

Universidade Federal do Rio de Janeiro

Escola Politécnica

Departamento de Eletrônica e de Computação

**Desenvolvimento de um Codificador de Vídeo HD em
Tempo Real com Complexidade Variável Usando a
Biblioteca x264**

Autor:

Allan Freitas da Silva

Orientador:

José Fernando Leite de Oliveira, D. Sc.

Orientador:

Prof. Eduardo Antônio Barros da Silva, Ph. D.

Examinador:

Prof. Fernando Manuel Bernardo Pereira, Ph. D.

Examinador:

Prof. Gelson Vieira Mendonça, Ph. D.

DEL

Agosto de 2013

Universidade Federal do Rio de Janeiro

Escola Politécnica

Departamento de Eletrônica e de Computação

**Desenvolvimento de um Codificador de Vídeo HD em
Tempo Real com Complexidade Variável Usando a
Biblioteca x264**

Autor:



Allan Freitas da Silva

Orientador:



José Fernando Leite de Oliveira, D. Sc.

Orientador:




Prof. Eduardo Antônio Barros da Silva, Ph. D.

Examinador:



Prof. Fernando Manuel Bernardo Pereira, Ph. D.

Examinador:



Prof. Gelson Vieira Mendonça, Ph. D.

DEL

Agosto de 2013

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Escola Politécnica - Departamento de Eletrônica e de Computação

Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária

Rio de Janeiro - RJ CEP 21949-900

Este exemplar é de propriedade da Universidade Federal do Rio de Janeiro, que poderá incluí-lo em base de dados, armazenar em computador, microfilmear ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es) e do(s) orientador(es).

AGRADECIMENTO

Agradeço primeiramente à minha família, que me deu formação educacional e apoio para que eu ingressasse na universidade. Também aprecio toda a preocupação que demonstraram em relação à conclusão do curso.

Agradeço aos meus amigos por todos os bons momentos ao longo desses anos. Agradeço em especial àqueles que contribuíram de alguma forma para este texto, com dicas de latex, traduções de textos e ajuda em geração de figuras.

Agradeço à minha namorada, que se mostrou paciente mesmo em meus momentos de pessimismo em relação aos prazos do trabalho. Sua atuação foi essencial para manter minha motivação.

Agradeço aos meus orientadores, que se mostraram sempre presentes, auxiliando quando necessário. Aprecio todo o esforço necessário para realizar as correções de forma rápida.

Finalmente, agradeço à banca examinadora, pela pronta resposta, e por ter se disposto a avaliar este trabalho.

RESUMO

Este trabalho tem por objetivo analisar a viabilidade da codificação de vídeos em *full HD* para o padrão H.264 via *software*, através da biblioteca x264. Esta biblioteca realiza a codificação a partir de um grande número de parâmetros, de modo que sua configuração não é trivial. Dessa forma, é proposta aqui uma metodologia de análise dos parâmetros, visando a determinar um subconjunto relevante dos mesmos que possibilite a codificação em tempo real com a máxima qualidade obtível para uma dada complexidade. Com base nos resultados, foi verificado que o sistema otimizado possui desempenho superior aos perfis de complexidade pré-definidos na biblioteca, com o ganho de qualidade, medido em termos de PSNR, superior a 1dB. Por fim, o sistema também se mostrou capaz de realizar a codificação com velocidade suficiente para que a exibição dos quadros seja feita em tempo real.

Palavras-Chave: vídeo digital, compressão de vídeo, H.264, x264, MEncoder.

ABSTRACT

This work has the objective of analysing the viability of full HD video encoding using the H.264 standard through a software based on x264 library. This library requires the configuration of several parameters in order to encode a video, and this configuration is not trivial. In this work we propose a method for parameter analysis in order to determine a meaningful subset which allows maximum quality for real time encoding, given complexity restrictions. Based on the obtained results, it has been verified that the optimized system has superior performance compared to pre-defined complexity profiles, with PSNR gain higher than 1 dB. Finally, the system proved itself capable of encoding with enough speed so that the frame reproduction can be done in real time.

Key-words: digital video, video compression, H.264, x264, MEncoder.

SIGLAS

UFRJ - Universidade Federal do Rio de Janeiro

DEL - Departamento de Engenharia Eletrônica

LPS - Laboratório de Processamento de Sinais

HD - *High Definition*

PSNR - *Peak Signal-to-Noise Ratio*

GOP - *Group of Pictures*

KLT - *Karhunen-Loève Transform*

DCT - *Discrete Cossine Transform*

EOB - *End of Block*

JVT - *Joint Video Team*

ITU-T - *Telecommunication Standardization Sector of the International Telecommunications Union*

MPEG - *Moving Pictures Experts Group*

CAVLC - *Context Adaptative Variable Length Coding*

CABAC - *Context Adaptative Binary Arithmetic Coding*

SAD - *Sum of Absolute Differences*

SATD - *Sum of Absolute Transformed Differences*

SSIM - *Structural Similarity*

Sumário

Lista de Figuras	x
Lista de Tabelas	xiii
1 Introdução	1
1.1 Tema	1
1.2 Delimitação	1
1.3 Justificativa	1
1.4 Objetivo	2
1.5 Metodologia	2
1.6 Descrição	3
2 Compressão de Vídeos	5
2.1 Conceitos Básicos	5
2.2 Predição de Movimentos	6
2.3 Transformação	8
2.4 Quantização	10
2.5 Codificação por Entropia	11
3 Padrão H.264	13
3.1 Visão Geral	13
3.2 Predição Intra	13
3.3 Predição Inter	16
3.3.1 Tamanhos das Partições	16
3.3.2 Interpolação a Nível de Subpíxel	16
3.3.3 Quadros de Referência	18

3.4	Transformadas	20
3.5	Codificação	21
3.6	Filtro de Desblocagem	22
3.7	Perfis	22
3.8	Níveis	24
4	Biblioteca x264	26
4.1	Definição	26
4.2	Controle de Taxa	27
4.2.1	Duas Passadas	27
4.2.2	Taxa de Bits Média Constante	27
4.2.3	Quantização Constante	28
4.2.4	Qualidade Constante	28
4.3	Algoritmos e Modos de Atuação	28
4.3.1	Partições	28
4.3.2	Estimação de Movimentos	29
4.3.3	Estimação a Nível de Subpíxel	39
4.3.4	Uso da Informação de Crominância	40
4.3.5	Número de Quadros de Referência	40
4.3.6	Referências Múltiplas	40
4.3.7	Predição Ponderada para Quadros P	40
4.3.8	Predição Ponderada para Quadros B	41
4.3.9	Número de Quadros B	41
4.3.10	Uso Adaptativo de Quadros B	41
4.3.11	Uso de Quadros B como Referência	42
4.3.12	Predição Direta	42
4.3.13	Transformadas 8×8	43
4.3.14	Codificação Aritmética	43
4.3.15	Árvore de Macroblocos	43
4.3.16	Número de Quadros para a Árvore de Macroblocos	44
4.3.17	Filtro de Desblocagem	44
4.3.18	Quantização Adaptativa	44
4.3.19	Codificação por Treliça	44

4.3.20	Perfis	45
4.3.21	Opções Pré-definidas	45
4.3.22	Multiprocessamento	45
4.3.23	PSNR	45
4.3.24	SSIM	46
4.4	Trabalhos Correlatos	46
5	Resultados Experimentais	47
5.1	Descrição do Experimento	47
5.2	Base de Dados	48
5.3	Problemas Encontrados	48
5.4	Análise Individual de Parâmetros	50
5.4.1	Estimação de Movimentos	50
5.4.2	Filtro de Desblocagem	51
5.4.3	Estimação a Nível de Subpíxel	51
5.4.4	Número de Quadros de Referência	52
5.4.5	Partições	52
5.4.6	Transformadas 8×8	54
5.4.7	Codificação Aritmética	55
5.4.8	Quadros B	55
5.4.9	Modos Ótimos	58
5.5	Análise Combinacional de Parâmetros	58
5.6	Codificação em Tempo Real	66
6	Conclusões	69
6.1	Considerações Finais	69
6.2	Trabalhos Futuros	70
	Referências Bibliográficas	71

Lista de Figuras

2.1	Diagrama de blocos do sistema de compressão de vídeos.	6
2.2	Exemplo de estimação de movimentos.	8
2.3	Estimação de movimentos.	9
2.4	Matrizes de base para um bloco de dimensões 8×8	10
2.5	Varredura Zig-zag.	11
3.1	Diagrama de blocos do codificador de H.264.	14
3.2	Blocos disponíveis para predição intra.	14
3.3	Modos de predição intra para blocos de 4×4 píxeis.	15
3.4	Modos de predição intra para blocos de 16×16 píxeis.	15
3.5	Modos de partições dos macroblocos.	16
3.6	Exemplo de divisão de um bloco em vários tipos de partição.	17
3.7	Interpolação dos píxeis.	17
3.8	Exemplos de estimação de movimentos a nível de subpíxel.	18
3.9	Estimação de movimentos para um quadro P . A lista 0 contém quadros passados. A lista 1, quadros futuros.	19
3.10	Estimação de movimentos para um quadro B . A lista 0 contém quadros passados. A lista 1, quadros futuros.	20
3.11	Exemplo de compressão de um quadro sem filtro de desblocagem. . .	23
3.12	Exemplo de compressão de um quadro com filtro de desblocagem. . .	23
3.13	Diagrama dos conjuntos de ferramentas pertencentes a cada perfil. . .	24
4.1	Passo a passo do algoritmo de estimação de movimentos <i>diamond</i> . . .	31
4.2	Passo a passo do algoritmo de estimação de movimentos <i>hexagon</i> . . .	32
4.3	1º passo do algoritmo de estimação de movimentos <i>uneven multi-hexagon</i> : busca <i>unsymmetrical cross</i>	33

