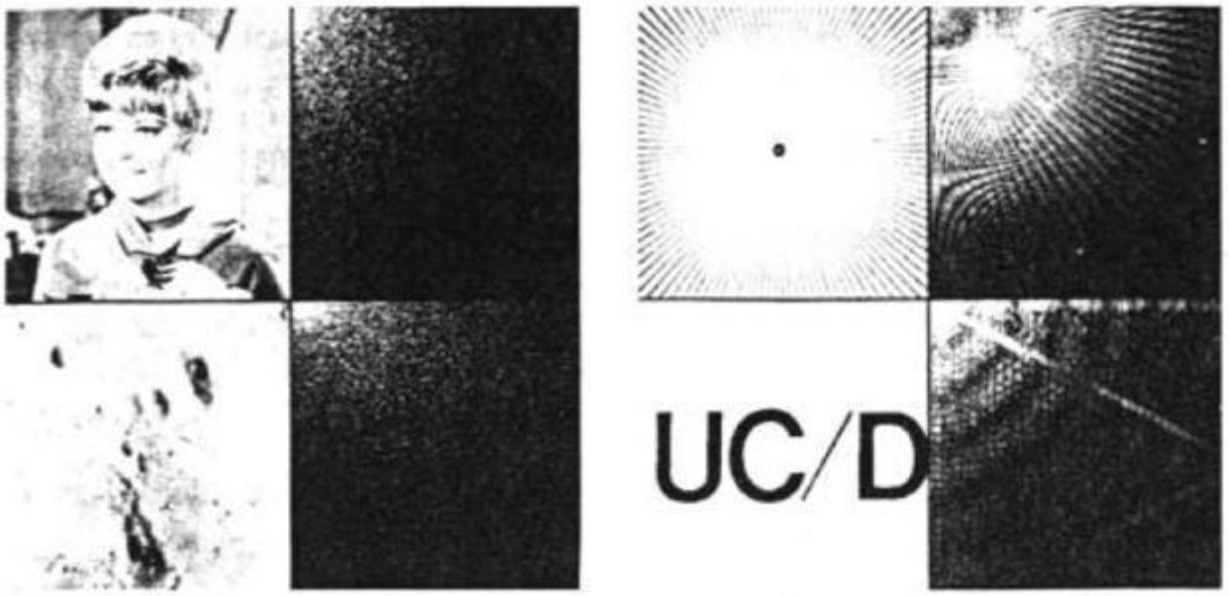


Figura 2.4: Exemplo de aplicação da DCT em imagens comuns (34).

No caso da aplicação da transformada na imagem completa, a mais popular transformação utilizada é a *Discrete Wavelet Transform* (DWT). Esta transformação é aplicada à imagem original de modo a decompô-la em imagens contendo sub-bandas de frequências, sendo que a cada estágio o número de coeficientes é metade do anterior. A DWT é utilizada no *codec* Dirac, que é estudado em detalhes neste trabalho.

O método é implementado da seguinte forma, como mostrado na Figura 2.5: filtram-se as linhas (colunas) inicialmente da imagem original, duas vezes – uma com um filtro passa-baixas e uma com um filtro passa-altas. As duas imagens resultantes filtradas têm, então, suas colunas (linhas) subamostradas, retornando ao tamanho da imagem inicial. Neste segundo momento, filtram-se as colunas (linhas) com os dois filtros e subamostram-se as linhas (colunas), obtendo quatro subimagens que, juntas, têm o tamanho da imagem original: LL - dois filtros passa-baixas

-, HL - passa-altas nas linhas (colunas) e passa-baixas nas colunas (linhas) -, LH – passa-baixas nas linhas (colunas) e passa-altas nas colunas (linhas) – e HH (dois filtros passa-altas). Estágios subsequentes ocorrem por meio da replicação deste processo na subimagem LL. Um exemplo da aplicação de três estágios da DWT é apresentado na Figura 2.6.

Os filtros utilizados na decomposição da imagem são chamados de filtros de análise, enquanto os utilizados na reconstrução são referidos como filtros de síntese. Estes processos de aplicação dos filtros visando à compressão e, posteriormente, reconstrução, foram definidos por Mallat (35), para a DWT. Uma representação simples em uma dimensão dos estágios de decomposição e de síntese é apresentada na Figura 2.7 e na Figura 2.8.

Como descrito para o caso de decomposição de imagens, na Figura 2.7 representa-se o mesmo processo: realiza-se a decomposição em frequências altas e baixas, sendo que as baixas continuam a ser decompostas nos diversos estágios, empregando também subamostragem a cada estágio. Na síntese, Figura 2.8, o processo é invertido, realizando-se inicialmente superamostragem, transformando-se os dados e em seguida somando os resultados ao fim de cada etapa.

Os filtros da Figura 2.7 são os de análise, os de síntese sendo os da Figura 2.8. Estes filtros são interdependentes: dado um de análise, obtém-se o de síntese, e vice-versa. No caso especial de as *wavelets* serem ortogonais (que é o caso representado nas figuras), os filtros passa-baixas de análise e de síntese são iguais, acontecendo o mesmo com os filtros passa-altas.

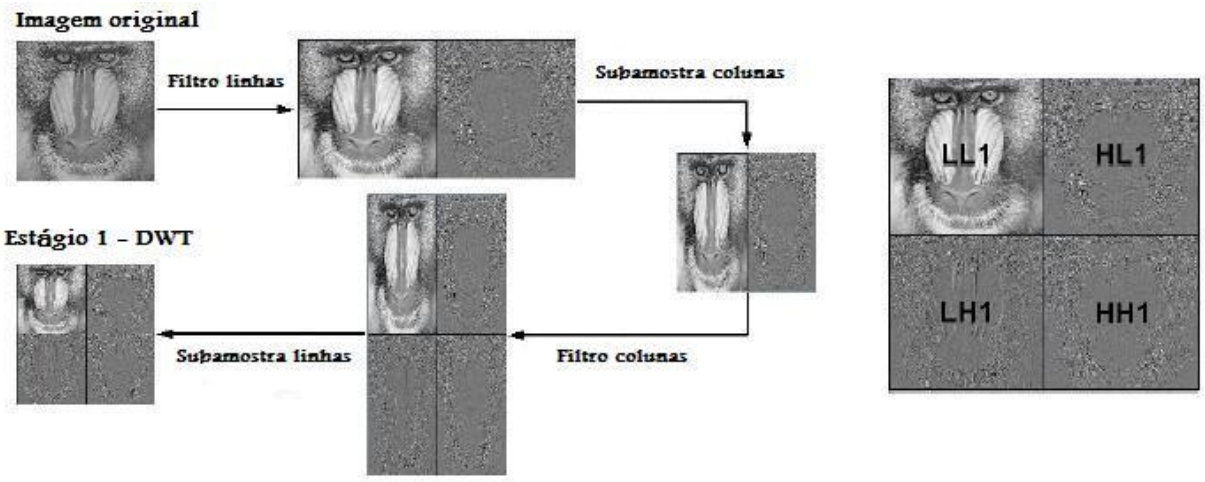


Figura 2.5: Aplicação de um estágio da DWT; à direita, o resultado deste estágio (36).

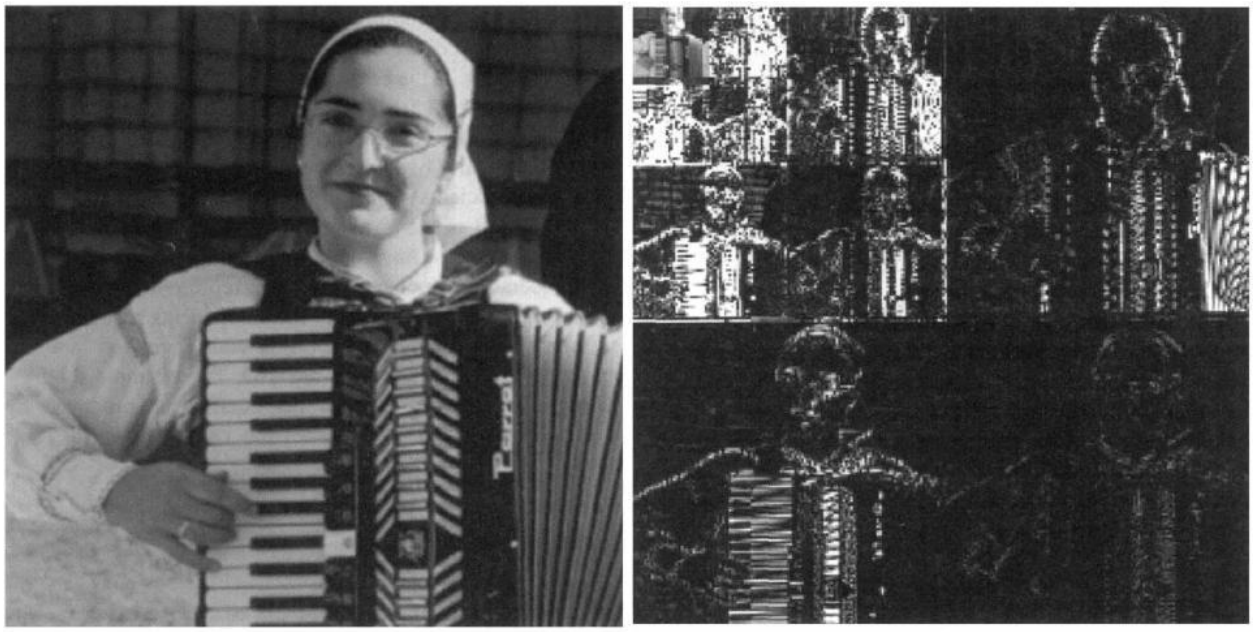


Figura 2.6: Exemplo de decomposição de imagens usando a DWT (1).

Percebe-se que muitos dos coeficientes de alta frequência resultantes são insignificantes, com valor zero ou próximos disto. Desta forma, desenvolveram-se técnicas que descartam inteligentemente estes coeficientes desprezíveis, com destaque para o SPIHT (*Set Partitioning Into Hierarchical Trees*) (37).

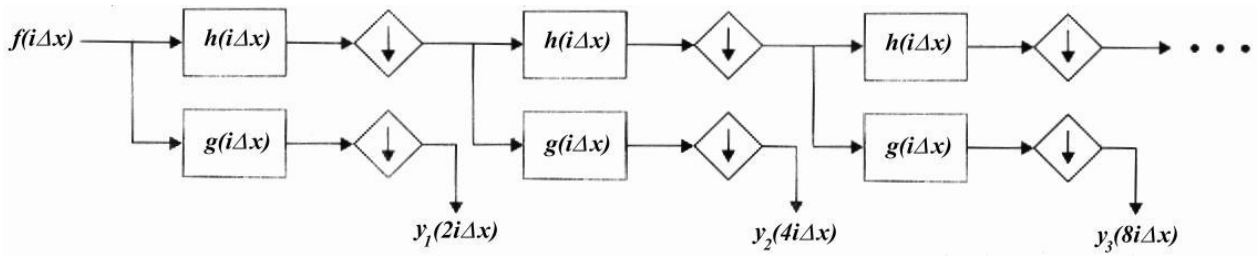


Figura 2.7: Esquema empregado na decomposição utilizando a DWT, para uma dimensão (38).

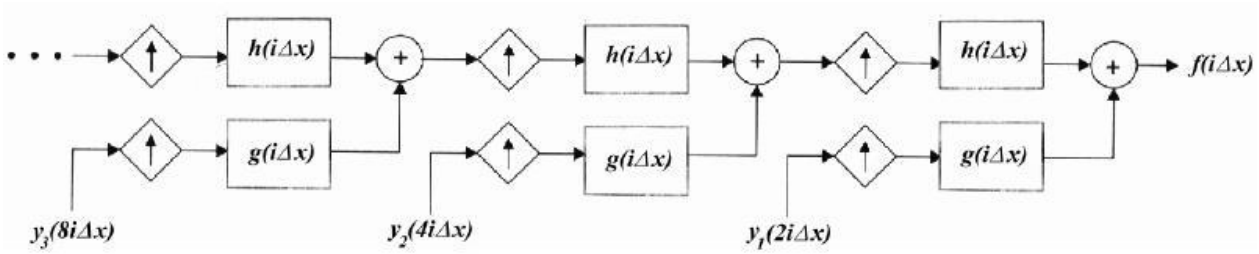


Figura 2.8: Esquema empregado na síntese utilizando a DWT, para uma dimensão (38).

Estas se baseiam na exploração da idéia da ligação entre coeficientes "pais" e "filhos", localizados na "raiz da árvore" (na região LL) e nos seus "galhos" (nas regiões LH, HL e HH), respectivamente, como mostrado na Figura 2.9. A análise de significância dos coeficientes começa nos "pais", seguindo-se possivelmente a cada geração de "filhos". Caso um coeficiente não seja considerado significativo, a análise deste ramo da árvore é interrompida – este e seus filhos serão descartados. A significância de um coeficiente é determinada por sua comparação com um limiar; caso este limiar seja alto, muitos coeficientes serão rejeitados, resultando em grande compressão, com mais perda. Caso contrário, com um limiar baixo, a maioria dos coeficientes será levada em conta e uma menor compressão resultará, com melhor qualidade da imagem resultante.

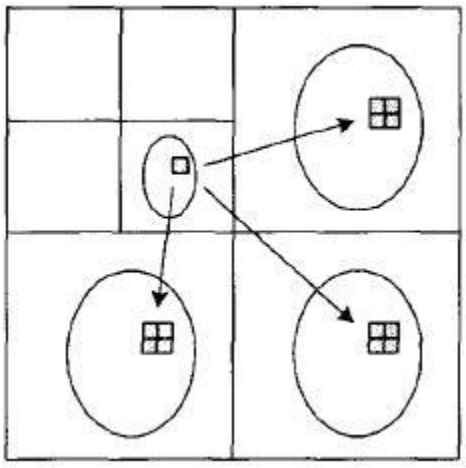


Figura 2.9: Relação "pai"-"filho" dos coeficientes da DWT (1).

Quantização

A quantização é um processo irreversível que, por esta razão, representa perdas inevitáveis na qualidade da imagem. Esta etapa deve ser pensada como um mapeamento entre dados de entrada, que podem apresentar qualquer valor num determinado intervalo e dados de saída, que estão restritos a alguns valores possíveis.

Os quantizadores escalares podem ser lineares ou não-lineares, como apresentado na Figura 2.10. Em geral, quantizadores não-lineares possuem uma zona dita "morta" – nesta, desproporcionalmente maior que os outros intervalos, uma larga gama de valores são mapeados a zero.

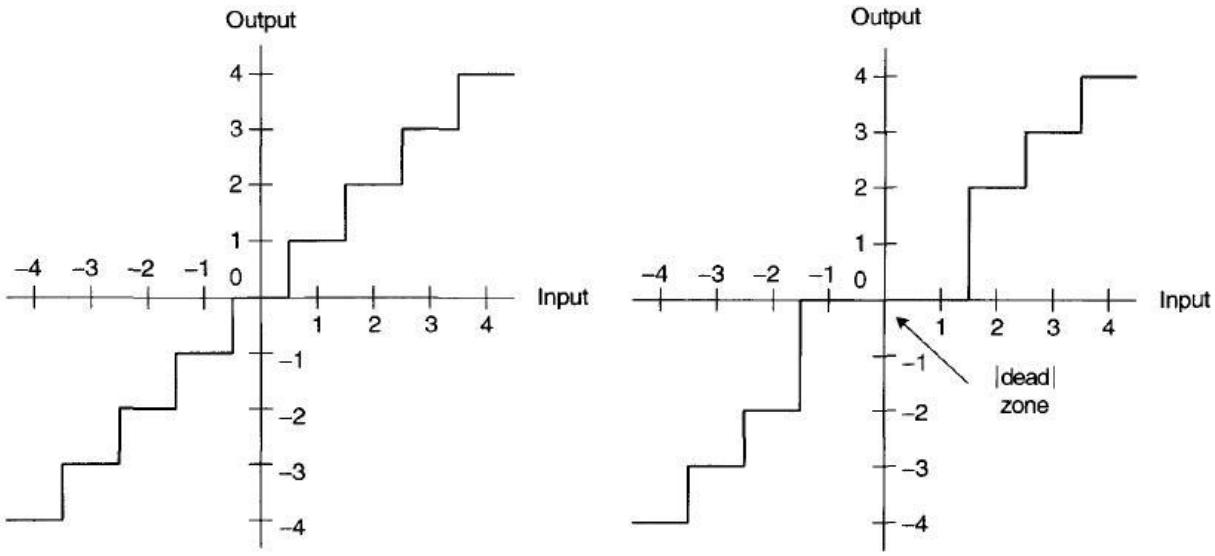


Figura 2.10: Quantizador linear (esquerda) e não-linear (direita), com zona morta (1).

Técnicas mais avançadas de quantização compreendem principalmente quantização vetorial. Entretanto, estas ainda não são muito utilizadas por *codecs* de imagem e vídeo.

Codificação de Entropia

Esta etapa consiste em pura compactação dos dados, utilizando-se de técnicas como: ordenação apropriada de coeficientes, codificação de "corrida de zeros", dentre outras, além dos algoritmos de codificação de entropia propriamente ditos.

A ordenação de coeficientes é utilizada principalmente em conjunto com a DCT, para que os coeficientes significativos (alocados na região superior esquerda da imagem transformada) sejam agrupados conjuntamente – utilizando-se ordenação em zig-zag, como na Figura 2.11. Procedendo desta forma, coeficientes insignificantes também serão agrupados conjuntamente – em particular, se observa a presença de muitos coeficientes quantizados de valor zero, as chamadas "corridas de zeros".

Estas "corridas de zeros" são codificadas geralmente pelo uso da técnica "*run-level*". Os dados de entrada são codificados em um par (*run*, *level*), correspondente ao número de zeros precedentes e ao valor que segue estes zeros. Por exemplo, o par (10,5) indica a presença de 10 zeros seguidos de um 5.

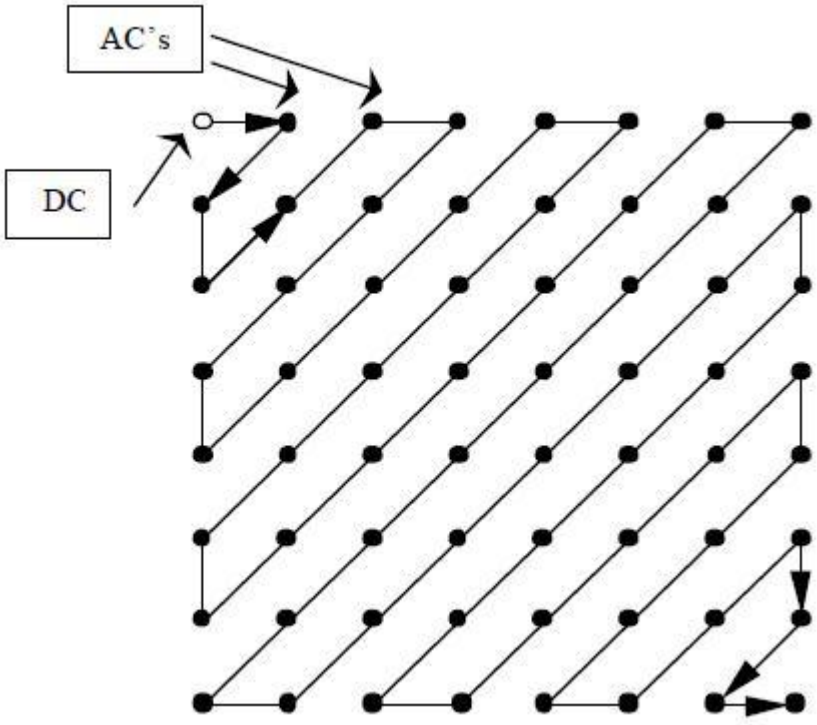


Figura 2.11: Ordenação em zigzag para coeficientes resultantes da DCT (33).

Os algoritmos de codificação de entropia propriamente dita mais utilizados são baseados na codificação aritmética e em códigos de Golomb, que proporcionam compressão muito próxima ao limiar estabelecido pela Teoria da Informação. Ambas as técnicas apresentam eficiência maior que a codificação de Huffman.

Para ilustrar seu funcionamento, um exemplo de codificação de Golomb será apresentado aqui. Esta codificação atua sobre uma sequência binária que passou por um pré-processamento para transformá-la em inteiros referentes a corridas de zeros. A ordem do código (m) é o parâmetro essencial a se definir – ele é calculado a partir das probabilidades dos bits. Para o exemplo que segue, utilizar-se-á m = 9.

m será o divisor de cada número da corrida de zeros a ser codificada. Sendo n tal número, ter-se-á $n = mq + r$, onde q é o quociente e r o resto – q e r serão os parâmetros a serem codificados.

Para uma sequência de bits:

000000000000010000100000000000000000100000000010000000000000000000

Codificada em *run-length* por:

13, 4, 17, 9, 19

Expressa em termos de quociente e resto (q,r):

(1,4), (0,4), (1,8), (1,0), (2,1)

r será representado por um código de Huffman (conforme Figura 2.12), enquanto q será representado por um código unário (o valor do quociente é codificado pela quantidade de bits 1). Entre quociente e resto, coloca-se um bit 0, de forma a separá-los.

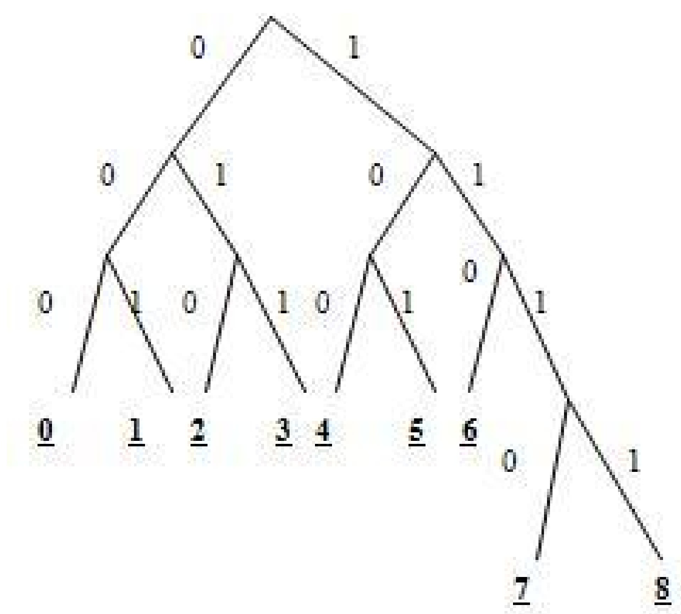


Figura 2.12: Árvore de atribuição de códigos para o resto - exemplo de utilização de códigos de Golomb.

O código de Golomb será, para o exemplo, então:

10100, 0100, 101111, 10000, 110001

Ou seja, a sequência inicial, de 66 bits, foi reduzida a esta de 26 – a mensagem final representa apenas 39,4% da inicial.

Decodificação

O processo de decodificação de imagens é basicamente o inverso do de codificação. A imagem reconstruída não poderá ser idêntica à original, devido à irreversibilidade da quantização.

Os passos básicos a se seguir são (1):

1. Extração de pares *run-level* da seqüência de bits, utilizando-se um decodificador de entropia;
2. Conversão nos devidos valores e reordenação de acordo com o zig-zag;
3. Multiplicação de cada coeficiente pelo seu fator de escala;
4. Aplicação da transformada inversa para reconstrução da imagem.

2.4 Compressão de vídeos

Vídeos digitais são compostos de uma sequência de imagens e, por isto, apresentam outra propriedade interessante a ser explorada, a de redundância temporal. Muita da informação presente em um quadro em um dado instante continuará no instante seguinte, o que proporciona uma repetição de informação muito significativa, evidenciando-se a possibilidade de se codificar apenas as mudanças ocorridas nas cenas. Este fato é o que distingue fundamentalmente a compressão de imagens e vídeos.

Duas opções de compressão existem para vídeos: *intra-frame* e *inter-frame*. A primeira consiste simplesmente na compressão de cada imagem independentemente das outras, sem se aproveitar da redundância temporal. Este método tem sido bastante empregado em sistemas de edição que necessitam de acesso rápido a qualquer quadro do vídeo.

Compressão mais eficiente pode ser obtida, entretanto, utilizando-se predição *inter-frame*, quando quadros anteriormente analisados são empregados como base para localização de partes da imagem do quadro atual. Neste caso, o decodificador tem a possibilidade de criar uma

predição da imagem a ser reconstruída, o que faz ser suficiente o envio de uma imagem residual (obtida pela diferença entre a imagem a ser decodificada e a predição gerada).

A imagem residual será codificada utilizando-se normalmente operações de transformada, quantização e codificação de entropia, da mesma forma que feito na compressão de imagens estáticas. O importante neste processo é a realização de uma predição eficiente – pois, desta forma, a imagem residual conterá muito pouca informação a ser codificada.

Estimação e compensação de movimento

Boa parte da informação presente em imagens previamente tratadas simplesmente se repete – o que ocorre principalmente quando a câmera está fixa -, o que possibilita o não-envio destas regiões. Entretanto, de forma mais geral, quando há uma mudança significativa de uma imagem para outra, geralmente o que ocorre é a movimentação dentro da cena e um simples deslocamento possibilita a localização do que se mexeu no quadro anterior.

Por esta razão, utilizam-se dois procedimentos, chamados de estimação e compensação de movimento, que permitem identificar a localização para onde os objetos se deslocaram (estimando o movimento) e efetivar este desconto para obter a imagem residual (compensando o movimento), necessitando-se apenas do envio/armazenagem do deslocamento a efetuar para achar a região na imagem-base (este deslocamento é comumente chamado de vetor de movimento - *motion vector*).

Estes vetores de movimento representam informações adicionais a serem transmitidas/armazenadas e, portanto, também são sujeitos a codificação de entropia. Normalmente, usam-se as mesmas técnicas empregadas para a codificação de pares *run-level*, baseando-se nos algoritmos aritméticos e de Golomb: aos vetores de movimento mais comuns são atribuídos códigos menores, enquanto os mais raros são codificados com códigos mais extensos.

A estimação de movimento é uma etapa computacionalmente muito intensiva e o design de um algoritmo eficiente impacta de forma dramática o desempenho do codificador. Esta etapa é objeto de estudo detalhado deste trabalho, e será tratada no capítulo seguinte.

Codecs híbridos

Os *codecs* mais bem-sucedidos atualmente são os chamados híbridos: eles lidam com técnicas avançadas de estimação e compensação de movimento, empregando também métodos de codificação de imagens estáticas (39). Um exemplo de tal codec é mostrado na Figura 2.13.

É importante notar que, mesmo no codificador, é necessário haver uma etapa de decodificação, realizada com os dados transmitidos/armazenados. Isto se dá em razão de a predição realizada no decodificador se basear nestes dados transmitidos/armazenados. Logo, se a predição fosse realizada com as imagens originais no codificador, a predição obtida no decodificador não seria a prevista, acarretando erros significativos na imagem reconstruída, visto que a imagem residual e os vetores de movimento transmitidos/armazenados se refeririam à imagem original, e não à reconstrução possível na recepção. Este erro, que pode ser pouco significativo no começo da transmissão, se acumulará durante a decodificação dos quadros seguintes, já que uma predição é sempre baseada nas imagens anteriormente decodificadas que estariam, neste caso, incorretas.

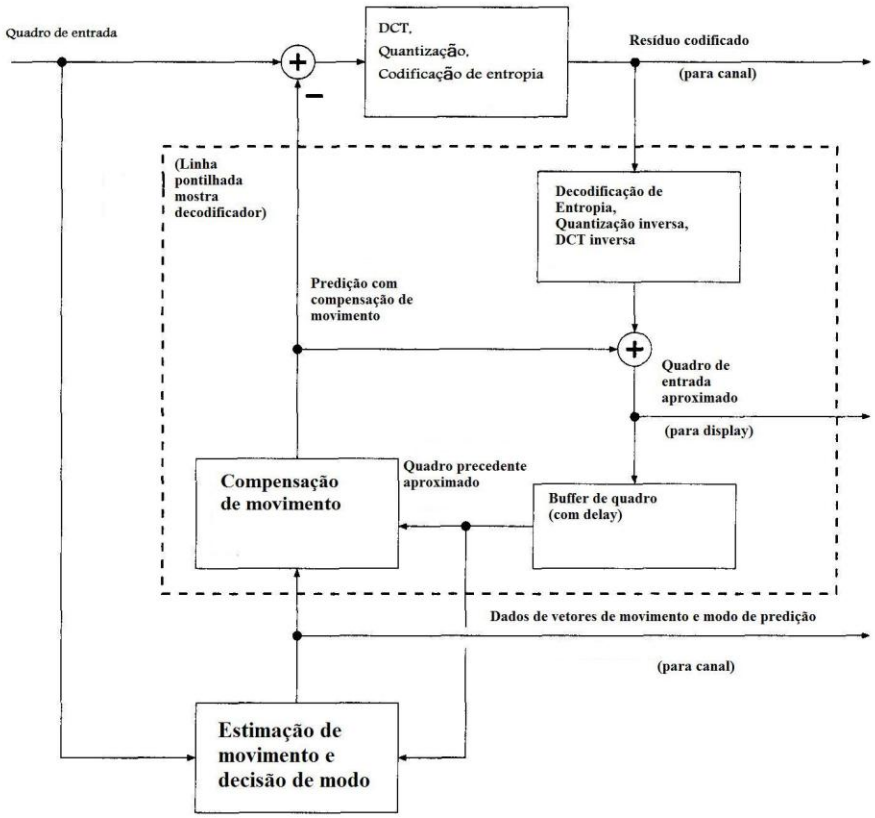


Figura 2.13: Exemplo de codificador híbrido utilizando DCT (39).

Tais *codecs* têm de lidar com uma série de problemas de otimização, tais como (39):

1. Como segmentar cada imagem;
2. Trocar uma área da figura ou não por conteúdo *intra-frame*;
3. Caso não decida por usar conteúdo *intra-frame*:
    a. Como realizar estimação de movimento;
    b. Como selecionar a aproximação a usar como refinamento da predição *inter-frame*;
4. Se trocar a área por conteúdo *intra-frame*, qual aproximação usar para fazê-lo.

Decodificação

O decodificador de vídeo é muito mais simples que seu codificador. A decodificação não necessita implementar a estimação de movimento, etapa bastante dispendiosa, já que os vetores de movimento são transmitidos no *bitstream*.

A diferença básica do decodificador de vídeo para o de imagem é a necessidade de construir uma predição da imagem para cada quadro *inter-frame* a ser decodificado. Esta operação é realizada a partir da utilização dos vetores de movimento e dos quadros de referência utilizados para a predição.

À parte este procedimento, empregam-se basicamente todas as operações básicas realizadas no decodificador de imagens estáticas, com a diferença de que a imagem preliminarmente reconstruída conterá apenas o resíduo a ser acrescentado à imagem predita, compondo assim o quadro completo decodificado.

A exceção a este caso se dá quando a imagem a ser decodificada é do tipo *intra-frame*. Nesta situação, não há a construção de uma predição e toda a informação necessária para sua decodificação não depende de quadros anteriores nem futuros.

Capítulo 3
Estimação de movimento

Os padrões de vídeo digital usualmente compreendem apenas a etapa de decodificação. Contanto que o *bitstream* esteja conforme o formato descrito pela especificação, ele será capaz de ser decodificado. Desta forma, há grande liberdade para o design do codificador.

Entretanto, constata-se que a possibilidade de melhorias nos últimos estágios dos *codecs* de vídeo é limitada, visto que a maioria das operações envolvidas nestas etapas (transformação, quantização e codificação de entropia) é bastante ligada ao formato obrigatório do *bitstream*. Por outro lado, a etapa de estimação de movimento, anterior a estas, apresenta possibilidade imensa de aperfeiçoamento.

Estimação de movimento eficiente reduz a energia do quadro residual a ser transmitido/armazenado e, desta forma, pode gerar ganhos de compressão muito altos. Esta etapa, entretanto, é, na maioria dos casos, bastante intensiva em complexidade computacional – de forma geral, uma compressão mais eficiente demanda mais processamento -, de modo que é essencial que sejam estudados estes *trade-offs* (1). A estimação de movimento é responsável por até 50% da computação encontrada em todo o sistema de codificação, sendo sua parte mais complexa (40).

Neste capítulo, tratar-se-á em mais detalhes este tema – estimação de movimento. Inicialmente, são apresentados princípios básicos, seguindo-se a descrição das principais técnicas utilizadas atualmente. Na última seção, um estudo com algoritmos de estado da arte, parte da contribuição deste trabalho, é apresentado.

3.1 Princípios básicos

O objetivo da estimação de movimento é a criação de um modelo do quadro a ser codificado, o mais preciso possível, utilizando-se, neste intuito, quadros previamente codificados de referência, que podem ser quadros do "passado" (*forward prediction*) ou do "futuro" (*backward prediction*). É essencial também que o algoritmo utilizado tenha complexidade computacional aceitável.

A compensação de movimento encarrega-se de efetuar a utilização do quadro predito (pela etapa de estimação de movimento), descontando-o do quadro original, de modo que será necessário o envio apenas do quadro residual (em conjunto com os parâmetros a serem utilizados na decodificação, evidentemente).

O esquema básico de estimação e compensação de movimento é apresentado na Figura 3.1. A estimação de movimento, como já descrito, cria um modelo a partir de quadros de referência para se aproximar o máximo possível do quadro a ser codificado – de acordo com um critério de semelhança a ser estabelecido previamente. Em seguida, este último é compensado, pela subtração do modelo, produzindo um quadro residual. Este será codificado e transmitido/armazenado, conjuntamente com os vetores de movimento, que indicarão os deslocamentos a serem realizados nos quadros de referência para a construção do modelo.

Ao mesmo tempo, o quadro residual é decodificado e adicionado ao modelo para reconstruir uma cópia do quadro que será montado no decodificador. Este quadro reconstruído será armazenado para ser utilizado como referência para quadros posteriores (1).

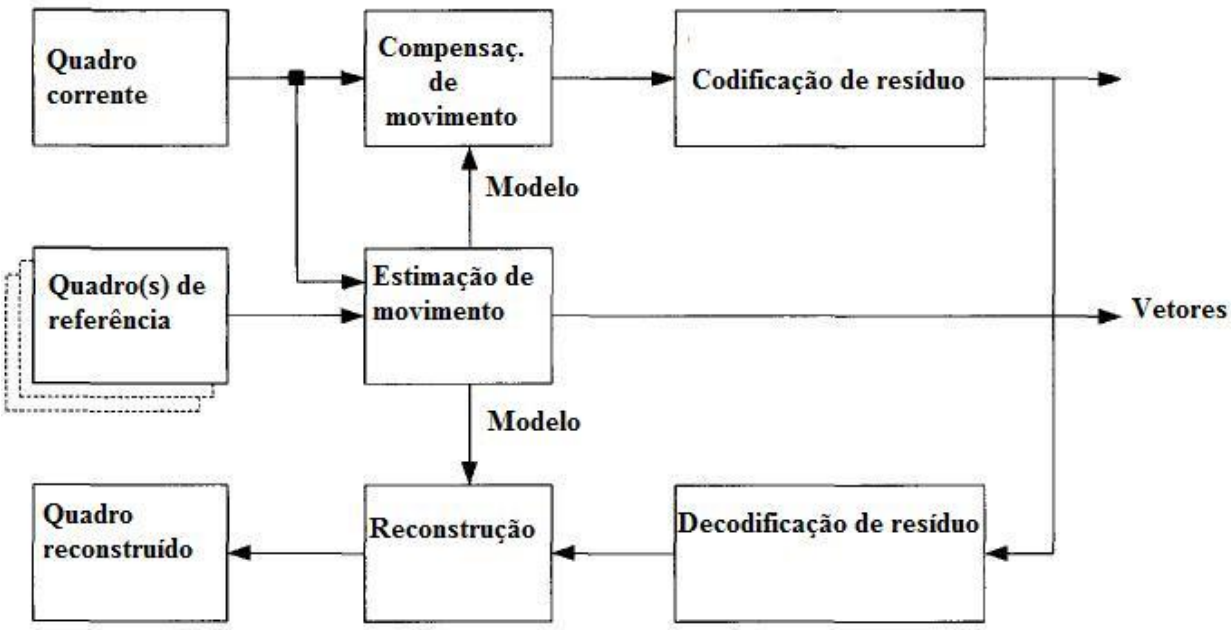


Figura 3.1: Diagrama de blocos de estimação e compensação de movimento (1).

3.2 Algoritmos baseados em comparação de blocos

Os algoritmos de estimação de movimento mais utilizados são aqueles baseados em comparação de blocos (*block-based motion estimation*). Estes são base para a grande maioria dos codificadores, senão todos, utilizados nos diversos padrões de codificação de vídeo.

O processo de comparação de blocos é empregado, na maioria dos casos, apenas para a componente de luminância, já que esta apresenta a maior parte da informação/energia de cada quadro. Esforços para a utilização das componentes de crominância são, em geral, desnecessários.

Após a definição do tamanho do bloco para o qual se procurará outros blocos similares nos quadros de referência, inicia-se o algoritmo de comparação. Usualmente, a procura por blocos similares começa por regiões vizinhas ao bloco em questão. Em muitos casos, utilizam-se os vetores de movimento de blocos adjacentes (para os quais já foram definidos vetores de movimento) como tentativa inicial de procura, já que o movimento de blocos vizinhos é bastante correlacionado.

Um ponto importante é a definição de um critério para efetuar as comparações. O mais empregado é o cálculo da Soma de Diferenças Absolutas (SAD, na sigla em inglês – *Sum of Absolute Differences*), que é bastante simples de calcular, conforme sua expressão:

$$SAD = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left| C_{ij} - R_{ij} \right| \tag{3.1}$$

onde C_{ij} e R_{ij} são as amostras do bloco sob análise e do bloco do quadro de referência - C_{00} e R_{00} são as amostras do canto superior esquerdo das imagens.

Outras opções compreendem o Erro Quadrático Médio (MSE – *Mean Square Error*) – já definido anteriormente na Equação 2.2 –, que fornece uma medida precisa da energia restante no bloco residual, e o Erro Absoluto Médio (MAE – *Mean Absolute Error*) – Equação 3.2.

$$MAE = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left| C_{ij} - R_{ij} \right| \tag{3.2}$$

O SAD fornece uma aproximação bastante razoável para a quantidade de energia restante e, por ser computacionalmente simples de calcular – necessita de menos operações que os outros critérios -, é o critério mais frequentemente utilizado.