4.4	2º passo do algoritmo de estimação de movimentos <i>uneven multi-hexagon</i> : busca exaustiva em um retângulo de 5 x 5 píxeis.	34
4.5	3º passo do algoritmo de estimação de movimentos <i>uneven multi-hexagon</i> : busca <i>uneven multi-hexagon</i>	35
4.6	4º passo do algoritmo de estimação de movimentos <i>uneven multi-hexagon</i> : busca <i>hexagon</i>	36
4.7	5º passo do algoritmo de estimação de movimentos <i>uneven multi-hexagon</i> : busca <i>hexagon</i>	37
4.8	6º passo do algoritmo de estimação de movimentos <i>uneven multi-hexagon</i> : busca <i>diamond</i>	38
5.1	Taxa de quadros atingida pelo codificador em função do número de quadros codificados, para diversas realizações.	49
5.2	Quantidade de CPU consumida pelo MEncoder, ao utilizar 4 <i>threads</i>	50
5.3	Nuvem de pontos e sua respectiva curva ótima para o vídeo 2, com taxa de 1 Mb/s.	59
5.4	Nuvem de pontos e sua respectiva curva ótima para o vídeo 2, com taxa de 10 Mb/s.	60
5.5	Nuvem de pontos e sua respectiva curva ótima para o vídeo 6, com taxa de 2 Mb/s.	61
5.6	Nuvem de pontos e sua respectiva curva ótima para o vídeo 6, com taxa de 5 Mb/s.	62
5.7	Nuvem de pontos e sua respectiva curva ótima para o vídeo 2, com taxa de 1 Mb/s.	62
5.8	Nuvem de pontos e sua respectiva curva ótima para o vídeo 2, com taxa de 10 Mb/s.	63
5.9	Nuvem de pontos e sua respectiva curva ótima para o vídeo 6, com taxa de 2 Mb/s.	63
5.10	Nuvem de pontos e sua respectiva curva ótima para o vídeo 6, com taxa de 5 Mb/s.	64
5.11	Exemplo da descontinuidade na qualidade obtida.	64
5.12	Configurações ótimas aplicadas no vídeo 2, com taxa de 1 Mb/s e 4 <i>threads</i>	66

5.13	Configurações ótimas aplicadas no vídeo 2, com taxa de 10 Mb/s e 4 <i>threads</i>	67
5.14	Configurações ótimas aplicadas no vídeo 6, com taxa de 2 Mb/s e 4 <i>threads</i>	67
5.15	Configurações ótimas aplicadas no vídeo 6, com taxa de 5 Mb/s e 4 <i>threads</i>	68

Lista de Tabelas

3.1	Níveis no padrão H.264	25
5.1	Valores obtidos de qualidade e velocidade de compressão para o algoritmo de estimação de movimentos, com o vídeo 1 e taxa de 10Mb/s .	51
5.2	Valores obtidos de qualidade e velocidade de compressão para o algoritmo de estimação de movimentos, com o vídeo 2 e taxa de 10Mb/s .	51
5.3	Valores obtidos de qualidade e velocidade de compressão para o filtro de desbloqueamento, com o vídeo 2 e taxa de 10Mb/s	51
5.4	Valores obtidos de qualidade e velocidade de compressão para diferentes modos de predição a nível de subpíxel, com o vídeo 3 e taxa de 10Mb/s	52
5.5	Valores obtidos de qualidade e velocidade de compressão para várias quantidades de quadros a serem usados como referência, com o vídeo 3 e taxa de 10Mb/s	52
5.6	Valores obtidos de qualidade e velocidade de compressão para várias quantidades de quadros a serem usados como referência, com o vídeo 4 e taxa de 10Mb/s	53
5.7	Valores obtidos de qualidade e velocidade de compressão para os modos de partição dos quadros, com o vídeo 2 e taxa de 10Mb/s	53
5.8	Valores obtidos de qualidade e velocidade de compressão para os modos de partição dos quadros, com o vídeo 3 e taxa de 10Mb/s	53
5.9	Valores obtidos de qualidade e velocidade de compressão para os modos de partição dos quadros, com o vídeo 4 e taxa de 10Mb/s	54
5.10	Valores obtidos de qualidade e velocidade de compressão para o uso de DCT com tamanho 8×8 , com o vídeo 3 e taxa de 10Mb/s	54

5.11	Valores obtidos de qualidade e velocidade de compressão para do CABAC, com o vídeo 3 e taxa de 10Mb/s	55
5.12	Valores obtidos de qualidade e velocidade de compressão para a máxima quantidade de quadros B consecutivos sem utilizar algoritmo adaptativo de decisão, com o vídeo 3 e taxa de 10Mb/s	55
5.13	Valores obtidos de qualidade e velocidade de compressão para a máxima quantidade de quadros B consecutivos utilizando o algoritmo adaptativo de decisão 1 , com o vídeo 3 e taxa de 10Mb/s	56
5.14	Valores obtidos de qualidade e velocidade de compressão para a máxima quantidade de quadros B consecutivos utilizando o algoritmo adaptativo de decisão 2 , com o vídeo 3 e taxa de 10Mb/s	56
5.15	Valores obtidos de qualidade e velocidade de compressão para a máxima quantidade de quadros B consecutivos permitindo que quadros B sejam usados como referência. com o vídeo 3 e taxa de 10Mb/s	57
5.16	Valores obtidos de qualidade e velocidade de compressão para a máxima quantidade de quadros B consecutivos permitindo que quadros B sejam usados como referência e utilizando o algoritmo adaptativo de decisão 1 , com o vídeo 1 e taxa de 10Mb/s	57
5.17	Valores obtidos de qualidade e velocidade de compressão para máxima a quantidade de quadros B consecutivos permitindo que quadros B sejam usados como referência e utilizando o algoritmo adaptativo de decisão 1 , com o vídeo 2 e taxa de 10Mb/s	57
5.18	Melhores modos de operação para os parâmetros testados.	58
5.19	Valores assumidos pelos parâmetros testados combinadamente.	59
5.20	Configurações ótimas para as taxas de <i>bits</i> testadas. Cada linha mostra as configurações obtidas para cada taxa em ordem decrescente de complexidade, da esquerda para a direita	65

Capítulo 1

Introdução

1.1 Tema

O tema deste trabalho trata da compressão de vídeo digital e uma de suas implementações. Assim, pretende-se avaliar a possibilidade de uma solução para a codificação de vídeos em alta resolução via *software*, para o padrão mais usado hoje nos sistemas de compressão de sinais de vídeos, o H.264/AVC [1]. A hipótese é que ao se fazer uso de bibliotecas mais modernas e de um computador suficientemente rápido, é possível realizar a compressão de vídeos em tempo real.

1.2 Delimitação

Este trabalho vai se focar no estudo do H.264, que é o padrão escolhido para integrar o sistema de TV digital no Brasil. Como implementação via *software*, o codec escolhido foi o x264 [2], a partir da versão presente no MEncoder [3], onde serão estudados e testados os principais parâmetros do mesmo. Também serão desenvolvidos pequenos *scripts* em C/C++ ou *bash*, de forma a facilitar e automatizar as simulações.

1.3 Justificativa

Na última década, o crescimento tecnológico causou um grande avanço na compressão de vídeos digitais, possibilitando seu uso em diversas aplicações. Essa

evolução, acompanhada da evolução na eletrônica, possibilitou o surgimento do H.264, que é considerado o estado da técnica em compressão de vídeos, e apresenta uma grande eficiência de compressão ao custo de poder computacional e complexidade nas técnicas.

O padrão citado apresenta diversos perfis de operação, podendo ser usado em sistemas com diversos valores de taxas de *bits* ou níveis de poder computacional, cada qual com uma certa eficiência. Devido a esse grande número de possibilidades, o processo de configuração de um codificador para H.264 não é trivial. Mas, superada essa dificuldade de configuração, o H.264 permite reduzir a complexidade computacional de codificação ao custo de eventuais perdas na qualidade do vídeo obtido.

Como implementação desse padrão, uma das principais é o x264. Dentre as vantagens presentes nesse programa, pode-se citar: código fonte aberto, alto desempenho em termos de velocidade [4], chegando a ser 50 vezes mais rápido que a referência padrão, o *Joint Model* [5], e possibilidade de ajuste dos vários modos de operação do H.264. Porém, este grande número de parâmetros também dificulta seu uso.

1.4 Objetivo

Este projeto se propõe então a fazer um estudo dos modos de operação da biblioteca x264, visando a estabelecer um conjunto de configurações que realizem uma codificação em H.264, com a restrição que seja em tempo real, e que seja próxima da ótima em termos de distorção *versus* complexidade para diversas taxas, adaptando a complexidade computacional requerida ao *hardware* em uso. Desse modo, a otimização do *software* permitirá que a codificação em tempo real para vídeos em resolução *full HD*, isto é, de 1920 x 1080 píxeis, seja realizada com a máxima qualidade obtenível.

1.5 Metodologia

O trabalho foi dividido em diversas etapas, realizadas de forma sequencial. Inicialmente foi realizado um estudo do padrão H.264 e dos diversos comandos dos

software x264 e MEncoder.

Uma maneira preliminar de encontrar a curva ótima seria fazer um levantamento dos valores de qualidade, em termos da relação sinal-ruído de pico (PSNR), e complexidade, em termos da velocidade em que o programa realizou a codificação (em quadros por segundo), obtidos por cada combinação entre os valores assumidos pelos parâmetros do sistema. Entretanto, o x264 apresenta tantos parâmetros que essa simulação não é viável de ser concluída em tempo hábil. Dessa forma, foi selecionado um subconjunto a ser testado, com diversos valores que os mesmos podem assumir, de modo a reduzir a quantidade total de combinações.

Essa etapa consistiu da análise individual dos parâmetros. A partir da variação dos mesmos, foram feitas medidas de qualidade e de complexidade, visando a uma codificação no perfil *high*. Com base nos resultados, poder-se-ão fixar alguns desses parâmetros em determinados valores, sem perda muito significativa de otimização do sistema. Além disso, também foram determinados quais parâmetros que, variando-os, permitem criar uma troca entre qualidade e velocidade de compressão.

De posse da informação de quais parâmetros devem ou não ser variados, foram feitos vários testes com as combinações relevantes. Gerou-se então uma nuvem de pontos no plano qualidade versus complexidade, onde foi encontrada a curva com as configurações mais eficientes no sentido de maximizar a qualidade e minimizar a complexidade.

1.6 Descrição

Este capítulo apresentou uma ambientação do trabalho apresentado. Foram mostrados os objetivos do trabalho e a metodologia para solução.

O Capítulo 2 mostra o básico da compressão de vídeos. São apresentadas as diversas técnicas presentes na codificação.

Já o Capítulo 3 apresenta um complemento ao Capítulo 2. São exibidos os aprimoramentos existentes no padrão H.264.

O Capítulo 4 trata da biblioteca x264. Esta biblioteca implementa diversos algoritmos relacionados à compressão de vídeos.

Por sua vez, os gráficos das simulações realizadas são exibidos no Capítulo 5.

São mostrados os resultados obtidos para cada parâmetro testado, as curvas ótimas e os problemas encontrados.

No Capítulo 6, são exibidos as conclusões e os possíveis desdobramentos de trabalhos futuros.

Capítulo 2

Compressão de Vídeos

Neste capítulo, serão apresentadas técnicas envolvidas na compressão de vídeos digitais.

2.1 Conceitos Básicos

Um vídeo digital é composto de quadros exibidos sequencialmente a uma certa velocidade, cada qual formado por uma quantidade de píxeis. Entretanto, este tipo de sinal normalmente carrega uma quantidade enorme de informação, de modo que é necessário um grande número de *bits* para representá-lo, como pode ser visto a partir de um cálculo simples. Considerando uma resolução de 640x480 píxeis com imagens em preto e branco, para quadros sendo exibidos a uma taxa de 30 fps (quadros por segundo), um vídeo de 1 hora de duração ocuparia aproximadamente 31 Gb, necessitando por exemplo de 7 DVDs para armazená-lo.