O algoritmo de comparação corresponde a uma sequência de operações bem definida estabelecida previamente. Há, na literatura, uma variedade colossal de algoritmos disponíveis. O algoritmo de comparação de blocos ótimo é bem conhecido e é chamado normalmente de *Full Search* (FS). Este realiza uma busca exaustiva em cada possível bloco do quadro de referência numa região de procura previamente estabelecide para encontrar a posição onde o critério de comparação fornece um valor mínimo. Logo, este método sempre encontrará o mínimo global na imagem usada como referência, dentro de sua região de busca. É importante notar que nem todos os algoritmos de estimação de movimento restringem a área de busca, o que pode resultar em alguns outros métodos com resultados melhores que o FS pelo simples fato de não apresentarem esta restrição (da mesma forma, se o FS for utilizado com região de busca que cubra toda a imagem, ele será inevitavelmente ótimo).

O FS, entretanto, é computacionalmente proibitivo, pois demanda um processamento muito intenso – por buscar todos os blocos do quadro de referência dada uma região de procura. Muitos algoritmos alternativos têm sido propostos nos últimos trinta anos para solucionar este problema.

A maioria dos algoritmos de comparação de blocos simplesmente reduz o número de operações pela seleção de alguns poucos lugares onde serão realizadas comparações. A localidade guardada é a que apresenta mínimo SAD. Então, um vetor de movimento é associado ao bloco sob análise, apontando para a referida localidade.

Alguns destes algoritmos sempre usam o mesmo padrão de procura para cada bloco. Exemplos incluem o *Three-Step Search* (TSS) (7), o *Diamond Search* (DS) (8), o *Four-Step Search* (FSS) (9), o *New Three-Step Search* (NTSS) (10) e o *Cross-Diamond Search* (CDS) (11). Métodos mais complexos utilizam alguns critérios para selecionar os lugares de procura, como correlação espacial, temporal ou de multi-resolução. Alguns exemplos destes são o *Adaptive Rood Pattern Search* (ARPS) (12), o *Advanced Diamond Zonal Search* (ADZS) (13), o *Motion Vector Field Adaptive Search* (MVFAST) (14), o *Multi-resolution Spatio-temporal Correlations* (MRST) (15) e o *Predictive Motion Vector Field Adaptive Search* (PMVFAST) (16).

Outros algoritmos pré-processam os quadros para simplificar a procura baseada em comparação de blocos a ser realizada no passo seguinte. A *One-Bit Transform* (1BT) (17), a

Modified One-Bit Transform (M1BT) (18) e a *Two-Bit Transform* (2BT) (19) são alguns exemplos destes. Há também outros que reduzem o custo computacional pela remoção de posições não ótimas de procura sem perder a otimalidade do algoritmo FS. Estes compreendem o *Successive Elimination Algorithm* (SEA) (20) e o *Multilevel Successive Elimination Algorithm* (MSEA) (21).

É muito frequente que algoritmos simples fiquem presos em regiões de mínimos locais das imagens, já que vão sendo guiados pelas avaliações realizadas a cada passo, não havendo garantias de que as primeiras regiões analisadas levarão ao mínimo global da imagem. Desta forma, para um objeto que se deslocou bastante dentro do quadro, possivelmente o algoritmo que o procurará será levado a escolher outras direções que não a que leva à verdadeira posição do objeto, devido a resultados obtidos nos arredores do bloco em questão. O mínimo global é certamente alcançado pelo algoritmo FS, já que este analisa todas as possíveis posições do quadro de referência.

Uma observação interessante a ser feita sobre os algoritmos de comparação de blocos é que eles, na prática, não necessariamente encontram verdadeiros movimentos, mas apenas blocos que são bastante similares ao que está sob análise no quadro atual. Desta forma, algumas vezes os vetores de movimento indicam não a direção de um verdadeiro movimento, como dito, mas simplesmente um bloco bastante similar ao bloco para o qual se está procurando movimento.

Os quadros de referência

Podem existir vantagens na escolha de quadros de referência que não sejam os imediatamente anteriores ao quadro sob análise. Se houver um corte ou uma mudança brusca significativa numa cena, por exemplo, fará mais sentido utilizar quadros do "futuro" como referência para a detecção de movimento. Esta estratégia também será eficiente quando um objeto deixa de encobrir uma parte escondida da imagem – a parte escondida não existia anteriormente. Esta técnica exige, portanto, que os quadros do "futuro" tenham sido codificados anteriormente ao quadro corrente.

Uma opção atualmente muito utilizada nos *codecs* mais avançados é a predição bidirecional: possibilita-se o emprego de predição baseada em quadros anteriores, posteriores, ou uma combinação entre eles – em geral, a média entre os dois. O codificador escolhe o modo de

predição a se utilizar, dependendo de qual oferecer melhores resultados. Esta é a base dos quadros classificados como do tipo "B" no codificador H.264, ou "L2", no caso do Dirac.

Alguns *codecs*, como o próprio H.264, permitem a utilização de muitas referências para predição – neste padrão, podem chegar a até 32. A Figura 3.2 apresenta um esquema que ilustra as possibilidades descritas.

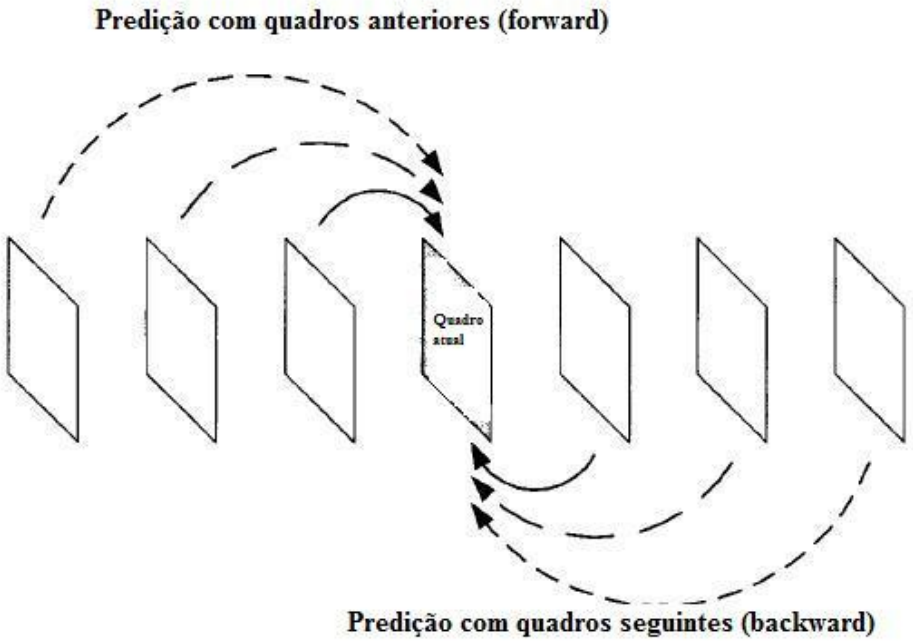


Figura 3.2: Opções de predição: *forward*, *backward*, ou uma combinação destes (1).

3.3 Técnicas utilizadas para aperfeiçoamento da predição

À parte os princípios básicos de estimação de movimento, discutidos anteriormente, existem algumas melhorias que já são empregadas há algum tempo em *codecs* avançados. Discutir-se-á, nesta seção, quatro delas: estimação de movimento de subpixel, compensação de movimento sobreposta, tamanho de bloco variável e extrapolação da imagem.

Estimação de movimento de subpixel

A estimação de movimento utilizando precisão não inteira de pixels é justificada pelo simples fato de que os objetos não necessariamente se moverão por um número inteiro de pixels. Na realidade, é muito provável que eles não o façam. Esta forma de estimação de movimento é realizada pela interpolação dos próprios pixels, de forma a chegar a uma resolução, inicialmente, de meio pixel, para, em seguida, um quarto de pixel, um oitavo de pixel, etc. O único fator restritivo à utilização de toda esta precisão é a complexidade computacional envolvida, que é significativamente maior que a que se tem quando se utiliza apenas estimação de movimento no nível de pixels. Em geral, obtêm-se melhores resultados ao se utilizar mais precisão. A Figura 3.3 ilustra a predição chegando a até um quarto de pixel, atingida por duas interpolações realizadas na região de procura.

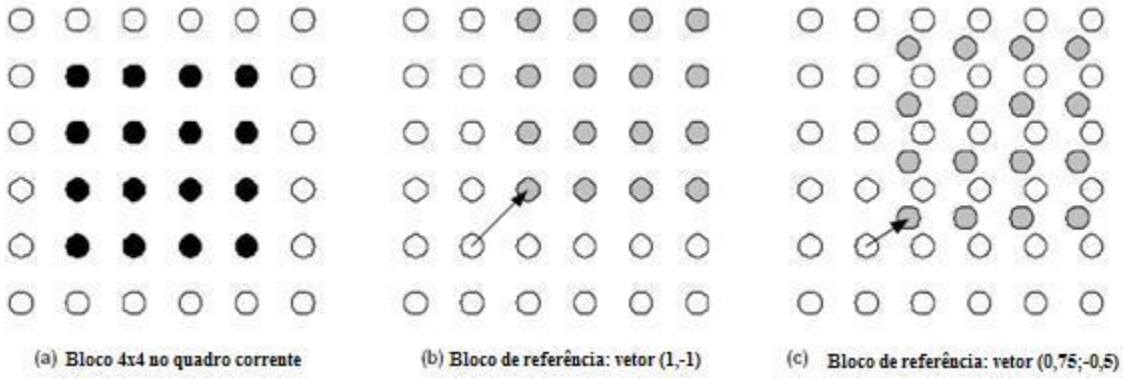


Figura 3.3: (a) Bloco para o qual se procura vetores de movimento. (b) Bloco similar encontrado por busca em número inteiro de pixels. (c) Solução ótima encontrada ao nível de 1/4 de pixel. (41)

É bastante comum que se realize primeiramente a etapa de estimação de movimento no nível de inteiros de pixel para, depois, na região encontrada, processar-se a interpolação e a procura por blocos que minimizem ainda mais a energia do quadro residual. Embora a região desta forma encontrada possa não ser de fato o mínimo global da imagem ao nível de subpixel analisado, este procedimento já fornece uma boa otimização complementar à busca em nível de inteiros de pixel.

Este procedimento de interpolação é mostrado na Figura 3.4. No caso da codificação H.264, por exemplo, um filtro de resposta impulsiva finita (FIR – *Finite Impulse Response*) pré-definido é utilizado. Tomando como exemplo os pixels da referida figura, o cálculo da amostra de meio-pixel identificada pela letra “b” seria realizado a partir da seguinte expressão:

$$b = round\left(\frac{E - 5F + 20G + 20H - 5I + J}{32}\right) \tag{3.3}$$

Como se nota, maior importância é dada a pixels que estão mais próximos ao que se pretende avaliar, pela atribuição de um peso significativamente superior ao dos demais pixels da linha. Regras similares são aplicadas ao cálculo das outras amostras não inteiras de pixel.

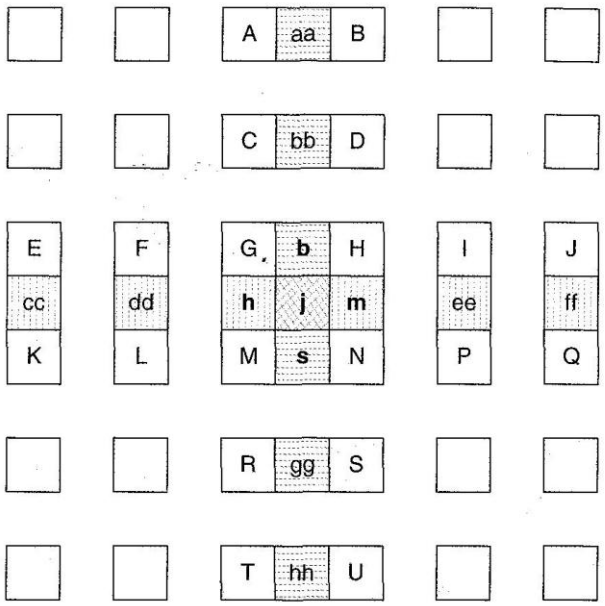


Figura 3.4: Interpolação de pixels. "b" é uma amostra de metade de pixel obtida a partir de filtragem sobre E, F, G, H, I e J. "s", "h", "m", "aa", "bb", "cc", "dd", "ee", "ff", "gg" e "hh" são obtidos de forma similar. "j" é obtido a partir da aplicação de filtragem sobre amostras de metade de pixel – "cc", "dd", "h", "m", "ee" e "ff" (41).

Compensação de movimento sobreposta

A técnica de compensação de movimento sobreposta (OBMC – *Overlapped Block Motion Compensation*) consiste na utilização de blocos superpostos quando da realização da estimação de movimento. Assim, atribuem-se pesos às diferentes predições realizadas, sendo que o valor do pixel que faz parte de diversos blocos ao mesmo tempo é dado pelo cálculo que leva estes pesos em conta – muitas vezes, uma simples média aritmética é realizada.

A OBMC permite uma suavização dos valores dos pixels nas bordas dos blocos, permitindo uma melhoria na precisão da predição. A Figura 3.5 ilustra um exemplo de utilização desta

técnica: a área sombreada faz parte de mais de um bloco; o pixel p participa de 4 blocos simultaneamente.

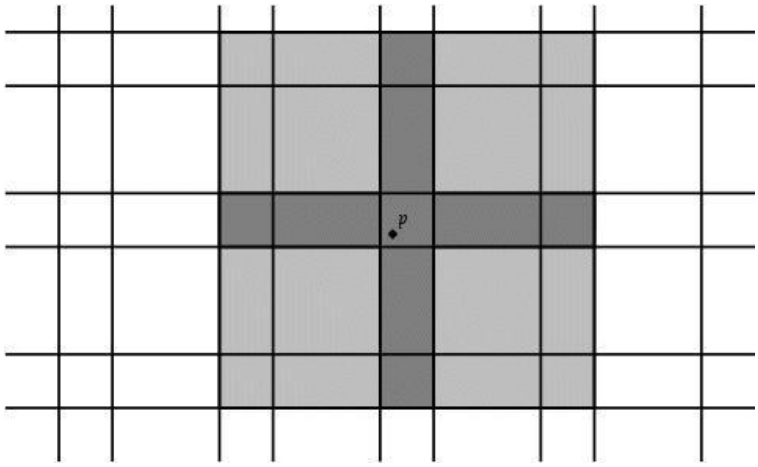


Figura 3.5: OBMC mostrando blocos superpostos (área sombreada) . O pixel p faz parte de quatro blocos (42).

Tamanho de bloco variável

Presente em quase todos os *codecs* utilizados na atualidade, a segmentação em blocos de diferentes tamanhos é bastante vantajosa quando o movimento se torna mais complexo. Em geral, uma partição grande é apropriada para áreas homogêneas, enquanto partições menores são mais apropriadas para áreas com bastantes detalhes.

A codificação H.264, por exemplo, utiliza particionamento de blocos estruturado em árvore. Os vetores de movimento podem ser encontrados para cada um deles ou para um grupo deles – compensação de movimento estruturada em árvore.

Para ilustrar o uso de compensação de movimento estruturada em árvore, vê-se na Figura 3.6 um quadro residual. Os tipos de partição usados são mostrados sobre a imagem. Em áreas onde há pouca diferença entre os *frames*, são usados blocos maiores, 16 x 16. Já em áreas onde há movimento mais detalhado, partições menores são mais eficientes.

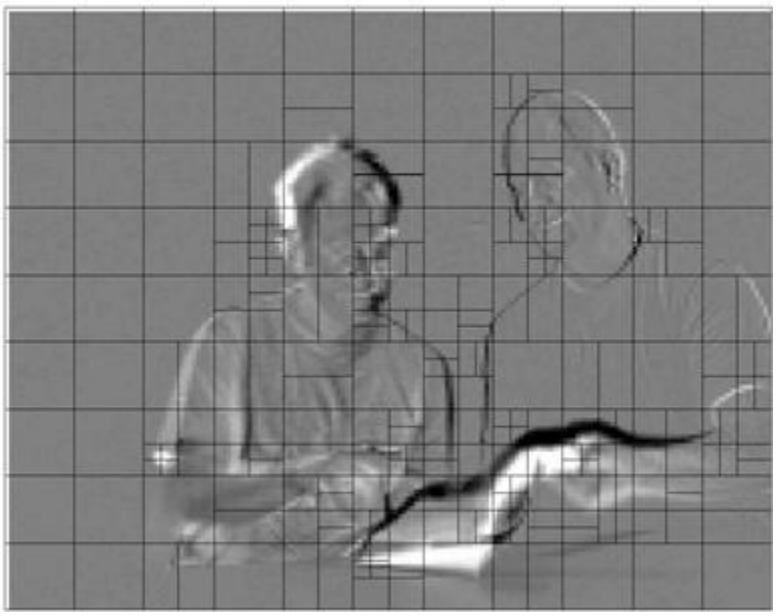


Figura 3.6: Quadro residual mostrando escolha ótima de partições (41).

Extrapolação da imagem

Em muitos casos, quando há movimentação próxima às bordas da imagem, torna-se interessante extrapolá-la, pois, nesta situação, o melhor bloco a utilizar pode ter parte no exterior do quadro.

Um exemplo prático apresenta-se na Figura 3.7: a bola que aparece no quadro corrente é apenas parcialmente visível no quadro de referência. Utilizando-se uma simples extrapolação linear da imagem, como na referida figura, encontra-se uma predição melhor para o bloco em questão.

3.4 Estudo comparativo de algoritmos de estado da arte

Um dos objetivos deste trabalho é o estudo comparativo de técnicas de estado da arte de estimação de movimento – tema desta seção.

Três algoritmos de estado da arte de estimação de movimento são estudados: *Adaptive Modified Two-Bit Transform* (AM2BT) (29), *Adaptive Multilevel Successive Elimination Algorithm* (AdaMSEA) (30) e *Enhanced Adaptive Rood Pattern Search* (EARPS) (28). Cada um destes analisa a estimação de movimento com uma abordagem diferente, representando a evolução de três distintas formas de estimar movimento.

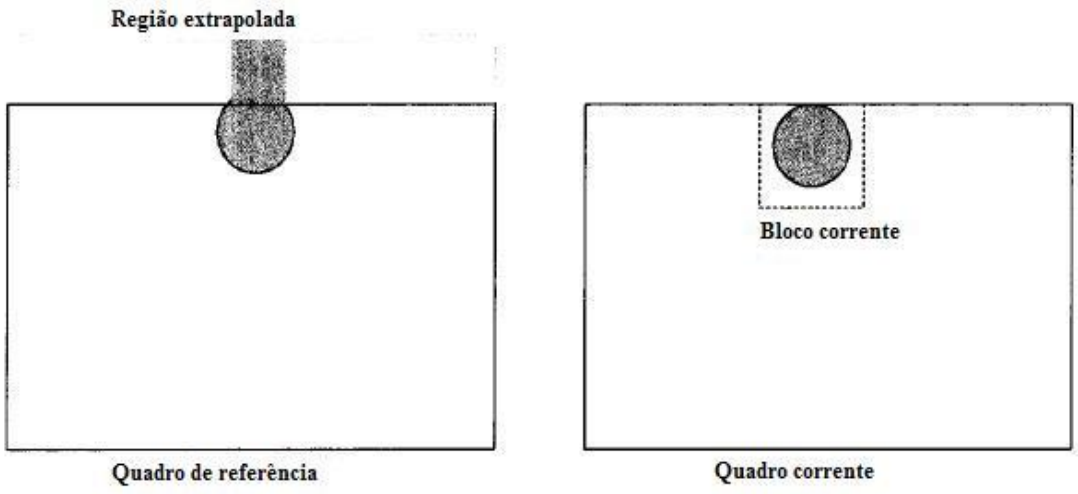


Figura 3.7: Caso em que a extrapolação da figura faz com que se encontre uma melhor predição (1).

A motivação para esta comparação vem do fato de inexistir na literatura uma comparação de métodos tão diferentes. Quando comparados com algoritmos mais simples, cada um destes apresenta resultados superiores (28) (29) (30).

Inicialmente, descrever-se-á em mais detalhes os algoritmos estudados. Na sequência, serão apresentados os procedimentos empregados e os resultados.

Adaptive Modified Two-Bit Transform

A *Adaptive Modified Two-Bit Transform* (AM2BT) (29) é baseada na *Two-Bit Transform* (19). Representações binárias dos *frames* são obtidas utilizando a média μ e o desvio-padrão σ de cada bloco:

$$B_1(i,j) = \begin{cases} 1, & se\ I(i,j) \geq \mu \\ 0, & caso\ contrário \end{cases} \tag{3.4}$$

$$B_2(i,j) = \begin{cases} 1, & se\ I(i,j) \geq \mu + \sigma\ ou\ I(i,j) \leq \mu - \sigma \\ 0, & caso\ contrário \end{cases} \tag{3.5}$$

onde $B_1(i,j)$ e $B_2(i,j)$ representam os dois planos de bits resultantes, usados para estimar movimento e $I(i,j)$ o brilho do pixel localizado na linha *i*, coluna *j*.

Inicialmente, um critério de similaridade baseado em operadores Booleanos (o que permite rápida execução e implementação simples) é usado para a procura em todos os locais do frame anterior. O critério é chamado *Number of Non-Matching Points* (NNMP), sendo definido como segue:

$$NNMP(m,n) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\{B_1^t(i,j) \oplus B_1^{t-1}(i+m,j+n)\}||\{B_2^t(i,j) \oplus B_2^{t-1}(i+m,j+n)\} \quad , \quad -s \le m,n \le s-1 \tag{3.6}$$

onde *m* e *n* são os deslocamentos (vetores de movimento), *N* é o tamanho do bloco, B_i^t as representações binárias do quadro corrente, B_i^{t-1} as do quadro anterior, *s* o espaço de procura, $\oplus$ o operador booleano OU exclusivo e $\|$ a operação OU.

Com o uso do NNMP, são encontrados o primeiro e o segundo melhores vetores de movimento (*mv1* e *mv2*, respectivamente). Em seguida, a distorção entre o bloco original e o predito por *mv1* é calculada e comparada com um limiar T (baseado em σ, calculado anteriormente). Se esta for menor que T, o vetor de movimento final escolhido é *mv1*. Caso contrário, o mesmo procedimento é realizado com *mv2*.

Se a distorção ainda for maior que T para *mv2*, uma procura ao redor das localidades apontadas por *mv1* e *mv2* é realizada utilizando-se uma procura de dois passos de múltiplos candidatos (*Multiple-candidate Two-Step Search*). A Figura 3.8 ilustra esta busca: inicialmente, pesquisam-se quatro pontos ao redor das localidades apontadas por *mv1* e *mv2*. O ponto com menor medida de distorção nesta etapa tem seus oito vizinhos pesquisados, determinando o ponto final com a menor medida de distorção da procura em dois passos.

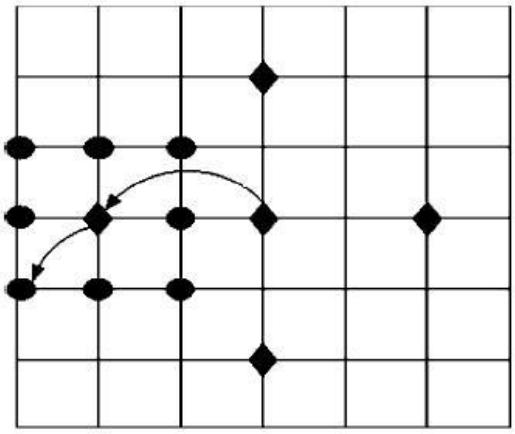


Figura 3.8: ***Multiple-candidate Two-Step Search*** **utilizada no AM2BT (29).**

Se a distorção encontrada para o melhor vetor de movimento desta etapa estiver abaixo de 2T, este é escolhido como vetor de movimento para o bloco corrente. Caso contrário, um terceiro estágio é computado utilizando-se FS e o melhor vetor de movimento encontrado é escolhido. O esquemático deste algoritmo é apresentado na Figura 3.9.

Resumindo, o algoritmo do AM2BT é o seguinte:

1. Construção de duas representações binárias dos quadros;
2. Cômputo de NNMP para a procura de blocos similares em quadros anteriores;
3. Primeiro e segundo melhores vetores de movimento encontrados: *mv1* e *mv2*;
4. Distorção de *mv1* < T? Se sim, *mv1* é o vetor escolhido;
5. Caso contrário: Distorção de *mv2* < T? Se sim, *mv2* é o vetor escolhido;
6. Uso de procura de dois passos de múltiplos candidatos no entorno de mv1 e no entorno de mv2;
7. Melhor vetor encontrado tem distorção < 2T? Se sim, ele é o vetor escolhido;
8. Uso de FS para achar melhor vetor;
9. Melhor vetor encontrado é escolhido.

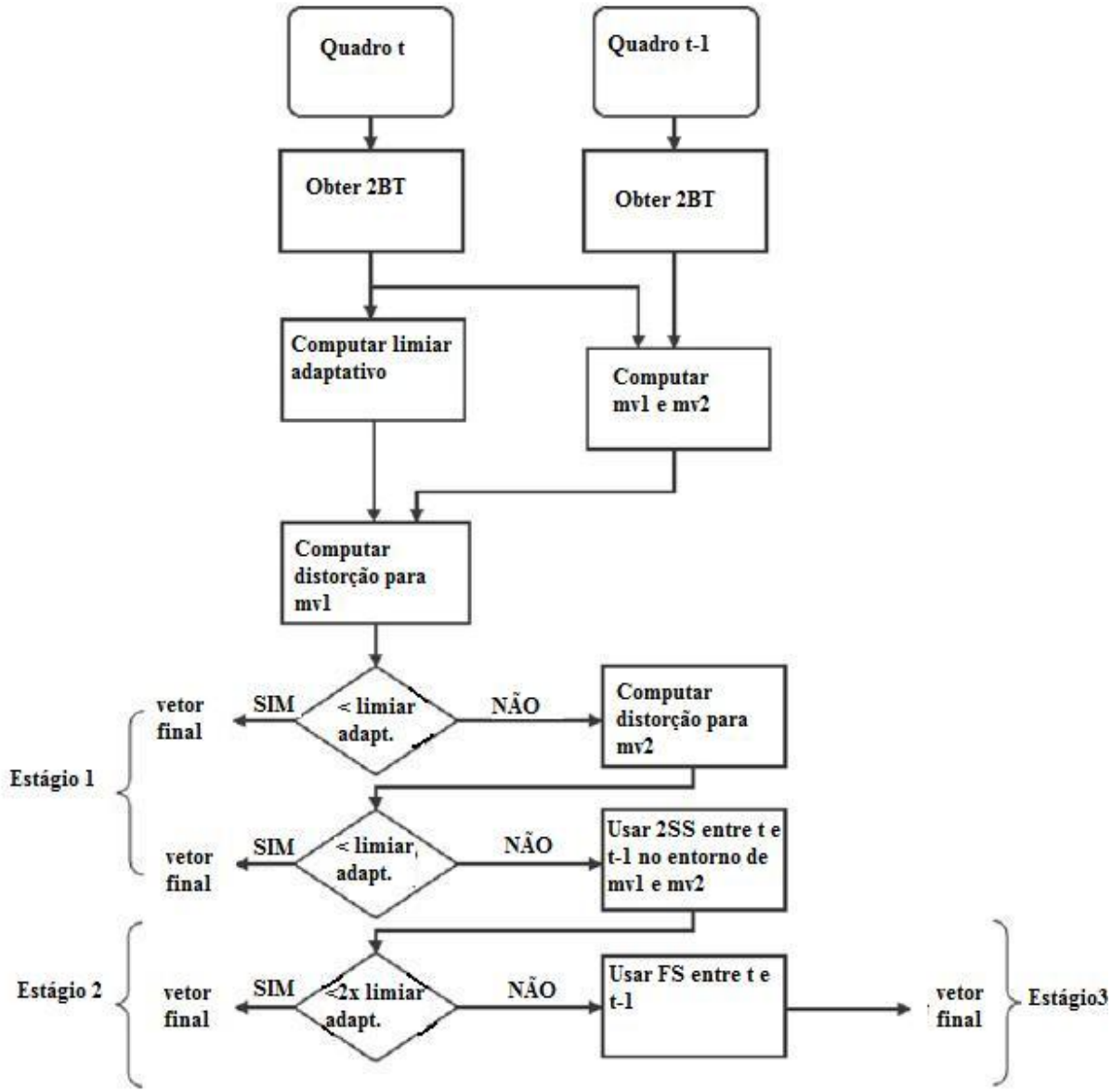


Figura 3.9: Esquema do AM2BT (29).

Adaptive Multilevel Successive Elimination Algorithm

O *Adaptive Multilevel Successive Elimination Algorithm* (AdaMSEA) (30) tem sua base no *Multilevel Successive Elimination Algorithm* (21), que reduz a carga computacional do FS pela remoção de muitos lugares de procura não ótimos.

Esta remoção se dá a partir da utilização da seguinte propriedade:

$$\left|C - R^{(u,v)}\right| \leq SAD(u,v) \tag{3.7}$$

onde C e $R^{(u,v)}$ representam as somas dos pixels do bloco corrente e do bloco de referência, respectivamente. $SAD(u,v)$ representa o valor do SAD para o vetor testado neste momento, (u,v).

Logo, é impossível que um bloco para o qual o termo do lado esquerdo (da expressão 3.7) seja maior que o valor mínimo corrente de SAD encontrado tenha um valor de SAD menor. Este bloco, portanto, não será comparado com o bloco de referência (seu SAD não será calculado). Caso contrário, se o bloco satisfizer esta propriedade, seu SAD é comparado com SAD_{min} atual e, caso seja menor, SAD_{min} é atualizado para este novo SAD. Este é o princípio do *Successive Elimination Algorithm* (20), em que algoritmos rápidos permitem o cômputo simples da expressão do lado esquerdo da Equação 3.7, o que impede um *overhead* significativo de operações realizadas para o processo de estimação de movimento.

Entretanto, muitos dos blocos para os quais a Equação 3.7 é satisfeita apresenta SAD maior que o SAD_{min} atual. Logo, uma das melhorias a serem aportadas é a da contrução de níveis entre esta primeira comparação e o cálculo do SAD. Para tal, os macroblocos são particionados – a cada partição, um nível superior é definido. A soma das diferenças referentes a cada partição é, então, calculada, seu valor sendo mais próximo do SAD, o que refina mais ainda a eliminação de blocos que não serão úteis:

$$\sum_{i=0}^{S} \left| C_l(i) - R_l{}^{(u,v)}(i) \right| \leq SAD_{min}(u,v) \tag{3.8}$$

onde i = 0,..,S representa o índice da partição (ou seja, somam-se as diferenças referentes a todas as partições). l representa o nível para o qual as somas de pixels das partições são calculadas. A expressão 3.7 representa, portanto, o cálculo efetuado no nível zero, quando o macrobloco ainda não apresenta nenhuma partição, S=0.

Desta forma, sabendo-se que estes cálculos se aproximam cada vez mais do valor SAD_{min}, define-se uma ordem de particionamento (eliminação), na qual os blocos serão eliminados caso a soma das diferenças seja maior que o valor SAD_{min} corrente.

Utilizando macroblocos de tamanho 16x16, AdaMSEA realiza o particionamento a partir do valor do gradiente horizontal (calculado de forma aproximada pela Equação 3.9), para todos os macroblocos de cada quadro. Cada macrobloco é dividido em quatro submacroblocos. A cada etapa, seleciona-se a partição com maior magnitude de gradiente: se esta é maior que um limiar T, este é dividido em outras quatro partes, e assim sucessivamente, até que não restem partições

com gradiente maior que T. Cada nova partição corresponde a um nível mais alto, e este procedimento determina a ordem de eliminação. A Figura 3.10 apresenta um exemplo da ordem de eliminação determinada pela soma das magnitudes de gradiente.

$$G_x(x,y) = |f(x+1,y) - f(x,y)| \tag{3.9}$$

onde $G_x(x,y)$ representa o gradiente horizontal do pixel de posição (x,y), e $f(x,y)$ o brilho do pixel nesta mesma posição.

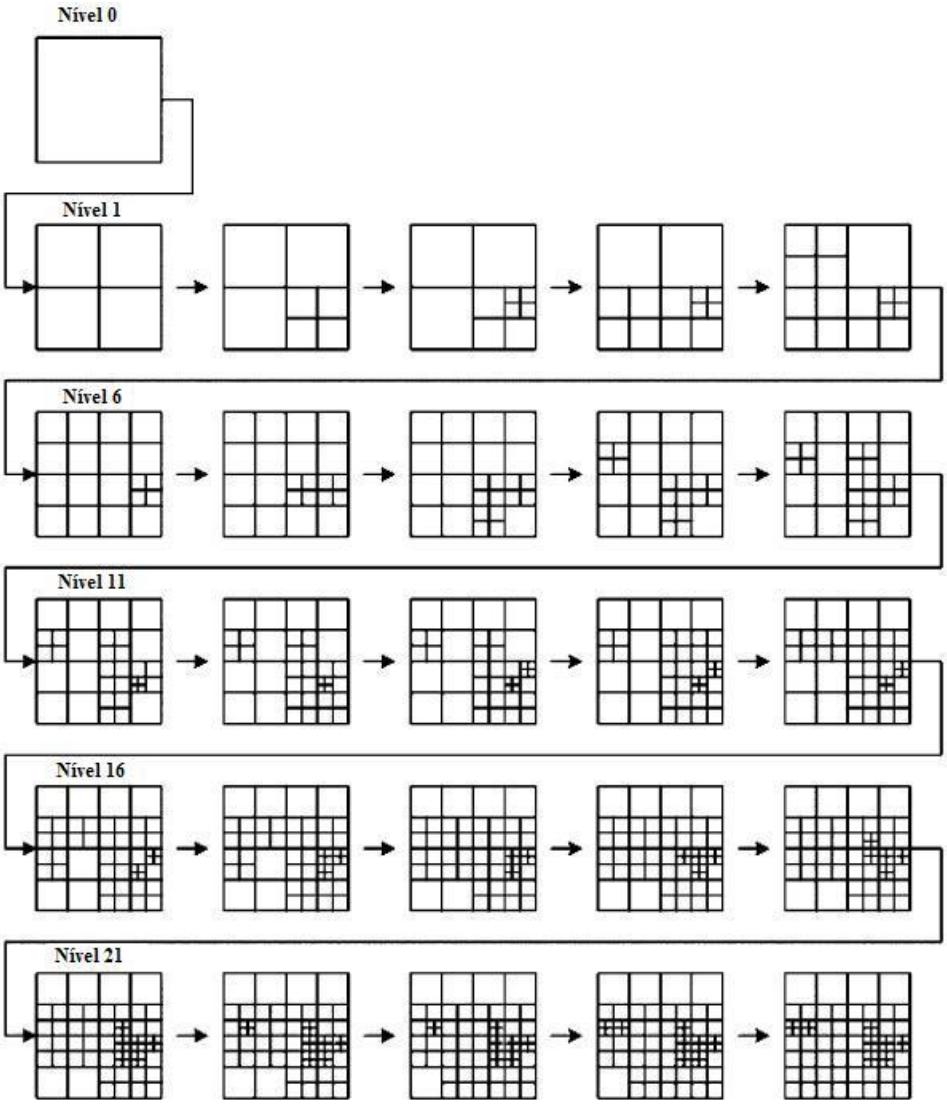


Figura 3.10:Exemplo da ordem de eliminação (30).

Em seguida, os cálculos começam no nível zero, com a soma das normas do candidato, para cada macrobloco. Se este resultado for maior que a menor SAD achada até o momento, o candidato é rejeitado e passa-se ao próximo bloco. Caso contrário, a soma das normas do primeiro nível é calculada e comparada com a SAD. Se este resultado for maior, o candidato é rejeitado. Este procedimento segue até que a soma de normas do nível mais alto seja calculada. Se esta for maior que a SAD, o candidato é rejeitado. Caso contrário, a SADc (SAD do candidato) é calculada. Finalmente, se a SAD é maior que a SADc, o bloco mais similar achado até o momento será este candidato, SAD = SADc, e a busca continuará com o próximo candidato.

Resumindo, duas rotinas principais são utilizadas no AdaMSEA:

Algoritmo para determinação de ordem de eliminação

Para cada macrobloco:

Colocar o macrobloco na fila

Repetir

1. Selecionar o bloco com maior soma de magnitudes de gradientes na fila
2. Dividir o bloco selecionado em quatro sub-blocos e calcular suas somas de magnitude de gradientes
3. Checar cada sub-bloco e colocá-lo na fila se sua soma de magnitude de gradientes for maior que o limiar T

Até a fila estar vazia

Fim (Para cada macrobloco)

Algoritmo para determinação de vetores de movimento

Para cada candidato:

Se SAD_{min} ainda for inexistente (primeiro candidato a se analisar), calcula-se o SAD do bloco atual e o define como SAD_{min}

Repetir, incrementando *l* de 0 até o último nível

Rejeita o candidato se a soma das diferenças para o nível *l* for maior que o SAD_{min}

Até o último nível

Calcula-se o SAD do candidato

Se este for menor que SAD_{min}, atualizar SAD_{min}

Fim (Para cada candidato)

Enhanced Adaptive Rood Pattern Search

O *Enhanced Adaptive Rood Pattern Search* (EARPS) (28) é um método recente rápido e eficiente baseado no *Adaptive Rood Pattern Search* (ARPS) (12). Este algoritmo usa três melhorias com relação ao ARPS: *Initial Search Centre Prediction* (ISC), *Adaptive Search Pattern* (ASP) e *Early Termination*. Os padrões de procura são apresentados na Figura 3.11.

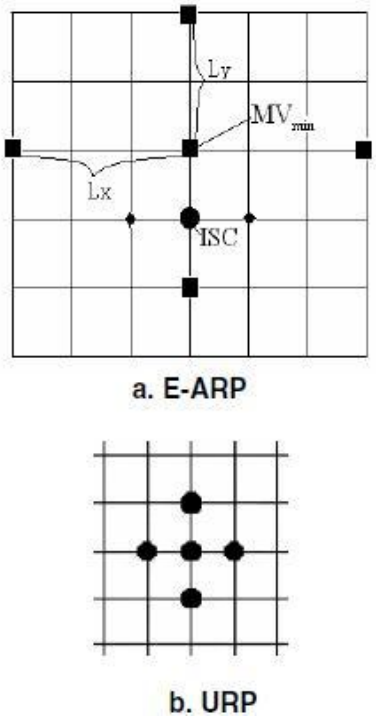


Figura 3.11: Padrões de busca utilizados no EARPS (28).

No primeiro passo, a SAD é calculada no local indicado por um estimador baseado na mediana dos vetores de movimento de três blocos adjacentes (SADp) e, se a condição de terminação for atendida (SAD calculado menor que um certo limiar T), a procura acaba (o vetor de movimento encontrado é associado a este bloco e passa-se ao próximo bloco). Caso contrário, a SAD é calculada no ponto (0,0) (SAD_0) e se esta for menor que T a procura é encerrada. Se esta for maior que T, o ISC é escolhido como a posição do menor valor entre SAD_0 e SADp (ponto de distorção mínima, *Minimum Distortion Block*, MDB).

Em seguida, o centro de um padrão de procura unitário (*Unit-size Rood Pattern*, URP), mostrado na Figura 3.11 (b), é colocado no ISC. A SAD é calculada para os locais indicados por este padrão e se o ponto de MDB encontrado neste passo for menor que T ou estiver localizado no ISC (ou seja, o ponto de MDB continua no centro da região procurada), a procura é terminada. Caso contrário, um padrão adaptado ao bloco (ASP), conforme as Equações 3.10 e 3.11, é utilizado com centro no ponto de MDB encontrado no passo anterior.

$$L_x = max(|MV_{L.x} - MV_{min.x}|, |MV_{T.x} - MV_{min.x}|, |MV_{TR.x} - MV_{min.x}|) \quad (3.10)$$

$$L_y = max(|MV_{L.y} - MV_{min.y}|, |MV_{T.y} - MV_{min.y}|, |MV_{TR.y} - MV_{min.y}|) \quad (3.11)$$

onde L_x e L_y são os comprimentos dos braços de procura do padrão (conforme Figura 3.11), MV_L, MV_T e MV_{TR} são os vetores de movimento dos blocos adjacentes à esquerda, acima e acima à direita do bloco corrente e MV_{min} é o vetor de movimento encontrado na procura realizada inicialmente com o URP.

A Figura 3.11 (a) ilustra ainda um caso onde a busca realizada pelo URP indica que o melhor ponto encontrado nesta etapa é o de cima, centro da próxima busca (realizada com padrão adaptativo com $L_x = 3$ e $L_y = 2$ neste caso).

Duas são as razões existentes para a forma de cálculo de L_x e L_y: após a realização de uma busca refinada no estágio anterior (URP), o que interessa neste momento é buscar um pouco mais longe e não continuar o refinamento de uma busca até então satisfatória – isto justifica o fato de buscar o mais distante vetor de movimento dentre os blocos adjacentes (uso do operador max). A segunda razão é a necessidade de subtrair do vetor indicado pelos blocos adjacentes o vetor que indica a localização onde a busca se encontra presentemente (caso contrário, estaria-se buscando em localidades diferentes das realmente indicadas pelos vetores), o que explica as diferenças entre vetores de movimento nas duas expressões.

Se a condição de terminação for satisfeita para algum dos pontos examinados, a procura é encerrada. Se esta não for atendida, o URP é colocado no ponto de MDB do passo anterior e é usado repetidamente até que a condição de terminação seja satisfeita ou que o ponto de MDB esteja no centro do URP. O diagrama de blocos do algoritmo é mostrado na Figura 3.12.

Resumindo, o algoritmo do EARPS é:

1. Para cada bloco, em *raster* order:
    1.1. Cálculo de SADp (local indicado por estimador). Se SADp < T, este vetor é atribuído ao bloco corrente.
    1.2. Cálculo de SAD0 (local (0,0)). Se SAD0 < T, este vetor é atribuído ao bloco corrente.
    1.3. Compara-se SAD0 com SADp, e a posição que fornecer o menor SAD é setada como o ISC, de onde a busca seguirá.
    1.4. Realizar busca utilizando URP, no ISC. Caso o ponto de menor distorção encontrado nesta etapa (MBD) tenha SAD menor que T ou se localize no ISC, este será atribuído ao bloco corrente.
    1.5. A busca continua a partir do MBD encontrado no passo anterior. Realizar busca utilizando padrão adaptativo. Caso o MBD encontrado neste passo tenha SAD menor que T ou se localize no centro do padrão adaptativo, este será atribuído ao bloco corrente.
    1.6. A busca continua a partir do MBD encontrado no passo anterior. Realizar busca utilizando URP repetidamente até a condição de término ser atingida (SAD menor que T ou MBD localizado no centro do URP).

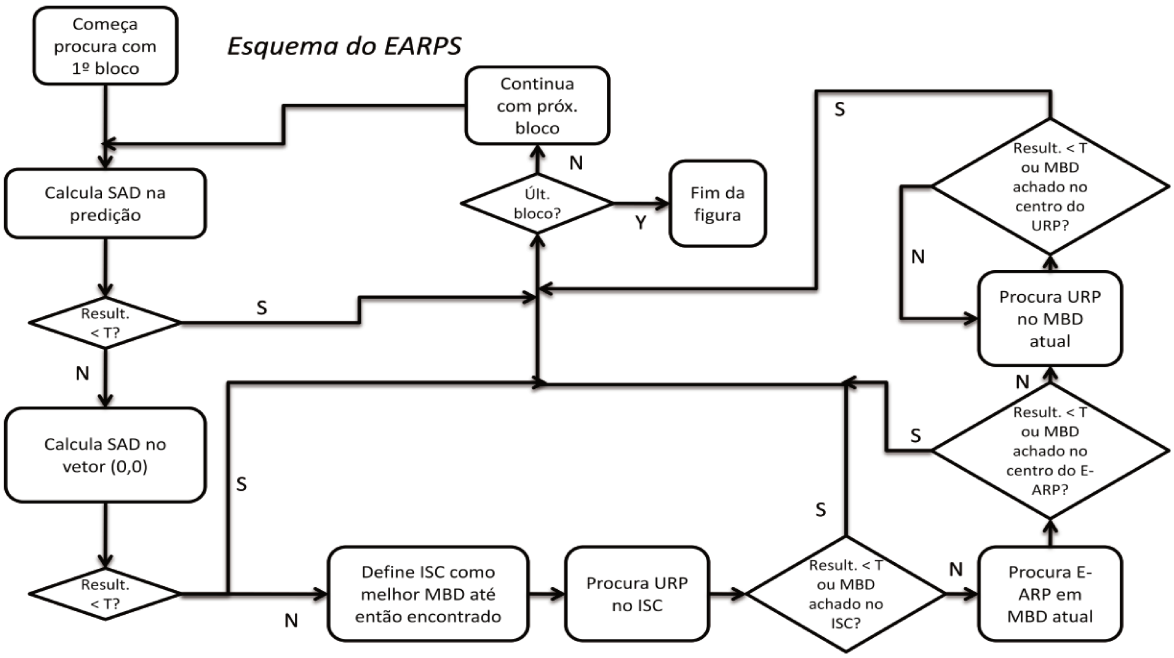


Figura 3.12: Diagrama de blocos do EARPS.

Testes, resultados e discussão

Os três algoritmos estudados foram implementados em ambiente MATLAB e simulados para quatro seqüências de imagens, com diferentes graus e tipos de movimento – "*football*" (125 frames de tamanho 352 x 240), "*tennis*" (112 frames de tamanho 352 x 240), "*foreman*" (299 frames de tamanho 352 x 288) e "*mobile*" (140 frames de tamanho 352 x 240). O método FS (ótimo) também foi implementado para realizar comparações. Os tamanhos de bloco usados são 16 x 16 e 8 x 8, com área de procura (*search range*) de 7 pixels na direção horizontal e vertical. A medida de distorção utilizada para todos os algoritmos implementados é a SAD.

Para obter uma comparação em termos de qualidade de reconstrução, o PSNR e o MSSIM foram computados para cada frame. O custo computacional foi medido pelo número de pontos de procura ("*search points*", correspondendo ao número de operações de cálculo de SAD realizadas – uma medida de comparação comumente empregada) por vetor de movimento gerado, isto é, por macrobloco.

Um exemplo de reconstrução de quadro é apresentado na Figura 3.13. O quadro orginal é apresentado, podendo ser comparado visualmente aos reconstruídos. Constata-se claramente distorção no canto superior direito das imagens, associando-a aos valores correspondentes de PSNR e MSSIM.

Os valores médios obtidos para o PSNR e MSSIM e o custo computacional para cada seqüência são apresentados na Tabela 3.1 e na Tabela 3.2, e plotados também nas Figura 3.14, Figura 3.15, Figura 3.16, Figura 3.17, Figura 3.18 e Figura 3.19

Os resultados obtidos para o AdaMSEA são idênticos aos do Full Search, como esperado, já que o primeiro apenas reduz o custo computacional do último sem, no entanto, desprezar nenhum lugar de procura. Apesar de apresentar excelentes resultados em termos de PSNR e MSSIM, o AdaMSEA é muito computacionalmente custoso, quando comparado com o EARPS ou com o AM2BT, as duas técnicas estudadas que alcançam maior redução de complexidade computacional.

Todos os algoritmos estudados provêem resultados bons. O EARPS provê excelentes resultados em termos de complexidade computacional para todas as sequências de imagens consideradas. Entretanto, para a sequência "football" com tamanho de bloco16x16, ele apresenta o pior resultado em termos de qualidade de reconstrução, porém ainda dentro de uma margem bastante razoável. Para vários quadros da sequência "foreman", notou-se que os resultados

obtidos para o EARPS foram melhores mesmo que os dados pelo FS (o que acarretou também no fato de a qualidade média de reconstrução ser também maior para o caso de blocos de tamanho 16x16). Isto se explica pelo fato de o EARPS não impor uma região de procura, enquanto todos os outros métodos analisados o fazem (como já notado anteriormente, o FS é ótimo apenas dentro da sua região de busca pré-determinada). Como o movimento ocorre muito rapidamente para esta sequência, os vetores de movimento necessitam ter valores bastante grandes, o que não pode ser alcançado pelos outros algoritmos, dada sua restrição de busca. A sequência "mobile", por outro lado, apresenta resultados similares em termos de qualidade de reconstrução para todos os algortimos empregados. A complexidade computacional é menor para esta sequência.

Constata-se a partir de todos os resultados apresentados que o algoritmo EARPS é o que provê melhor compromisso entre qualidade de reconstrução e complexidade computacional.Este é indiscutivelmente o que apresenta menor complexidade computacional, com qualidade de reconstrução bastante próxima à do algoritmo FS.

Por fim, é importante notar que outros parâmetros influenciam a complexidade computacional, que devem ser levados em conta quando se implementa um destes métodos. Neste trabalho, a influência do número de pontos de procura é analisada. Operações de pré-processamento não foram levadas em conta, e deve ser dito que o AM2BT e o AdaMSEA possuem uma quantidade razoável destas. Assim, o EARPS possui indubitavelmente o menor custo computacional dentre os algoritmos analisados. Entretanto, se uma implementação em *hardware* for desejada, é extremamente recomendável que a arquitetura deste seja bem compreendida, visto que alguns algoritmos podem ser mais facilmente implementados em certas confirgurações de *hardware*.

Tabela 3.1: Valores médios de PSNR, MSSIM e número de pontos de procura obtidos, com tamanho de macrobloco 16x16.

	Tennis			Football			Foreman			Mobile		
	PSNR	MSSIM	Nº pontos	PSNR	MSSIM	Nº pontos	PSNR	MSSIM	Nº pontos	PSNR	MSSIM	Nº pontos
FS	29,410	0,8819	202,049	22,466	0,7695	202,049	31,202	0,8969	204,283	22,976	0,8865	202,049
EARPS	28,677	0,8735	5,620	21,842	0,7586	8,494	31,468	0,9156	5,885	22,942	0,8863	5,612
AdaMSEA	29,410	0,8819	50,824	22,466	0,7695	35,898	31,202	0,8969	38,517	22,976	0,8865	17,375
AM2BT	29,038	0,8735	5,264	22,315	0,7617	20,587	29,740	0,8720	5,472	22,823	0,8813	7,468

Tabela 3.2: Valores médios de PSNR, MSSIM e número de pontos de procura obtidos, com tamanho de macrobloco 8x8.

	Tennis			Football			Foreman			Mobile		
	PSNR	MSSIM	Nº pontos	PSNR	MSSIM	Nº pontos	PSNR	MSSIM	Nº pontos	PSNR	MSSIM	Nº pontos
FS	31,166	0,9083	213,376	24,643	0,8311	213,376	32,701	0,9165	214,518	23,873	0,9033	213,376
EARPS	29,967	0,8680	2,430	23,584	0,8002	5,343	31,691	0,9021	2,209	23,625	0,8974	3,389
AdaMSEA	31,166	0,9083	59,167	24,643	0,8311	44,341	32,701	0,9165	42,162	23,873	0,9033	30,247
AM2BT	29,532	0,8170	8,781	23,752	0,7801	32,281	29,921	0,8605	7,552	23,441	0,8841	11,052

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figura 3.13: Quadro #150 da sequência "foreman". (a) Original. (b) Reconstruído usando FS com macrobloco de tamanho 16x16. PSNR = 31,780, MSSIM = 0,9282. (c) AM2BT, 16x16. PSNR = 28,612, MSSIM = 0,8774. (d) AdaMSEA, 16x16. PSNR = 31,780, MSSIM = 0,9282. (e) E-ARPS, 16x16. PSNR = 30,787, MSSIM = 0,9156. (f) FS, 8x8. PSNR = 33,768, MSSIM = 0,9473. (g) AM2BT, 8x8. PSNR = 29,461, MSSIM = 0,8521. (h) AdaMSEA, 8x8. PSNR = 33,768, MSSIM = 0,9473. (i) E-ARPS, 8x8. PSNR = 31,089, MSSIM = 0,8859.

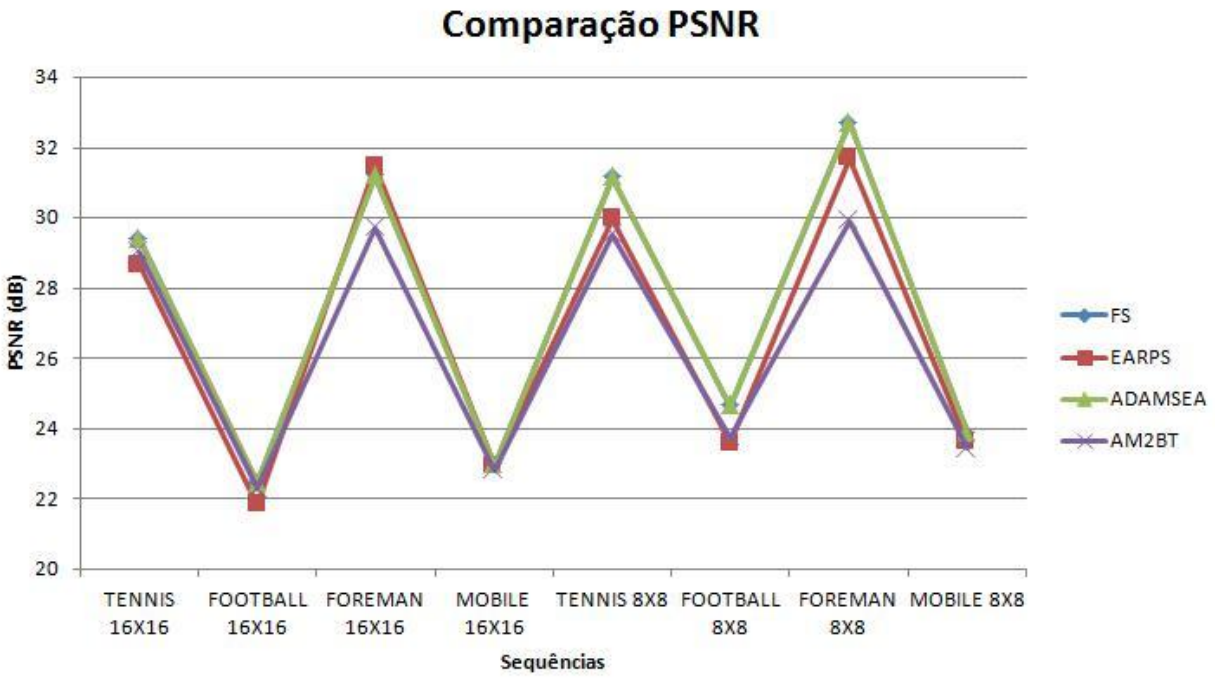


Figura 3.14: Comparação PSNR para as diversas sequências com distintos tamanhos de bloco.

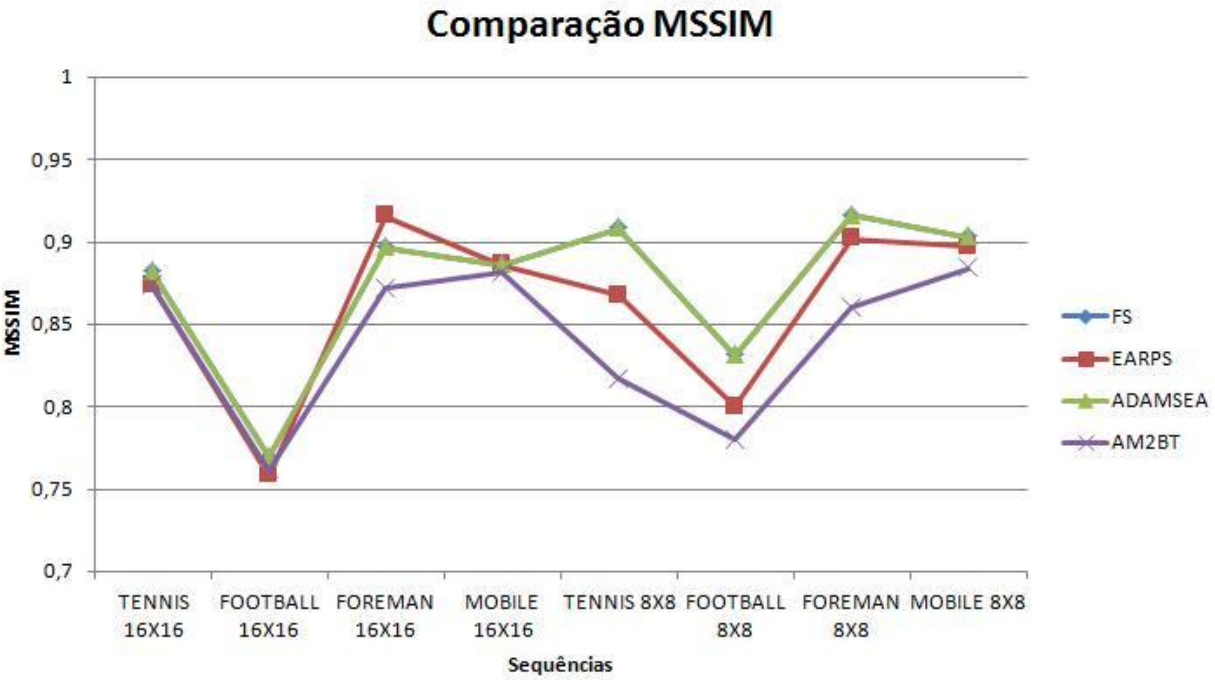


Figura 3.15: Comparação MSSIM para as diversas sequências com distintos tamanhos de bloco.

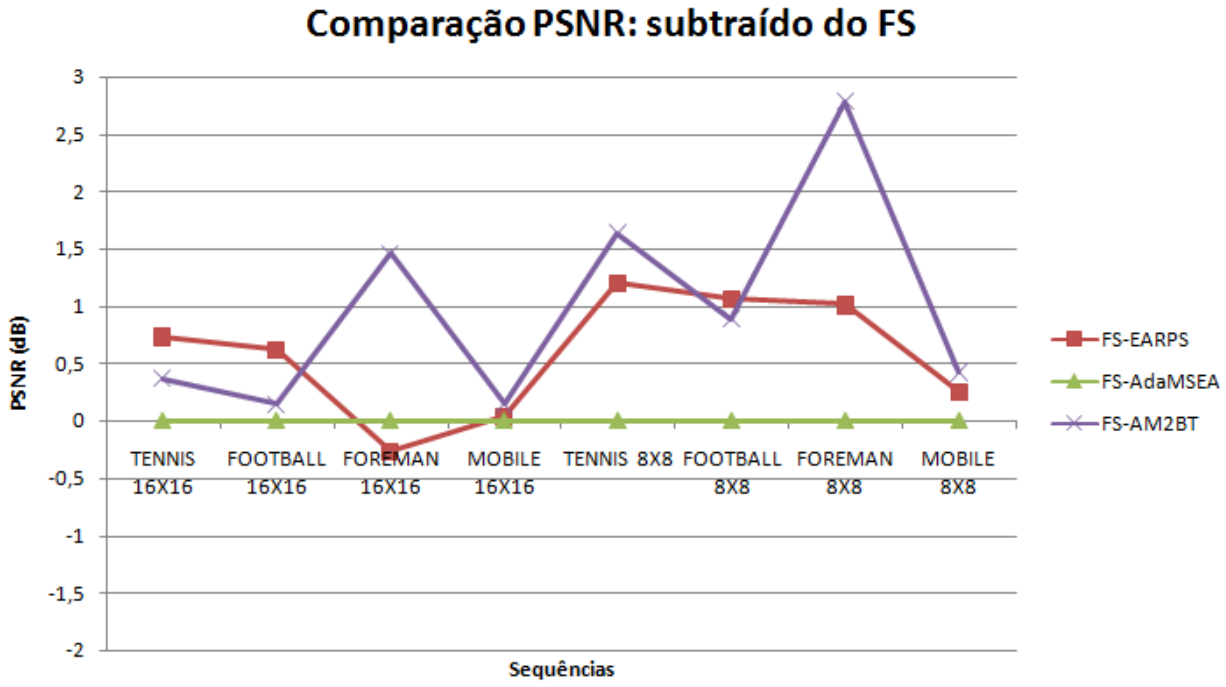


Figura 3.16: Comparação PSNR para as diversas sequências com distintos tamanhos de bloco, com referência ao desempenho do algoritmo FS.

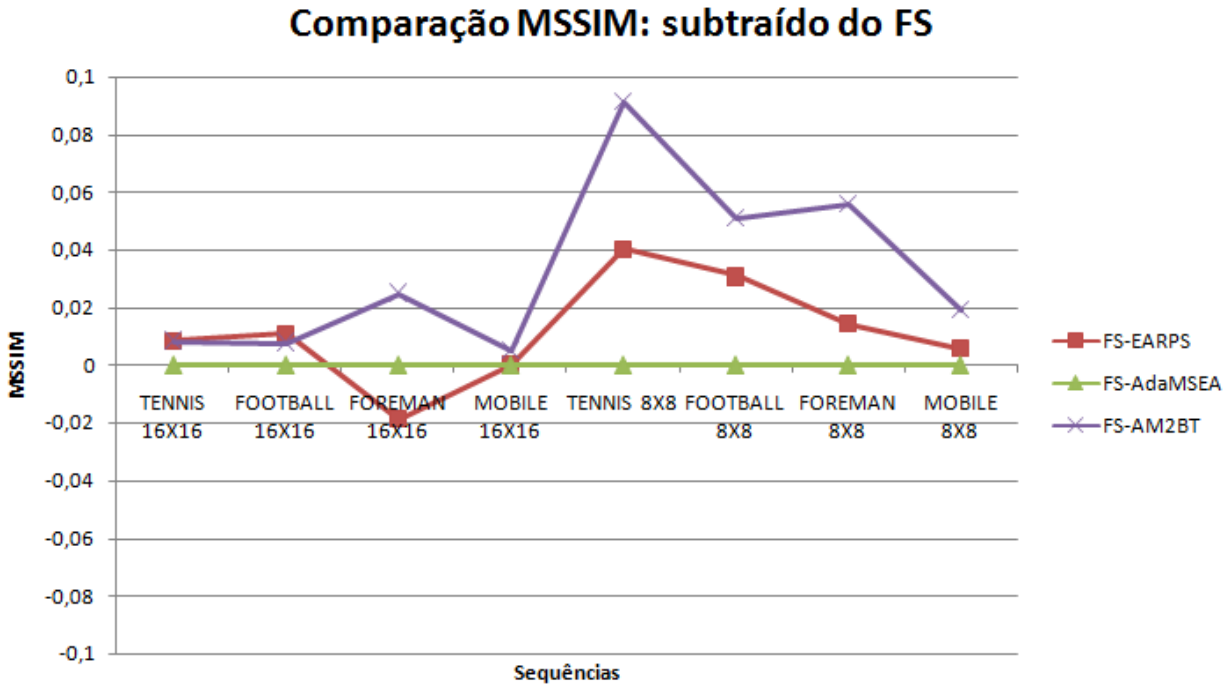


Figura 3.17: Comparação MSSIM para as diversas sequências com distintos tamanhos de bloco, com referência ao desempenho do algoritmo FS.

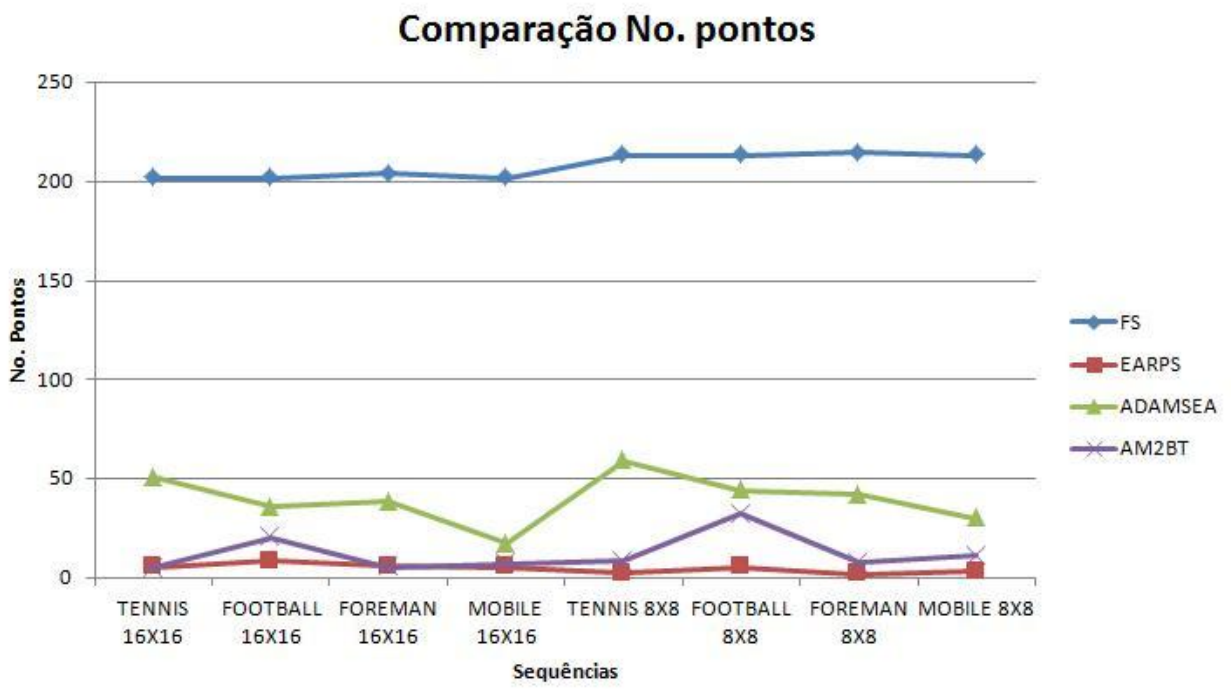


Figura 3.18:Comparação de número de pontos de busca para as diversas sequências com distintos tamanhos de bloco.

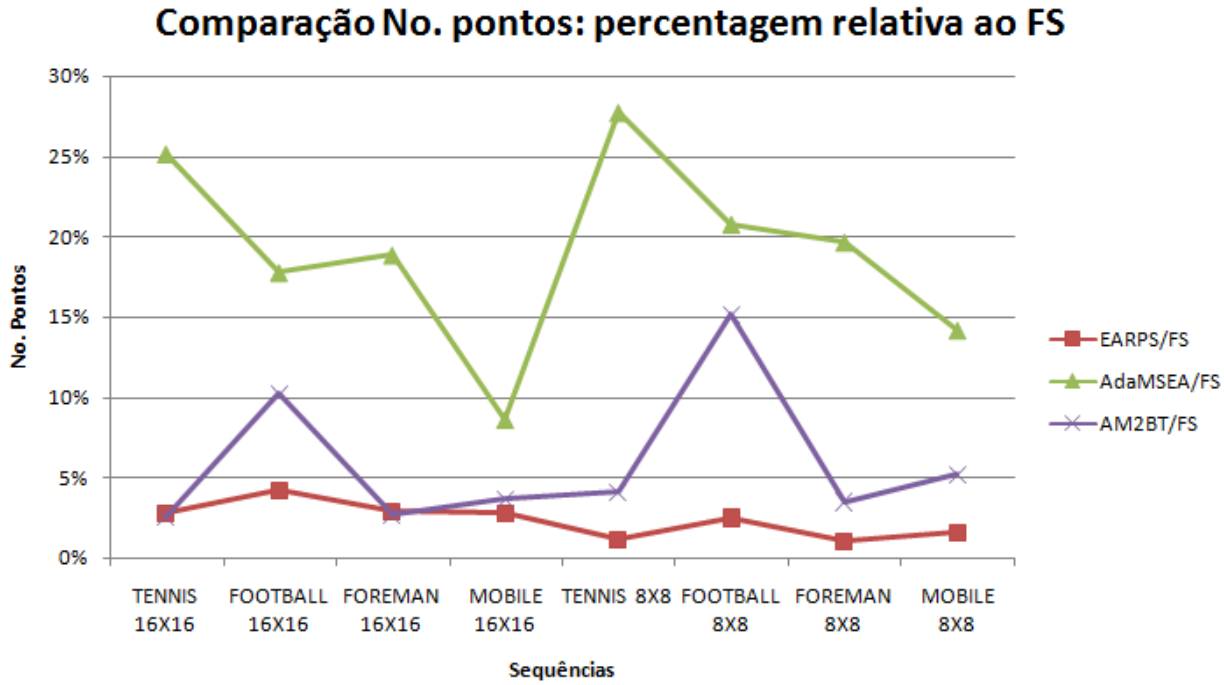


Figura 3.19: Comparação de número de pontos de busca para as diversas sequências com distintos tamanhos de bloco, em termos de percentagem relativa ao desempenho do algoritmo FS.

Capítulo 4

O *codec* Dirac

O Dirac é um *codec* aberto de vídeo digital, originalmente desenvolvido pela BBC. Provendo grande eficiência de compressão, sua arquitetura básica é similar à de um *codec* híbrido convencional, com a diferença de utilizar a Transformada *Wavelet* Discreta (DWT, *Discrete Wavelet Transform*) em lugar das usuais transformadas de bloco – como a Transformada Cosseno Discreto (DCT, *Discrete Cosine Transform*).

Neste capítulo, será explorado em mais detalhes o funcionamento deste *codec*. Inicialmente, os principais módulos do codificador são apresentados, um a um. Na seqüência, descreve-se um estudo realizado com o objetivo de comparar o desempenho deste *codec* com o do H.264/AVC, um dos métodos de compressão mais eficientes atualmente.

4.1 Arquitetura básica

Como já descrito, o Dirac é um *codec* de vídeo dito híbrido, pelo fato de empregar técnicas de compressão de imagens estáticas e de compressão de movimento. A Figura 4.1 ilustra sua arquitetura básica.

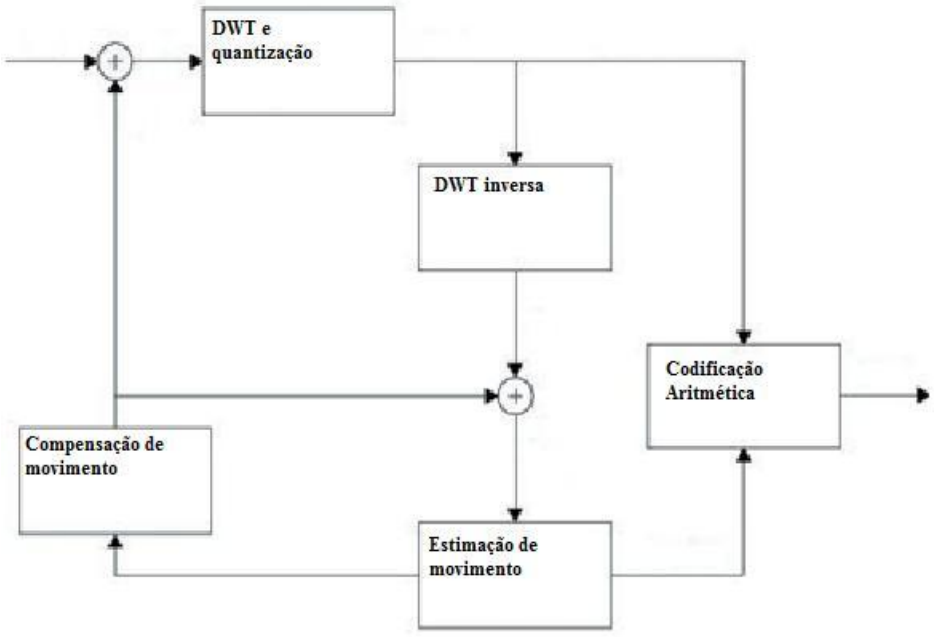


Figura 4.1: Arquitetura básica do codificador Dirac (42).

Um quadro a ser codificado na entrada é subtraído de sua predição, gerando um quadro residual, que será transformado (utilizando a DWT) e quantizado antes de passar pela codificação de entropia e ser enviado/armazenado.

A predição, realizada na etapa de estimação de movimento, gera outros dados a serem enviados (vetores de movimento), que também passarão pela codificação de entropia e serão transmitidos/armazenados. Os dados necessários para a estimação de movimento são o quadro que será comprimido e os quadros de referência reconstruídos. Os quadros de referência são armazenados em um *buffer* interno, depois de passarem pela transformação e quantização inversas. Este estágio se ocupa do *trade-off* entre precisão de predição e taxa de *bits*.

A compensação de movimento gera o quadro reconstruído, que será obtido pela utilização dos vetores de movimento para sua reconstrução. Esta predição é subtraída do quadro original, restando o residual que, como já dito, será transformado e quantizado antes de passar pela etapa de codificação de entropia para ser enviado/armazenado.

A codificação de entropia no Dirac, também representada na Figura 4.1, consiste na binarização, modelagem de contexto e no uso de codificador aritmético. Esta etapa realiza compactação dos dados – não ocorrendo perdas.

Nas seções seguintes, são descritas em mais detalhes as principais etapas deste algoritmo de codificação.

4.2 Transformação e quantização

O Dirac utiliza a DWT como transformada, se diferenciando dos *codecs* mais usuais – como toda a linha MPEG -, que utilizam a DCT. Isto possibilita o uso de qualquer dimensão para os quadros – o emprego de *padding* assegura a correta aplicação da DWT.

Aplica-se a DWT em quadros de tipo Inter e Intra: os Intra são diretamente transformados, enquanto os Inter passam previamente por compensação de movimento, resultando num quadro residual a ser transformado. Três componentes (um de luminância e outros dois de crominância), na forma de vetores bidimensionais, são utilizados em ambos os casos.

Três estágios são empregados na codificação dos quadros. Inicialmente, os vetores de dados são transformados utilizando a DWT com filtros separáveis, resultando em dados divididos em sub-bandas. Em seguida, estes são quantizados e ulteriormente passam pela etapa de codificação de entropia. A Figura 4.2 ilustra estas etapas.

Como observado nesta, a quantização e a codificação de entropia de uma sub-banda dependem da quantização e codificação de sub-bandas anteriores. Este aspecto é devido às relações entre "pais" e "filhos" existentes entre os coeficientes, que permitem a eliminação de redundância dos dados.

As codificações de um quadro Intra e do quadro residual Inter diferem apenas no uso de predição na sub-banda DC no quadro Intra.

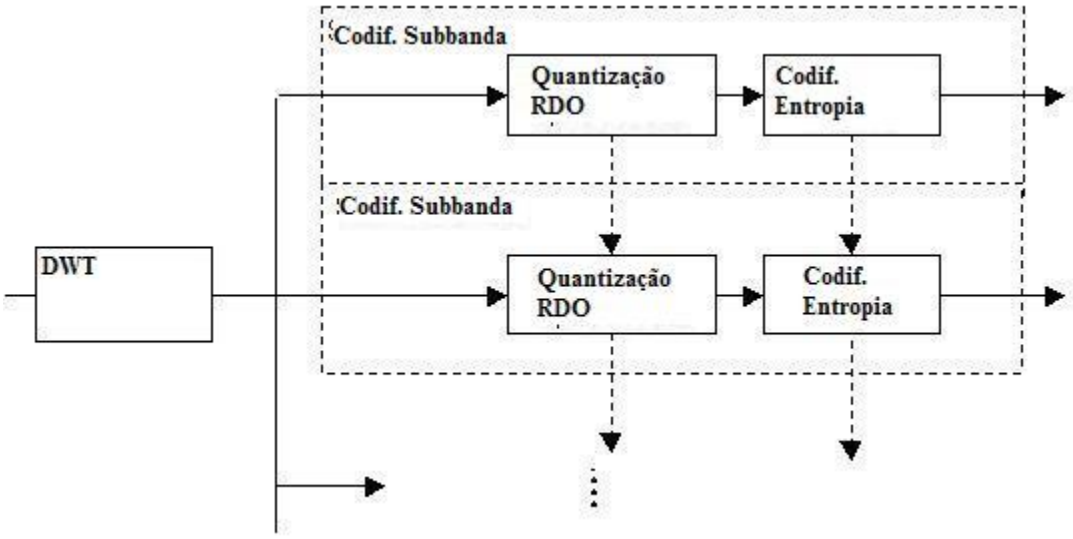


Figura 4.2: Etapas realizadas na codificação dos dados dos quadros (42).

A DWT já foi introduzida na seção 2.3. Ela é utilizada com o propósito de eliminar dados de menor relevância das imagens utilizadas, sendo realizada na prática a partir de dois filtros, passa-baixas e passa-altas, seguidos de subamostragem por um fator de 2. Estes procedimentos são aplicados nas linhas e nas colunas, sendo bem ilustrados na Figura 2.5 e na Figura 2.6.

A escolha dos filtros *wavelet* a serem utilizados impacta o resultado da compressão obtido. É importante constatar o *trade-off* em termos de complexidade e performance para o uso de cada *wavelet*. Este se dá principalmente devido às características de suporte compacto e de regularidade das *wavelets*, que são conflitantes (38). Quanto mais extenso o filtro, mais regular, favorecendo o processo de compressão (performance); entretanto, pior será o aspecto de implementação do ponto de vista de velocidade (complexidade).

Uma quantidade razoável de filtros *wavelet* é suportada pelo *codec* Dirac (a nomenclatura (m,n) se refere ao comprimento do filtro de análise passa-baixas e passa-altas):

1. Deslauriers-Dubuc (9,7);
2. LeGall (5,3);
3. Deslauriers-Dubuc (13,7);
4. Haar sem *shift*;
5. Haar com *shift* simples por nível;
6. Fidelity;
7. Daubechies (9,7) – aproximação inteira.

Todos os filtros suportados pelo *codec* são implementados utilizando-se técnicas de *lifting*, para otimização de velocidade a partir da simplificação da implementação devido a certas propriedades matemáticas do processo.