Por essa razão, o uso dos vídeos digitais só foi popularizado com o surgimento dos primeiros padrões de compressão, principalmente o MPEG-2 [6]. Estes padrões especificam técnicas que visam a diminuir a quantidade de *bits* necessária para representar um vídeo digital, a partir da redução na redundância de informação presente em diversas características das sequências de vídeos, e eliminando parte dessa informação da forma menos perceptível possível.

Os diversos padrões de codificação apresentam diferentes técnicas, porque o crescimento tecnológico permitiu que os padrões posteriores adotassem algoritmos mais complexos. Porém, em geral estes são derivados a partir das primeiras técnicas

encontradas no padrão H.261 [7]. Com isso, podemos dividir o processo de compressão de vídeos em 4 grandes blocos: predição de movimentos, transformação, quantização e codificação por entropia. Um diagrama de blocos do sistema simplificado está presente na Figura 2.1.

Estes blocos visam a reduzir a quantidade de informação, explorando as diversas formas de redundância presentes nos dados, ou descartando parte da informação de forma menos perceptível. A redundância temporal surge porque existe grande similaridade entre quadros próximos, uma vez que não existem grandes deslocamentos para intervalos curtos de tempo. Já a redundância espacial está presente porque píxeis vizinhos tendem a possuir valores próximos de cor, já que os objetos normalmente representados não possuem variações bruscas de cor píxel a píxel. A redundância psicoacústica existe porque nem todos os detalhes presentes são percebidos pelo sistema visual humano. Por fim, a redundância entrópica está relacionada às probabilidades de ocorrência dos diversos símbolos, pois, quanto mais provável é um símbolo, menos informação nova ele transmite.

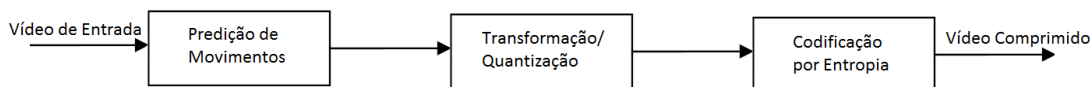


Figura 2.1: Diagrama de blocos do sistema de compressão de vídeos.

2.2 Predição de Movimentos

Para reduzir a redundância temporal, utiliza-se a chamada codificação inter-quadros, isto é, realiza-se para cada quadro uma codificação que utilize informações referentes a outros quadros do vídeo, eliminando a informação em comum entre os mesmos. Este processo é um dos que mais contribuem para as taxas de compressão de vídeos digitais, apresentando desempenho superior à simples codificação individual das imagens.

A Figura 2.2 ilustra o conceito explorado pela codificação inter-quadros. Sendo as Figuras 2.2a e 2.2b dois quadros em sequência, a diferença entre os mesmos está retratada na Figura 2.2c, em escala de cinza, onde o nível de cinza dominante

representa as diferenças próximas de zero, e os níveis próximos de branco e de preto representam, respectivamente, as diferenças muito positivas e muito negativas. A esta diferença damos o nome de resíduo. A figura mostra que a maior parte dos píxeis apresenta um tom de cinza. Assim, uma vez que o primeiro quadro já tenha sido obtido, pouca informação precisa ser adicionada para gerar o segundo quadro.

A análise da Figura 2.2 também mostra que os trechos que possuem maior diferença são os em que ocorre maior quantidade de movimento entre um quadro e outro, ao passo que as diferenças pequenas ocorrem basicamente nos trechos de fundo estático. Assim, o processo de codificação pode ser aprimorado se for possível realizar uma compensação do movimento executado pelos objetos antes do cálculo da diferença entre os quadros. Esta técnica de busca dos deslocamentos entre um quadro e outro é chamada de predição ou estimação de movimentos.

Deve-se ressaltar que esta técnica não pode ser aplicada em todos os quadros. Alguns quadros, denominados quadros **I**, devem ser codificados de forma independente, para impedir que um erro de transmissão em quadros anteriores se propague para quadros posteriores, e também para permitir acesso aleatório, ou seja, que a decodificação a partir de um quadro **I** qualquer não dependa de informações anteriores. Definiu-se o intervalo entre dois quadros **I** por GOP (*group of pictures*).

O processo de estimação de movimentos é elucidado pela Figura 2.3. O quadro que se deseja codificar com esta técnica, denominado quadro **P**, é dividido em macroblocos de 16×16 píxeis, que são subdivididos em blocos 8×8 . Para cada bloco, é executado um algoritmo que faça uma busca em um quadro de referência anterior, para encontrar um bloco que seja bastante similar ao bloco que está sendo analisado. Nesse caso, o codificador precisa enviar, além da diferença, alguma informação sobre qual bloco do quadro anterior foi utilizado. A essa informação foi designado o nome de vetor de movimento. Posteriormente, também foi definido um tipo de quadro que pode utilizar a predição a partir de quadros **P** ou **I** futuros, passados ou uma média de ambos, denominado quadro **B**. A Figura 2.2d mostra um exemplo de estimação de movimento, que deve ser comparada com a Figura 2.2c.



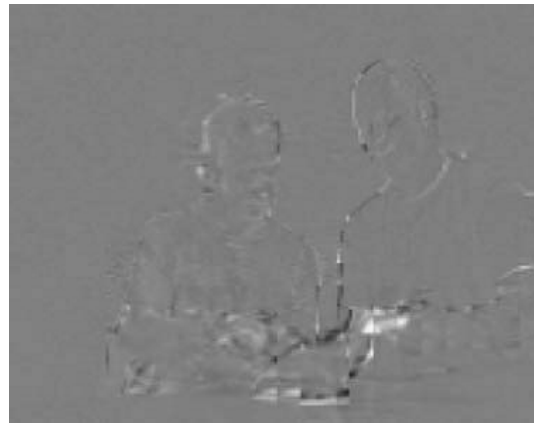
(a) Quadro n



(b) Quadro n+1



(c) Diferença entre os quadros



(d) Diferença entre os quadros, após estimativa de movimentos com blocos 8 x 8

Figura 2.2: Exemplo de estimação de movimentos. [11]

2.3 Transformação

A compressão de vídeos também utiliza de codificação por transformadas, de maneira semelhante ao existente em compressão de imagens. Esta técnica tem por objetivo eliminar dados redundantes entre os píxeis. Isto ocorre a partir de um mapeamento dos píxeis em alguns coeficientes, de forma a concentrar a energia em alguns destes, e descorrelacionar as informações dos píxeis, reduzindo a redundância espacial. Os coeficientes são então quantizados, de modo que alguns possam ser descartados sem perda significativa de qualidade.

A transformada ótima para esta aplicação é a Transformada de Karhunen-

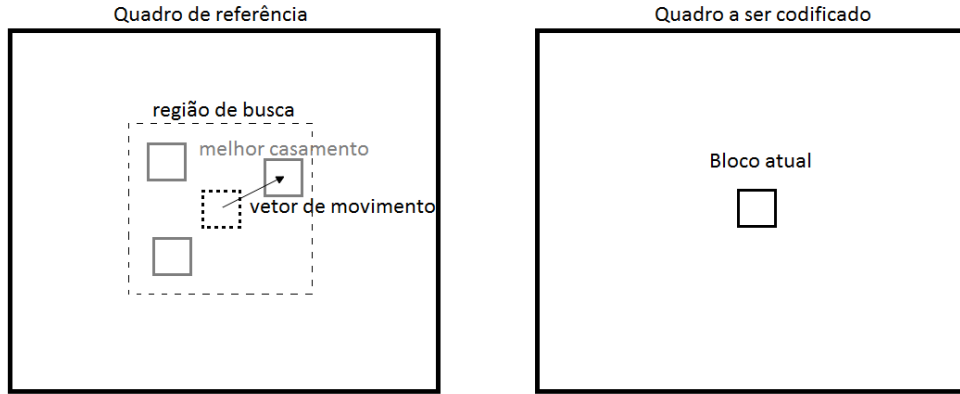


Figura 2.3: Estimação de movimentos.

Loève, mais conhecida por KLT [8]. Entretanto, a implementação desta transformada não é trivial, visto que ela precisa ser calculada para cada quadro. Dessa forma, comumente se utiliza a DCT [9], isto é, Transformada Cosseno Discreta, que se aproxima da ótima para o conjunto típico de imagens normalmente utilizadas, e é fixa, sendo portanto de fácil implementação. Após dividir a um quadro em blocos, a DCT é aplicada bi-dimensionalmente a cada bloco. Para um bloco de dimensões $N \times N$, ela pode ser descrita por uma matriz de transformação \mathbf{M} da seguinte forma:

$$\mathbf{M}_{ij} = \begin{cases} \sqrt{\frac{1}{N}} \cos \frac{(2j+1)i\pi}{2N}, & i = 0; j = 0,1,\dots,N-1 \\ \sqrt{\frac{2}{N}} \cos \frac{(2j+1)i\pi}{2N}, & i = 1,2,\dots,N-1; j = 0,1,\dots,N-1. \end{cases} \quad (2.1)$$

A matriz de transformação dada pela Eq. 2.1, quando aplicada nas linhas e nas colunas de um bloco $N \times N$, gera uma matriz de mesma dimensão, cujos elementos podem ser considerados pesos de um conjunto de matrizes de base. Dessa forma, a transformação gera uma projeção das imagens em um conjunto de bases, ou de maneira análoga, os diversos blocos da a imagem podem ser representados pela combinação linear de diversos blocos-bases, cada qual com um determinado peso. A Figura 2.4 exhibe um exemplo de conjunto de matrizes de base, para blocos 8×8 . Uma análise desta figura mostra que a decomposição nas bases mostradas se assemelha a uma decomposição espectral da imagem, já que o conjunto de bases apresenta componentes de maior frequência horizontal e/ou vertical quando se varia, respectivamente, a posição da coluna e/ou da linha do peso correspondente.