Para a correta aplicação de k estágios da DWT, faz-se necessário que 2^k seja divisor das dimensões dos quadros de todas as componentes. Caso estas não atendam a esse critério, realiza-se *padding* (com os valores das extremidades, para melhor performance de compressão) para que este seja atendido.

Outra técnica empregada no Dirac é a de explorar as relações existentes entre coeficientes "pais" e "filhos" – tópico também já citado na seção 2.3. Codificando-se primeiramente os coeficientes de baixas frequências, exerce-se a opção de codificar ou não os "filhos", dependendo do valor dos "pais", comparados com um limiar. Estas relações "pais"-"filhos" são reapresentadas na Figura 4.3.

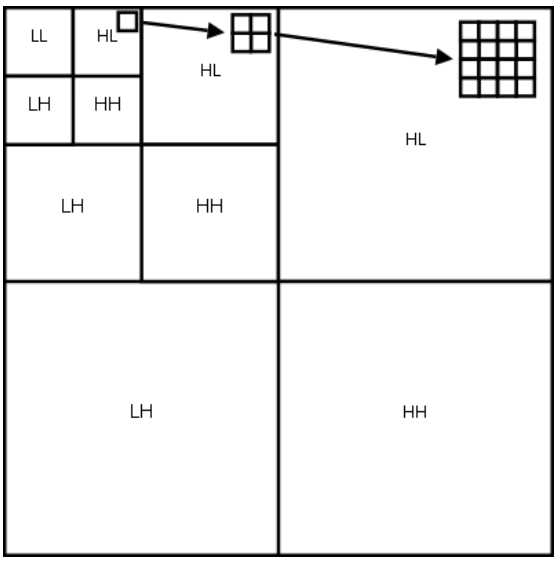


Figura 4.3: Relações "pais"-"filhos" dos coeficientes resultantes da DWT (42).

A quantização realizada pelo Dirac baseia-se num esquema de “zona morta”, caracterizada pelo fato de o intervalo empregado para a quantização utilizando o valor nulo ser duas vezes maior que o intervalo normal. A quantização aplicada é, desta forma, muito simples de ser implementada: basta dividir o valor pelo fator de quantização (*quantiser*) e arredondar para o inteiro mais próximo de zero – conforme a equação 4.1. Este procedimento, ilustrado na Figura 4.4, é bastante austero com os valores pequenos – o que permite uma operação de eliminação de ruído eficaz.

$$N = Q(v) = \begin{cases} \left\lfloor \frac{|v|}{D} \right\rfloor, se\ v \geq 0 \\ -\left\lfloor \frac{|v|}{D} \right\rfloor, se\ v < 0 \end{cases} \tag{4.1}$$

onde v é o valor a ser quantizado por N = Q(v), e D é o fator de quantização.

Assim, a representação de qualquer valor pertencente a um determinado intervalo é feita por um valor pré-definido deste próprio intervalo, sendo o valor médio o que é escolhido normalmente. Para o caso do quantizador utilizado no Dirac, isto é de fato utilizado para a codificação Intra. Entretanto, no caso de quadros de tipo Inter, pelo fato de os coeficientes transformados terem distribuição centrada muito próximo do zero, decaindo rapidamente para

valores maiores – o que implica que é mais provável que os valores se situem na primeira metade do intervalo de quantização, e por isso um valor mais próximo do limiar inferior do intervalo é empregado. O valor reconstruído é dado pela expressão 4.2. No caso Intra, X = 0,5. No caso Inter, X = 0,375, refletindo a característica citada.

$$v_{rec} = \begin{cases} 0, se\ N = 0 \\ (N + X)D, se\ N > 0 \\ (N - X)D, se\ N < 0 \end{cases} \quad (4.2)$$

onde v_{rec}é o valor reconstruído do coeficiente e X determina o ponto de reconstrução, variando entre 0 e 1.

Figura 4.4: (a) Quantização uniforme. (b) Quantização utilizando "zona morta", utilizada pelo Dirac (42).

A escolha dos chamados *quantisers*, ou fatores de quantização, é uma etapa também essencial para o trabalho do codificador. O Dirac utiliza uma técnica bastante conhecida, de minimização de uma expressão Lagrangiana (equação 4.3) envolvendo a taxa e a distorção.

$$J = D(Q) + \lambda R(Q) \quad (4.3)$$

onde J é a expressão a ser minimizada, D a distorção, R a taxa de *bits*, Q o fator de quantização e λ o parâmetro que determinará o compromisso entre a taxa e a distorção utilizadas.

Logo, a minimização de 4.3 leva à alocação de *bits* conforme a importância dada à taxa e à distorção: quanto maior λ, maior é a importância conferida à taxa; quanto menor seu valor, mais

relevância é atribuída à distorção. Uma implementação correta deste *trade-off* é essencial para o bom funcionamento do codificador.

A taxa e a distorção são, na prática, diretamente dependentes do fator de quantização; desta forma, diversos valores para este parâmetro são testados e o que minimiza J escolhido. A taxa é estimada por uma medida da entropia dos símbolos quantizados com o fator Q. A distorção, por outro lado, é medida em termos da seguinte expressão:

$$E = \left(\sum_{i,j} \left| p_{i,j} - q(i,j) \right|^4 \right)^{1/4} \tag{4.4}$$

onde q(i,j) representa os coeficientes quantizados, que são comparados aos valores não-quantizados, $p_{i,j}$.

A equação 4.3 fica, então:

$$J = \left(\frac{E^2}{w} \right) + \lambda . C . Ent(Q) \tag{4.5}$$

onde w é o peso atribuído à sub-banda sob análise (maiores frequências tendo maiores valores de w), C é um fator de correção – necessário pelo fato de o cálculo de entropia aqui empregado não levar em conta dependências entre coeficientes que são importantes na etapa de codificação de entropia – e Ent(Q) a entropia calculada levando-se em conta a quantização com fator Q.

Os fatores de quantização são incrementados em quartos de potências de 2 – ou seja, Q é uma aproximação inteira de $2^{n/4}$, para inteiros n. Colocado de outra forma, os fatores de quantização representam magnitudes de coeficientes a precisão variável em incrementos de um quarto de *bit*.

4.3 Estimação e compensação de movimento

Específica ao codificador, a etapa de estimação de movimento compreende a parte mais complexa de qualquer *codec* de vídeo.

Três tipos de quadro são utilizados no Dirac: I, L1 e L2, conforme apresentado na Figura 4.5. Os primeiros são codificados sem referência a outros quadros (predição Intra-frame), enquanto os outros realizam eliminação de redundância devida ao movimento (pelo uso de quadros previamente codificados como referência): os L1 se utilizam apenas de referências

temporalmente anteriores apenas (*forward prediction*, similares ao tipo P comumente utilizado nos padrões MPEG) e os L2 empregam predições cuja referência pode estar temporalmente antes ou depois (utilizando também *backward prediction*, similares ao tipo B presente nos padrões MPEG).

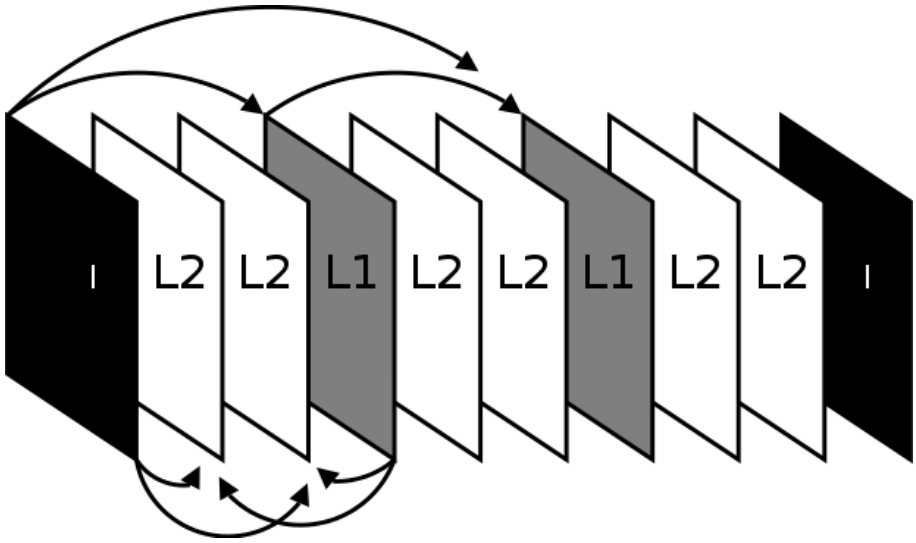


Figura 4.5: Estrutura temporal utilizada: quadros I, L1 e L2 (42).

O *buffer* empregado no tratamento dos quadros, no momento de decodificação, mantém na sua estrutura apenas aqueles que ainda apresentam alguma utilidade. Isto é verificado pelo *header* de cada quadro, que especifica seu número na ordem de visualização, os números dos quadros que servem como referência e o tempo que este deve permanecer no *buffer*. Assim, o decodificador possui todas as informações necessárias para o correto gerenciamento dos quadros, procurando os que servem de referência, exibindo os que são demandados e eliminando os que tiverem expirado.

O número de quadros de tipo L1 e I pode ser especificado para o codificador, assim como a separação entre os quadros L1. Assim, há a possibilidade de se construir diversos Grupos de Figuras distintos (GOP, *Group of Pictures*), não apenas os utilizados pelos padrões MPEG. Permite-se também a utilização de estruturas GOP variantes, devido à flexibilidade existente do *buffer*.

Uma possibilidade que existe no codificador é a de definir o número de quadros L1 como sendo zero. Neste caso, realiza-se codificação com quadros apenas de tipo I, ou seja, utiliza-se

somente predição intra. A compressão realizada desta forma é útil para um leque variado de aplicações, como a edição, na qual acesso rápido a todas os quadros é necessário.

Outras possibilidades para melhorar ainda mais a codificação são já previstas no *codec*, mas ainda não suportadas. Duas destas são o uso de um modelo parametrizado do movimento e a alternativa de saltar alguns quadros.

Como explicado na seção 3.2, faz-se importante a definição do critério de seleção utilizado no momento da procura para encontrar o melhor vetor de movimento possível. Este critério, ao qual se demanda comumente a minimização da energia residual a ser codificada, deve, por outro lado, verificar se os vetores de movimento atribuídos serão, da mesma forma, codificados com um número reduzido de bits. Isto porque a utilização de vetores de movimento diferentes para cada bloco demanda mais taxa para o *bitstream*, já que sua codificação é dificultada, o que nem sempre resulta em ganho de qualidade, pois estes são tão pequenos que acabam sendo facilmente corrompidos por ruído.

Desta forma, o Dirac adapta o SAD (Equação 3.1), no momento da decisão de modo, levando em conta a suavização resultante da mudança de vetores de movimento – esta sendo medida pela diferença entre cada vetor de movimento candidato e a mediana daqueles definidos para seus pixels vizinhos (como a análise é feita em *raster order*, os vetores V1, V2, V3 e V4 já estarão definidos no momento do cômputo de V - Figura 4.6. Entretanto, apenas V2, V3 e V4 são utilizados no Dirac).

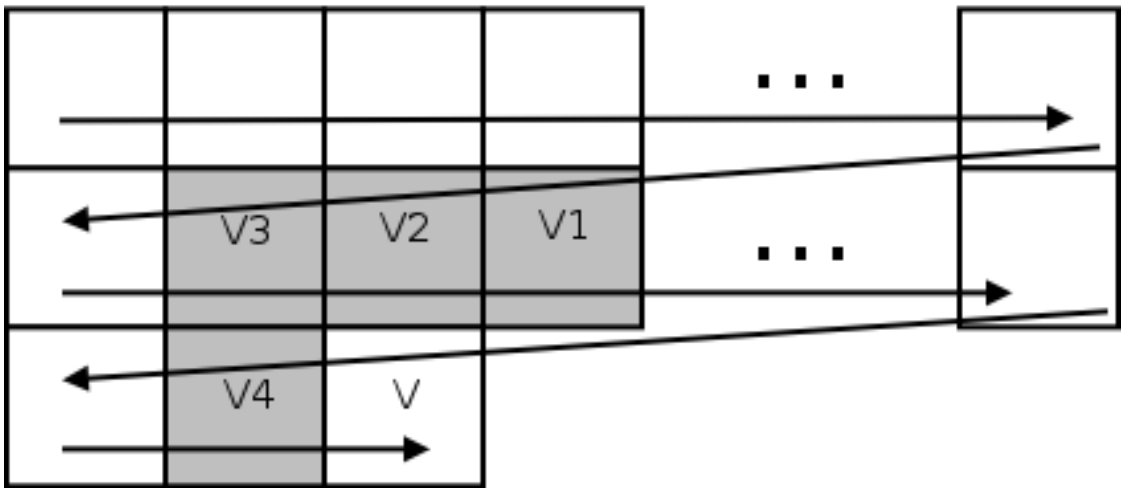


Figura 4.6: Vetores de movimento vizinhos anteriormente definidos (42).

A métrica resultante a ser minimizada será, então:

$$SAD + \lambda \,.\, min\left(|V_x - pred_x| + |V_y - pred_y|, |V_x| + |V_y|\right) \quad (4.6)$$

onde $pred$ é o vetor predito pela mediana dos vizinhos e λ um parâmetro de codificação.

O parâmetro λ é usado para controlar o *trade-off* entre precisão do bloco encontrado e suavidade de mudança de vetor. Quanto maior seu valor, mais importância será acordada ao último – o vetor escolhido será, no limite, o mais similar a seus vizinhos. Inversamente, quando λ for pouco significativo, esta métrica será dominada pelo SAD.

Usa-se o valor $|V_x| + |V_y|$, na mesma expressão, para previnir a utilização de um excesso de suavização, pois quando há, de fato, uma mudança significativa de movimento, o vetor terá de ser trocado – este mecanismo evita seu excesso de penalização.

A estimação de movimento no Dirac é realizada por meio de três etapas básicas, utilizando apenas a componente de luminância. Primeiramente, realiza-se a busca pelos vetores de movimento para cada bloco, com precisão inteira de pixel, empregando um método hierárquico. Em seguida, estes vetores são refinados para precisão de sub-pixel – suportando até 1/8 de pixel. Na terceira etapa, realiza-se a decisão de qual modo será utilizado (*mode decision*), levando-se em conta as diferentes formas de predição a utilizar.

O Dirac adota estruturas de macroblocos para possibilitar a adaptação ao movimento contido no vídeo, por permitir a variação do tamanho dos blocos. O estágio de estimação de movimento da codificação é organizado por macroblocos, e cada combinação de tamanho de bloco e de modo de predição é tentada - a decisão do modo a utilizar é, desta forma, realizada macrobloco por macrobloco.

Na compensação de movimento, é permitido o uso de compensação de movimento sobreposta (OBMC – *Overlapped Block Motion Compensation*), na qual algumas regiões dos quadros podem pertencer a mais de um bloco para o qual se procurará vetores de movimento.

Na sequência, examinam-se as técnicas utilizadas nestas etapas em mais detalhes.

Busca pelos vetores de movimento com precisão inteira de pixel

O codificador Dirac implementa estimação de movimento hierárquica, que consiste basicamente na realização de estimação de movimento preliminarmente com versões menores (subamostradas) do quadro corrente e da referência. Desta forma, vetores encontrados para níveis

mais altos (versões reduzidas dos quadros) são utilizados como base para a procura nos níveis subsequentes. A Figura 4.7 ilustra esta busca hierárquica.

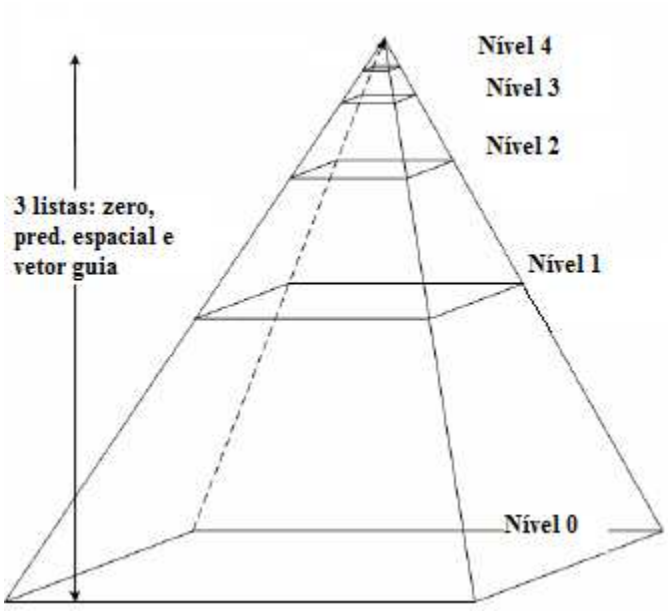


Figura 4.7: Estimação hierárquica utilizada no Dirac: aos níveis mais altos, correspondem as menores resoluções (adaptado de (26)).

O tamanho do bloco utilizado em todos os níveis é constante, o que acarreta no fato de um nível possuir um quarto da quantidade de blocos que o imediatamente superior. Assim, o vetor encontrado em um dado nível para um certo bloco (chamado vetor guia) será a referência para quatro blocos no nível acima.

A cada nível, vetores de movimento são buscados em uma região no entorno da que é apontada pelo vetor guia. São também levados em conta o vetor (0,0) e outro derivado de uma predição dos anteriormente definidos. A procura de vetores em todas estas regiões evita que o algoritmo se prenda numa região de mínimo local. Entretanto, o fato de utilizar busca hierárquica faz com que esta técnica seja ainda bastante simples, não consumindo um tempo extraordinário de processamento para codificação. O Capítulo 5 explora em mais detalhes o algoritmo utilizado para busca hierárquica.

Refinamento para precisão de sub-pixel

A busca por vetores de movimento com precisão de sub-pixel é realizada como um refinamento da busca de vetores de movimento de precisão inteira. Ela é realizada também de forma hierárquica. O processo é ilustrado na Figura 4.8.

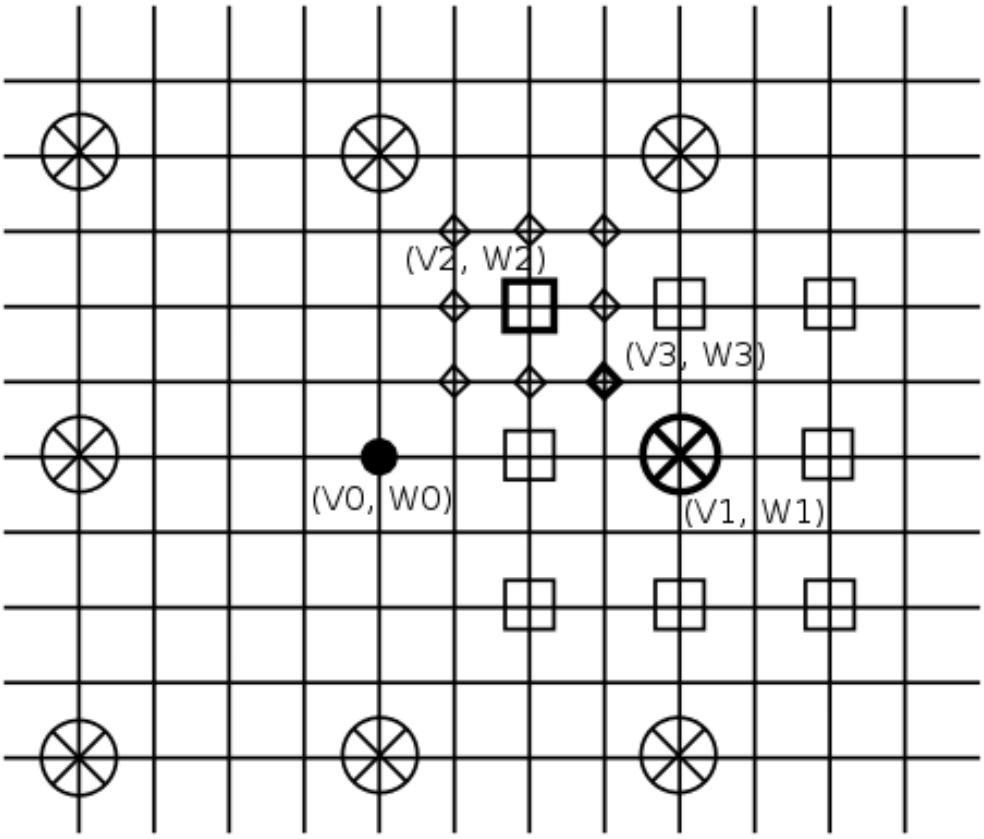


Figura 4.8: Exemplo de estimação de movimento com precisão de 1/8 de sub-pixel (42).

Inicialmente, encontra-se o melhor vetor para precisão de ½ pixel, entre o vetor de precisão inteira encontrado anteriormente - representado por (V_0, W_0) na Figura 4.8 – e seus oito vizinhos. Conforme a figura, (V_1, W_1) é encontrado e, a partir dele, realiza-se a busca por vetores de precisão de ¼ de pixel – sendo (V_2, W_2) o escolhido, e assim em diante.

Apesar de ser possível realizar a conversão dos quadros de um fator de 8 em cada dimensão, não é isto que é realizado na prática. Isto evita o excesso de cálculos que poderia aumentar bastante a complexidade computacional do *codec*.

A referência é convertida apenas de um fator de 2, sendo as seguintes conversões realizadas à medida em que se precisa dos valores, por interpolação linear. Na sequência, um filtro com

características de passa-baixas é utilizado para eliminar ruídos possivelmente resultantes das interpolações.

Estruturas de macroblocos e decisão de modo

O Dirac apresenta a possibilidade da realização de estimação e compensação de movimento utilizando blocos de tamanho variável, isto é, a unidade para a qual será realizada a estimação e compensação de movimento não terá tamanho nem modo de predição pré-estabelecidos.

Algumas possibilidades existem para cada uma destas unidades. Cada macrobloco pode ser dividido de três formas, dependendo do nível de particionamento (*splitting level*) selecionado, sendo que um macrobloco representa uma matriz 4x4 de blocos (os blocos podem ter qualquer tamanho, desde que obedeçam alguns critérios necessários para que a compensação de movimento sobreposta seja realizada adequadamente). Estas possibilidades estão representadas na Figura 4.9:

- Nível de particionamento 0: sem subdivisão, apenas um vetor de movimento por quadro de referência para o macrobloco;
- Nível de particionamento 1: divisão em 4 submacroblocos (matriz 2x2 de blocos), cada um possuindo um vetor de movimento por referência;
- Nível de particionamento 2: divisão nos 16 blocos, cada um possuindo um vetor de movimento por referência.

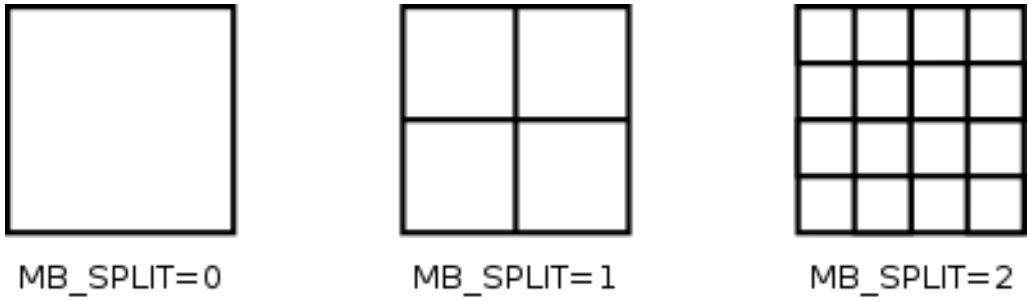


Figura 4.9: Formas de particionar macroblocos (42).

A escolha do nível de particionamento é realizada a partir da estimação de movimento para os submacroblocos e macroblocos. Ao mesmo tempo, realiza-se a definição do modo de predição

mais adequado para cada unidade de predição (bloco, submacrobloco, macrobloco), sendo que há 4 disponíveis:

- Intra: predição por um valor DC (valor médio);
- Ref1_only: predição realizada apenas com a referência 1;
- Ref2_only: predição realizada apenas com a referência 2;
- Ref1and2: predição realizada com ambas referências.

Desta forma, para cada macrobloco, uma combinação destes parâmetros (nível de particionamento e modos) é testada, calculando-se seu custo e comparando-o com o das outras possibilidades, para selecionar a que oferece melhor desempenho. Este custo é calculado com base na Equação 4.6, ajustado para levar em conta os diferentes tamanhos dos blocos, submacroblocos e macroblocos utilizados.

Outros parâmetros relacionados a cada unidade de predição são:

- Ref1_x: componente horizontal do vetor de movimento da primeira referência;
- Ref1_y: componente vertical do vetor de movimento da primeira referência;
- Ref2_x: componente horizontal do vetor de movimento da segunda referência;
- Ref2_y: componente vertical do vetor de movimento da segunda referência;
- DC: valor médio da unidade de predição para cada componente (Y, U e V), codificado com precisão de 8 bits.

Evidentemente, apenas alguns destes valores estarão presentes no *bitstream*, pois, por exemplo, não há necessidade de enviar o valor DC caso a predição seja inter-frame.

Compensação de movimento sobreposta

Esta técnica, já descrita no item 3.3, permite a realização de estimação e compensação de movimento se utilizando de blocos que se sobrepõem. As partes sobrepostas atribuem, então, um determinado peso ao valor estimado por cada bloco, suavizando assim o efeito de bloco resultante, o que facilita sua codificação por *wavelets*.

Portanto, o valor predito de um pixel que pertença a mais de um bloco será dado por:

$$p_p(x, y, t) = \sum_i w_i p(x - V_i, y - W_i, t') \tag{4.7}$$

onde p_p é o pixel predito, w_i os pesos atribuídos a cada bloco i (tal que $\sum_i w_i = 1$), (V_i,W_i) o vetor de movimento encontrado no bloco i da referência t' para o quadro corrente t.

Quaisquer tamanho de blocos e grau de sobreposição podem ser utilizados – estes parâmetros são configuráveis no codificador e transmitidos ao decodificador

4.4 Codificação de entropia

Esta etapa é realizada tanto para os coeficientes resultantes de transformação e quantização, quanto para os dados de movimento de cada bloco. O esquema básico implementado é o mesmo em ambos os casos – apresentado na Figura 4.10.

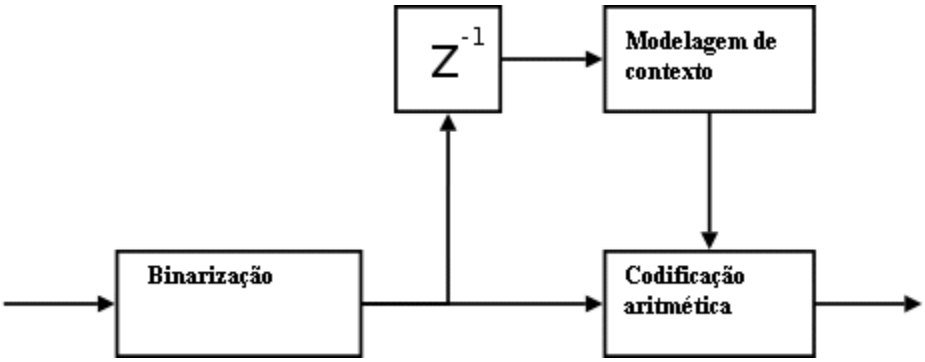


Figura 4.10: Esquema básico de codificação de entropia no Dirac (42).

Codificação de entropia de coeficientes wavelet

O Dirac realiza a codificação de coeficientes de sub-bandas utilizando uma ordem zig-zag, começando com frequências baixas até atingir as mais elevadas. Dentro de cada sub-banda, os coeficientes são divididos em blocos – assim, cada bloco é codificado em *raster order* e, dentro dele, seus coeficientes são codificados também com este tipo de ordenamento. Um bloco pode ser, alternativamente, pulado – o que o decodificador interpreta atribuindo-se zeros para todos os coeficientes deste bloco.

A binarização tem o propósito de facilitar a análise estatística do *bitstream*, pelo fato de transformar os coeficientes, com valores os mais diversos, em bits. Como a maioria destes valores ocorre raramente, se não houvesse a binarização pouca compactação seria possível.

Este *codec* utiliza binarização entrelaçada exp-Golomb. Os bits de dados são entrelaçados com zeros, colocando-se um ‘1’ no final da sequência, indicativo de terminação.

A modelagem de contexto segue, estimando as probabilidades necessárias ao codificador aritmético a partir das dependências existentes na sequência de bits, relacionadas às relações “pais”-”filhos” e às relações de vizinhança. Em particular, coeficientes com valor zero tendem a se localizar numa mesma região – e esta característica deve ser aproveitada para melhorar ainda mais a compactação.

O codificador aritmético apresenta uma performance excelente, muito próxima ao limiar teórico determinado pela Teoria da Informação. Ilustrar-se-á o funcionamento deste codificador a partir de um exemplo.

Considere-se que o bit zero tenha probabilidade 0,7 e o bit 1 tenha probabilidade 0,3. Codificar-se-á a sequência (10001). A cada um dos bits é associado um intervalo, conforme a Tabela 4.1.

Tabela 4.1: Intervalos para cada bit para o codificador aritmético do exemplo.

Bit	**Probabilidade**	**Intervalo**
0	0,7	0-0,7
1	0,3	0,7-1,0

O procedimento utilizado para codificação é ilustrado na Tabela 4.2.

Cada vez que um símbolo é codificado, o intervalo se torna progressivamente menor, tendo como resultado certo subintervalo. Finalmente, qualquer valor neste (0,77203-08029) poderá ser enviado, que toda a sequência de símbolos poderá ser decodificada corretamente no receptor. Ilustrar-se-á, também, na sequência, o procedimento de decodificação.

Codificação de entropia de dados de movimento

O que distingue a codificação de entropia dos coeficientes wavelet com relação à dos dados de movimento é sua preparação antes da binarização. Implementam-se estratégias de predição dos vetores de movimento e de modo de particionamento, por exemplo, o que permite diminuir a quantidade de informações a serem enviadas.

O nível de particionamento é predito a partir dos níveis de macroblocos vizinhos da abertura do macrobloco corrente (isto é, os três macroblocos à esquerda, acima e acima à esquerda).

Tabela 4.2: Procedimento de codificação do codificador aritmético para o exemplo dado.

Procedimento	Intervalo	Símbolo	Intervalo	Notas
Setar intervalo inicial	0-1,0			
Para o próximo símbolo, achar intervalo correspondente		1	0,7-1,0	Intervalo entre 0-1,0
Setar novo intervalo	0,7-1,0			
Para o próximo símbolo, achar intervalo correspondente		0	0-0,7	Intervalo entre 0-1,0
Setar novo intervalo	0,7-0,91			0,91 é 70% do intervalo
Para o próximo símbolo, achar intervalo correspondente		0	0-0,7	Intervalo entre 0-1,0
Setar novo intervalo	0,7-0,847			0,847 é 70% do intervalo
Para o próximo símbolo, achar intervalo correspondente		0	0-0,7	Intervalo entre 0-1,0
Setar novo intervalo	0,7-0,8029			0,8029 é 70% do intervalo
Para o próximo símbolo, achar intervalo correspondente		1	0,7-1,0	Intervalo entre 0-1,0
Setar novo intervalo	0,77203-0,8029			0,77203 é 70% do intervalo

Tabela 4.3: Procedimento de decodificação do codificador aritmético para o exemplo dado.

Procedimento	Intervalo	Subintervalo	Símbolo decodificado
Setar intervalo inicial	0-1,0		
Achar subintervalo no qual símbolo está		0,7-1,0	1
Setar novo intervalo	0,7-1,0		
Achar subintervalo no qual símbolo está		0,7-0,91	0
Setar novo intervalo	0,7-0,91		
Achar subintervalo no qual símbolo está		0,7-0,847	0
Setar novo intervalo	0,7-0,847		
Achar subintervalo no qual símbolo está		0,7-0,8029	0
Setar novo intervalo	0,7-0,8029		
Achar subintervalo no qual símbolo está		0,77203-0,8029	1
Setar novo intervalo	0,77203-0,8029		

Dois bits são utilizados para indicar o modo que, similarmente, é predito pela abertura do bloco (ou submacrobloco ou macrobloco): o primeiro, quando 1, indica a utilização de predição com a referência 1 e o segundo, da mesma forma, indica a utilização da referência 2. Ou seja, quando o valor dos dois bits é:

- 0, utiliza-se modo Intra;
- 1, utiliza-se modo Ref1_only;
- 2, utiliza-se modo Ref2_only;
- 3, utiliza-se modo Ref1and2.

Já os vetores de movimento têm suas componentes horizontal e vertical preditas separadamente, a partir dos blocos da abertura, pela mediana destes valores – a convenção de *raster order* é utilizada para a ordem dos macroblocos e para a ordem dos blocos dentro dos macroblocos. Os valores DC são preditos a partir do valor médio dos mesmos vizinhos.

Para os valores DC e vetores de movimento, quando um dos blocos da abertura não está presente, simplesmente se considera apenas os dois restantes (para o caso dos vetores de movimento, a mediana vira uma média). Quando há apenas um bloco disponível, este será o valor utilizado. Se não houver blocos disponíveis, nenhuma predição é feita para os vetores, enquanto o valor DC é setado por *default* em zero.

Na sequência, são realizadas as etapas de binarização, modelagem de contexto e codificação aritmética que, como já dito, são bastante similares às realizadas para codificação de coeficientes wavelet, já descritas.

4.5 Estudo comparativo de desempenho

A fim de avaliar o desempenho do codificador Dirac, realizou-se um estudo, comparando sua performance com a do codificador H.264/AVC, um dos mais eficientes existentes atualmente.

Os dois codecs foram usados para comprimir 8 vídeos, compreendendo diferentes graus e formas de movimento com vários tamanhos. Seus detalhes são apresentados na Tabela 4.4.

Tabela 4.4: Vídeos utilizados para os experimentos

Vídeo	Resolução	Número de quadros	Taxa de quadros (quadros por seg.)
canal352	352x288 (CIF)	68	12,5
snow352	352x288 (CIF)	75	12,5
squirrel352	352x288 (CIF)	75	12,5
canal720	720x576 (SDTV 576p)	50	25
snow720	720x576 (SDTV 576p)	50	25
squirrel720	720x576 (SDTV 576p)	50	25
snow1280	1280x720 (HDTV 720p)	30	60
squirrel1280	1280x720 (HDTV 720p)	30	60

Estes vídeos estão disponíveis em formato RGB em (43) e foram convertidos para o formato YUV, com amostragem de crominância 4:2:2. Cada vídeo foi comprimido diversas vezes com o mesmo codificador para diferentes taxas, com o objetivo de analisar o desempenho restringido a cada taxa.

Os codificadores usados foram os de referência (versões de pesquisa) para ambos os *codecs*, implementados em C++. Para o Dirac, a versão 1.0.2 foi empregada (44) enquanto o JM 15.1 foi usado para o H.264 (45) (*main profile*). As simulações foram realizadas utilizando um PC equipado com Intel Pentium M 740, 1,73GHz, 768MB RAM.

O codificador Dirac foi utilizado com seus parâmetros de referência, como recomendado pelos autores do software. Cada formato de vídeo implica diversos *presets*, ajustes nos parâmetros para a adequação a cada formato. Dois destes parâmetros são o tamanho e a separação entre blocos: para as resoluções CIF e SDTV 576p, os blocos são 12x12, com separação de 8 em cada direção; para a resolução HDTV 720p, os blocos são 16x16, com separação de 12 em cada direção (estes são os tamanhos e separações para o nível de particionamento 2, conforme seção 4.3).

Para prover comparação de qualidade visual, o PSNR e o MSSIM foram novamente utilizados. Eles foram computados para os coeficientes de luminância de cada quadro em cada vídeo e os valores médios de todo o vídeo foram levados em conta.

A complexidade computacional de cada codificador é analisada a partir do tempo de codificação por quadro. Embora as implementações neste trabalho não sejam otimizadas, elas provêem uma boa ideia do custo computacional de cada codificador.

Uma comparação de dois *codecs* deve ser submetida a certos critérios, para que seja efetiva. Objetiva-se sempre a obtenção da variação da qualidade visual dada uma restrição de taxa imposta ao *codec*. Entretanto, alguns parâmetros importantes são cruciais para que a comparação realizada seja realmente ilustrativa do desempenho de cada um destes.

A definição do método de estimação de movimento influencia decisivamente nos resultados obtidos. Deste modo, para as simulações realizadas, utilizou-se os dois *codecs* com estimação de movimento *Full Search* com área de procura de tamanho 32x32. O desempenho do Dirac com estimação hierárquica também foi avaliado, o que ilustra a variação que pode haver ao mudar-se este tipo de algoritmo.

Outro parâmetro a ser configurado é o tipo e a sequência dos quadros utilizados, ou seja, o Grupo de Figuras. Para ambos os codificadores, procurou-se estabelecer a distância de 3 quadros entre os de tipo P (ou L1, no caso do Dirac), o que, entretanto, nem sempre é respeitado, dadas as decisões do controle de taxa versus distorção de cada codificador. Por exemplo, o Dirac implementa um método de detecção de cortes de cenas. Nestes casos, ele insere um quadro intra-frame, já que a realização de estimação de movimento para esta quadro seria muito pouco efetiva.

Resultados e discussão

Os resultados obtidos são dados no Apêndice I: são apresentadas as curvas de qualidade de reconstrução objetiva (PSNR e MSSIM) por taxa de *bits*, assim como a tabela onde todos os resultados são encontrados, inclusive os que se referem ao tempo de processamento de cada quadro. A Tabela 4.5 apresenta um resumo dos resultados de complexidade computacional, extraídos da tabela do Apêndice I (o caso do Dirac com estimação de movimento hierárquica não é dado, já que esta comparação não é relevante). Apresenta-se também no Apêndice II exemplos de quadros reconstruídos, que podem ser visualmente comparados aos originais.

No que se refere ao desempenho em termos de qualidade de reconstrução, nota-se claramente que o *codec* H.264 é superior, quando a medida utilizada é o PSNR, assim como quando se usa o MSSIM. A diferença em termos de PSNR chega a atingir em alguns casos um pouco mais de 3dB, o que é bastante significativo. Entretanto, fica claro também que, para os resultados em termos de MSSIM, as diferenças entre as curvas são menores; ou seja, a diferença de qualidade bastante elevada constatada pelos resultados de PSNR não se confirma com as curvas de MSSIM.

A análise dos resultados fornecidos pelo Dirac por meio de indicadores objetivos já é bastante discutível; seus criadores argumentam que ele não foi desenvolvido para minimizar as diferenças medidas por tais métodos, mas apenas por medidas subjetivas, mais complexas a serem realizadas. De fato, ao se analisar vários quadros resultantes da codificação e, em seguida, decodificação realizadas por cada um dos *codecs*, constatam-se indícios de que há melhor qualidade visual nos tratados pelo Dirac, quando comparados aos tratados pelo H.264.

Exemplos são apresentados na Figura 4.11, na Figura 4.12 e na Figura 4.13. Nas duas primeiras, apresentam-se seções do 10º quadro da sequência snow1280, enquanto na seguinte uma seção do 10º quadro da sequência squirrel1280.

Na Figura 4.11, o rapaz que pratica *snowboarding* é o principal foco de atenção do espectador, enquanto o resto da cena é composta da paisagem, com árvores desfolhadas e bastante neve. Nota-se claramente que a imagem tratada pelo Dirac apresenta melhor qualidade – o rosto do rapaz é mais bem representado, assim como a parte esquerda de sua prancha (claramente borrada na imagem H.264, sendo impossível de perceber suas nuances). Todavia, esta apresenta um PSNR de apenas 32,5573 dB (e um MSSIM de 0,899686), enquanto o do quadro tratado pelo H.264 obteve 34,7512 dB (e um MSSIM de 0,929964).

Na Figura 4.12, mostra-se outra seção do mesmo quadro da Figura 4.11. As árvores desfolhadas apresentadas aparecem muito mais nítidas na imagem tratada pelo Dirac, enquanto aparecem muito borradas na codificada pelo H.264.

Tabela 4.5: Principais resultados de complexidade computacional: comparação entre o Dirac e o H.264.

Vídeo	**H.264 – Média de tempo de codif. por quadro (% comp. H.264)**	**Dirac - Média de tempo de codif. por quadro (% comp. H.264)**
canal352	38,08 (100)	20,93 (55)
snow352	38,98 (100)	21,78 (56)
squirrel352	39,73 (100)	20,89 (53)
canal720	148,01 (100)	45,32 (31)
snow720	153,06 (100)	73,23 (48)
squirrel720	157,24 (100)	51,67 (33)
snow1280	393,33 (100)	104,37 (27)
squirrel1280	412,57 (100)	71,66 (17)
Média global de percentagem	***100%***	***40%***

Na Figura 4.13, apresenta-se uma seção do 10º quadro da sequência squirrel1280 que, como foco principal, tem um esquilo se mexendo na grama, rodeado por árvores. Nota-se mais uma vez como a codificação pelo Dirac, apesar de obter PSNR e MSSIM menos significativos, são muito mais bem avaliadas subjetivamente por um observador humano. Nesta imagem, a pelugem do esquilo é muito mais bem representada pelo Dirac que pelo H.264, nesta estando nitidamente borrada, mais uma vez.

Enfim, há indícios significativos de que as imagens são muito mais naturais quando tratadas pelo Dirac, apesar de objetivamente obterem PSNR e MSSIM menores.

Isto acontece simplesmente pelo fato de as avaliações objetivas serem na prática "cegas" a muitos aspectos que são valorizados por um espectador que realmente observa a cena. No caso da

prancha do rapaz, na Figura 4.11, por exemplo, é bem provável que o borrão apresente menor erro que possa ser objetivamente calculado que as nuances apresentadas pela imagem superior – da mesma forma para o rosto no mesmo quadro e para os outros casos apresentados nas duas figuras subsequentes.

Figura 4.11: Seção do 10º quadro do vídeo snow1280. Acima, imagem tratada pelo Dirac (Hierárquico), com PSNR = 32,5573 e MSSIM = 0,899686. Abaixo, pelo H.264, com PSNR = 34,7512 e MSSIM = 0,929964.

Figura 4.12: Outra seção do 10º quadro do vídeo snow280. Acima, imagem tratada pelo Dirac (Hierárquico), com PSNR = 32,5573 e MSSIM = 0,899686. Abaixo, pelo H.264, com PSNR = 34,7512 e MSSIM = 0,929964.

Figura 4.13: Seção do 10º quadro do vídeo squirrel1280. Acima, imagem tratada pelo Dirac (Hierárquico), com PSNR = 30,3596 e MSSIM = 0,876668. Abaixo, pelo H.264, com PSNR = 33,5563 e MSSIM = 0,929297.

Este impasse é característico da diferença entre as avaliações objetiva e subjetiva, que podem ser significativas, como no caso apresentado e outras muitas vezes nos vídeos ora analisados. Pode-se afirmar, enfim, que os criadores do Dirac estão corretos ao não visar apenas a atingir certo nível de um indicador claramente pouco fiel à realidade.

Em algumas das curvas apresentadas no Apêndice I (particularmente as referentes ao vídeo snow720), observa-se que o desempenho do Dirac com *Full Search* é inferior ao deste *codec* com estimação hierárquica. Isto se deve ao fato de a busca hierárquica poder se estender além da região de busca estabelecida na busca exaustiva (como já notado anteriormente, o FS é ótimo apenas dentro da sua região de busca pré-determinada).

Considerando o tempo de processamento de cada *codec*, nota-se, pela Tabela I. 1 do Apêndice I, que o H.264 é muito mais "pesado" que o Dirac. Quando ambos os codificadores são utilizados com busca exaustiva, a latência observada pelo H.264 é de 1,7 a 7 vezes a do Dirac, dependendo do vídeo tratado. O tempo de processamento obtido para o Dirac com estimação de movimento hierárquica é bastante inferior às duas outras implementações; o que é natural, já que a busca realizada é muito mais simples.

A Tabela 4.5 sumariza os resultados de tempo de processamento: em média, o Dirac é 60% mais rápido que o H.264. O *speedup* atingido é significativo para todos os vídeos tratados, tendo variação de 44% até 83%, o que deixa clara a simplicidade e eficiência do Dirac frente à complexidade do H.264.

Por fim, tem-se que o Dirac é um *codec* muito bom, mesmo estando apenas em sua segunda versão, a mais recente atualmente disponível. Apesar de apresentar qualidade objetiva (PSNR, MSSIM) ainda bastante inferior à dada pelo H.264, constatam-se indícios relevantes de que a avaliação subjetiva lhe é favorável. Da mesma forma, a qualidade avaliada pelo MSSIM indica que a diferença de qualidade não é tão elevada quanto as medidas de PSNR deixam crer. Uma comparação baseada em análise subjetiva dos dois codecs seria certamente muito relevante, sendo um ótimo tema a ser explorado em trabalhos futuros.

Por outro lado, o Dirac é certamente muito mais simples, o que se verifica pela comparação em termos de tempo de processamento por quadro. Isto já era esperado, dada sua estrutura bastante simples apresentada neste capítulo. O fato de ele ser um *codec* livre de *royalties* é outro ponto importante. O H.264, por exemplo, exigirá o pagamento de *royalties* a partir do ano de 2011 – o que pode, em muitos casos, inviabilizar sua utilização.

Capítulo 5

Proposta de algoritmo de estimação de movimento para o codificador Dirac

Esta parte visa a oferecer a contribuição mais significativa deste trabalho. Após ter analisado técnicas modernas de estimação de movimento, compreendido em detalhes o funcionamento do *codec* Dirac (e comparado-o com o H.264, um dos *codecs* de melhor desempenho da atualidade), criou-se uma base importante de conhecimento para poder efetivamente contribuir com um novo e relevante algoritmo de estimação de movimento para o codificador Dirac.

Neste capítulo, será detalhado o algoritmo proposto, assim como as outras possibilidades de interesse para as técnicas estudadas. Testes realizados demonstram a superioridade do algoritmo proposto em termos de complexidade computacional.

5.1 Elaboração do algoritmo proposto

No Capítulo 3, foram estudados algoritmos de estado da arte para estimação de movimento, com resultados favoráveis para o algoritmo *Enhanced Adaptive Rood Pattern Search* (EARPS). Desta forma, a primeira possibilidade estudada para a melhoria do algoritmo de estimação de movimento do codificador Dirac foi a utilização deste método.

O algoritmo utilizado pelo Dirac para estimar movimento, como explicado no Capítulo 4, baseia-se em algumas etapas. A proposta de melhoria aqui apresentada contempla a etapa de busca dos vetores de movimento com precisão inteira de pixel. Esta é fundamental para o bom desempenho do *codec*, visto que as etapas seguintes baseiam-se nos seus resultados.

Como forma de aperfeiçoar ainda mais o desempenho da estimação de movimento para este codificador, foram realizadas alterações, baseando-se na eficiência da busca hierárquica realizada pelo Dirac:

- Implementação do algoritmo EARPS nos diversos níveis de busca, substituindo a busca computacionalmente custosa em cada nível utilizada pelo Dirac;
- Eliminação de um dos passos do algoritmo EARPS para tornar a busca ainda mais rápida em cada nível.

Desta forma, além do EARPS, detalhado no Capítulo 3, dois novos algoritmos foram implementados. A seguir, estes serão descritos em detalhes:

Hierarchical Enhanced Adaptive Rood Pattern Search (HEARPS)

Esta implementação realiza a busca hierárquica, utilizando em cada nível o padrão de busca do EARPS. Os vetores de movimento encontrados em cada nível são repassados ao nível seguinte (inferior), multiplicados por 2 – devido à mudança de resolução. O diagrama de blocos deste algoritmo é dado na Figura 5.1.

Assim, este novo algoritmo, denominado HEARPS, consiste nos seguintes passos (utiliza-se a mesma notação da seção 3.4, onde se descreve o algoritmo do EARPS):

1. Realizar subamostragem do quadro atual e dos quadros de referência. A profundidade máxima atingida nesta etapa é dada pela expressão:

$$depth = min\left[log_2\left(\frac{width}{12}\right), log_2\left(\frac{height}{12}\right)\right] \quad (5.1)$$

   onde *width* e *height* são os valores referentes ao comprimento e altura dos quadros analisados, respectivamente.
2. Para cada bloco, em *raster order*:
   2.1. Para cada nível (iniciando no mais profundo, até o próprio quadro), fazer:
      2.1.1. Cálculo de SADp (locais de busca: i) resultante do estimador e ii) resultante da busca no nível imediatamente superior multiplicado por 2). Se SADp < T, este vetor é atribuído ao bloco corrente e é salvo para utilização em próximos níveis, se nível atual não for zero.
      2.1.2. Cálculo de SAD0 (local (0,0)). Se SAD0 < T, este vetor é atribuído ao bloco corrente e é salvo para utilização em próximos níveis, se nível atual não for zero.

2.1.3. Compara-se SAD0 com SADp, e a posição que fornecer o menor SAD é setada como o ISC, de onde a busca seguirá.

2.1.4. Realizar busca utilizando URP, no ISC. Caso o ponto de menor distorção encontrado nesta etapa (MBD) tenha SAD menor que T ou se localize no ISC, este será atribuído ao bloco corrente e é salvo para utilização em próximos níveis, se nível atual não for zero.

2.1.5. A busca continua a partir do MBD encontrado neste passo, realizando busca com padrão adaptativo. Caso o MBD encontrado nesta etapa tenha SAD menor que T ou se localize no centro do E-ARP, este será atribuído ao bloco corrente e é salvo para utilização em próximos níveis, se nível atual não for zero.

2.1.6. Realizar busca utilizando URP, no MBD atual. Realizar a busca repetidamente até que o MBD encontrado tenha SAD menor que T ou se localize no centro do URP. O vetor encontrado será atribuído ao bloco corrente e é salvo para utilização em próximos níveis, se nível atual não for zero.

Figura 5.1: Diagrama de blocos do HEARPS.

Modified Hierarchical Enhanced Adaptive Rood Pattern Search (MHEARPS)

Esta implementação modifica o HEARPS pela supressão do quinto passo da busca do EARPS. Isto é realizado em decorrência da pouca utilidade que ele apresenta: após a realização dos passos 2.1.1 a 2.1.5, a continuidade da busca não provê melhoria de resultados muito significativa. O diagrama de blocos deste algoritmo é dado na Figura 5.2.

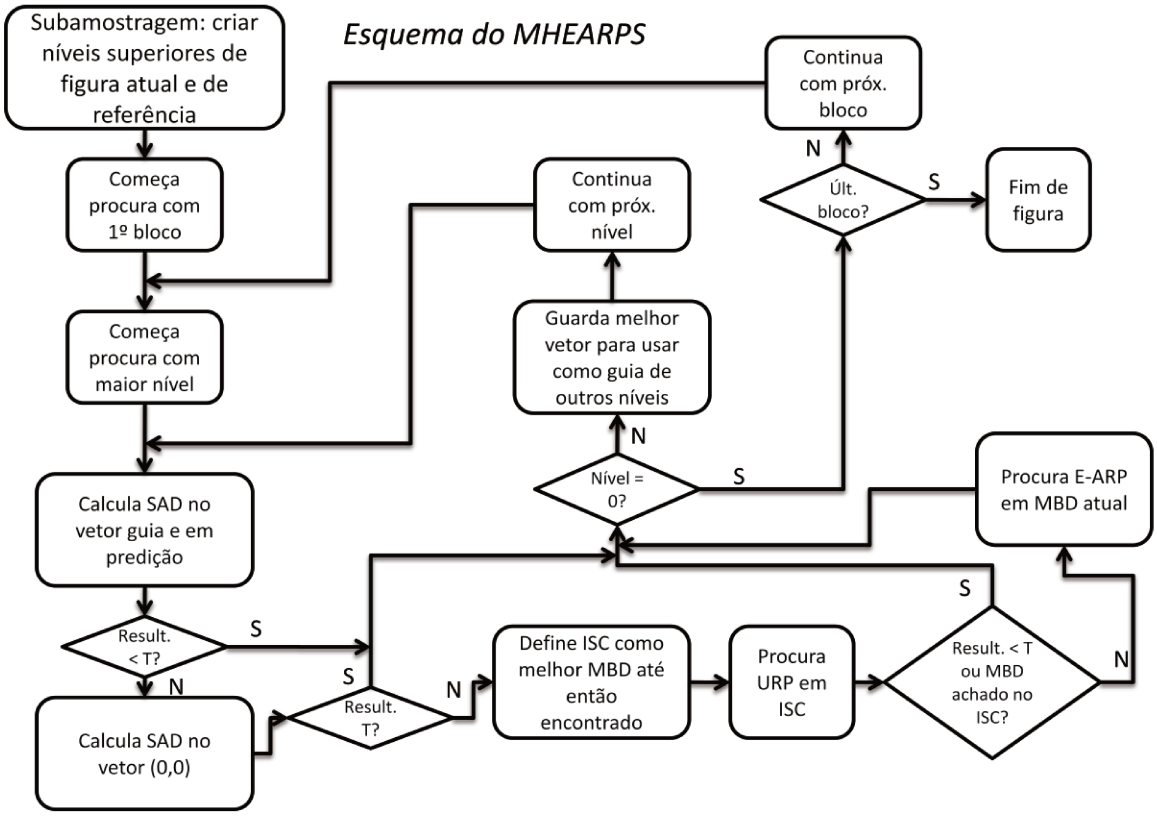


Figura 5.2: Diagrama de blocos do MHEARPS

Desta forma, este algoritmo, ao qual se denominou MHEARPS, consiste em:

1. Realizar subamostragem do quadro atual e dos quadros de referência. A profundidade máxima atingida nesta etapa é dada pela expressão 5.1.
2. Para cada bloco, em *raster order*:
    2.1. Para cada nível (iniciando no mais profundo, até o próprio quadro), fazer:
        2.1.1. Cálculo de SADp (locais de busca: i) resultante do estimador e ii) resultante da busca no nível imediatamente superior multiplicado por 2). Se SADp < T, este vetor é atribuído ao bloco corrente e é salvo para utilização em próximos níveis, se nível atual não for zero.
        2.1.2. Cálculo de SAD0 (local (0,0)). Se SAD0 < T, este vetor é atribuído ao bloco corrente e é salvo para utilização em próximos níveis, se nível atual não for zero.

2.1.3. Compara-se SAD0 com SADp, e a posição que fornecer o menor SAD é setada como o ISC, de onde a busca seguirá.

2.1.4. Realizar busca utilizando URP, no ISC. Caso o ponto de menor distorção encontrado nesta etapa (MBD) tenha SAD menor que T ou se localize no ISC, este será atribuído ao bloco corrente e é salvo para utilização em próximos níveis, se nível atual não for zero.

2.1.5. A busca continua a partir do MBD encontrado neste passo, realizando busca com padrão adaptativo. Vetor encontrado é salvo para utilização em próximos níveis, se nível atual não for zero.

Detalhamento da busca hierárquica realizada pelo Dirac

O algoritmo hierárquico de busca utilizado pelo Dirac já foi apresentado na seção 4.3. Entretanto, para que se possa comparar em detalhes com os outros algoritmos aqui propostos, detalhar-se-á este método. O padrão de busca aplicado em cada nível é quadrado; todos os vetores no interior ou no limite deste quadrado têm seu SAD calculados. O diagrama de blocos da busca hierárquica do Dirac é dado na Figura 5.3.

Desta forma, este algoritmo consiste em:

1. Realizar subamostragem do quadro atual e dos quadros de referência. A profundidade máxima atingida nesta etapa é dada pela expressão 5.1.
2. Para cada bloco, em *raster order*:
   2.1. Para cada nível (iniciando no mais profundo, até o próprio quadro):
      2.1.1. Calcular o tamanho do lado do quadrado utilizado como padrão de busca

$$m_{xr} = 1 + 2(min[(level + 1)\,, 5]) \qquad (5.2)$$

      onde m_{xr} é o lado do quadrado utilizado como padrão de busca, e *level* o nível corrente.

      2.1.2. Calcular o SAD para todos os vetores presentes nas três listas seguintes (cada uma destas contém todos os vetores possíveis na região quadrada):

         i. Vetores centrados na posição (0,0);

ii. Vetores centrados na posição indicada pela predição espacial feita por mediana;

iii. Vetores centrados na posição indicada pelo vetor guia.

2.1.3. Armazenar melhor vetor encontrado para utilização no nível inferior subseqüente (caso não se esteja no nível 0).

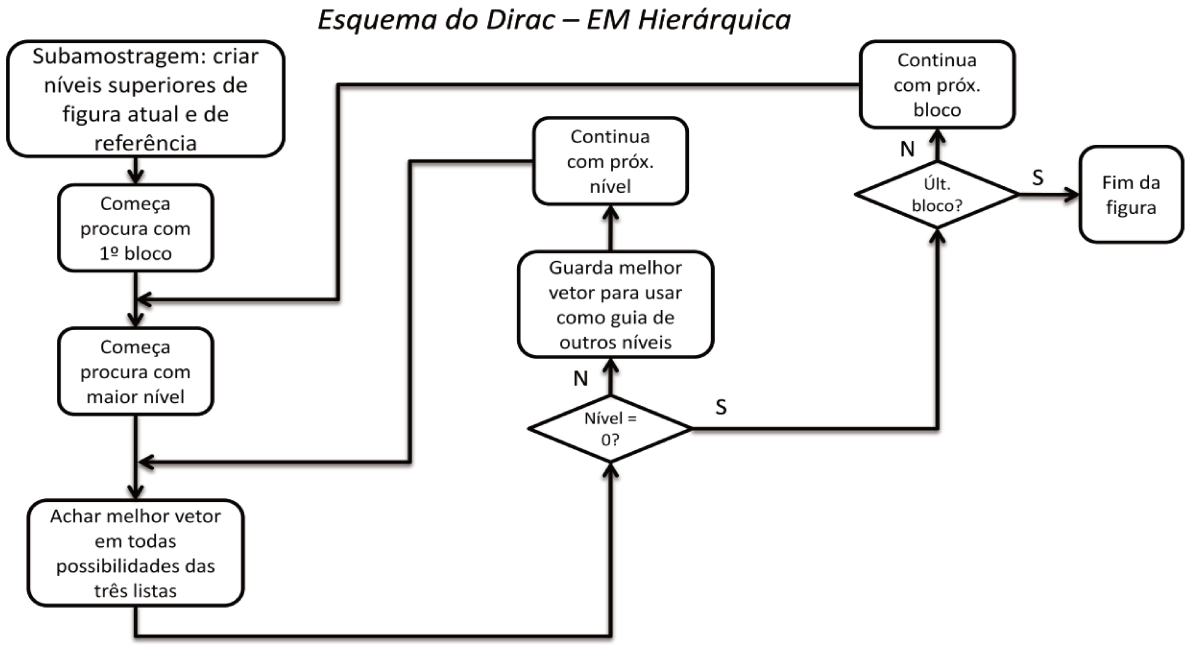


Figura 5.3: Diagrama de blocos da busca hierárquica realizada pelo Dirac.

Complexidade dos novos algoritmos implementados

A complexidade dos algoritmos apresentados é avaliada, neste caso, pelo número da quantidade de cálculos de SAD, já que o número de operações realizadas em cada um destes cálculos é constante.

Na sequência, analisa-se os melhores e piores casos da quantidade de cálculos de SAD por bloco que podem ser obtidos com cada um dos algoritmos utilizados na comparação (assume-se que apenas uma referência é utilizada; no caso de existirem duas, é suficiente a multiplicação dos resultados abaixo por 2):

- EARPS
  - **Melhor caso: 1 cálculo de SAD por bloco.**

Busca procede sem subamostragem, utilizando quadros e referências originais. A comparação com o limiar permite o encerramento rápido do algoritmo.

- **Pior caso: Ilimitado.**

  Como o último passo determina que a busca seja concluída apenas com a condição de limiar satisfeita ou se o melhor vetor encontrado estiver no mesmo local do melhor vetor encontrado no passo precedente, não há limites para este número de operações.

- HEARPS
  - **Melhor caso: Dependente do número de níveis.**

    Busca em níveis, mas com terminação rápida. No caso do primeiro bloco, no nível mais profundo o melhor caso é o de terminação rápida quando comparado com (0,0). Neste mesmo bloco, nos outros níveis, o melhor caso é de terminação rápida quando comparado com o vetor guia.

    352x288 (CIF): 5 cálculos de SAD por bloco.

    720x576 (SDTV 576p): 6 cálculos de SAD por bloco.

    352x288 (HDTV 720p): 6 cálculos de SAD por bloco.
  - **Pior caso: Ilimitado.**

    Como o último passo determina que a busca seja concluída apenas com a condição de limiar satisfeita ou se o melhor vetor encontrado estiver no mesmo local do melhor vetor encontrado no passo precedente, não há limites para este número de operações, em nenhum nível.
- MHEARPS
  - **Melhor caso: Dependente do número de níveis.**

    Busca em níveis, mas com terminação rápida. No caso do primeiro bloco, no nível mais profundo o melhor caso é o de terminação rápida quando comparado com (0,0). Neste mesmo bloco, nos outros níveis, o melhor caso é de terminação rápida quando comparado com o vetor guia.

    Para os tamanhos de quadro utilizados:

    352x288 (CIF): 5 cálculos de SAD por bloco.

    720x576 (SDTV 576p): 6 cálculos de SAD por bloco.

352x288 (HDTV 720p): 6 cálculos de SAD por bloco.

- **Pior caso: Dependente do número de níveis.**

  Com este algoritmo, garante-se uma quantidade limitada de cálculos de SAD por nível, podendo-se estimar o melhor caso.

  352x288 (CIF): 55 cálculos de SAD por bloco.

  720x576 (SDTV 576p): 66 cálculos de SAD por bloco.

  352x288 (HDTV 720p): 66 cálculos de SAD por bloco.

- Dirac Hierárquico
  - **Melhor caso: Dependente da resolução.**

    Busca em níveis, com padrão quadrado cujo tamanho do lado depende diretamente do nível. Melhor caso atingido quando três listas analisadas são idênticas:

    352x288 (CIF): 285 cálculos de SAD por bloco.

    720x576 (SDTV 576p): 406 cálculos de SAD por bloco.

    352x288 (HDTV 720p): 406 cálculos de SAD por bloco.
  - **Pior caso: Dependente da resolução.**

    Pior caso atingido quando três listas analisadas são inteiramente diferentes:

    352x288 (CIF): 855 cálculos de SAD por bloco.

    720x576 (SDTV 576p): 1218 cálculos de SAD por bloco.

    352x288 (HDTV 720p): 1218 cálculos de SAD por bloco.

A partir destes cálculos, observa-se que o Dirac Hierárquico apresenta complexidades mínimas e máximas altas quando comparado aos outros algoritmos propostos. Para tirar qualquer conclusão, no entanto, deve-se realizar testes para observar o comportamento comparativo dos métodos analisados.

5.2 Testes, resultados e discussão

Os quatro algoritmos analisados foram usados no codificador Dirac para comprimir 8 vídeos (os mesmos do Capítulo 4), compreendendo diferentes graus e formas de movimento com vários tamanhos, como já apresentado na Tabela 4.4.

Como comentado no Capítulo 4, estes estão disponíveis em formato RGB em (36) e foram convertidos para o formato YUV, com amostragem de crominância 4:2:2 antes da compressão. Cada vídeo foi comprimido diversas vezes com o mesmo método para diferentes taxas, com o objetivo de analisar o desempenho restringido a cada taxa.

O codificador usado é a versão de pesquisa para o Dirac, implementado em C++, versão 1.0.2 (37). As simulações foram realizadas utilizando um PC equipado com Intel Pentium M 740, 1,73GHz, 768MB RAM.

Mais uma vez, o codificador Dirac foi utilizado com seus parâmetros de referência, como recomendado pelos autores do software. Cada formato de vídeo implica diversos presets, ajustes nos parâmetros para a adequação a cada formato.

Da mesma forma, a comparação de qualidade de reconstrução foi realizada a partir das avaliações objetivas do PSNR e do MSSIM. Eles foram computados para os coeficientes de luminância de cada quadro em cada vídeo e os valores médios de todo o vídeo foram levados em conta.

Desta vez, a principal medida de complexidade computacional de cada codificador é o número de cálculos de SAD por quadro. Ademais, é também provida a quantidade de tempo média decorrida por quadro para cada vídeo.

Para os três algoritmos baseados no EARPS, foi utilizado o mesmo valor de limiar: 512. Este foi considerado como um bom compromisso para obter uma boa qualidade visual, como utilizado para o algoritmo EARPS (28).

Os resultados obtidos são dados no Apêndice III: são apresentadas as curvas de qualidade de reconstrução (PSNR e MSSIM) por taxa de *bits*, assim como a tabela onde todos os resultados são encontrados. A Tabela 5.1 apresenta resultados sumarizados acerca da complexidade computacional.

No que se refere ao desempenho em termos de qualidade visual, nota-se que os algoritmos MHEARPS, HEARPS e Hierárquico apresentam desempenho bastante similar em todos os casos. O EARPS é nitidamente inferior em alguns casos, como para os vídeos snow352, snow720 e

snow1280, atingindo até 2dB a menos que os outros. Isto pode ser justificado pela observação de que o EARPS é o único algoritmo que não implementa a busca hierárquica, ou seja, não consegue realizar a busca em localidades mais distantes, o que é realmente imprescindível para os vídeos snow, que apresentam grande movimentação.

Constata-se também, pela tabela fornecida no Apêndice III, que os algoritmos MHEARPS, HEARPS e EARPS apresentam complexidade computacional muito inferior à do algoritmo Hierárquico padrão do Dirac.

Tabela 5.1: Principais resultados em termos de complexidade computacional para os testes realizados.

Vídeo	**Dirac – Média de SAD/quadro (% comp. Dirac)**	**EARPS - Média de SAD/quadro (% comp. Dirac)**	**HEARPS - Média de SAD/quadro (% comp. Dirac)**	**MHEARPS - Média de SAD/quadro (% comp. Dirac)**
canal352	117.444 (100)	16.440 (14)	19.736 (17)	18.743 (16)
snow352	111.717 (100)	18.508 (17)	20.822 (19)	18.859 (17)
squirrel352	95.590 (100)	23.258 (24)	28.603 (30)	25.419 (27)
canal720	418.337 (100)	45.869 (11)	59.098 (14)	58.561 (14)
snow720	520.380 (100)	110.412 (21)	99.410 (19)	86.054 (17)
squirrel720	368.698 (100)	96.677 (26)	110.617 (30)	94.647 (26)
snow1280	490.278 (100)	126.084 (26)	115.891 (24)	97.146 (20)
squirrel1280	368.308 (100)	104.376 (28)	120.882 (33)	104.451 (28)
Média global de percentagem	***100%***	***22%***	***23%***	***21%***

O algoritmo que provê melhores resultados é o MHEARPS, reduzindo o número de cálculos de SAD em média 79%, atingindo em alguns casos 86%. Ou seja, uma redução drástica na complexidade computacional do Dirac, sem perder qualidade visual. Esta redução implica, naturalmente, na redução também significativa em termos de tempo de processamento médio por quadro, como observado na referida tabela.

O algoritmo HEARPS apresenta resultados pouco piores, atingindo 77% de economia de cálculos de SAD, com pico de 86%. Neste momento, fica claro que o passo 2.5 do HEARPS não é necessário: os vetores de movimento neste ponto têm pouca chance de melhorias, devido ao extenso processo de busca hierárquica adicionado ao algoritmo. A exclusão deste também

implica na limitação do número de operações para o pior caso, o que é importante para evitar o excesso de computações. O algoritmo MHEARPS é a principal contribuição deste trabalho.

Capítulo 6

Conclusão

O foco deste trabalho é a codificação de vídeo, técnica de grande importância para os sistemas de telecomunicações modernos, nos quais se observa um aumento significativo do tráfego deste tipo de sinal. A codificação, que viabiliza a compressão do vídeo, torna-se indispensável para que os recursos de armazenamento e transmissão limitados sejam utilizados com eficiência.

A principal contribuição aportada neste trabalho é a proposta de um algoritmo mais eficiente para a realização de estimação de movimento no codificador do *codec* Dirac. O método (*Modified Hierarchical Enhanced Adaptive Rood Pattern Search* - MHEARPS) proposto traz ganhos evidentes em termos de complexidade computacional, atingindo em média 79% de redução de cálculos de comparação de blocos, uma das operações mais dispendiosas de um codificador de vídeo. É crucial que uma qualidade boa seja atingida com taxas de bits aceitáveis, com o mínimo de processamento possível – e este trabalho contribui com a busca por esta eficiência.

Inicialmente, estudam-se comparativamente técnicas de estimação de movimento recentes, ilustrando formas de estado da arte de realizar esta operação, essencial a qualquer *codec* de vídeo. Num segundo momento, analisa-se o *codec* Dirac, recentemente lançado, baseado na Transformada Wavelet, e com diversas características bem particulares. Por fim, o algoritmo é proposto com o objetivo de tornar mais eficiente a estimação de movimento do Dirac.

Na primeira parte do trabalho, constatou-se o desempenho superior de um algoritmo simples e eficaz, o *Enhanced Adaptive Rood Pattern Search* (EARPS) (28). Quando comparado a outros métodos de estado da arte (29)(30), baseados em outras formas de estimar o movimento, fica claro que o compromisso obtido com este é mais interessante.

O *codec* Dirac é estudado na sequência, e é avaliado comparativamente ao H.264, um dos mais eficientes *codecs* de vídeo da atualidade. Esta análise difere significativamente das que estão disponíveis na literatura atualmente (22)(23)(24)(25), que não levam em conta diversos parâmetros cruciais para uma comparação justa, como o Grupo de Figuras a ser implementado, ou o algoritmo de estimação de movimento que se utiliza. Caso estes parâmetros não sejam

igualmente configurados nos dois codificadores, não será obtida uma comparação justa. É também imprescindível a utilização de algumas sequências de vídeo, com diferentes graus e tipos de movimento.

Observa-se que o Dirac não alcança a performance do H.264 em termos da avaliação objetiva, com o uso do PSNR e do MSSIM. Entretanto, é importante mencionar que foram constatados indícios de que uma avaliação subjetiva pode ser favorável ao Dirac. Ademais, o Dirac se apresenta em média 60% mais rápido que o H.264. A inclusão de mais algumas funcionalidades neste *codec* nos próximos anos deverá aperfeiçoá-lo cada vez mais: estão previstas as implantações de codificação de região de interesse, modo *skip*, modelagem de movimento global, dentre outras.

Outro ponto importante a ressaltar é o fato de o Dirac ser um *codec* livre para qualquer utilização, aberto, não exigindo o pagamento de *royalties*. O Dirac pretende não infringir nenhuma patente de terceiros, o que permitirá seu uso público sem restrições. Isto contrasta com o H.264, cujo detentor da patente, MPEG LA, começará a exigir o pagamento de *royalties* já em 2011.

Na última parte do trabalho, objetiva-se agregar o conhecimento acumulado nas duas etapas anteriores visando a propor um algoritmo mais eficiente de estado da arte para o *codec* Dirac. As modificações sucessivas que levaram ao desenvolvimento do *Modified Hierarchical Enhanced Adaptive Rood Pattern Search* (MHEARPS) são explicitadas no Capítulo 5. O desempenho superior deste é comprovado pelos testes realizados. Esta constitui a principal contribuição desta dissertação.

Observa-se na literatura de referência alguns artigos recentes acerca de uma melhoria na estimação de movimento do Dirac, atingindo até 60% de ganho em termos de cálculos de SAD (26)(27). Este ganho se efetiva de forma relativamente diferente da que é empregada nesta dissertação. Uma possível conjugação destas duas abordagens pode resultar em um ganho ainda mais significativo para o codificador Dirac em termos de complexidade computacional.

Outras publicações de extrema relevância bastante recentes (46)(47) contribuem enormemente com a compreensão de algoritmos de estimação de movimento baseados em padrões de busca (*Pattern-based Block Motion Estimation*, PBME), por detalhar um modelo estatístico que leva em conta as características intrínsecas dos métodos PBME. Apesar de o

algoritmo aqui proposto não ser exatamente um PBME, a extensão deste tipo de resultados para agregar a busca hierárquica tem grande potencial de importante contribuição.

6.1 Sugestões para trabalhos futuros

Alguns pontos explorados neste trabalho apresentam potencial para continuidade, por haver grande possibilidade de se obter resultados interessantes.

Com relação à comparação entre o Dirac e o H.264, faz-se cada vez mais necessária a realização de uma comparação conforme métodos subjetivos de avaliação. Os indícios aqui levantados acerca de um possível desempenho superior do Dirac devem ser checados.

Outra possibilidade de trabalho futuro é a utilização do MHEARPS com modificações introduzidas por (26) e por (27) (implementação de estimação semi-hierárquica, utilização de vetores de movimento de blocos temporalmente vizinhos, dentre outras). Pelo fato de este e os outros trabalhos otimizarem a busca comparativa de estimação de movimento baseados em princípios relativamente diferentes, há potencial para uma junção destes métodos objetivando melhorar ainda mais a performance global do codificador.

Ademais, como já dito, a extensão dos métodos de análise e *design* de algoritmos de estimação de movimento baseados em (47) e (46) tem grande potencial de aportar uma abordagem compreensiva de extrema relevância para algoritmos PBME com busca hierárquica, como o MHEARPS.

Apêndice I

Resultados da seção 4.5 – Análise comparativa: Dirac x H.264

- Curvas de desempenho comparativo:

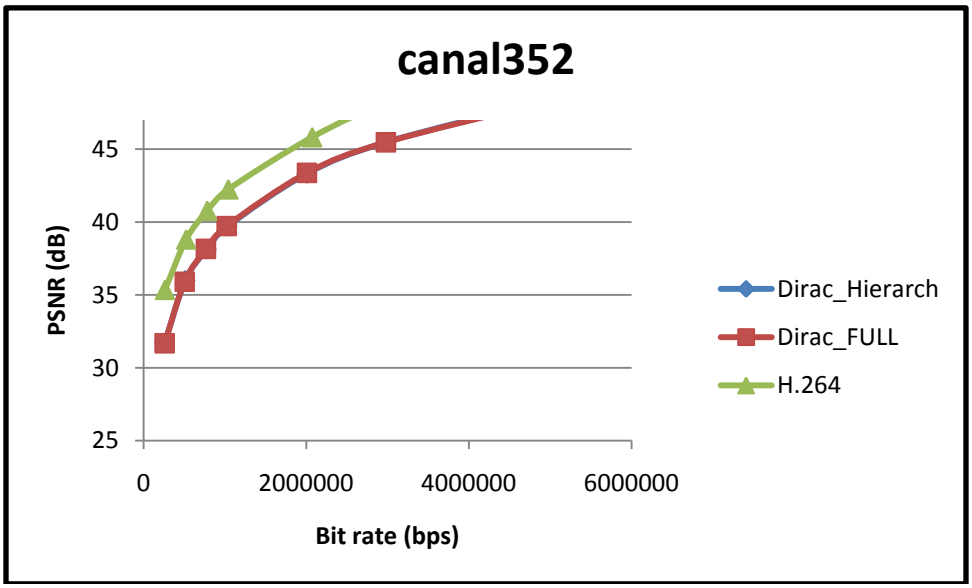


Figura I. 1: Curva de desempenho comparativo (PSNR x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo canal352.

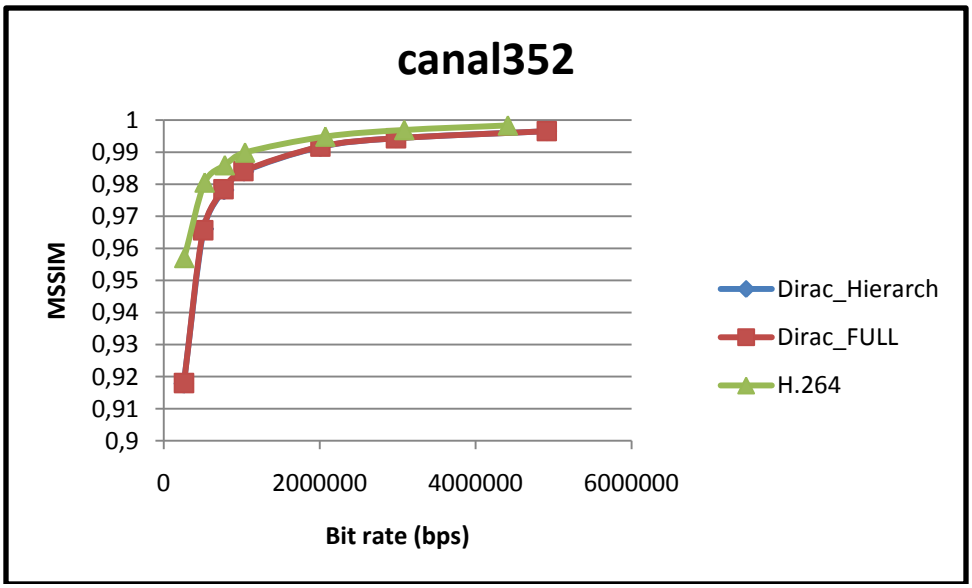


Figura I. 2: Curva de desempenho comparativo (MSSIM x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo canal352.