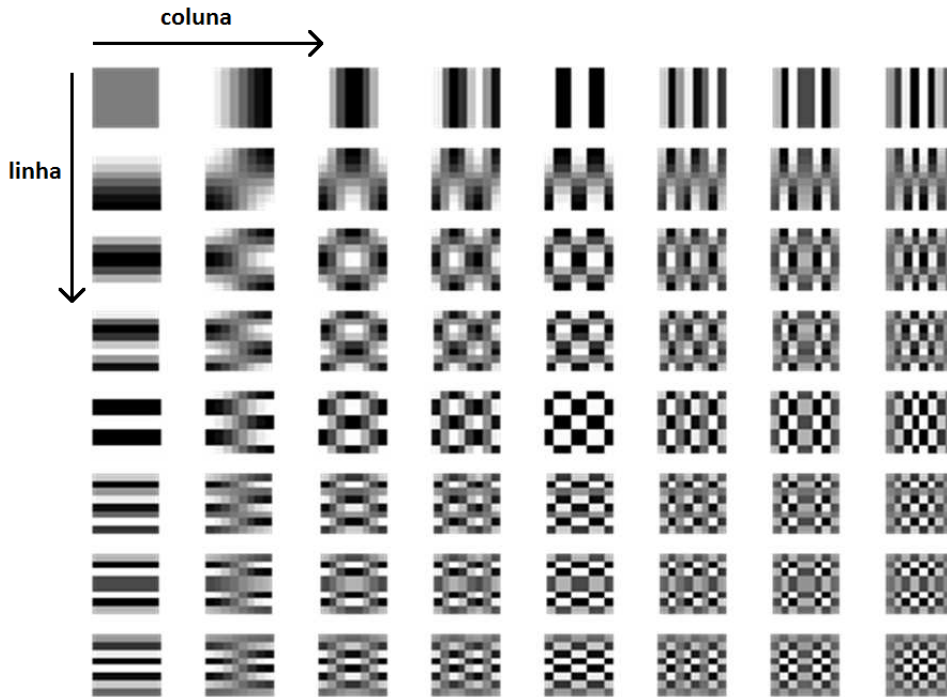


Figura 2.4: Matrizes de base para um bloco de dimensões 8×8 .

2.4 Quantização

Os coeficientes obtidos pela DCT foram gerados a partir de uma matriz, dada pela Equação 2.1, que pode assumir diversos valores reais, de modo que o resultado da transformação em geral também segue este padrão. De maneira a utilizar estes valores no âmbito digital, é necessário que os mesmos sejam quantizados, para posteriormente serem digitalizados. Além disso, a quantização também permite descartar parte da informação, gerando uma compressão com perdas, mas que permite atingir uma taxa de bits menor que a entropia. A maneira mais simples de se fazer isto é utilizando um passo de quantização constante, e arredondando os coeficientes em relação a este passo. Por exemplo, supondo que se tenha obtido a seguinte resposta para os coeficientes da DCT:

514, 0.5, 0.7, 28.6, 1.1, 16.9, 0.2, 2.4, 1.3, 0.8, 17.3, 0.4, 0.2, 19.2, 1.5, 0.2, 0.3, 0.1, ...

Utilizando passo de quantização valendo 16, os valores quantizados dos coeficientes serão:

512, 0, 0, 32, 0, 16, 0, 0, 0, 0, 16, 0, 0, 16, 0, 0, 0, 0, ...

2.5 Codificação por Entropia

O bloco de transformação a partir da DCT foi responsável por gerar coeficientes que representam um sistema de frequências espaciais, que depois foram quantizados. Para imagens naturais, em geral, os coeficientes relativos às menores frequências espaciais vão ter maior energia, ao passo que os outros terão maior probabilidade de ter valores pequenos. De forma a aproveitar esta estatística, os coeficientes são reordenados, para agrupar os coeficientes com maior probabilidade de serem não-nulos. A Figura 2.5 mostra o método utilizado em processamento de imagens para a reordenação: a Varredura Zig-zag, que pode ser aplicada tanto em sistemas com varredura progressiva quanto em varredura entrelaçada.

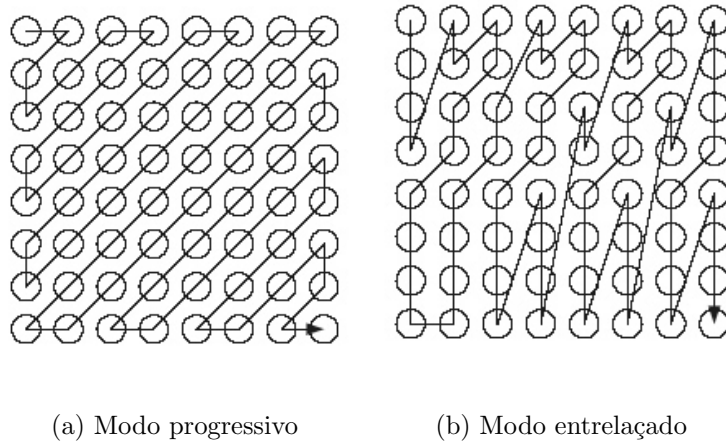


Figura 2.5: Varredura Zig-zag.

Feito isso, a codificação ocorre a partir de um algoritmo de Zeros Corridos ou Zeros Consecutivos, onde se conta a quantidade de coeficientes nulos entre coeficientes não-nulos, com um símbolo representando que todos os coeficientes à direita são nulos, chamado *end of block* ou EOB. Assim, tomando o exemplo anterior, se os coeficientes quantizados forem reordenados da seguinte forma:

512, 0, 0, 32, 0, 16, 0, 0, 0, 0, 16, 0, 0, 16, 0, 0, 0, 0, ...

A codificação será:

$$(0, 512)(2, 32)(1, 16)(4, 16)(2, 16)(EOB)$$

O processo de compressão de imagens e vídeos é concluído com uma codificação eficiente dos dados gerados. Todos os símbolos e parâmetros devem ser codificados em *bits*. Assim, o codificador utiliza técnicas para maximizar a quantidade de informação por *bit*, reduzindo a redundância entrópica. Isto é feito a partir de códigos de comprimento variável, como o Código de Huffman.

Capítulo 3

Padrão H.264

3.1 Visão Geral

O padrão de codificação H.264 foi desenvolvido pela *Joint Video Team (JVT)*, formada pela colaboração entre membros dos grupos *ITU-T Video Coding Experts Group* e *MPEG Moving Pictures Experts Group*. O objetivo era desenvolver um padrão que apresentasse uma melhoria de até 50 % em economia de taxa com relação aos padrões anteriores, mas com a mesma qualidade percebida, valendo-se de mais capacidade computacional nos codificadores e decodificadores. Além disso, o padrão deveria ser capaz de oferecer suporte a diversos tipos de aplicação, entre as quais podemos citar: aplicações com diferentes taxas de *bits*, com baixo ou alto retardo ou com maiores taxa de erros. Em outubro de 2003, foi aprovada a versão final do padrão citado.

O H.264 possui estrutura semelhante aos padrões anteriores, com diversos refinamentos. A Figura 3.1 apresenta um diagrama de blocos do codificador, que é uma evolução do mostrado na Figura 2.1. As seções a seguir mostram as melhorias implementadas no sistema. Maiores informações sobre o H.264 podem ser vistas em [11].

3.2 Predição Intra

Uma das principais adições à compressão é a possibilidade de realizar uma codificação eficiente sem utilizar as informações de outros quadros, o que é bastante

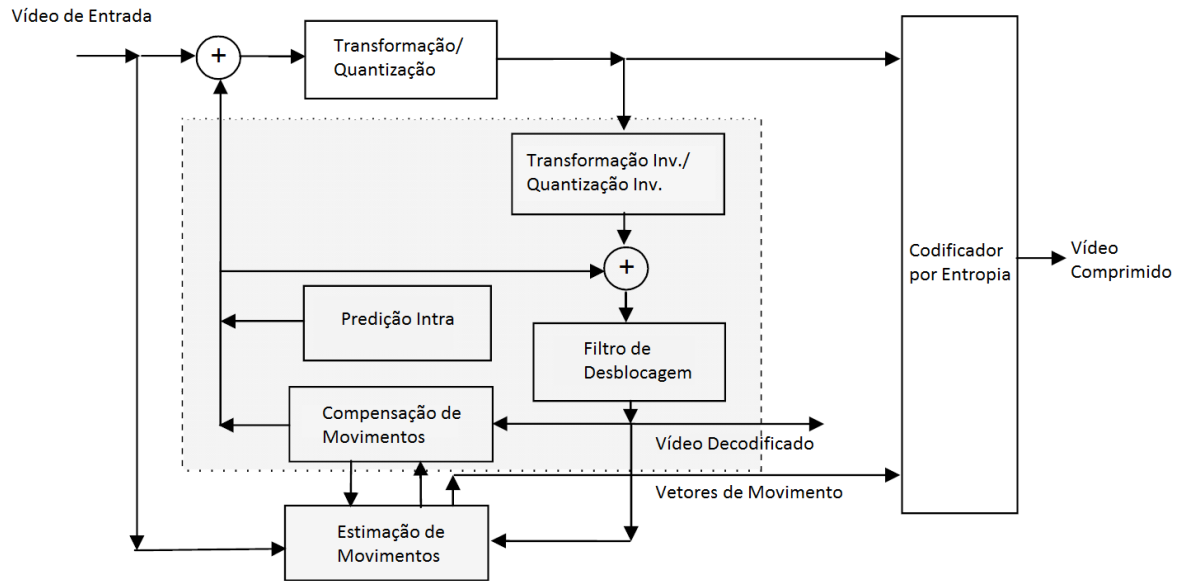


Figura 3.1: Diagrama de blocos do codificador de H.264.

útil nos quadros **I**. Na codificação, a imagem é dividida em macroblocos de 16×16 píxeis. Estes macroblocos podem ser divididos em blocos 4×4 , 8×8 ou mantidos com o tamanho original, à critério do codificador.

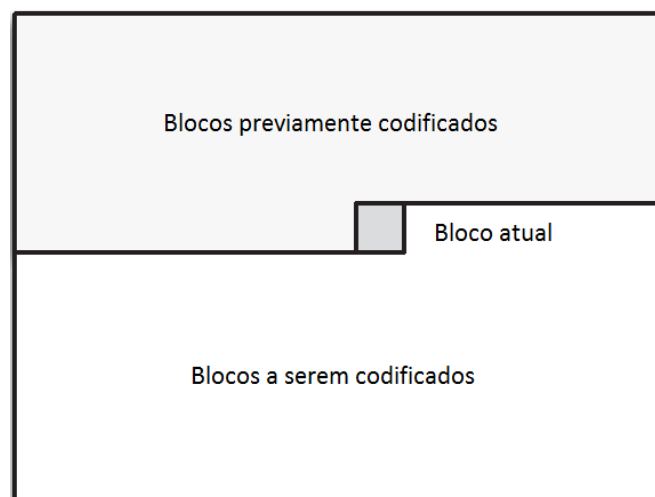


Figura 3.2: Blocos disponíveis para predição intra.

Como os macroblocos são transmitidos sequencialmente, na reconstrução de um determinado bloco já foram obtidos os blocos à esquerda e acima na imagem, tal como mostra a Figura 3.2. A predição intra consiste no uso da informação contida nesses blocos previamente decodificados para realizar uma estimativa do bloco a ser

codificado, que é descontada do bloco original, e só a diferença é quantizada. Isso fornece um método mais simples de reduzir a redundância espacial.