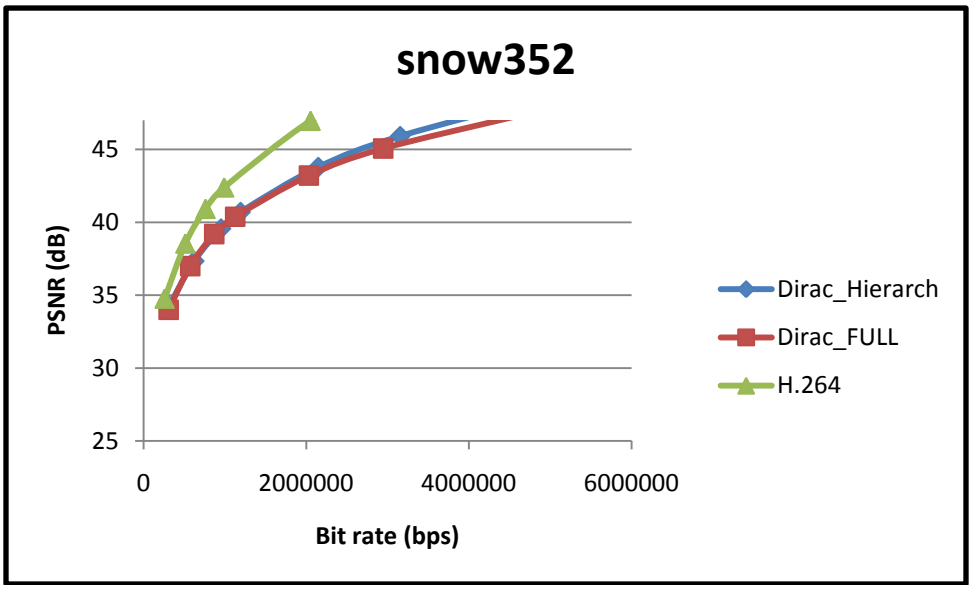


Figura I. 3: Curva de desempenho comparativo (PSNR x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo snow352.

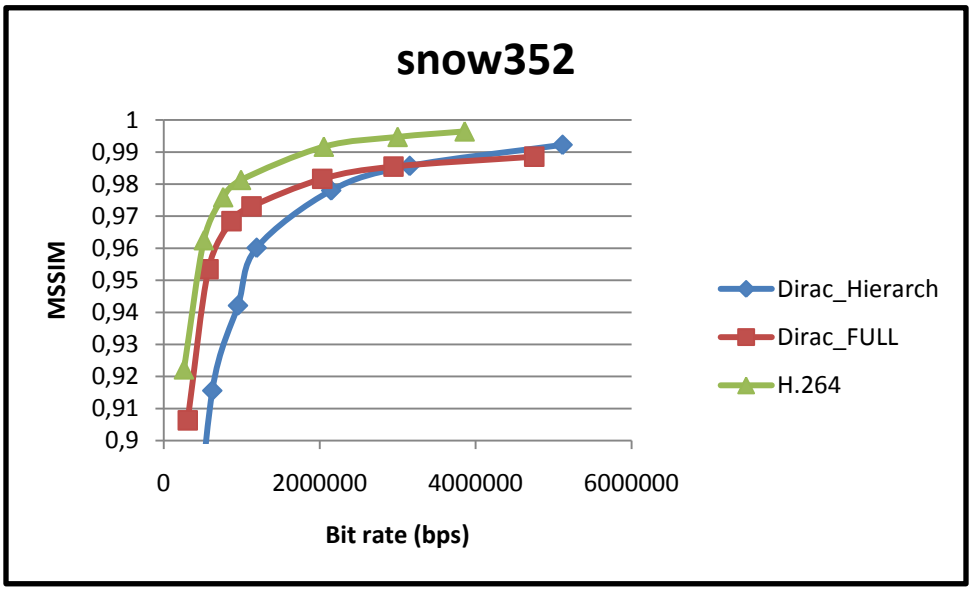


Figura I. 4: Curva de desempenho comparativo (MSSIM x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo snow352.

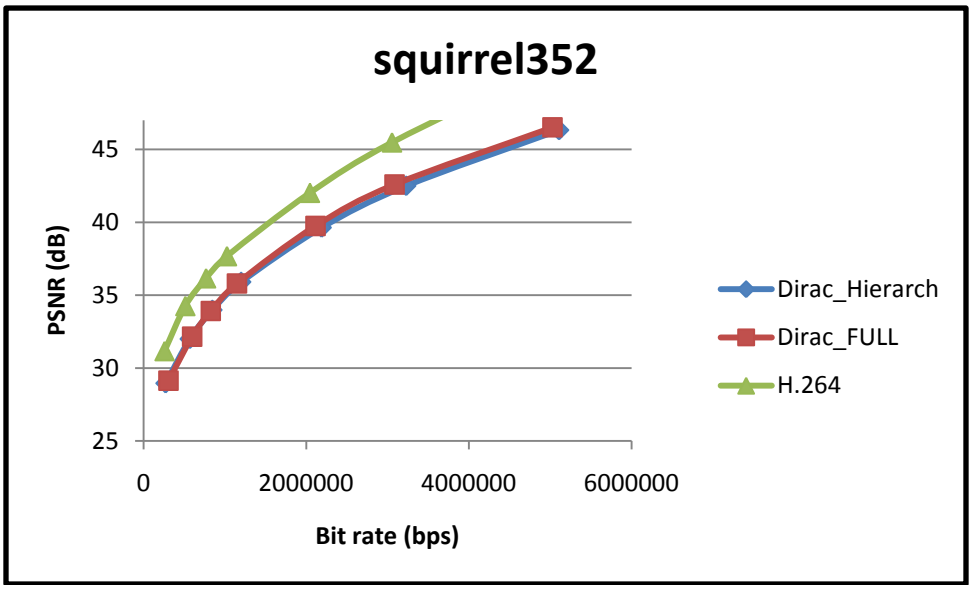


Figura I. 5: Curva de desempenho comparativo (PSNR x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo squirrel352.

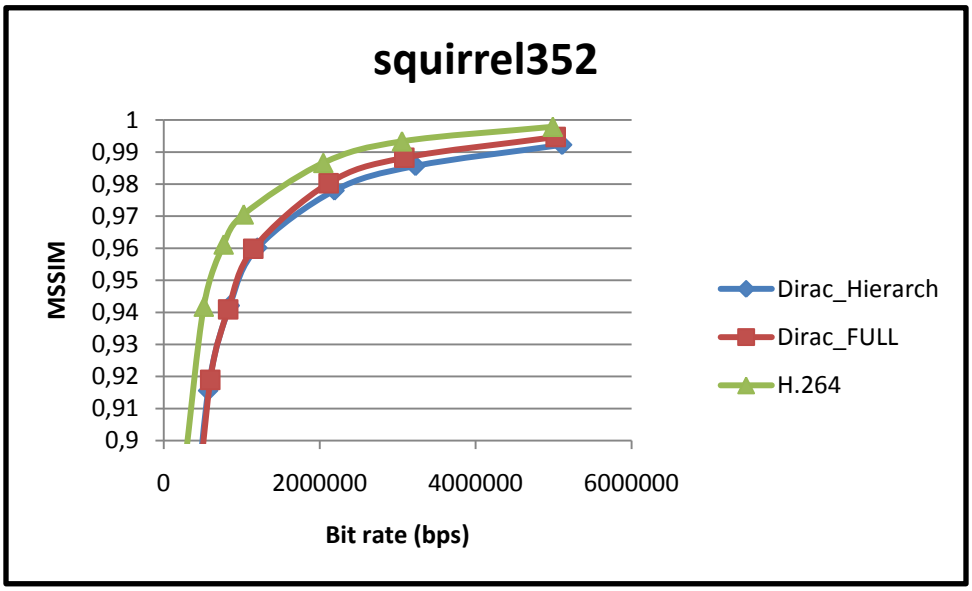


Figura I. 6: Curva de desempenho comparativo (MSSIM x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo squirrel352.

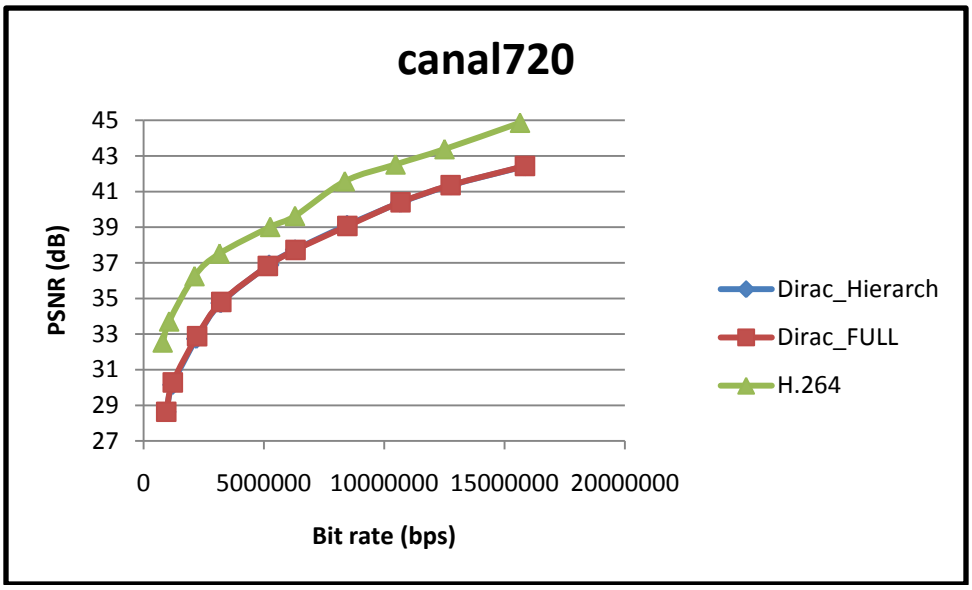


Figura I. 7: Curva de desempenho comparativo (PSNR x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo canal720.

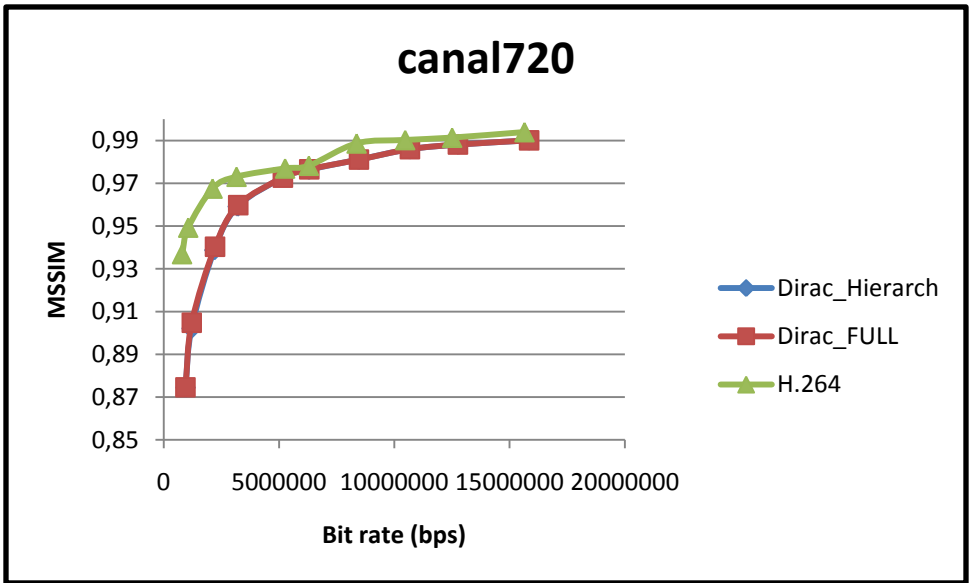


Figura I. 8: Curva de desempenho comparativo (MSSIM x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo canal720.

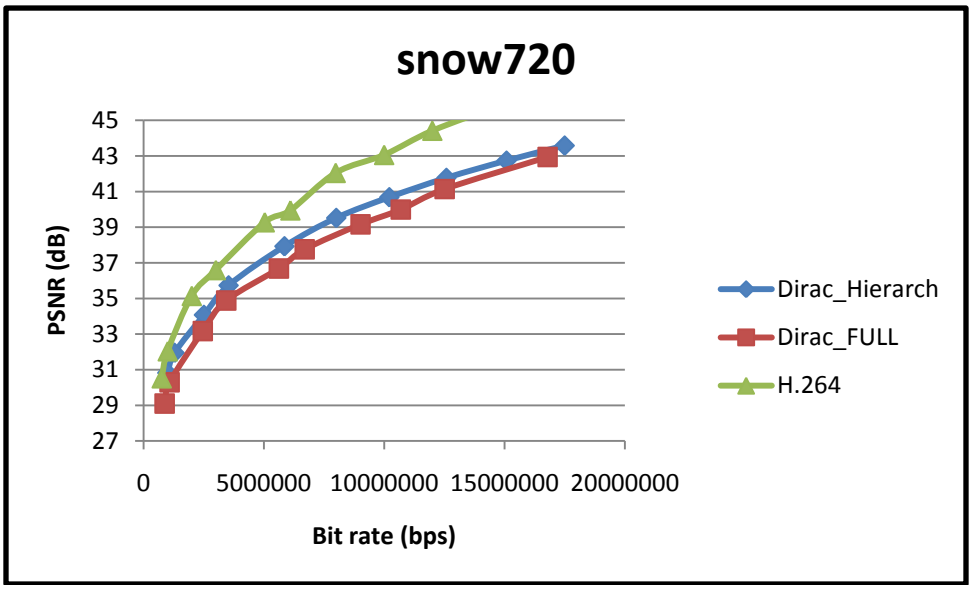


Figura I. 9: Curva de desempenho comparativo (PSNR x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo snow720.

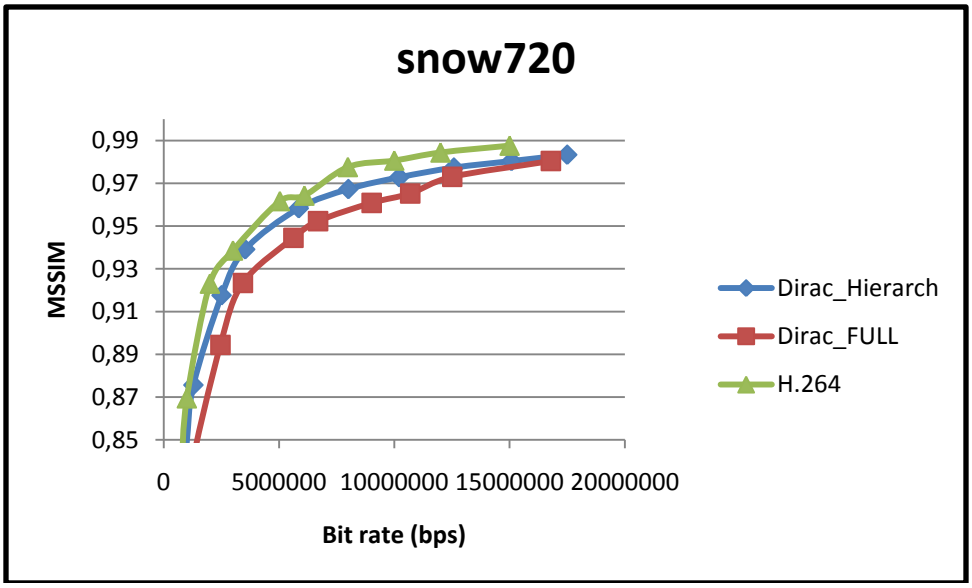


Figura I. 10: Curva de desempenho comparativo (MSSIM x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo snow720.

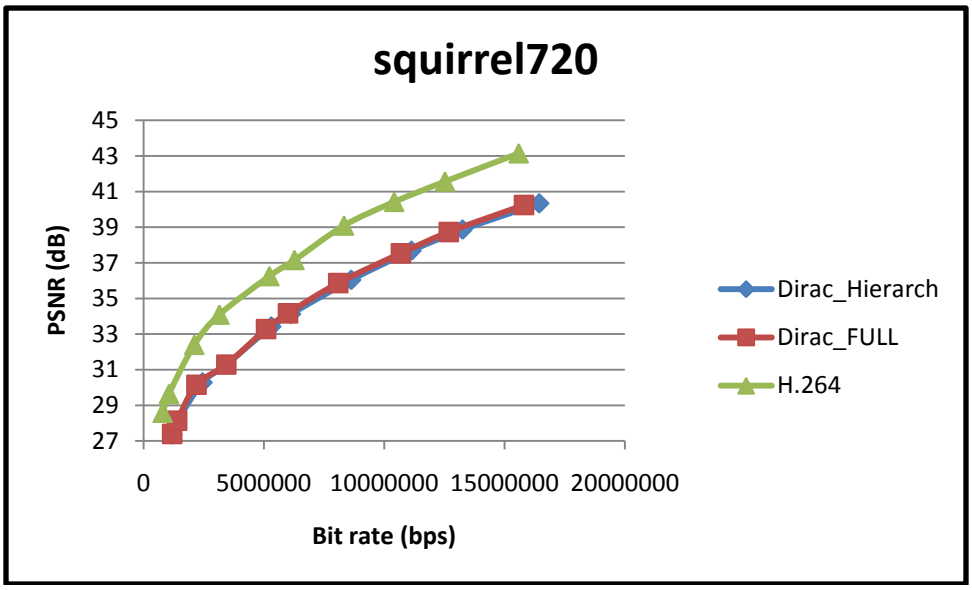


Figura I. 11: Curva de desempenho comparativo (PSNR x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo squirrel720.

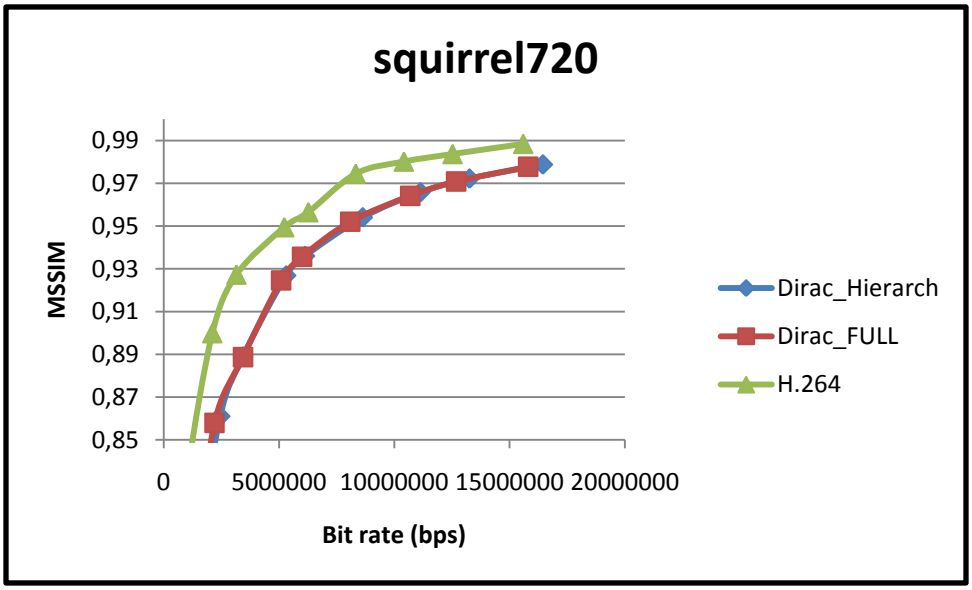


Figura I. 12: Curva de desempenho comparativo (MSSIM x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo squirrel720.

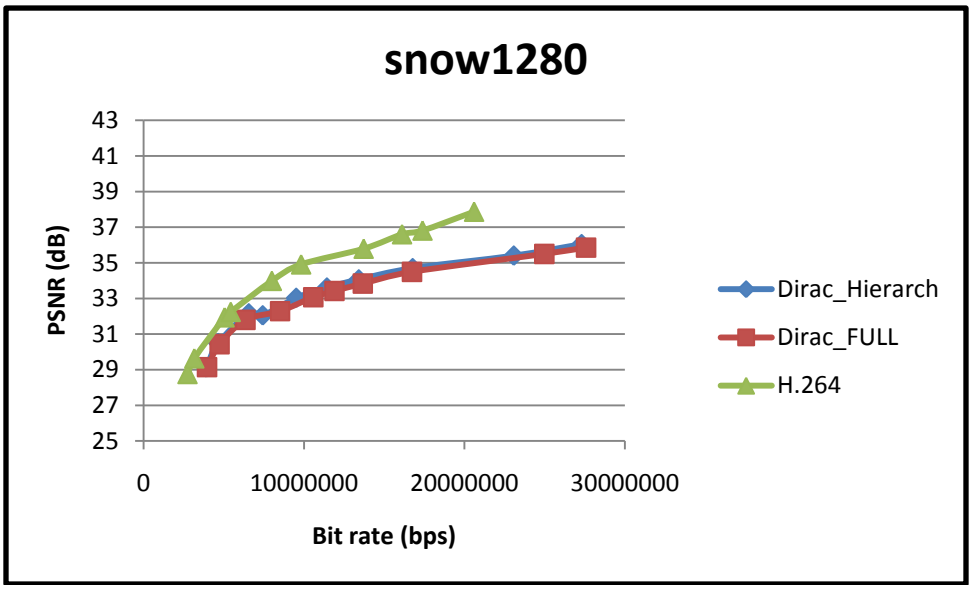


Figura I. 13: Curva de desempenho comparativo (PSNR x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo snow1280.

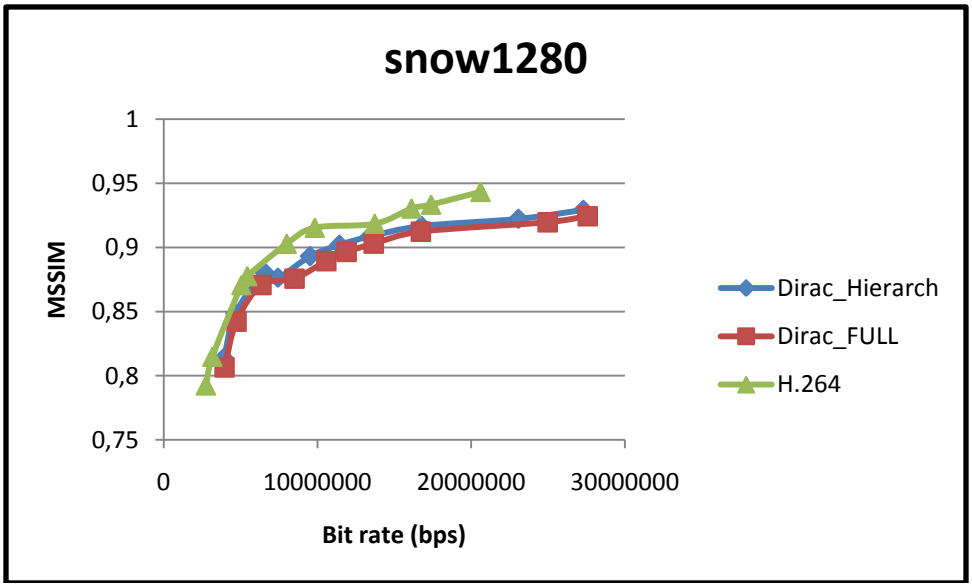


Figura I. 14: Curva de desempenho comparativo (MSSIM x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo snow1280.

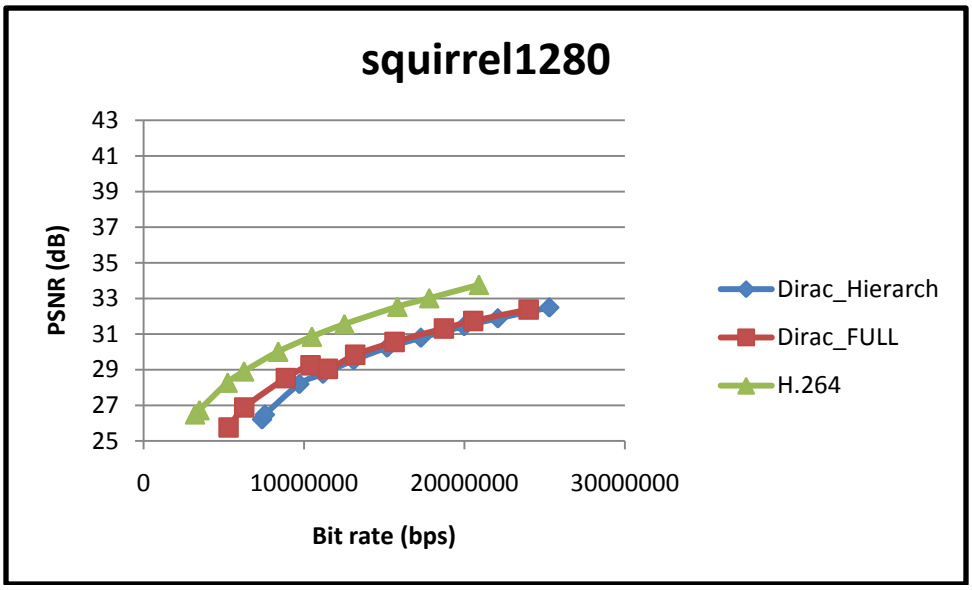


Figura I. 15: Curva de desempenho comparativo (PSNR x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo squirrel1280.

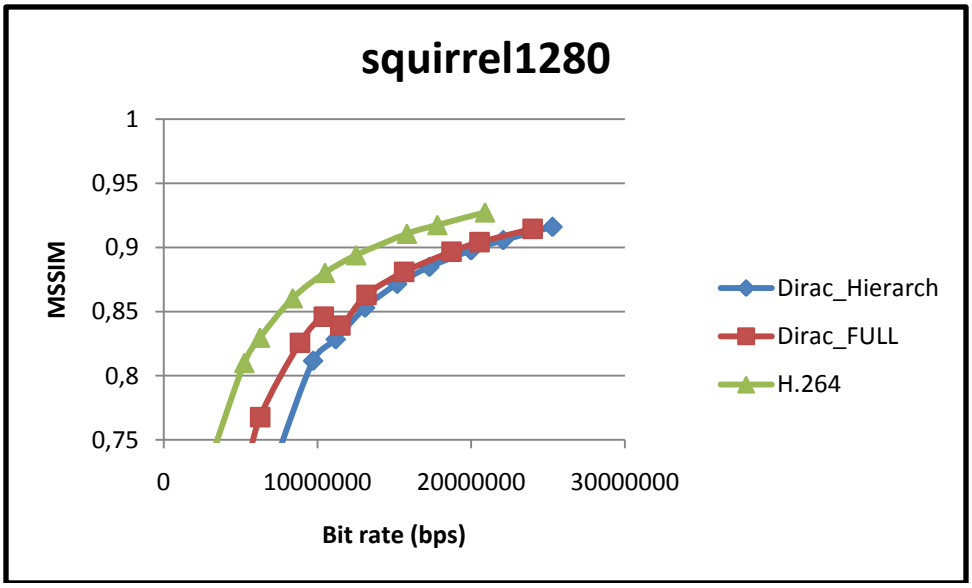


Figura I. 16: Curva de desempenho comparativo (MSSIM x Taxa de bits): Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - vídeo squirrel1280.

Tabela I. 1: Resultados obtidos com os algoritmos Dirac (EM Hierárquica), Dirac (FS) e H.264 (FS) - seção 4.5.

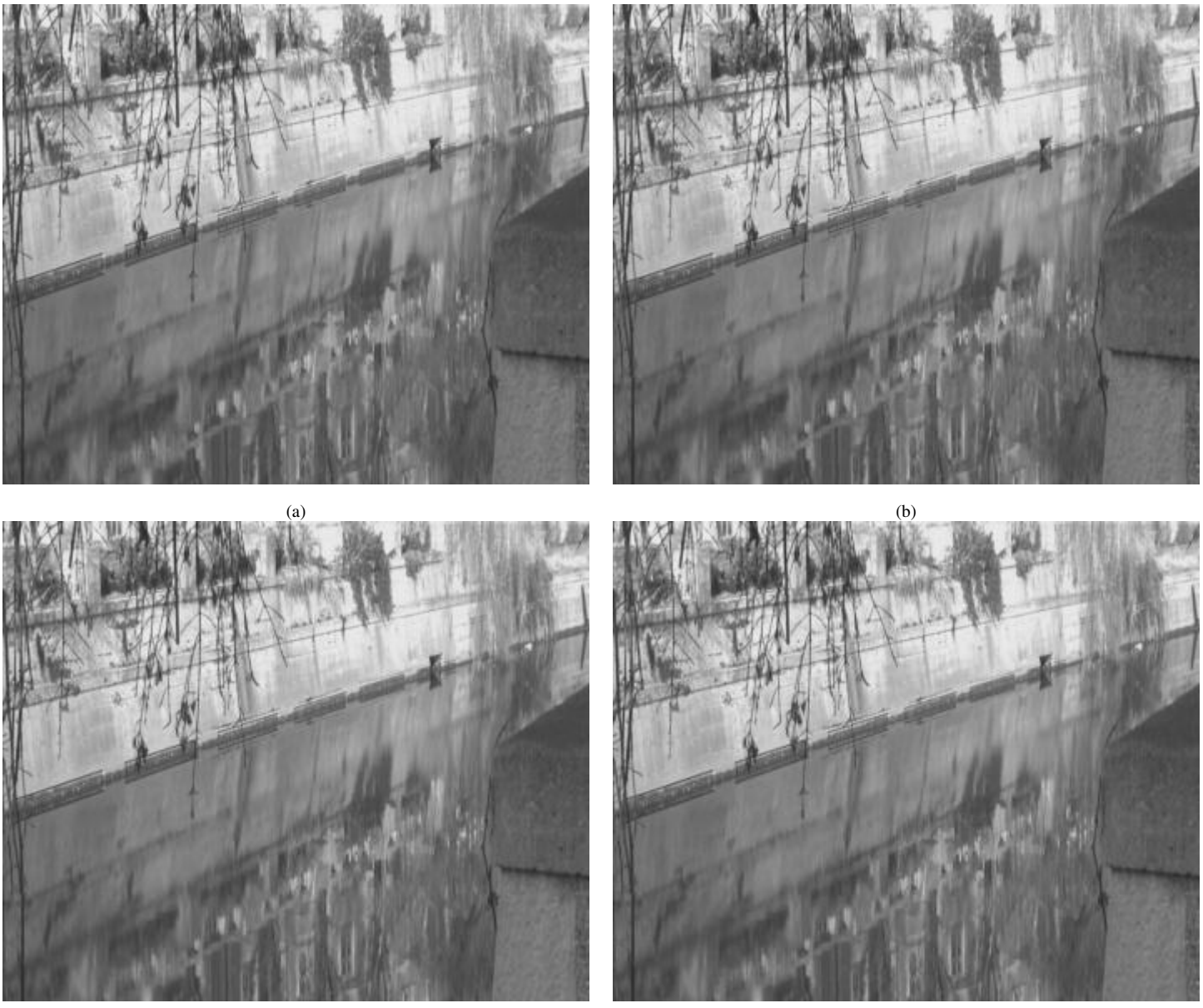
Video	Target Bit rate (bps)	Dirac Bit Rate (bps)	Dirac PSNR (dB)	Dirac MSSIM	Dirac tempo/quadro (s)	Dirac FULL Bit Rate (bps)	Dirac FULL PSNR (dB)	Dirac FULL MSSIM	Dirac FULL tempo/quadro (s)	H264 Bit rate (bps)	H264 PSNR (dB)	H264 MSSIM	H264 tempo/quadro (s)
canal352	250	255895	31,6874	0,917847	0,30675	258305	31,684	0,917998	20,8874	261850	35,3495	0,957	37,189
canal352	500	512466	35,991	0,966072	0,309735	504836	35,903	0,965666	20,8518	522800	38,8093	0,980534	37,11833824
canal352	750	772011	38,1141	0,978209	0,312956	766182	38,1578	0,978461	20,8977	781790	40,7836	0,985944	37,39633824
canal352	1000	1027794	39,6959	0,983978	0,317103	1021466	39,7258	0,984119	20,9343	1041000	42,2477	0,989896	37,57826471
canal352	2000	2006892	43,3238	0,991658	0,327662	2008158	43,3782	0,991736	21,0648	2071640	45,8287	0,994873	38,45270588
canal352	3000	2981827	45,481	0,994366	0,338691	2978729	45,4706	0,994356	20,9216	3083790	48,5524	0,996929	39,07220588
canal352	5000	4914011	48,2936	0,996567	0,357074	4912933	48,2052	0,996564	20,9632	4412300	51,7457	0,99835	39,77495588
snow352	250	310346	34,273	0,840163	0,255827	306948	34	0,906322	19,53	258150	34,7672	0,922239	38,59825333
snow352	500	622184	37,3441	0,915587	0,31396	573174	36,9968	0,953459	22,1296	513260	38,5506	0,962521	38,34169333
snow352	750	951412	39,5734	0,942129	0,319373	867668	39,1919	0,96846	22,0935	761130	40,9298	0,975944	38,41756
snow352	1000	1191361	40,7086	0,960153	0,323747	1125161	40,383	0,973041	22,1	990070	42,3997	0,981317	38,54270667
snow352	2000	2147089	43,7932	0,978036	0,334787	2032101	43,2107	0,981647	22,2519	2052250	46,9719	0,991692	39,19614667
snow352	3000	3153932	45,9149	0,985699	0,345413	2946234	45,0645	0,985477	22,2015	2999400	49,6229	0,994733	39,63950667
snow352	5000	5114361	48,8415	0,992278	0,3648	4751090	47,4979	0,988569	22,1713	3858940	51,7874	0,996455	40,14714667
squirrel352	250	270136	28,954	0,840163	0,229373	304980	29,1368	0,848129	19,1594	256990	31,1624	0,889947	39,05725333
squirrel352	500	567936	31,9991	0,915587	0,295627	595925	32,1558	0,918966	21,6323	512280	34,2555	0,941799	38,99753333
squirrel352	750	845269	33,9863	0,942129	0,29896	825721	33,9163	0,940923	21,574	767360	36,1589	0,961189	39,04504
squirrel352	1000	1199354	35,9093	0,960153	0,304787	1147806	35,807	0,95985	20,9254	1024490	37,6771	0,970571	39,21976
squirrel352	2000	2188674	39,6426	0,978036	0,316467	2113822	39,7545	0,980295	20,9735	2045410	42,0217	0,986634	39,86042667
squirrel352	3000	3228101	42,5054	0,985699	0,327493	3084998	42,5956	0,988237	20,9875	3055200	45,4801	0,993356	40,38097333
squirrel352	5000	5108093	46,333	0,992278	0,345613	5027296	46,5187	0,994713	20,9762	4990400	50,7259	0,997907	41,55325333
canal720	750	940424	28,6267	0,874431	0,6922	938764	28,6354	0,874568	32,4113	791500	32,5314	0,936891	146,99098
canal720	1000	1202892	30,1491	0,902108	0,6597	1211616	30,2878	0,904796	32,3578	1058620	33,7122	0,9493	146,5748
canal720	2000	2191732	32,7339	0,938768	1,14406	2221492	32,8793	0,940304	48,4078	2115100	36,2547	0,967562	145,30168
canal720	3000	3203316	34,7391	0,95918	1,1522	3227016	34,8043	0,959764	48,5809	3154920	37,5218	0,973097	146,5669
canal720	5000	5216904	36,8819	0,972892	1,1722	5163068	36,8236	0,972632	48,7891	5260520	39,0167	0,97697	147,5331
canal720	6000	6297860	37,7313	0,976397	1,1772	6312492	37,7218	0,976546	48,6559	6291750	39,6302	0,978257	147,6353
canal720	8000	8455088	39,1074	0,981066	1,19406	8467120	39,0727	0,981097	48,4697	8361170	41,5777	0,988681	148,44902
canal720	10000	10671040	40,3785	0,985945	1,20624	10677968	40,3999	0,98598	48,5041	10470850	42,5233	0,990264	149,48786

Video	Target Bit rate (bps)	Dirac Bit Rate (bps)	Dirac PSNR (dB)	Dirac MSSIM	Dirac tempo/quadro (s)	Dirac FULL Bit Rate (bps)	Dirac FULL PSNR (dB)	Dirac FULL MSSIM	Dirac FULL tempo/quadro (s)	H264 Bit rate (bps)	H264 PSNR (dB)	H264 MSSIM	H264 tempo/quadro (s)
canal720	12000	12776096	41,3541	0,988099	1,21686	12759684	41,3625	0,988127	48,4831	12506910	43,382	0,991373	150,1546
canal720	15000	15837692	42,418	0,990079	1,23282	15849776	42,4401	0,990097	48,5053	15644920	44,8632	0,993971	151,3718
snow720	750	984816	30,8202	0,847833	0,90938	874468	29,0973	0,794104	55,8822	748800	30,5184	0,834003	151,99692
snow720	1000	1284628	31,9471	0,875648	0,91844	1078784	30,2923	0,830138	55,7491	999020	32,0362	0,869395	151,94416
snow720	2000	2506824	34,0751	0,917605	1,39968	2457896	33,1701	0,894401	76,8488	1998490	35,1441	0,923063	151,667
snow720	3000	3531964	35,7315	0,939121	1,40938	3434240	34,8817	0,92338	76,5169	2999920	36,5951	0,938557	151,91048
snow720	5000	5854976	37,9293	0,958325	1,42312	5621488	36,6794	0,94449	77,96	5033940	39,2791	0,961662	154,03566
snow720	6000	8008324	39,5208	0,9673	1,43906	6693940	37,7589	0,952317	78,1962	6099700	39,9477	0,964242	153,0608
snow720	8000	10214592	40,6824	0,972825	1,4525	9016772	39,1599	0,960844	78,1538	7983740	42,0509	0,977607	153,69088
snow720	10000	12587092	41,7697	0,977356	1,47844	10688780	39,9908	0,96521	77,7513	9998670	43,0607	0,980633	152,83344
snow720	12000	15086724	42,737	0,98043	1,48282	12512284	41,1415	0,973028	77,5463	12003070	44,42	0,984411	154,23946
snow720	15000	17503260	43,5841	0,983372	1,49656	16790896	42,9424	0,980436	77,7075	15001880	45,9305	0,987624	155,19844
squirrel720	750	1190072	27,3572	0,764656	0,70436	1184368	27,3928	0,764603	40,3988	790650	28,588	0,80271	155,48622
squirrel720	1000	1382824	28,0729	0,790958	0,6828	1391928	28,139	0,792774	40,3706	1053280	29,6414	0,836567	157,05476
squirrel720	2000	2447088	30,2835	0,861047	0,76062	2196476	30,1545	0,857914	42,8169	2103790	32,3964	0,899917	158,80386
squirrel720	3000	3431100	31,2775	0,888929	1,165	3436764	31,2874	0,888688	56,4691	3149200	34,0959	0,927307	158,4866
squirrel720	5000	5305656	33,4207	0,926887	1,1803	5090124	33,2799	0,924579	55,9738	5227890	36,2556	0,949479	157,455
squirrel720	6000	6122996	34,1259	0,936007	1,18562	6000044	34,1681	0,935641	56,015	6270990	37,1595	0,956502	156,27828
squirrel720	8000	8627840	36,0396	0,954084	1,20468	8089620	35,865	0,952044	56,0159	8328220	39,0936	0,974519	156,67428
squirrel720	10000	11132148	37,6767	0,965825	1,22124	10692476	37,5357	0,964134	56,0722	10416690	40,422	0,980175	156,53748
squirrel720	12000	13259012	38,8618	0,972232	1,24656	12684924	38,7287	0,970856	56,2209	12527260	41,5673	0,983707	157,20124
squirrel720	15000	16443448	40,335	0,978819	1,25094	15808304	40,2451	0,977763	56,3175	15593880	43,1473	0,988449	158,40108
snow1280	1000	3900867	29,127	0,813652	1,4875	3951792	29,1429	0,806421	92,3083	2735010	28,7634	0,792546	377,8566333
snow1280	3000	4649702	30,49	0,848455	1,41093	4734145	30,4321	0,842181	90,9807	3160320	29,6401	0,814913	411,2230667
snow1280	5000	6554853	32,1637	0,880496	1,41823	6350017	31,7969	0,87072	91,1453	5047680	31,9426	0,870692	390,0949667
snow1280	6000	7423296	32,0511	0,876319	1,92447	8505398	32,2879	0,875663	110,529	5431980	32,2466	0,87769	369,4706667
snow1280	8000	9512135	33,0354	0,893163	1,93853	10573442	33,0678	0,889353	110,942	7995330	33,9885	0,902888	390,8147
snow1280	10000	11427420	33,611	0,902021	1,94477	11898661	33,4188	0,896496	109,565	9821810	34,9119	0,915397	390,7934667
snow1280	12000	13413962	34,0682	0,908522	1,95363	13662769	33,8333	0,902974	109,509	13720160	35,7916	0,918432	403,3100667
snow1280	15000	16770957	34,7009	0,916653	1,9646	16728647	34,4935	0,912302	109,528	16113600	36,604	0,930364	401,7546667

Video	Target Bit rate (bps)	Dirac Bit Rate (bps)	Dirac PSNR (dB)	Dirac MSSIM	Dirac tempo/quadro (s)	Dirac FULL Bit Rate (bps)	Dirac FULL PSNR (dB)	Dirac FULL MSSIM	Dirac FULL tempo/quadro (s)	H264 Bit rate (bps)	H264 PSNR (dB)	H264 MSSIM	H264 tempo/quadro (s)
snow1280	17000	23076011	35,4061	0,922266	1,99893	24979948	35,4956	0,919631	109,587	17389100	36,8106	0,933378	416,0111
snow1280	20000	27304967	36,05	0,929217	1,99737	27588411	35,8528	0,924323	109,56	20605460	37,868	0,943306	381,9458667
squirrel1280	1000	7386357	26,2104	0,731612	1,32917	5295440	25,7561	0,719264	61,8932	3478620	26,7334	0,748185	406,2516333
squirrel1280	3000	7563476	26,4763	0,743082	1,25937	6271376	26,8794	0,767696	61,6427	3199970	26,5003	0,740073	393,8964667
squirrel1280	5000	9711936	28,1969	0,811522	1,27137	8865054	28,5288	0,825634	61,6411	5234860	28,2771	0,809993	401,0471333
squirrel1280	6000	11183216	28,7694	0,828308	1,50103	10409622	29,2441	0,846063	61,6255	6257390	28,9093	0,829775	425,9852333
squirrel1280	8000	13097830	29,5379	0,85296	1,64633	11488111	29,0477	0,839091	78,4812	8386560	30,0125	0,860514	424,7825
squirrel1280	10000	15184927	30,2403	0,871547	1,6573	13188939	29,8377	0,862974	78,1453	10485100	30,8573	0,880267	417,5964333
squirrel1280	12000	17283292	30,7942	0,884797	1,68123	15640695	30,5603	0,880983	78,2219	12515580	31,5576	0,894033	424,9964
squirrel1280	15000	19990377	31,4446	0,897696	1,69217	18722509	31,3147	0,896718	78,1115	15821490	32,5383	0,910731	414,8110667
squirrel1280	17000	22085802	31,8843	0,9057	1,69633	20546925	31,7341	0,904154	78,2984	17805220	33,017	0,917583	412,5450333
squirrel1280	20000	25293314	32,4924	0,916053	1,6927	24002573	32,377	0,914555	78,537	20901020	33,7639	0,927409	403,7538667

Apêndice II

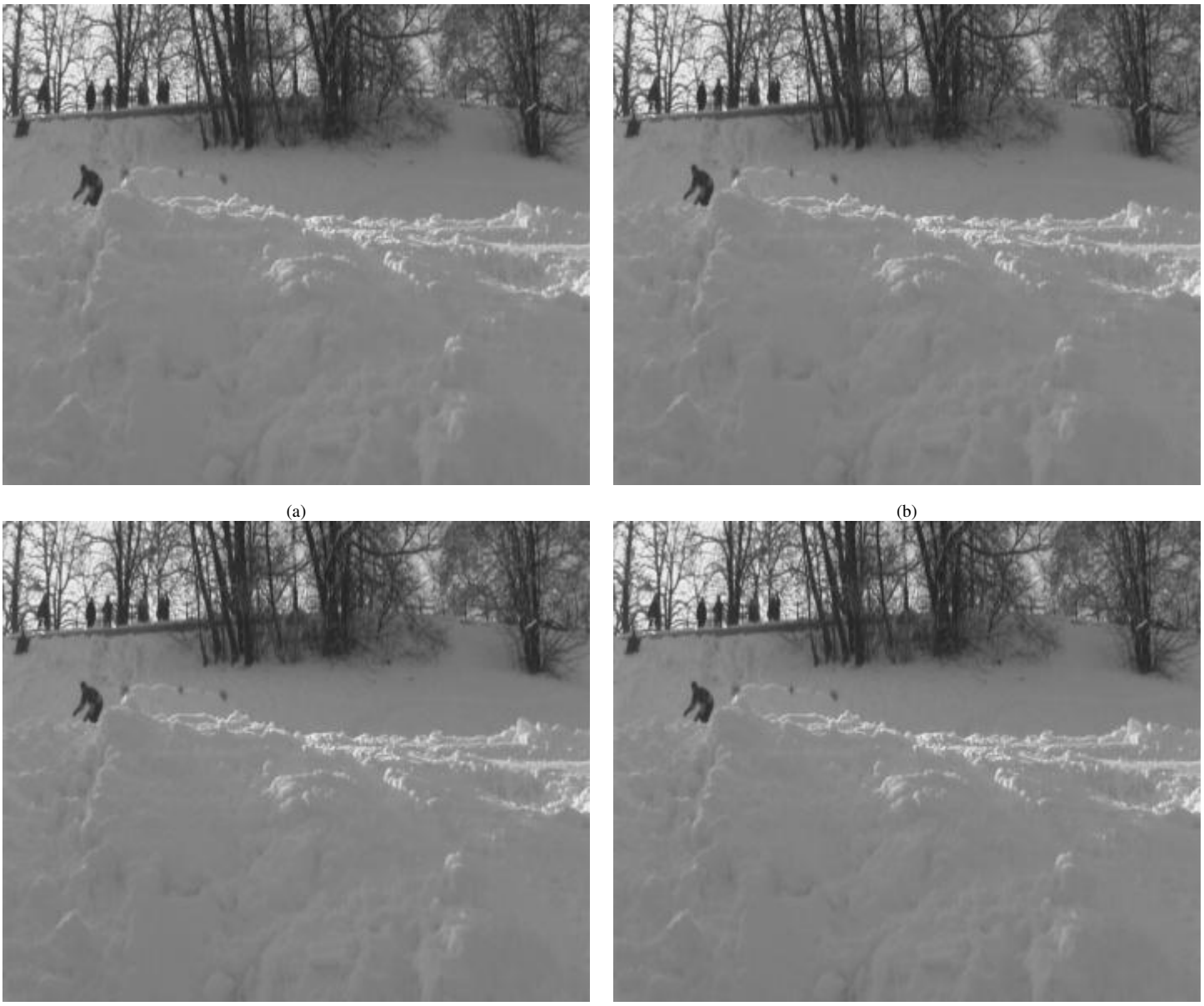
Resultados do item 4.5 – Análise comparativa: Dirac x H.264: Exemplos de quadros reconstruídos



(a) (b)

(c) (d)

Figura II. 1: 10º quadro do vídeo canal352. (a) Original. (b) H.264, PSNR = 48,1653, MSSIM = 0,995003. (c) Dirac Hierárquico, PSNR = 45,5379, MSSIM = 0,992588. (d) Dirac FULL, PSNR = 46,3505, MSSIM = 0,993982.



(a) (b)

(c) (d)

Figura II. 2: 10º quadro do vídeo snow352. (a) Original. (b) H.264, PSNR = 52,3403, MSSIM = 0,996076. (c) Dirac Hierárquico, PSNR = 49,2899, MSSIM = 0,994792. (d) Dirac FULL, PSNR = 49,2736, MSSIM = 0,994804.

(a) (b)

(c) (d)

Figura II. 3: 10º quadro do vídeo squirrel352. (a) Original. (b) H.264, PSNR = 46,8547, MSSIM = 0,995747. (c) Dirac Hierárquico, PSNR = 46,1565, MSSIM = 0,995813. (d) Dirac FULL, PSNR = 46,0943, MSSIM = 0,995822.

Apêndice III

Resultados da seção 5.2 – Testes, resultados e discussão

- Curvas de desempenho comparativo:

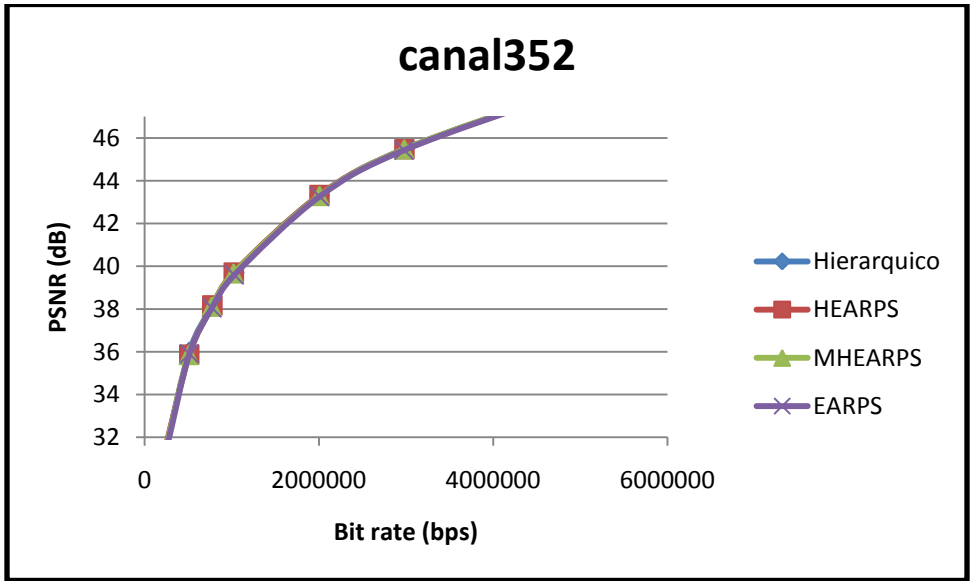


Figura III. 1: Comparação de performance (PSNR x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo canal352.

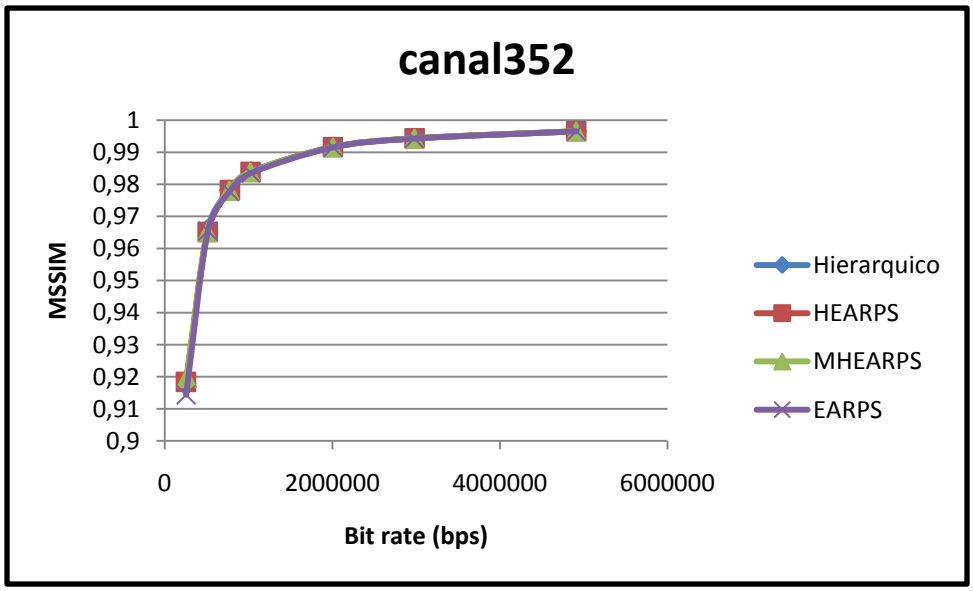


Figura III. 2: Comparação de performance (MSSIM x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo canal352.

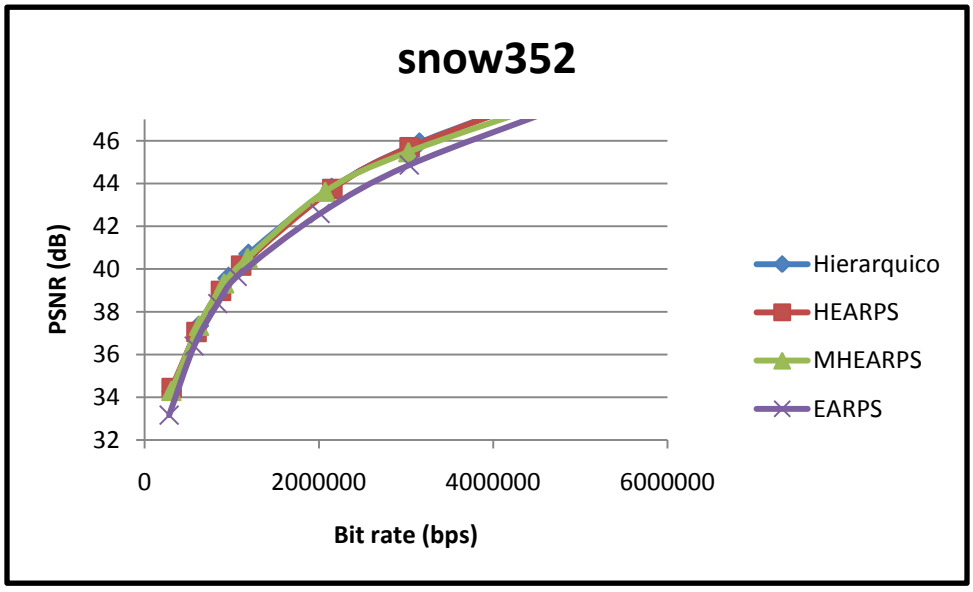


Figura III. 3: Comparação de performance (PSNR x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo snow352

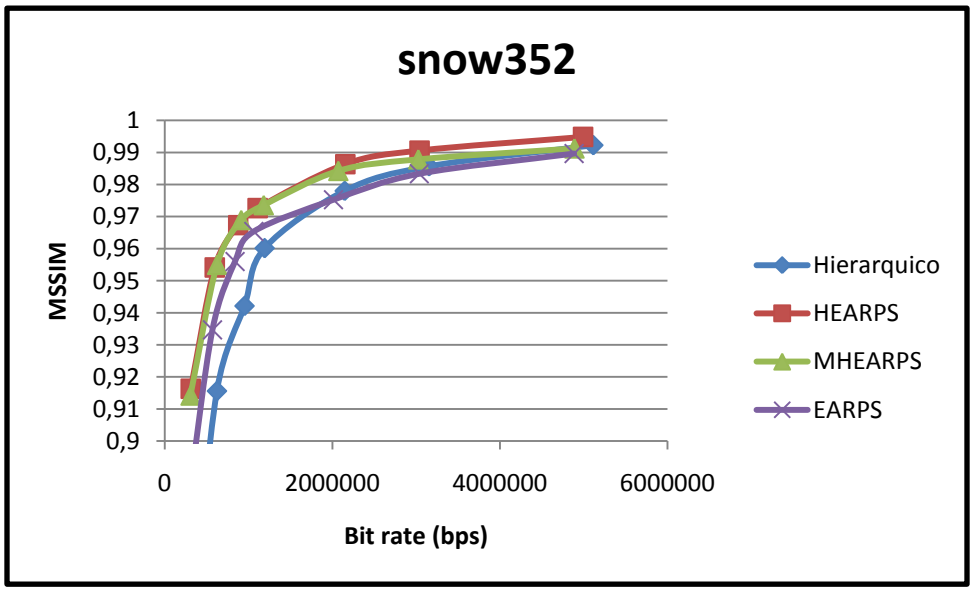


Figura III. 4: Comparação de performance (MSSIM x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo snow352.

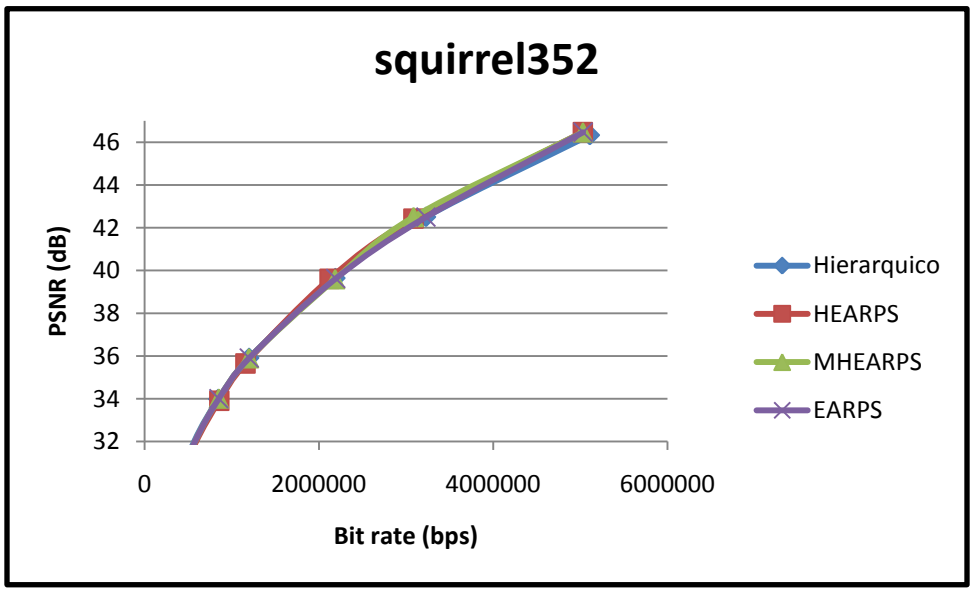


Figura III. 5: Comparação de performance (PSNR x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo squirrel352.

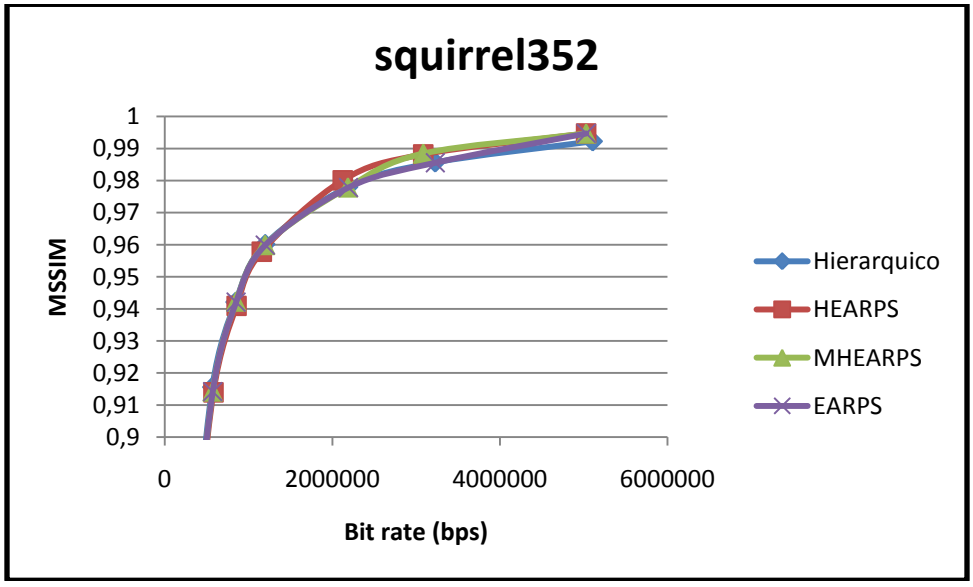


Figura III. 6: Comparação de performance (MSSIM x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo squirrel352.

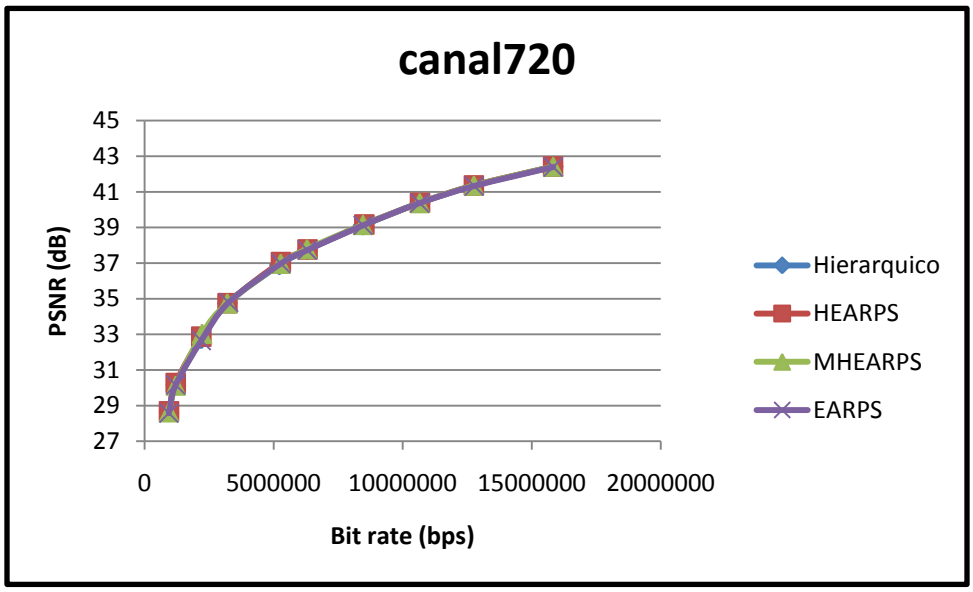


Figura III. 7: Comparação de performance (PSNR x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo canal720.

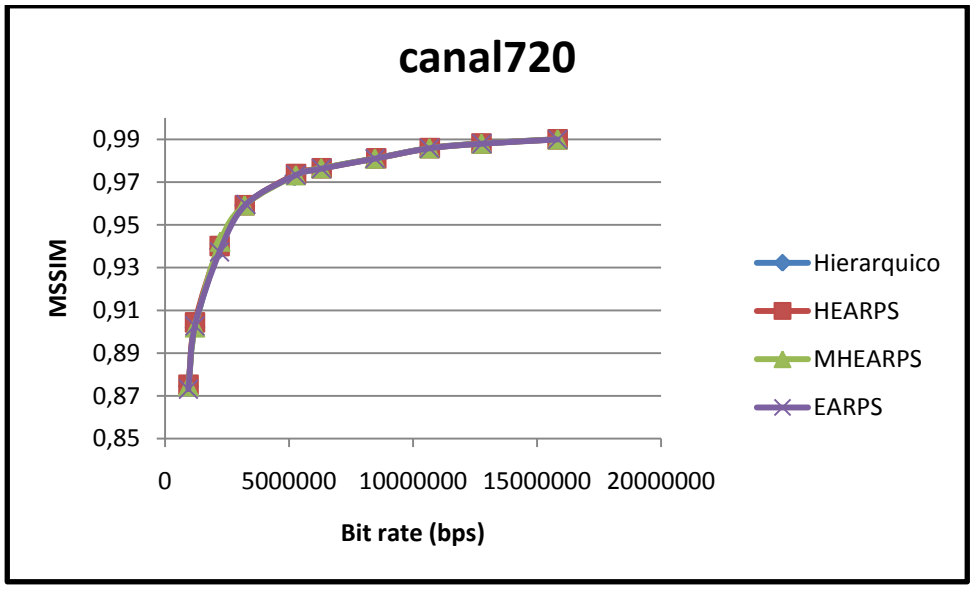


Figura III. 8: Comparação de performance (MSSIM x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo canal720.

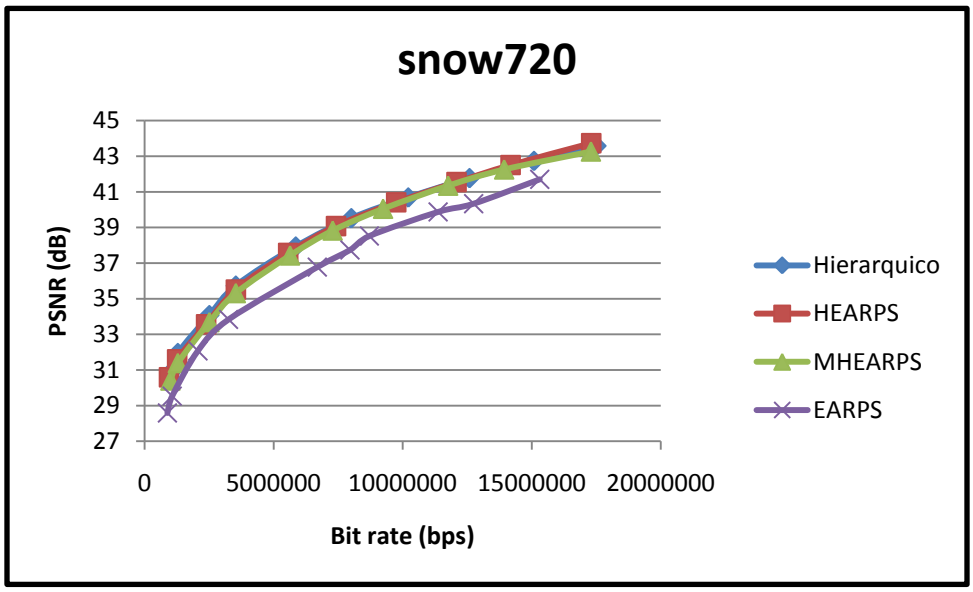


Figura III. 9: Comparação de performance (PSNR x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo snow720.

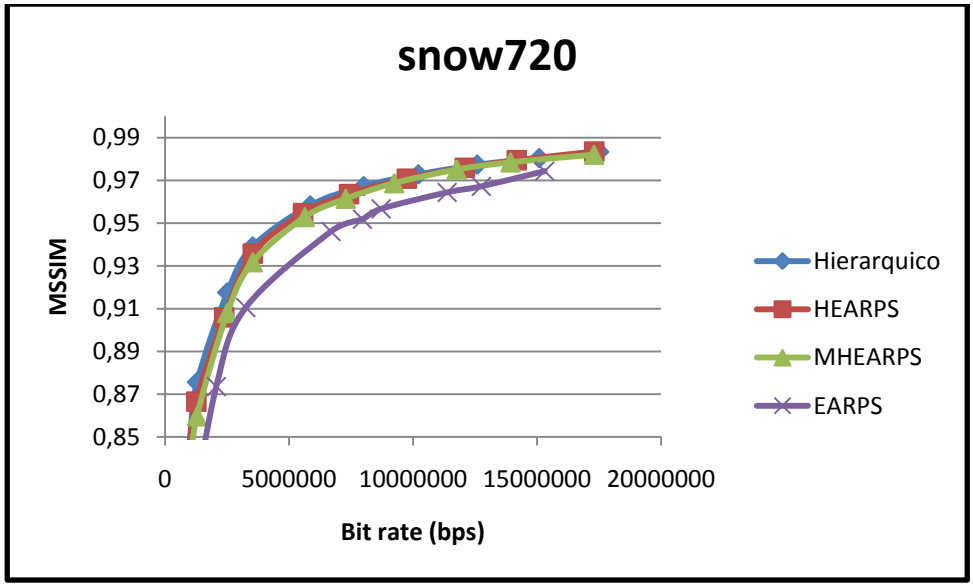


Figura III. 10: Comparação de performance (MSSIM x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo snow720.

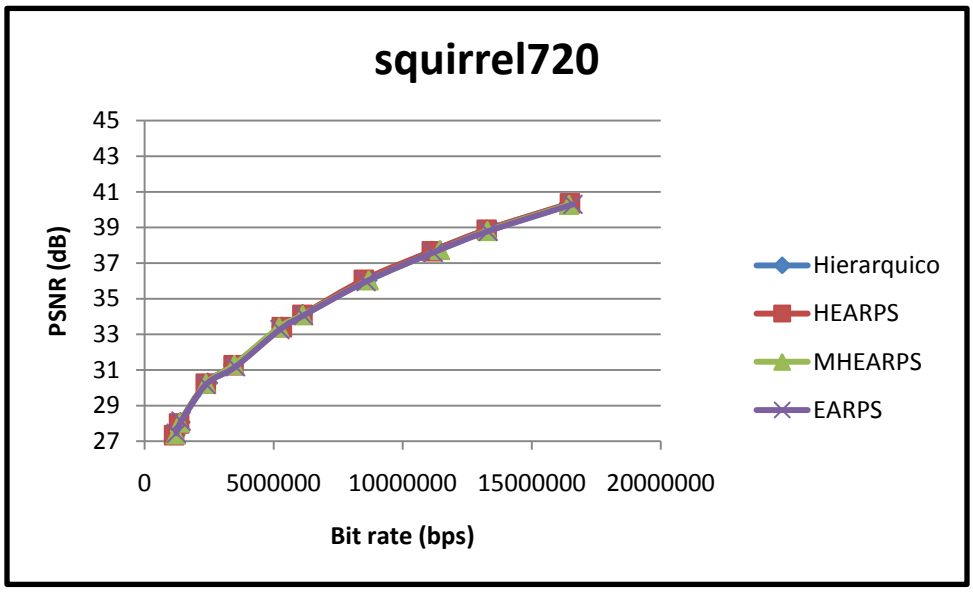


Figura III. 11: Comparação de performance (PSNR x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo squirrel720.

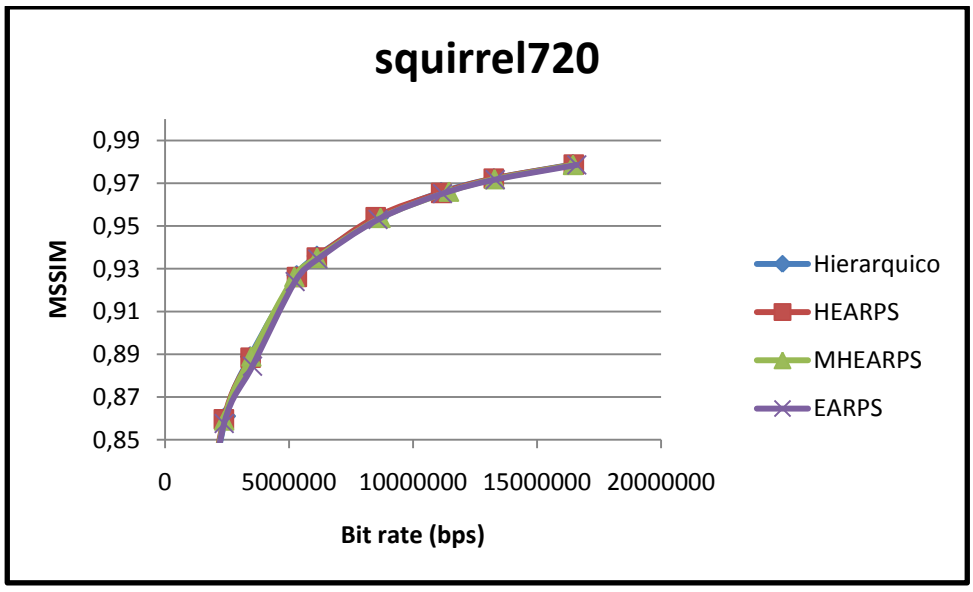


Figura III. 12: Comparação de performance (MSSIM x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo squirrel720.

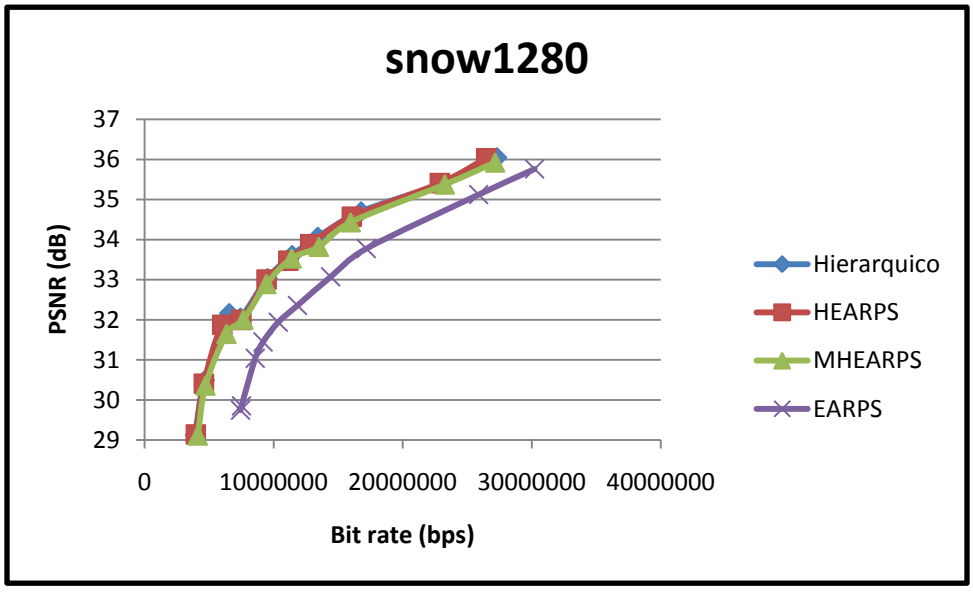


Figura III. 13: Comparação de performance (PSNR x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo snow1280.

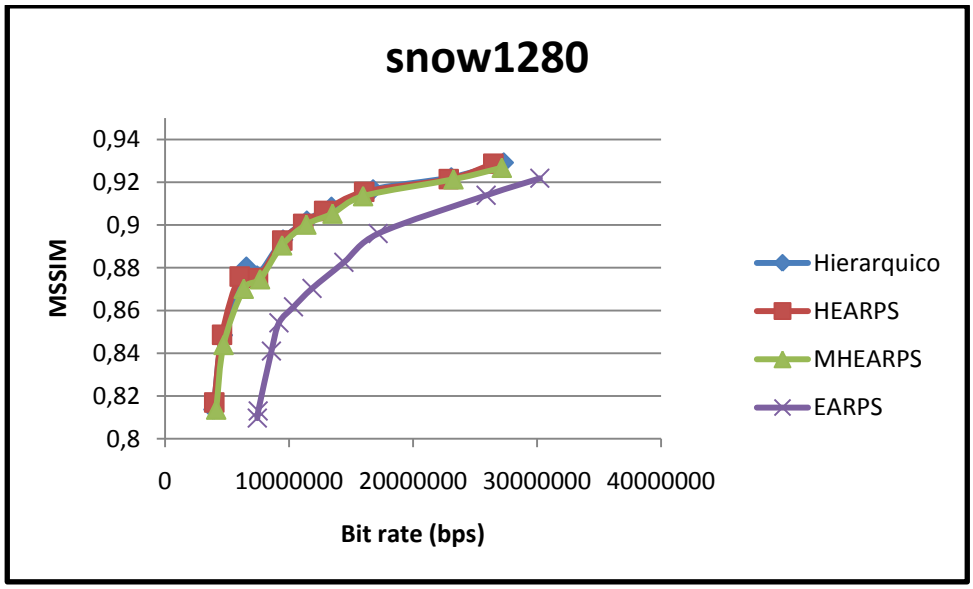


Figura III. 14: Comparação de performance (MSSIM x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo snow1280.

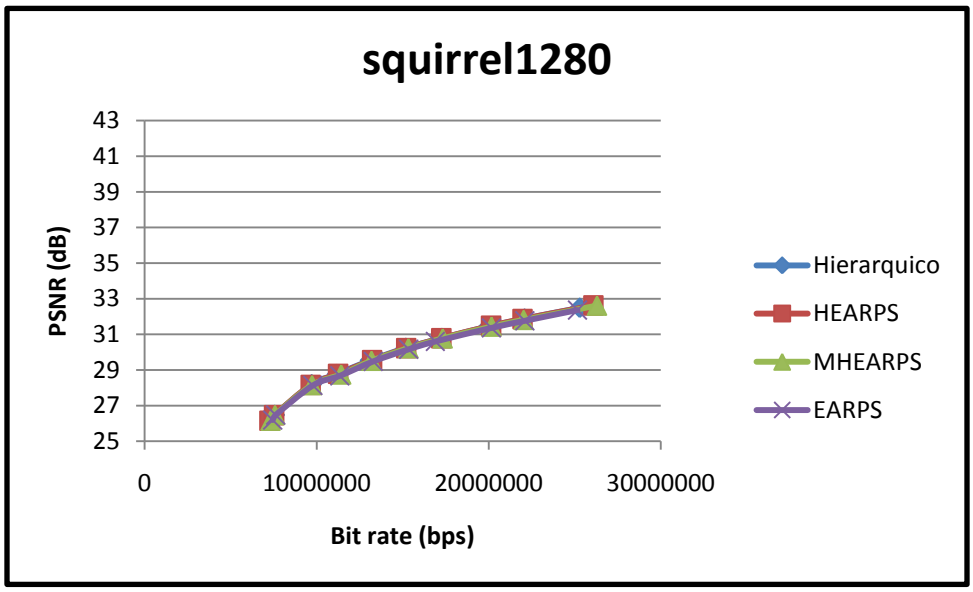


Figura III. 15: Comparação de performance (PSNR x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo squirrel1280.