O padrão H.264 especifica alguns modos de predição a partir dos píxeis acima e à esquerda dos blocos. Na Figura 3.3, estão contidos todos os modos de predição para blocos 4×4 de luminância, que são análogos para blocos 8×8 . Já a Figura 3.4 mostra os modos de predição para blocos 16×16 . Para a crominância, os blocos de 8×8 têm predição igual a dos blocos 16×16 da luminancia, e os blocos menores têm predição igual a dos modos 4×4 de luminância.

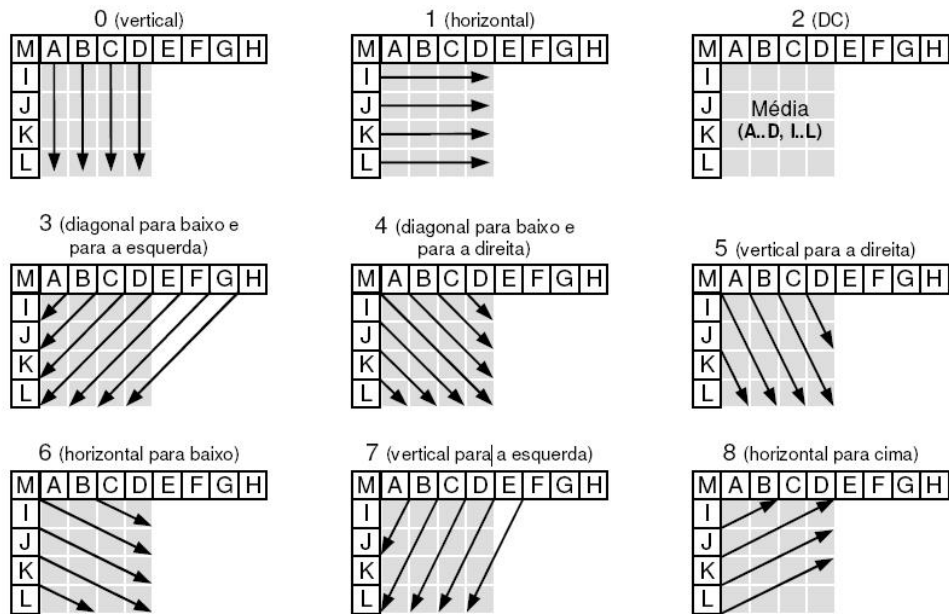


Figura 3.3: Modos de predição intra para blocos de 4×4 píxeis.

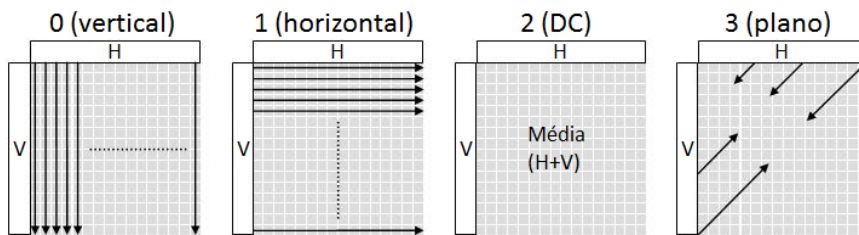


Figura 3.4: Modos de predição intra para blocos de 16×16 píxeis.

3.3 Predição Inter

O módulo de predição de movimentos, ou inter quadros, também sofreu uma série de melhorias em sua implementação, com a adição de novos modos de operação, que serão abordados nas subseções a seguir:

3.3.1 Tamanhos das Partições

Um dos avanços na eficiência de compressão surgiu a partir da possibilidade de escolher diversas partições dos macroblocos. Cada macrobloco pode ser dividido em alguns tipos de bloco, até o tamanho de 8×8 píxeis, e os blocos 8×8 também podem ser subdivididos em blocos menores de até 4×4 , como mostra a Figura 3.5. As opções de escolha permitem ao codificador se adequar aos detalhes e movimentos existentes no vídeo, aperfeiçoando a compressão, como mostra a Figura 3.6, mas sem gastar *bits* desnecessários com informações de controle, que ocorreria se todos os blocos fossem do menor tamanho possível.

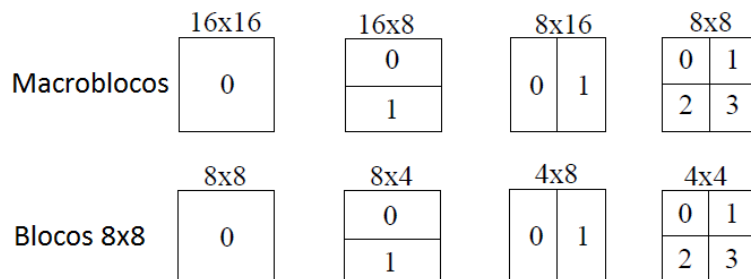


Figura 3.5: Modos de partições dos macroblocos.

3.3.2 Interpolação a Nível de Subpíxel

Também para melhoria da estimação de movimentos, é prevista uma interpolação dos píxeis dos quadros de modo aumentar a precisão dos vetores de movimento, para valores menores que o píxel inteiro, podendo ser de $1/2$ ou $1/4$ de píxel. Esta medida acarreta em um aumento tanto na quantidade de informação presente no vetor de movimentos quanto no tempo necessário para encontrar o melhor casamento entre os blocos. Tomando por base o exemplo apresentado na Figura 3.7, para $1/2$ píxel o cálculo realizado para estimar o valor de \mathbf{b} é:

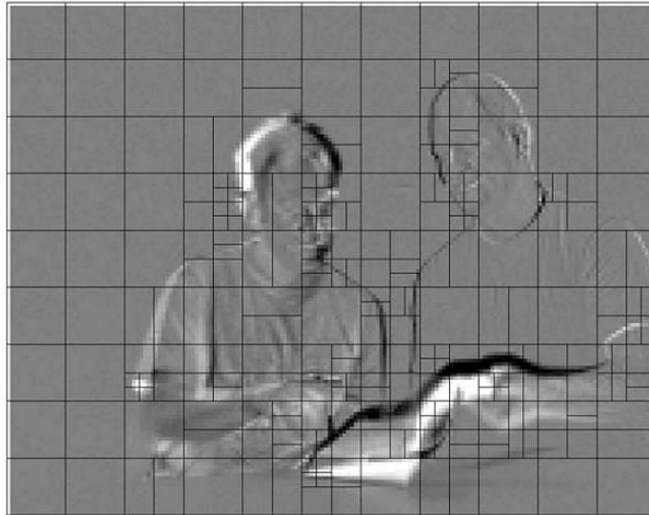


Figura 3.6: Exemplo de divisão de um bloco em vários tipos de partição. [11]

$$\mathbf{b} = (\mathbf{E} - 5\mathbf{F} + 20\mathbf{G} + 20\mathbf{H} - 5\mathbf{I} + \mathbf{J})/32$$

Já para 1/4 de píxel, realiza-se a média entre os píxeis vizinhos. Assim, para estimar o valor de **a**:

$$\mathbf{a} = (\mathbf{G} + \mathbf{b})/2$$

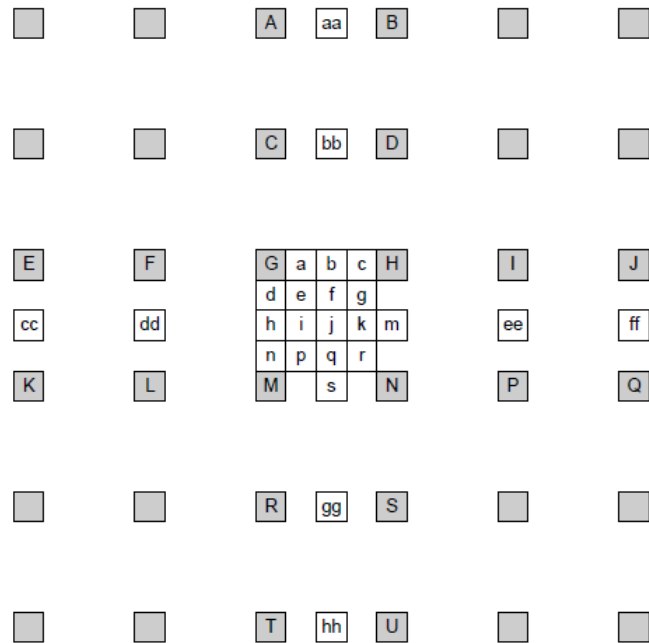
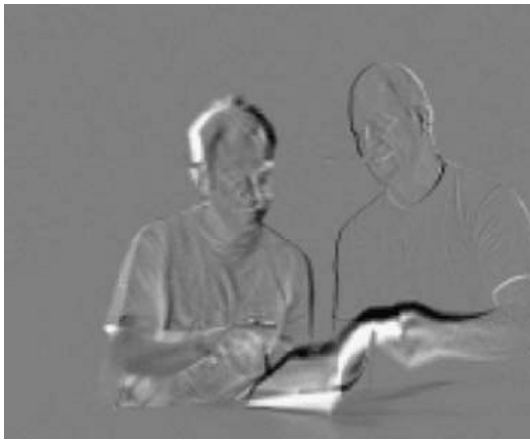


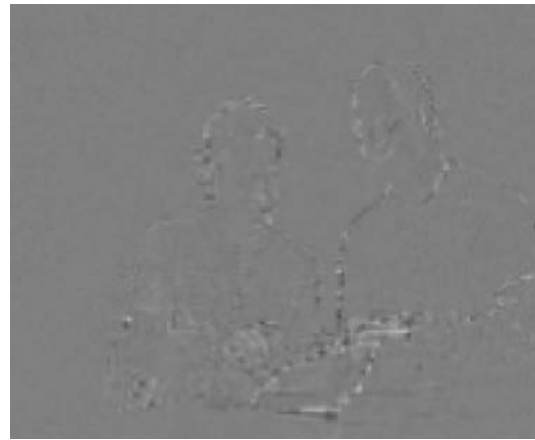
Figura 3.7: Interpolação dos píxeis. [11]

A Figura 3.8 mostra o resultado da estimação para interpolação de 1/2 e de 1/4 de píxel, em comparação à simples diferença entre os quadros e à estimação de

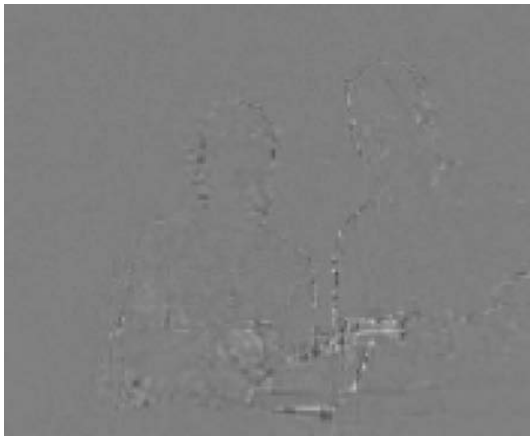
píxel inteiro. Uma análise detalhada mostra que a interpolação proporciona uma melhoria significativa na estimação de movimentos.