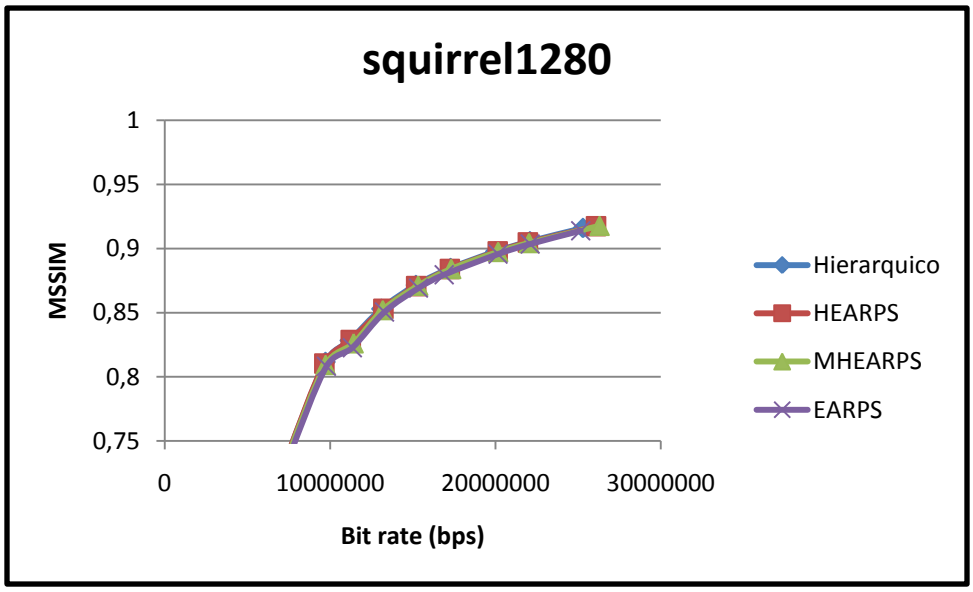


Figura III. 16: Comparação de performance (MSSIM x Taxa de bits) do Dirac com os quatro algoritmos diferentes de estimação de movimento - vídeo squirrel1280.

Tabela III. 1: Tabela com resultados obtidos para os algoritmos Hierárquico (Dirac) e EARPS - seção 5.2.

Video	Target Bit rate (bps)	Dirac Bit Rate (bps)	Dirac PSNR (dB)	Dirac MSSIM	Dirac tempo/quadro (s)	Dirac Nº SAD/quadro	EARPS Bit Rate (bps)	EARPS PSNR (dB)	EARPS MSSIM	EARPS tempo/quadro (s)	Comp. Tempo: EARPS/Dirac	EARPS Nº SAD/quadro	Comp. Nº SAD: EARPS/Dirac
canal352	250	255895	31,6874	0,917847	0,30675	117444	253836	31,4999	0,914354	0,201529	0,656981255	16440,2	0,139983311
canal352	500	512466	35,991	0,966072	0,309735	117444	513551	35,9029	0,965457	0,195309	0,630568066	16440,2	0,139983311
canal352	750	772011	38,1141	0,978209	0,312956	117444	772420	38,0552	0,977799	0,206118	0,658616547	16440,2	0,139983311
canal352	1000	1027794	39,6959	0,983978	0,317103	117444	1028489	39,5956	0,983519	0,2075	0,654361517	16440,2	0,139983311
canal352	2000	2006892	43,3238	0,991658	0,327662	117444	2006858	43,2632	0,991541	0,219206	0,669000372	16440,2	0,139983311
canal352	3000	2981827	45,481	0,994366	0,338691	117444	2976642	45,4322	0,994286	0,228632	0,675045986	16440,2	0,139983311
canal352	5000	4914011	48,2936	0,996567	0,357074	117444	4912500	48,2814	0,996556	0,247941	0,694368674	16440,2	0,139983311
snow352	250	310346	34,273	0,840163	0,255827	79760	281794	33,1641	0,881816	0,167507	0,654766698	13820,7	0,173278586
snow352	500	622184	37,3441	0,915587	0,31396	116271	568732	36,3961	0,934661	0,156867	0,499640082	16203,4	0,139358912
snow352	750	951412	39,5734	0,942129	0,319373	116674	838664	38,3814	0,955989	0,189787	0,594248731	19732,3	0,169123369
snow352	1000	1191361	40,7086	0,960153	0,323747	117203	1061742	39,6546	0,965363	0,198547	0,61327827	19732,3	0,168360025
snow352	2000	2147089	43,7932	0,978036	0,334787	117203	2013212	42,595	0,975261	0,20792	0,621051594	19732,3	0,168360025
snow352	3000	3153932	45,9149	0,985699	0,345413	117203	3037502	44,8669	0,983364	0,227707	0,659231123	20030,8	0,170906888
snow352	5000	5114361	48,8415	0,992278	0,3648	117708	4884738	47,7401	0,989669	0,251667	0,689876645	20304,7	0,172500595
squirrel352	250	270136	28,954	0,840163	0,229373	59709,8	292004	29,1536	0,845521	0,170827	0,744756358	14895,4	0,249463237
squirrel352	500	567936	31,9991	0,915587	0,295627	101163	571678	31,9486	0,914189	0,210413	0,711751633	24510,3	0,242285223
squirrel352	750	845269	33,9863	0,942129	0,29896	101163	851170	34,0037	0,942117	0,2148	0,718490768	24510,3	0,242285223
squirrel352	1000	1199354	35,9093	0,960153	0,304787	101163	1199906	35,9063	0,959896	0,231453	0,759392625	24510,3	0,242285223
squirrel352	2000	2188674	39,6426	0,978036	0,316467	101979	2192848	39,6247	0,977835	0,349373	1,103979246	24703,9	0,242244972
squirrel352	3000	3228101	42,5054	0,985699	0,327493	101979	3228598	42,4993	0,985556	0,279587	0,853719011	24703,9	0,242244972
squirrel352	5000	5108093	46,333	0,992278	0,345613	101979	5030814	46,4668	0,99469	0,268747	0,777595171	24972,2	0,244875906
canal720	750	940424	28,6267	0,874431	0,6922	162139	938992	28,5778	0,873092	0,50124	0,724125975	24268,5	0,149677129
canal720	1000	1202892	30,1491	0,902108	0,6597	162139	1202616	30,1554	0,902749	0,50874	0,771168713	24268,5	0,149677129
canal720	2000	2191732	32,7339	0,938768	1,14406	482387	2196364	32,6462	0,937329	0,71656	0,626330787	51269,9	0,106283751
canal720	3000	3203316	34,7391	0,95918	1,1522	482387	3257392	34,7896	0,95954	0,71968	0,624613782	51269,9	0,106283751
canal720	5000	5216904	36,8819	0,972892	1,1722	482387	5281240	36,9722	0,973361	0,72688	0,620098959	51269,9	0,106283751
canal720	6000	6297860	37,7313	0,976397	1,1772	482387	6303596	37,7258	0,976316	0,74312	0,631260618	51269,9	0,106283751
canal720	8000	8455088	39,1074	0,981066	1,19406	482387	8463048	39,1185	0,981011	0,75438	0,631777298	51269,9	0,106283751
canal720	10000	10671040	40,3785	0,985945	1,20624	482387	10660644	40,3681	0,985841	0,77376	0,641464385	51269,9	0,106283751

Video	Target Bit rate (bps)	Dirac Bit Rate (bps)	Dirac PSNR (dB)	Dirac MSSIM	Dirac tempo/quadro (s)	Dirac N° SAD/quadro	EARPS Bit Rate (bps)	EARPS PSNR (dB)	EARPS MSSIM	EARPS tempo/quadro (s)	Comp. Tempo: EARPS/Dirac	EARPS N° SAD/quadro	Comp. N° SAD: EARPS/Dirac
canal720	12000	12776096	41,3541	0,988099	1,21686	482387	12775716	41,3273	0,987997	0,7897	0,64896537	51269,9	0,106283751
canal720	15000	15837692	42,418	0,990079	1,23282	482387	15830596	42,3905	0,989982	0,88158	0,715092228	51269,9	0,106283751
snow720	750	984816	30,8202	0,847833	0,90938	215491	880556	28,5851	0,78038	0,6375	0,701027073	55108	0,255732258
snow720	1000	1284628	31,9471	0,875648	0,91844	246496	1068084	29,5358	0,808768	0,55532	0,604633945	60847	0,246847819
snow720	2000	2506824	34,0751	0,917605	1,39968	581522	2063784	32,0749	0,87354	0,57906	0,413708848	71549,6	0,123038509
snow720	3000	3531964	35,7315	0,939121	1,40938	586448	3252492	33,8365	0,910525	0,7572	0,537257517	123128	0,209955529
snow720	5000	5854976	37,9293	0,958325	1,42312	591611	6689884	36,7757	0,946101	0,86376	0,606948114	130647	0,220832608
snow720	6000	8008324	39,5208	0,9673	1,43906	591611	7957624	37,7363	0,951878	0,8125	0,564604672	130647	0,220832608
snow720	8000	10214592	40,6824	0,972825	1,4525	596316	8709464	38,5205	0,956711	0,79938	0,550347676	129601	0,21733611
snow720	10000	12587092	41,7697	0,977356	1,47844	596316	11372408	39,874	0,964387	0,84032	0,56838289	131039	0,219747583
snow720	12000	15086724	42,737	0,98043	1,48282	596316	12749580	40,3364	0,967198	0,86062	0,580394114	133748	0,224290477
snow720	15000	17503260	43,5841	0,983372	1,49656	601678	15308048	41,7025	0,97435	0,89562	0,598452451	137809	0,229041115
squirrel720	750	1190072	27,3572	0,764656	0,70436	143660	1228784	27,4466	0,767011	0,73344	1,041285706	43385,3	0,301999861
squirrel720	1000	1382824	28,0729	0,790958	0,6828	145488	1416612	28,1284	0,792032	0,52812	0,773462214	43839,1	0,301324508
squirrel720	2000	2447088	30,2835	0,861047	0,76062	194414	2381248	30,1888	0,857638	0,56874	0,747732113	56026,7	0,288182435
squirrel720	3000	3431100	31,2775	0,888929	1,165	455799	3521880	31,18	0,884391	0,78656	0,675158798	115660	0,253752202
squirrel720	5000	5305656	33,4207	0,926887	1,1803	455799	5242800	33,2667	0,924011	0,82124	0,695789206	117472	0,257727639
squirrel720	6000	6122996	34,1259	0,936007	1,18562	455799	6157232	34,0565	0,934524	0,82624	0,696884331	117472	0,257727639
squirrel720	8000	8627840	36,0396	0,954084	1,20468	455799	8585020	35,9647	0,953042	0,84314	0,699887107	118231	0,259392846
squirrel720	10000	11132148	37,6767	0,965825	1,22124	460077	11185876	37,5913	0,965132	0,86658	0,709590253	118231	0,256980897
squirrel720	12000	13259012	38,8618	0,972232	1,24656	460077	13311044	38,7841	0,971699	0,89718	0,719724682	118231	0,256980897
squirrel720	15000	16443448	40,335	0,978819	1,25094	460077	16599428	40,296	0,978577	0,96688	0,772922762	118231	0,256980897
snow1280	1000	3900867	29,127	0,813652	1,4875	222874	7431432	29,7458	0,809671	1,26357	0,849458824	89541	0,401756149
snow1280	3000	4649702	30,49	0,848455	1,41093	236218	7507324	29,853	0,813027	1,1531	0,817262373	89541	0,379060868
snow1280	5000	6554853	32,1637	0,880496	1,41823	236218	8570437	31,0384	0,841029	1,10523	0,77930237	91147,5	0,385861789
snow1280	6000	7423296	32,0511	0,876319	1,92447	598421	9176695	31,4472	0,854224	1,13283	0,588645185	94730,8	0,158301263
snow1280	8000	9512135	33,0354	0,893163	1,93853	598421	10354013	31,9392	0,861718	1,17083	0,603978272	105829	0,176847069
snow1280	10000	11427420	33,611	0,902021	1,94477	598421	11845434	32,3583	0,870335	1,43073	0,735680826	132107	0,220759298
snow1280	12000	13413962	34,0682	0,908522	1,95363	598421	14413394	33,0729	0,882538	1,44793	0,741148529	162162	0,270983137
snow1280	15000	16770957	34,7009	0,916653	1,9646	598421	17181026	33,7762	0,89601	1,38073	0,702804642	162162	0,270983137

Video	Target Bit rate (bps)	Dirac Bit Rate (bps)	Dirac PSNR (dB)	Dirac MSSIM	Dirac tempo/quadro (s)	Dirac N° SAD/quadro	EARPS Bit Rate (bps)	EARPS PSNR (dB)	EARPS MSSIM	EARPS tempo/quadro (s)	Comp. Tempo: EARPS/Dirac	EARPS N° SAD/quadro	Comp. N° SAD: EARPS/Dirac
snow1280	17000	23076011	35,4061	0,922266	1,99893	607683	25889966	35,125	0,913895	1,54113	0,770977473	166812	0,274504964
snow1280	20000	27304967	36,05	0,929217	1,99737	607683	30220227	35,7633	0,921872	1,3724	0,687103541	166812	0,274504964
squirrel1280	1000	7386357	26,2104	0,731612	1,32917	214086	7440863	26,1884	0,727783	1,03957	0,782119669	63435,4	0,296308026
squirrel1280	3000	7563476	26,4763	0,743082	1,25937	218787	7618541	26,4732	0,740674	1,00157	0,795294473	64722,7	0,295825163
squirrel1280	5000	9711936	28,1969	0,811522	1,27137	218787	9778845	28,1297	0,808193	1,00157	0,787787977	64722,7	0,295825163
squirrel1280	6000	11183216	28,7694	0,828308	1,50103	363024	11343600	28,6692	0,822722	1,16147	0,773782003	95066	0,261872493
squirrel1280	8000	13097830	29,5379	0,85296	1,64633	444733	13286633	29,4801	0,850505	1,24377	0,755480371	125970	0,283248601
squirrel1280	10000	15184927	30,2403	0,871547	1,6573	444733	15347532	30,1613	0,869173	1,25833	0,75926507	125970	0,283248601
squirrel1280	12000	17283292	30,7942	0,884797	1,68123	444733	16897006	30,5998	0,879586	1,275	0,758373334	125970	0,283248601
squirrel1280	15000	19990377	31,4446	0,897696	1,69217	444733	20156899	31,3615	0,895651	1,33437	0,788555523	125970	0,283248601
squirrel1280	17000	22085802	31,8843	0,9057	1,69633	444733	22101034	31,762	0,90351	1,525	0,898999605	125970	0,283248601
squirrel1280	20000	25293314	32,4924	0,916053	1,6927	444733	25159128	32,3651	0,913855	1,32917	0,785236604	125970	0,283248601

Tabela III. 2: Tabela com resultados obtidos para os algoritmos HEARPS e MHEARPS - seção 5.2.

Video	Target Bit rate (bps)	HEARPS Bit Rate (bps)	HEARPS PSNR (dB)	HEARPS MSSIM	HEARPS tempo/quadro (s)	Comp. Tempo: HEARPS/Dirac	HEARPS N° SAD/quadro	Comp. N° SAD: HEARPS/Dirac	MHEARPS Bit Rate (bps)	MHEARPS PSNR (dB)	MHEARPS MSSIM	MHEARPS tempo/quadro (s)	Comp. Tempo: MHEARPS/Dirac	MHEARPS N° SAD/quadro	Comp. N° SAD: MHEARPS/Dirac
canal352	250	254977	31,7273	0,9184	0,201059	0,655449063	19736	0,168046047	256511	31,7803	0,919645	0,199221	0,649457213	18743,5	0,159595211
canal352	500	500442	35,8636	0,965146	0,205647	0,663944985	19736	0,168046047	500039	35,855	0,965142	0,206809	0,667696579	18743,5	0,159595211
canal352	750	769313	38,1662	0,978239	0,205647	0,657111543	19736	0,168046047	765186	38,1152	0,978078	0,212765	0,679855954	18743,5	0,159595211
canal352	1000	1021017	39,7047	0,983901	0,214603	0,676761179	19736	0,168046047	1021675	39,6724	0,983836	0,215294	0,678940281	18743,5	0,159595211
canal352	2000	2006361	43,3283	0,991634	0,227265	0,69359584	19736	0,168046047	2004477	43,2938	0,99162	0,228397	0,697050619	18743,5	0,159595211
canal352	3000	2979133	45,4759	0,99435	0,238279	0,703529176	19736	0,168046047	2977892	45,4811	0,994354	0,238279	0,703529176	18743,5	0,159595211
canal352	5000	4910204	48,2802	0,996569	0,258971	0,72525863	19736	0,168046047	4911111	48,3167	0,996578	0,259426	0,726532876	18743,5	0,159595211
snow352	250	315184	34,4339	0,916321	0,179587	0,701986108	15637,7	0,196059428	313524	34,303	0,914204	0,17228	0,673423837	13933,8	0,17469659
snow352	500	594838	37,0549	0,954122	0,2	0,637023825	21538,4	0,185243096	626902	37,3507	0,954958	0,205827	0,655583514	19573,2	0,168341203
snow352	750	862960	38,9787	0,967323	0,207293	0,649062382	21590,7	0,185051511	905225	39,3331	0,968911	0,212507	0,66538812	19573,2	0,167759741
snow352	1000	1116272	40,1601	0,972658	0,21728	0,671141354	21590,7	0,184216274	1155598	40,517	0,973558	0,216867	0,669865667	19638,1	0,167556291
snow352	2000	2141757	43,7403	0,986349	0,23272	0,695128544	21699,9	0,185147991	2104605	43,6245	0,984233	0,22772	0,680193675	19717,3	0,168232042
snow352	3000	3041798	45,6961	0,990587	0,245	0,709295828	21850	0,186428675	3027160	45,4748	0,987834	0,244573	0,708059627	19789,9	0,16885148
snow352	5000	4992210	48,673	0,994838	0,29104	0,797807018	21850	0,185628844	4890722	48,0954	0,991243	0,259573	0,711548794	19789,9	0,16812706
squirrel352	250	289272	29,0976	0,844446	0,19604	0,854677752	18572,1	0,311039394	287737	29,0636	0,843718	0,183747	0,801083824	16343,6	0,273717212
squirrel352	500	579617	31,9069	0,913972	0,228333	0,772368559	30120,9	0,297746212	571496	31,9425	0,91425	0,22208	0,751216905	26773,6	0,264658027
squirrel352	750	856474	33,909	0,940933	0,245413	0,820889082	30120,9	0,297746212	851397	34,0089	0,94229	0,2852	0,953973776	26773,6	0,264658027
squirrel352	1000	1176401	35,6529	0,957819	0,245	0,803840059	30120,9	0,297746212	1198100	35,8959	0,95993	0,231867	0,760750951	26773,6	0,264658027
squirrel352	2000	2125418	39,6106	0,980081	0,258333	0,816303122	30429,1	0,298385942	2185812	39,5952	0,977886	0,24396	0,770886064	26972,6	0,264491709
squirrel352	3000	3085690	42,429	0,988176	0,26208	0,80026138	30429,1	0,298385942	3144792	42,5053	0,98843	0,26896	0,821269462	27150,2	0,266233244
squirrel352	5000	5027884	46,4737	0,994684	0,278947	0,80710795	30429,1	0,298385942	5025214	46,4705	0,994693	0,27604	0,798696808	27150,2	0,266233244
canal720	750	942296	28,6786	0,87524	0,51562	0,744900318	32992,7	0,203484048	939092	28,6329	0,874585	0,5653	0,816671482	32835,1	0,202512042
canal720	1000	1201780	30,2656	0,904378	0,48686	0,738002122	32992,7	0,203484048	1217000	30,131	0,902071	0,48186	0,73042292	32835,1	0,202512042
canal720	2000	2217612	32,8692	0,940108	0,75344	0,658566858	65625,1	0,136042431	2248372	33,0333	0,942204	0,73812	0,645175952	64992,5	0,134731035
canal720	3000	3186556	34,7492	0,959182	0,76188	0,661239368	65625,1	0,136042431	3188000	34,744	0,959185	0,75186	0,652542961	64992,5	0,134731035
canal720	5000	5299152	37,0622	0,973817	0,78282	0,667821191	65625,1	0,136042431	5273732	36,9663	0,973262	0,77062	0,657413411	64992,5	0,134731035
canal720	6000	6322112	37,7694	0,976472	0,85906	0,729748556	65625,1	0,136042431	6357336	37,791	0,976536	0,77342	0,65699966	64992,5	0,134731035
canal720	8000	8536308	39,1671	0,981125	0,80782	0,676532168	65625,1	0,136042431	8521784	39,1467	0,981062	0,79374	0,664740465	64992,5	0,134731035

Video	Target Bit rate (bps)	HEARPS Bit Rate (bps)	HEARPS PSNR (dB)	HEARPS MSSIM	HEARPS tempo/quadro (s)	Comp. Tempo: HEARPS/Dirac	HEARPS N° SAD/quadro	Comp. N° SAD: HEARPS/Dirac	MHEARPS Bit Rate (bps)	MHEARPS PSNR (dB)	MHEARPS MSSIM	MHEARPS tempo/quadro (s)	Comp. Tempo: MHEARPS/Dirac	MHEARPS N° SAD/quadro	Comp. N° SAD: MHEARPS/Dirac
canal720	10000	10661848	40,3847	0,985899	0,80812	0,669949595	65625,1	0,136042431	10662252	40,373	0,98588	0,8025	0,665290489	64992,5	0,134731035
canal720	12000	12776536	41,3606	0,98806	0,80562	0,662048222	65625,1	0,136042431	12760404	41,3547	0,988064	0,81438	0,669247079	64992,5	0,134731035
canal720	15000	15829364	42,4204	0,990047	0,8125	0,659058094	65625,1	0,136042431	15831812	42,4185	0,990051	0,83282	0,67554063	64992,5	0,134731035
snow720	750	932688	30,591	0,841309	0,7775	0,854978117	48868,8	0,226778845	947740	30,4063	0,834129	0,77656	0,853944446	38809,7	0,180098937
snow720	1000	1248340	31,6002	0,866533	0,69844	0,760463394	57372,6	0,232752661	1241224	31,3885	0,85971	0,56876	0,619267454	46534,7	0,188784808
snow720	2000	2364188	33,5626	0,905982	0,81344	0,581161408	107673	0,185157225	2470596	33,6852	0,908067	0,8272	0,590992227	95075,7	0,163494588
snow720	3000	3533820	35,5153	0,935771	0,82186	0,583135847	109782	0,187198183	3545152	35,3152	0,931786	0,78874	0,559636152	95075,7	0,162121279
snow720	5000	5547396	37,5679	0,954677	0,82938	0,582789926	110697	0,187111125	5628168	37,4278	0,952917	0,79686	0,559938726	95730,3	0,161812914
snow720	6000	7427756	39,0805	0,963613	0,8275	0,575028143	110697	0,187111125	7291600	38,8267	0,961493	0,81874	0,568940836	95730,3	0,161812914
snow720	8000	9732068	40,4137	0,970828	0,85562	0,589067126	111520	0,187014938	9389664	40,048	0,968746	0,86186	0,593363167	96578,6	0,16195876
snow720	10000	12061076	41,5469	0,975925	0,8775	0,593531019	111520	0,187014938	11862136	41,3447	0,975147	0,8825	0,596912962	98587,8	0,165328115
snow720	12000	14224276	42,5066	0,979452	0,89374	0,602729934	112985	0,18947169	13970320	42,2557	0,978482	0,88874	0,59935798	98587,8	0,165328115
snow720	15000	17879496	43,7335	0,983516	0,92594	0,618712247	112985	0,187783166	16606344	43,2626	0,982092	0,9175	0,613072647	99831,2	0,165921307
squirrel720	750	1142080	27,3381	0,763149	0,75124	1,066556874	50460,6	0,351250174	1208828	27,3902	0,765782	0,79	1,121585553	40528	0,282110539
squirrel720	1000	1335516	27,997	0,787997	0,54094	0,792237844	51109,9	0,351299764	1368268	28,0525	0,789715	0,54468	0,79771529	41372	0,284367096
squirrel720	2000	2358912	30,2298	0,859547	0,59188	0,778154663	64375,2	0,331124302	2376700	30,2309	0,858988	0,58968	0,775262286	53684,8	0,276136492
squirrel720	3000	3445948	31,2664	0,888375	0,84156	0,722369099	133783	0,293513149	3478252	31,2968	0,888967	0,84094	0,72183691	115576	0,25356791
squirrel720	5000	5310372	33,3997	0,926137	0,8553	0,724646276	133783	0,293513149	5328488	33,3972	0,926154	0,93344	0,790849784	115576	0,25356791
squirrel720	6000	6122132	34,0982	0,935268	0,86376	0,72853022	133783	0,293513149	6116992	34,0863	0,935142	0,8578	0,723503315	115576	0,25356791
squirrel720	8000	8530704	36,0649	0,954119	0,88874	0,737739483	134720	0,29556888	8692336	36,0427	0,953777	0,87374	0,725288043	115576	0,25356791
squirrel720	10000	11124828	37,6637	0,965501	0,91344	0,747961089	134720	0,29282055	11434132	37,7439	0,966065	0,88874	0,727735744	115576	0,251210124
squirrel720	12000	13254416	38,878	0,97217	0,91936	0,737517649	134720	0,29282055	13283732	38,814	0,971897	0,89718	0,719724682	116507	0,253233698
squirrel720	15000	16471120	40,3529	0,978713	1,01436	0,81087822	134720	0,29282055	16448336	40,2788	0,978554	0,91812	0,733944074	116507	0,253233698
snow1280	1000	3941994	29,1377	0,816987	1,15677	0,777660504	65868,9	0,295543222	4032623	29,1141	0,813675	1,20417	0,80952605	51347,5	0,230388022
snow1280	3000	4609438	30,4065	0,848614	1,025	0,726471193	66983,4	0,283566028	4780531	30,3602	0,843919	1,01667	0,720567285	52382,1	0,221753211
snow1280	5000	6059844	31,8757	0,875887	1,04687	0,738152486	66983,4	0,283566028	6249350	31,6508	0,870204	1,013	0,714270605	53367,3	0,225923935
snow1280	6000	7492651	31,9981	0,875164	1,25833	0,653857945	135353	0,226183573	7654345	32,0058	0,874636	1,2427	0,645736229	115154	0,192429744
snow1280	8000	9476347	33,0056	0,892739	1,27553	0,657988269	135353	0,226183573	9551792	32,8941	0,89049	1,25	0,644818496	115154	0,192429744
snow1280	10000	11171548	33,4754	0,900553	1,38853	0,713981602	137675	0,230063785	11678801	33,5333	0,900211	1,2604	0,648097204	115154	0,192429744

Video	Target Bit rate (bps)	HEARPS Bit Rate (bps)	HEARPS PSNR (dB)	HEARPS MSSIM	HEARPS tempo/quadro (s)	Comp. Tempo: HEARPS/Dirac	HEARPS N° SAD/quadro	Comp. N° SAD: HEARPS/Dirac	MHEARPS Bit Rate (bps)	MHEARPS PSNR (dB)	MHEARPS MSSIM	MHEARPS tempo/quadro (s)	Comp. Tempo: MHEARPS/Dirac	MHEARPS N° SAD/quadro	Comp. N° SAD: MHEARPS/Dirac
snow1280	12000	12831152	33,8845	0,906592	1,3021	0,666502869	137675	0,230063785	12980651	33,8222	0,905328	1,27917	0,654765744	117227	0,195893861
snow1280	15000	16081614	34,5713	0,91551	1,40417	0,714735824	137675	0,230063785	16005130	34,4345	0,91351	1,2906	0,656927619	117227	0,195893861
snow1280	17000	23044075	35,409	0,921475	1,36457	0,682650218	137675	0,226557268	23283468	35,377	0,921387	1,3	0,650347936	117227	0,192908145
snow1280	20000	27241974	36,0294	0,928682	1,36617	0,68398444	137675	0,226557268	27356739	35,9258	0,926773	1,32707	0,664408697	117227	0,192908145
squirrel1280	1000	7252139	26,1509	0,728694	1,4031	1,055621177	73682,7	0,34417337	7314477	26,1583	0,728707	1,4396	1,083081923	62788,7	0,293287277
squirrel1280	3000	7524059	26,4589	0,74207	1,04167	0,82713579	73682,7	0,336778236	7575176	26,4773	0,742411	1,15207	0,914798669	62788,7	0,286985516
squirrel1280	5000	9651372	28,1632	0,810313	1,14167	0,897984064	73682,7	0,336778236	9728959	28,1535	0,809376	1,04583	0,822600816	62788,7	0,286985516
squirrel1280	6000	11239720	28,7826	0,828686	1,2052	0,802915331	118728	0,327052757	11427500	28,7511	0,826195	1,18593	0,79007748	92236,8	0,254079069
squirrel1280	8000	13223768	29,5493	0,853	1,3224	0,803241148	144842	0,32568305	13196819	29,5196	0,852087	1,28543	0,78078514	127318	0,286279633
squirrel1280	10000	15189066	30,2266	0,870727	1,3151	0,79351958	144842	0,32568305	15334489	30,225	0,871061	1,38853	0,837826585	127318	0,286279633
squirrel1280	12000	17222073	30,785	0,884058	1,32397	0,787500818	144842	0,32568305	17338325	30,7744	0,884108	1,3031	0,775087287	127318	0,286279633
squirrel1280	15000	20138693	31,483	0,897868	1,35833	0,802714857	144842	0,32568305	20172035	31,4355	0,897407	1,32133	0,780849442	127318	0,286279633
squirrel1280	17000	21975464	31,8623	0,904794	1,37447	0,810260975	144842	0,32568305	21950193	31,8132	0,904307	1,3104	0,772491202	127318	0,286279633
squirrel1280	20000	26088279	32,6216	0,917447	1,35573	0,800927512	144842	0,32568305	26316883	32,614	0,917725	1,33437	0,788308619	127318	0,286279633

Referências Bibliográficas

1. **Richardson, I. E. G.** *Video Codec Design.* s.l. : John Wiley & Sons, 2002.
2. **J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, D. J. LeGall.** *MPEG Video Compression Standard.* s.l. : Kluwer Academic Publishers, 1996.
3. **Associação Brasileira de Normas Técnicas.** *Norma Brasileira ABNT NBR 15602-1 - "Televisão digital terrestre - Codificação de vídeo, áudio e multiplexação. Parte 1: Codificação de vídeo".* 2007.
4. *Performance comparison of intra-only H.264/AVC HP and JPEG2000 for a set of monochrome ISO/IEC test images.* **D. Marpe, V. George, T. Wiegand.** Palma de Mallorca, Espanha : s.n., 2004. 13th Meeting. Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG.
5. *A comparative study of JPEG2000, AVC/H.264, and HD photo.* **F. De Simone, M. Ouaret, F. Dufaux, A. G. Tescher, T. Ebrahimi.** 2007. Applications of Digital Image Processing - Proceedings of the SPIE.
6. *Performance comparison of the emerging H.264 video coding standard with the existing standards.* **N. Kamaci, Y. Altunbasak.** 2003. Proceedings of the 2003 International Conference on Multimedia and Exp. Vol. 2.
7. *Motion compensated inter-frame coding for video conferencing.* **Koga, T., Iinuma, K., Hirano, A., Iijima, Y., Ishiguro, T.** New Orleans, LA : s.n., 1981. Proc. NTC'81. pp. G5.3.1-G5.3.5.
8. *A new diamond search algorithm for fast block matching motion estimation.* **Zhu, S., Ma, K. K.** 1997. Proc. of Int. Conf. Information, Communications and Signal Processing. Vol. 1, pp. 292-296.
9. *A Novel Four-Step Search Algorithm for Fast Block Motion Estimation.* **Po, L. M., Ma, W. C.** 1996, IEEE Trans. on Circuits and Systems for Video Technology, Vol. 6, pp. 313-317. 3.
10. *A new three-step search algorithm for block motion estimation.* **Li, R., Zeng, B., Liou, M. L.** 1991, IEEE Trans. on Circuits and Systems for Video Technology, Vol. 4, pp. 438-442.
11. *A Novel Cross-Diamond Search Algorithm for Fast Block Motion Estimation.* **Cheung, C. H., Po, L. M.** 2002, IEEE Trans. on Circuits and Systems for Video Technology, Vol. 12.
12. *Adaptive Rood Pattern Search for Fast Block-Matching Motion Estimation.* **Nie, Y., Ma, K. K.** 2002, IEEE Trans. on Image Processing, Vol. 11.
13. *Optimizing the MPEG-4 Encoder-Advanced Diamond Zonal Search.* **Tourapis, A. M., Au, O. C., Liou, M. L., Shen, G., Ahmad, I.** Genebra : s.n., 2000. IEEE International Symposium on Circuits and Systems.
14. *Motion Vector Field Adaptive Fast Motion Estimation.* **Hosur, P.I., Ma, K. K.** Cingapura : s.n., 1999. Second International Conference on Information, Communications and Signal Processing.
15. *Fast Motion Vector Estimation Using Multiresolution-Spatio-Temporal Correlations.* **Chalidabhongse, J., Jay Kuo, C. C.** 1997, IEEE Trans. on Circuits and Systems for Video Technology, Vol. 7.
16. *Predictive Motion Vector Field Adaptive Search Technique (PMVFAST)-Enhancing Block Based Motion Estimation.* **Tourapis, A. M., Au, O. C., Liou, M. L.** San Jose, CA : s.n., 2001. Proceedings of Visual Communications and Image Processing 2001.

17. *Low-Complexity Block-Based Motion Estimation via One-Bit Transforms.* **Natarajan, B., Bhaskaran, V., Konstantinides, K.** 1997, IEEE Trans. on Circuits and Systems for Video Technology, Vol. 7.
18. *Modified One-Bit Transform for Motion Estimation.* **Wong, P. H. W., Au, O. C.** 1999, IEEE Trans. on Circuits and Systems for Video Technology, Vol. 9.
19. *Two-bit transform for binary block motion estimation.* **Ertürk, A., Ertürk, S.** 2005, IEEE Trans. on Circuits and Systems for Video Technol., Vol. 15, pp. 938-946.
20. *Successive elimination algorithm for motion estimation.* **Li, W., Salari, E.** 1995, IEEE Trans. on Image Processing, Vol. 4, pp. 105-107.
21. *A multilevel successive elimination algorithm for block matching motion estimation.* **Gao, X. Q., Duanmu, C. J., Zou, C. R.** 2000, IEEE Trans. on Image Processing, Vol. 9, pp. 501–504.
22. **Ravi, A.** *Performance analysis and comparison of Dirac video codec with H.264 / MPEG-4 Part 10 AVC.* Univesity of Texas at Arlington. Arlington : s.n., 2009. Dissertação de Mestrado.
23. *Comparison of Dirac and H.264/AVC Coding Quality Using Objective Video Quality Measures.* **Emil Dumic, Mario Mustra, Sonja Grgic.** Chalkida : s.n., 2009. 16th International Conference on Systems, Signals and Image Processing - IWSSIP 2009.
24. *Comparison of Open and Free Video Compression Systems.* **Halbach, Till.** 2009. International Conference on Image Theory and Applications.
25. *Performance Comparison of Emerging Dirac Video Codec with H.264/AVC.* **Onthriar, K. Loo, K.K. Xue, Z.** Cote d'Azur : s.n., 2006. International Conference on Digital Telecommunications - ICDT '06.
26. *Semi-hierarchical motion estimation for the Dirac video codec.* **Tun, M., Loo, K.K., Cosmas, J.** Las Vegas, NV : s.n., 2008. 2008 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting.
27. *Performance Improvement in Motion Estimation of Dirac Wavelet based Video Codec.* **Ali, A., Ali, S.F., Khan, N., Masud, S.** Icheon : s.n., 2009. 9th International Symposium on Communications and Information Technology.
28. *An Enhanced Adaptive Rood Pattern Search Algorithm for Fast Block-Matching Motion Estimation.* **Zhao, H., Yu, X., Sun, J., Sun, C., Cong, H.** 2008. Congress on Image and Signal Processing (CISP'08).
29. *Block motion estimation using adaptive modified two-bit transform.* **Demir, B., Ertürk, S.** 2007, IET Image Processing, Vol. 1, pp. 215-222.
30. *Fast Optimal Motion Estimation Based on Gradient-Based Adaptive Multilevel Successive Elimination.* **Liu, S. W., Wei, S. D., Lai, S. H.** 2008, IEEE Trans. Circuits Syst. Video Technol., Vol. 18, pp. 263-267.
31. **Haykin, S.** *Sistemas de Comunicação.* s.l. : Bookman, 2004.
32. *Image Quality Assessment: From Error Visibility to Structural Similarity.* **Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P.** 2004, IEEE Trans. on Image Processing, Vol. 13, pp. 600-612.
33. **Silva, Fernando Silvestre da.** *Procedimentos para Medição e Minimização do Efeito de Bloco Decorrente do Processamento Digital de Imagens.* 2001. Tese de Mestrado.
34. **Clarke, R. J.** *Transform Coding of Images.* s.l. : Academic Press, 1985.
35. *A Theory for Multiresolution Signal Decomposition: the Wavelet Representation.* **Mallat, S.** 1989, IEEE Transactions on PAMI, Vol. 11, pp. 674-693.
36. **Gray, Karen L.** *The JPEG2000 Standard.* Technische Universitat Munchen.

37. *A New, Fast and Efficient Image Coded Based on Set Partitioning in Hierarchical Trees.* **Said, A. e Pearlman, W. A.** Junho de 1996, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 6, pp. 243-250. 3.
38. **Silva, Ana Lúcia Mendes Cruz Silvestre da.** *Procedimentos para Método Híbrido de Compressão de Imagens Digitais Utilizando Transformadas Wavelet e Codificação Fractal.* 2005.
39. *Rate-Distortion Optimization for Video Compression.* **Sullivan, G. J., Wiegand, T.** 1998, IEEE Signal Processing Magazine.
40. *Low-power VLSI design for motion estimation using adaptive pixel truncation.* **He, Z. L., Tsui, C. Y., Chan, K. K., Liou, M. L.** 2000, IEEE Trans. Circuits Syst. Video Technoloy, pp. 669-678.
41. **Richardson, I. E. G.** *H.264 and MPEG-4 Video Compression.* s.l. : John Wiley &Sons, 2004.
42. **BBC R&D.** The technology behind Dirac. [Online] [Citado em: 9 de Maio de 2010.] http://www.bbc.co.uk/rd/projects/dirac/technology.shtml.
43. —. Dirac Project at SourceForge. [Online] [Citado em: 9 de Maio de 2010.] http://sourceforge.net/project/showfiles.php?group_id=102564&package_id=119507.
44. —. Dirac Reference Software 1.0.2. [Online] [Citado em: 9 de Maio de 2010.] http://sourceforge.net/project/showfiles.php?group_id=102564&package_id=112141.
45. **Fraunhofer Institute for Telecommunications - Heinrich Hertz Institute.** H.264/AVC JM15.1 Reference Software. [Online] 9 de Junho de 2009. http://iphome.hhi.de/suehring/tml.
46. *On the Design of Pattern-based Block Motion.* **Hang, Jang-Jer Tsai and Hsueh-Ming.** 1, Janeiro de 2010, IEEE Trans. Circuits Syst. Video Technol., Vol. 20.
47. *Modeling of pattern-based block motion estimation and its applications.* **Hang, J.-J. Tsai and H.-M.** 1, Janeiro de 2009, IEEE Trans. Circuits Syst. Video Technol, Vol. 19, pp. 108-113.
48. **BBC R&D.** *Dirac Specification.* 2008. Versão 2.3.3.