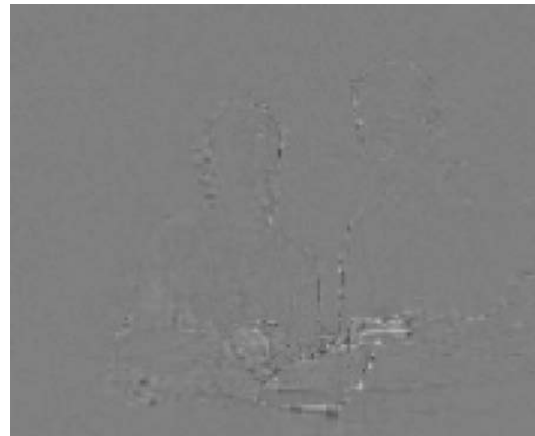
(a) Diferença entre os quadros



(b) Estimação de movimentos com blocos 4×4



(c) Estimação de movimentos com blocos 4×4 e interpolação de 1/2 de píxel



(d) Estimação de movimentos com blocos 4×4 e interpolação de 1/4 de píxel

Figura 3.8: Exemplos de estimação de movimentos a nível de subpíxel. [11]

3.3.3 Quadros de Referência

A estimação de movimentos adquire maior flexibilidade com uma mudança nas restrições impostas aos quadros de referência. Neste novo paradigma, os macroblocos de um determinado quadro podem fazer referência a macroblocos de diferentes quadros. Também foi adicionada a opção de os blocos serem preditos a partir de

uma combinação linear de diversas predições, com os pesos variando com a distância temporal entre os quadros atual e de referência.

O conceito de passado e futuro é generalizado para dois conjuntos de quadros. A lista 0 contém os quadros passados em relação ao quadro atual no início da lista, e quadros futuros no final, de modo que para uma predição relacionada à lista 0 é menos custoso utilizar os quadros passados. Por consequência, a lista 1 é composta por quadros que são futuro em relação ao quadro atual no início, e os futuros no final.

A definição de quadro **P** passa a ser os quadros compostos de macroblocos que utilizem uma referência para predição de um quadro da lista 0, como mostra a Figura 3.9, ou predição intra. Já os quadros **B** são aqueles cujos macroblocos utilizem referências de quadros da lista 0 e/ou da lista 1 ou predição intra, representado pela Figura 3.10. Para estes, ainda existe a possibilidade de realizar a estimação sem transmitir os vetores de movimento, em um modo denominado estimação direta. Outra propriedade dos quadros **B** é que também se permite utilizá-los como referência para outros quadros, o que pode permitir um melhor casamento para um bloco a ser estimado. Em ambos os tipos de quadro, também está previsto o uso de predição ponderada, onde um fator de escalamento e outro de soma são aplicados no quadro estimado, o que pode ser útil nos casos de transição de cena.

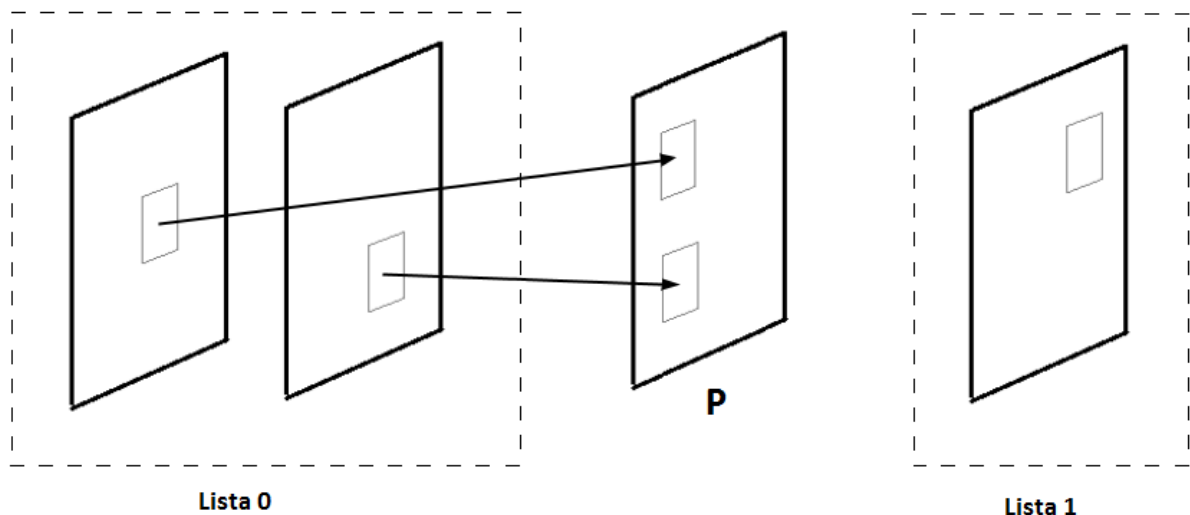


Figura 3.9: Estimação de movimentos para um quadro **P**. A lista 0 contém quadros passados. A lista 1, quadros futuros.

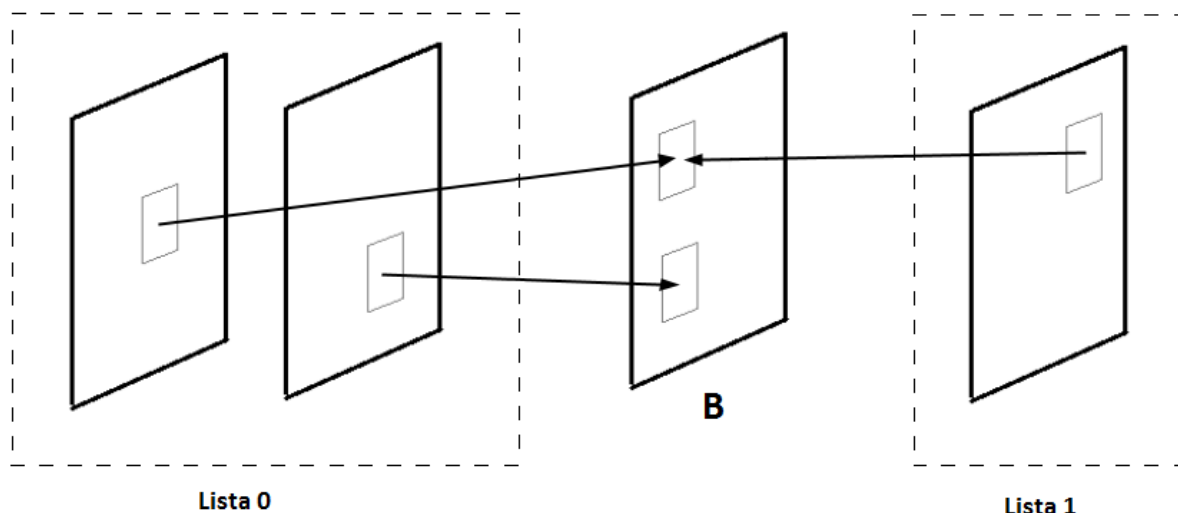


Figura 3.10: Estimação de movimentos para um quadro **B**. A lista 0 contém quadros passados. A lista 1, quadros futuros.

Por fim, o número de quadros que podem servir de referência também foi ampliado, para até 16 quadros. Isto acarreta em poder ocorrer uma predição a partir de um quadro anterior a um quadro **I**. Neste caso, o quadro **I** deixa de permitir acesso aleatório, e um erro anterior a ele pode influenciar o restante do sinal de vídeo. Assim, foi definido um subgrupo dos quadros **I** denominado quadro **IDR**, cuja função é forçar que esses problemas não aconteçam, de forma que não possa ocorrer nenhuma predição de um quadro posterior a um quadro **IDR** que faça referência a um quadro anterior ao mesmo.

3.4 Transformadas

A DCT, da forma que havia sido implementada, poderia causar um descasamento entre o codificador e o decodificador, visto que ela apresentava coeficientes reais, que podiam ser aproximados de maneiras diferentes. Para contornar este problema, foram definidas transformadas com coeficientes inteiros e propriedades semelhantes à DCT. Além disso, como visto na Figura 3.5, os blocos utilizados na predição de movimentos podem ter dimensões de até 4×4 . Para codificar melhor os resíduos gerados na predição, possibilitou-se também o uso de janelas 4×4 para o cálculo da transformada. Dessa forma, têm-se as seguintes matrizes de trans-

formação, para blocos 4×4 :

$$\mathbf{T}_{4 \times 4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & 1 & 2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \quad (3.1)$$

E para blocos 8×8 :

$$\mathbf{T}_{8 \times 8} = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix} \quad (3.2)$$

Outro tipo de transformada foi incluída no padrão H.264. Para os coeficientes DC obtidos em uma predição intra com blocos de 16×16 , assim como para todos os coeficientes DC nos blocos de croma de todos os tipos de quadros, é aplicada uma segunda transformação do tipo Hadamard [10].

3.5 Codificação

O padrão H.264 determina métodos mais sofisticados para codificação dos parâmetros e dados gerados na compressão dos vídeos. Para codificação de alguns parâmetros, um método simples utiliza um código de Exp-Golomb [11], que é um tipo de código de comprimento variável.

Uma técnica um pouco mais complexa envolve fazer um levantamento da estatística dos quadros conforme eles vão sendo codificados, ao invés de fazer um levantamento de antemão da mesma. Com isso, a codificação vai se adequando aos dados transmitidos, e também é possível fazer um modelo estatístico para sinais não-estacionários.

Foram criados métodos de codificação que se adaptam ao contexto, denominados *Context Adaptive Variable Length Coding* (CAVLC) e *Context Adaptive Binary Arithmetic Coding* (CABAC). No primeiro, códigos de comprimento variável são usados, baseados em um conjunto de tabelas que são escolhidas a partir dos elementos já transmitidos, logo, o contexto. Já o CABAC atua também utilizando os quadros transmitidos como modelo para a codificação, porém, utilizando de codificação aritmética, que permite o uso de números não-inteiros de *bits* por símbolo, o que é vantajoso para probabilidades muito altas (acima de 50 %).

3.6 Filtro de Desblocagem

Já que codificador pode utilizar técnicas de compressão diferentes para cada bloco, o resultado normalmente apresenta distorções ao remontar os quadros a partir dos blocos comprimidos, como mostra na Figura 3.11. Para suavizar tais imperfeições, o padrão H.264 reintegrou em sua estrutura um módulo de filtragem, conhecido por filtro de desblocagem. Este filtro, que é implementado de forma adaptativa, é controlado pelos diversos parâmetros pertinentes à codificação utilizada, como tipo de predição, passo de quantização, vetores de movimento obtidos, etc, com o objetivo de diminuir o efeito dos artefatos, mantendo as verdadeiras bordas das imagens. Como pode ser visto pelo resultado da Figura 3.12, os artefatos são consideravelmente reduzidos após a filtragem.

3.7 Perfis

Como o padrão H.264 foi desenvolvido para vários tipos de operação, cada qual com suas características, foram especificados diversos perfis de operação. Cada perfil define um subconjunto de ferramentas a ser utilizadas. Dessa forma, um decodificador não precisa se comprometer a ser compatível com todas as ferramentas disponíveis, mas sim, àquelas pertencentes a determinado perfil.

Nas primeiras versões, foram definidos os perfis **Baseline**, **Main**, e **Extended**. A partir do adendo [12], foram também adicionados perfis de tipo **High**, com base no **Main**. A Figura 3.13 mostra os principais perfis e as técnicas previstas para cada um deles, e as principais aplicações dos mesmos são listadas a seguir:



Figura 3.11: Exemplo de compressão de um quadro sem filtro de desblocagem. [11]



Figura 3.12: Exemplo de compressão de um quadro com filtro de desblocagem. [11]

- **Baseline** - indicado para sistemas com baixo poder computacional, ou onde o consumo de potência é um fator crítico, como equipamentos de videoconferência e dispositivos móveis.
- **Main** - inicialmente planejado para atender à maior parte dos consumidores, por oferecer um bom compromisso entre qualidade e recurso computacional. Era mais indicado para aplicações de transmissão e armazenamento, porém, foi sendo substituído pelo perfil **High**.

- **Extended** - projetado para atender à demanda do *streaming*, este perfil apresenta ferramentas que permitem alto poder de compressão e maior robustez a erros.
- **High** - possui as mesmas características do perfil **Main**, porém voltado a aplicações de alta definição. Foi adotado nos discos de HD DVD e Blu-Ray, assim como nas transmissões de TV de resolução HD (1280 x 720).
- **High 10** - perfil baseado no **High**, que adiciona o suporte à codificação com 10 bits por quadro.
- **High 4:2:2** - desenvolvido para aplicações de alta qualidade, a partir do perfil **High**, mas com suporte a video entrelaçado e ao formato de crominância 4:2:2.

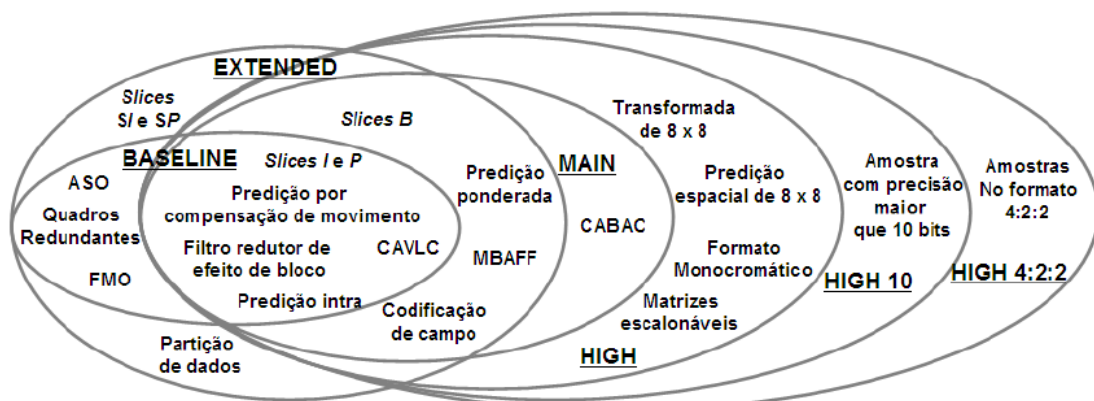


Figura 3.13: Diagrama dos conjuntos de ferramentas pertencentes a cada perfil.

3.8 Níveis

Os codificadores também podem ser divididos em níveis, que possuem 5 divisões e diversas subdivisões. Estes conjuntos impõem restrições relativas principalmente à taxa de quadros e à resolução dos quadros. A Tabela 3.1 mostra os valores especificados para cada nível.

Tabela 3.1: Níveis no padrão H.264

Resolução	Taxa de quadros/s	Número máximo de quadros de referência	Nível
QCIF (176x144)	15	4	1, 1b
	30	9	1.1
CIF (352x288)	7.5	2	1.1
	15	6	1.2
	30	6	1.3, 2
HHR (352x480)	30	6	2.1
HHR (352x576)	25	6	2.1
525 SD (720x480)	15	5	2.2
	30	5	3
625 SD (720x576)	25	5	3
720p HD (1280x720)	30	5	3.1
	60	4	3.2
1080p HD (1920x1080)	30	4	4, 4.1
	60	4	4.2
2Kx1K (2048x1024)	72	5	5
	120	5	5.1
4Kx2K (4096x2048)	30	5	5.1

Capítulo 4

Biblioteca x264

Neste capítulo, será apresentada a biblioteca x264 para codificação de vídeos no padrão H.264.

4.1 Definição

Os padrões de vídeo citados anteriormente especificam um modelo que deve ser seguido para um decodificador conseguir reconhecer determinado conjunto de *bits*. Assim, cabe ao codificador definir quais técnicas devem ou não ser usadas, além de implementar algoritmos que realizem as mesmas.

A biblioteca x264 é um *software* de código aberto em constante desenvolvimento, criado para realizar a codificação de sequências de vídeo no formato H.264. Seu desenvolvimento foi realizado visando a um bom desempenho em termos de velocidade. Para atingir tal resultado, esta biblioteca faz uso de linguagem *Assembly* em conjunto com C, realiza processamento *multithreaded*, além de implementar algoritmos sofisticados de estimação de movimentos e lógicas para a parada antecipada nas tomadas de decisão dos diversos algoritmos. Esta biblioteca ganhou diversos prêmios de comparação entre *codecs* [13], de modo que seu desempenho é comparável à outros codificadores comerciais de código fechado.

Neste trabalho, o x264 foi empregado a partir da versão presente no MEncoder [3], que é um *software* por linha de comando que faz uso de diversas bibliotecas para codificação de áudio e vídeo. Para seu uso, foram utilizados comandos como no seguinte exemplo:

```
mencoder input.avi -ovc x264 -x264encopts parametro1=valor1:  
parametro2=valor2:parametro3=valor3 -o output.avi
```

4.2 Controle de Taxa

De forma a impor uma restrição nas características da compressão, mas maximizando a qualidade obtida, são implementados algoritmos de controle de taxa. Estes algoritmos vão realizar o controle da quantização realizada, visando obedecer alguma condição imposta em algum parâmetro. Os modos de controle de taxa serão mostrados a seguir, assim como o comando necessário para sua utilização.

4.2.1 Duas Passadas

Com o controle implementado desta forma, o sistema realiza a compressão para obter dados relevantes. A partir destes dados, é realizada uma nova compressão, otimizando a distribuição de *bits*. Este modo é controlado pelo parâmetro **pass**.

No modo **1**, o programa gera, durante a codificação, um arquivo com características da mesma, como por exemplo os tipos de quadro utilizado. Dessa forma, ao se realizar a codificação novamente com o vídeo original, utilizando o modo **2**, o codificador teria acesso a informações referentes a quadros que ainda não foram analisados, podendo assim otimizar algumas etapas. Já no modo **3**, realiza-se a codificação lendo um arquivo previamente gerado, da mesma forma que o modo **2**, mas atualizando o arquivo de características com as mudanças geradas nessa nova codificação. Assim, este modo permite gerar múltiplas passadas.

Modo de uso: **pass=1, 2** ou **3**.

4.2.2 Taxa de Bits Média Constante

Neste modo, o codificador busca obter uma taxa de *bits* que, na média, possua determinado valor. Para tanto, ele realiza uma estimativa da complexidade dos quadros e da taxa de *bits* necessária para representá-los, enquanto compensa os desvios em relação à taxa alvo. Dessa forma, a taxa pode assumir uma certa variação ao redor da média. O parâmetro que define este modo é o **bitrate**.

Modo de uso: **bitrate=<taxa>**, em Kb/s.

4.2.3 Quantização Constante

Para esta abordagem, o codificador atribui um passo de quantização constante, a partir do parâmetro **qp**, aos quadros **P**, ou para os quadros **I** e **B**, pelos parâmetros **ipratio** e **bpratio**, respectivamente. Por consequência, a taxa final de *bits* não pode ser determinada com exatidão.

Modo de uso: **qp**=<passo>, de **0** a **51**, para especificar o passo de quantização dos quadros **P**.

4.2.4 Qualidade Constante

Este modo atua de forma semelhante ao controle pela taxa de *bits*, porém, visando a manter constante uma medida da qualidade dos quadros comprimidos. Dessa forma, este tipo de controle apresenta resultado de compressão superior ao controle pela quantização constante, por usar de informação subjetiva para eliminar *bits* de forma que a perda de informação seja menos perceptível. No x264, é regido pelo parâmetro **crf**.

Modo de uso: **crf**=<valor>, entre **0** e **51**.

4.3 Algoritmos e Modos de Atuação

Como previamente mencionado, o x264 possibilita o controle de vários dos possíveis modos de operação previstos pelo H.264. A seguir, serão descritos vários parâmetros que possibilitam este controle e como os mesmos devem ser utilizados.

4.3.1 Partições

São especificados vários tipos de partições possíveis para cada tipo de macrobloco, como mostrado na Seção 3.2 e na Subseção 3.3.1, a partir do parâmetro **partitions**. Para quadros **P**, este parâmetro habilita a partição para blocos de tamanho até 8×8 píxeis, pelo valor **p8x8**, e tamanho 4×4 para o valor **p4x4**. Da mesma forma, para quadros **I**, temos **i8x8** e **i4x4**. Para quadros **B**, somente está disponível o tamanho até 8×8 , pelo valor **b8x8**.

Para decidir os tipos de partições a serem utilizadas, o x264 utiliza o seguinte

algoritmo. Para predição inter, são analisados os custos em *bits* de se usar estimação de movimentos com blocos de 8×8 e 16×16 com todos os quadros de referência possíveis, e, com base nos resultados, prevêem-se os custos de utilizar blocos de 8×16 e 16×8 . Se o custo estimado destes blocos for pior que para o bloco de 16×16 píxeis, estas partições são abandonadas. Então, realiza-se a estimação para os blocos de 8×16 e 16×8 utilizando os mesmos quadros de referência obtidos pelos blocos 8×8 que os compõem, e o tipo de partição utilizada é aquela que obtiver o menor custo final. Caso sejam escolhidos os blocos 8×8 , este algoritmo se repete, para determinar as partições de até 4×4 píxeis. Todos os custos são analisados a partir da soma absoluta da diferença das transformadas (SATD).

Para predição intra, primeiramente toma-se a decisão pelo melhor tipo de estimação para blocos de 16×16 , visto na Figura 3.4. Realizando a estimação, caso nenhum dos macroblocos vizinhos utilize predição intra, e o custo de usar a mesma seja pior que para predição inter, a predição intra é abandonada. Caso contrário, também se analisa o custo de utilizar blocos 8×8 e 4×4 , e o tipo de partição com o menor custo é escolhido. Por fim, para ambos os tipos de predição são realizadas otimizações de taxa distorção nas decisões e nos vetores de movimento.

Modo de uso: **partitions**=<partições>, sendo <partições> os modos que se deseja habilitar, separados por vírgula. Também é possível assumir o valor **none** para nenhuma das partições mencionadas, ou **all** para todas.

4.3.2 Estimação de Movimentos

O parâmetro **me** escolhe o algoritmo que realiza a estimação de movimentos. As opções disponíveis são: *diamond*, *hexagon*, *uneven multi-hexagon*, *exhaustive* e *hadamard exhaustive*.

O algoritmo de busca *diamond*, dado pelo valor **dia**, implementa a busca que teoricamente é mais rápida e menos eficiente. Ele se inicia a partir de uma estimativa do vetor de movimento, que pode ser nulo, para nenhum deslocamento, ou uma predição do mesmo a partir de blocos vizinhos. Realiza-se então uma busca em forma de cruz, comparando o bloco do quadro que se quer estimar e os blocos do quadro de referência equivalentes a um deslocamento de um píxel acima, abaixo, à direita e à esquerda da estimativa inicial. Caso algum dos blocos deslocados apresente melhor

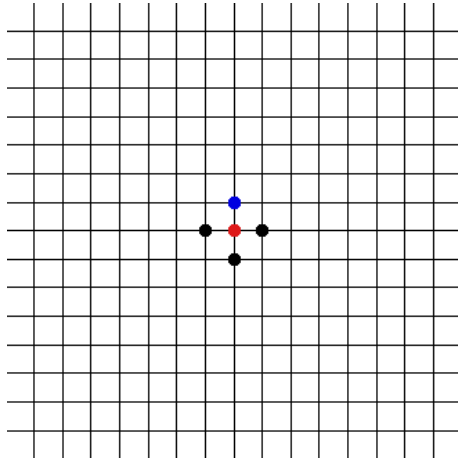
casamento com o bloco a ser estimado, este bloco passa a ser a nova estimativa do bloco atual, e o algoritmo se reinicia. Já caso isto não ocorra, a estimativa do bloco no início da iteração do algoritmo é dada como resultado final. Dessa forma, o algoritmo vai aos poucos convergindo para um mínimo, com complexidade $O(n)$ no pior caso. A Figura 4.1 mostra o passo a passo deste algoritmo. Uma desvantagem do mesmo é que ele pode eventualmente convergir para um mínimo local.

Outro algoritmo de busca presente é o *hexagon*, habilitado por **hex**. Este funciona deslocando os blocos em 6 píxeis próximos, num formato hexagonal, exceto no último passo do algoritmo, que realiza um deslocamento idêntico ao implementado pelo *diamond* para refinamento. Este algoritmo também possui complexidade $O(n)$ no pior caso. Um exemplo deste algoritmo está presente na Figura 4.2

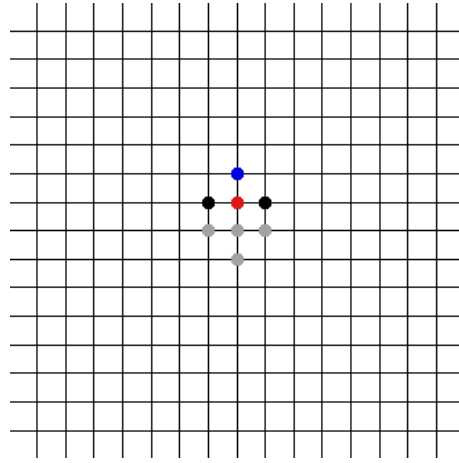
Já o algoritmo *uneven multi-hexagon*, dado por **umh**, envolve técnicas mais complexas e engloba, em parte, a busca realizada pelos algoritmos anteriores. Num primeiro passo, é realizada uma busca chamada de *unsymmetrical cross*, que analisa deslocamentos horizontais e verticais de diversos píxeis, para encontrar um novo ponto inicial. A seguir, realiza-se uma busca exaustiva em todos os deslocamentos ao redor de um pequeno retângulo. Se a busca não atingir nenhum resultado abaixo de um limite estabelecido, procura-se os deslocamentos seguindo um padrão composto de diversos hexágonos concêntricos contendo 16 píxeis, cada qual com um determinado tamanho, sendo o tamanho máximo controlado pelo parâmetro **merange**, o que determina a área de busca. Após encontrar o melhor vetor de movimento, realiza-se um refinamento ao redor deste ponto utilizando o algoritmo *hexagon*. Apesar da maior complexidade, esta não é maior do que $O(n)$. As Figuras 4.3 a 4.8 mostram um exemplo do andamento deste algoritmo.

Por fim, também são implementados algoritmos de busca exaustiva em todos os píxeis dentro da área de busca, a partir das técnicas *exhaustive* e *hadamard exhaustive*, habilitadas respectivamente por **esa** e **tesa**. Para estas técnicas, que só são recomendadas quando a velocidade de codificação não possui nenhuma restrição, a complexidade é $O(n^2)$.

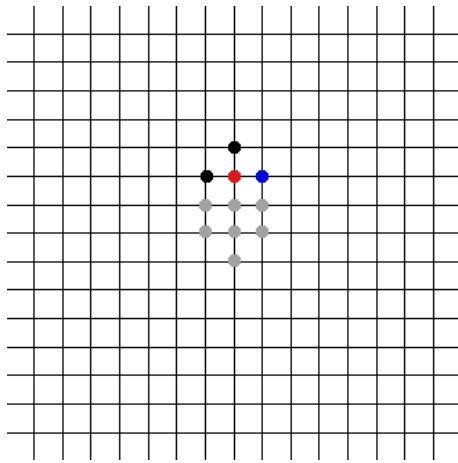
Modo de uso: **me**=<algoritmo>, para escolher algum dos algoritmos descritos acima.



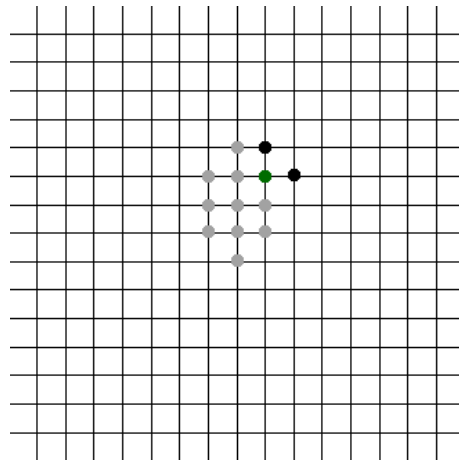
(a) 1º passo: busca *diamond*.



(b) 2º passo: busca *diamond*.

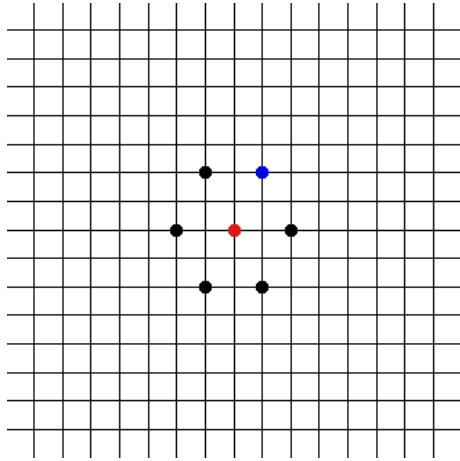


(c) 3º passo: busca *diamond*.

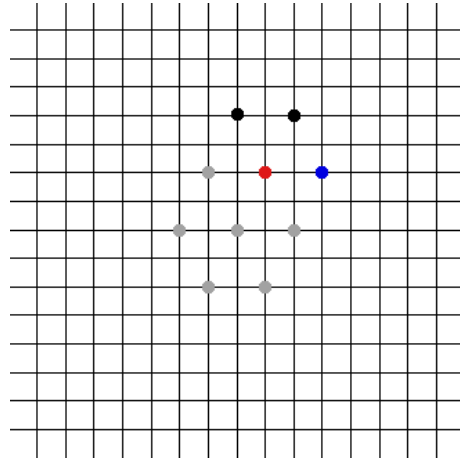


(d) 4º passo: busca *diamond*.

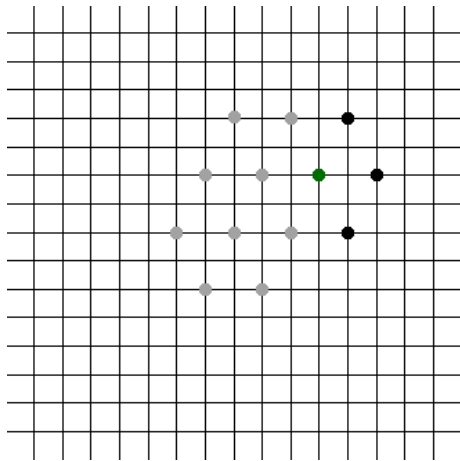
Figura 4.1: Passo a passo do algoritmo de estimação de movimentos *diamond*. O ponto vermelho representa o ponto inicial do passo, os pretos representam os deslocamentos sendo analisados, o azul representa o deslocamento de melhor casamento, o verde ocorre quando o melhor casamento se encontra no ponto inicial, e os pontos cinza representam os deslocamentos já analisados.



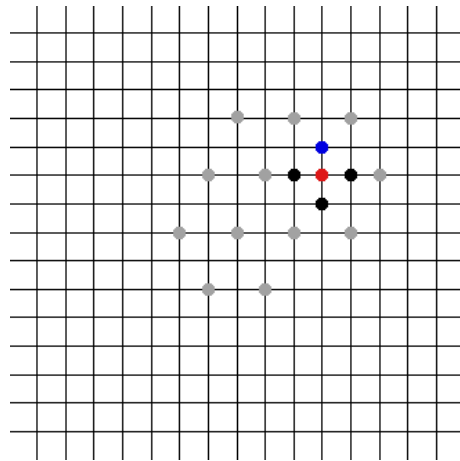
(a) 1º passo: busca *hexagon*.



(b) 2º passo: busca *hexagon*.



(c) 3º passo: busca *hexagon*.



(d) 4º passo: busca *diamond*.

Figura 4.2: Passo a passo do algoritmo de estimação de movimentos *hexagon*. O ponto vermelho representa o ponto inicial do passo, os pretos representam os deslocamentos sendo analisados, o azul representa o deslocamento de melhor casamento, o verde ocorre quando o melhor casamento se encontra no ponto inicial, e os pontos cinza representam os deslocamentos já analisados.

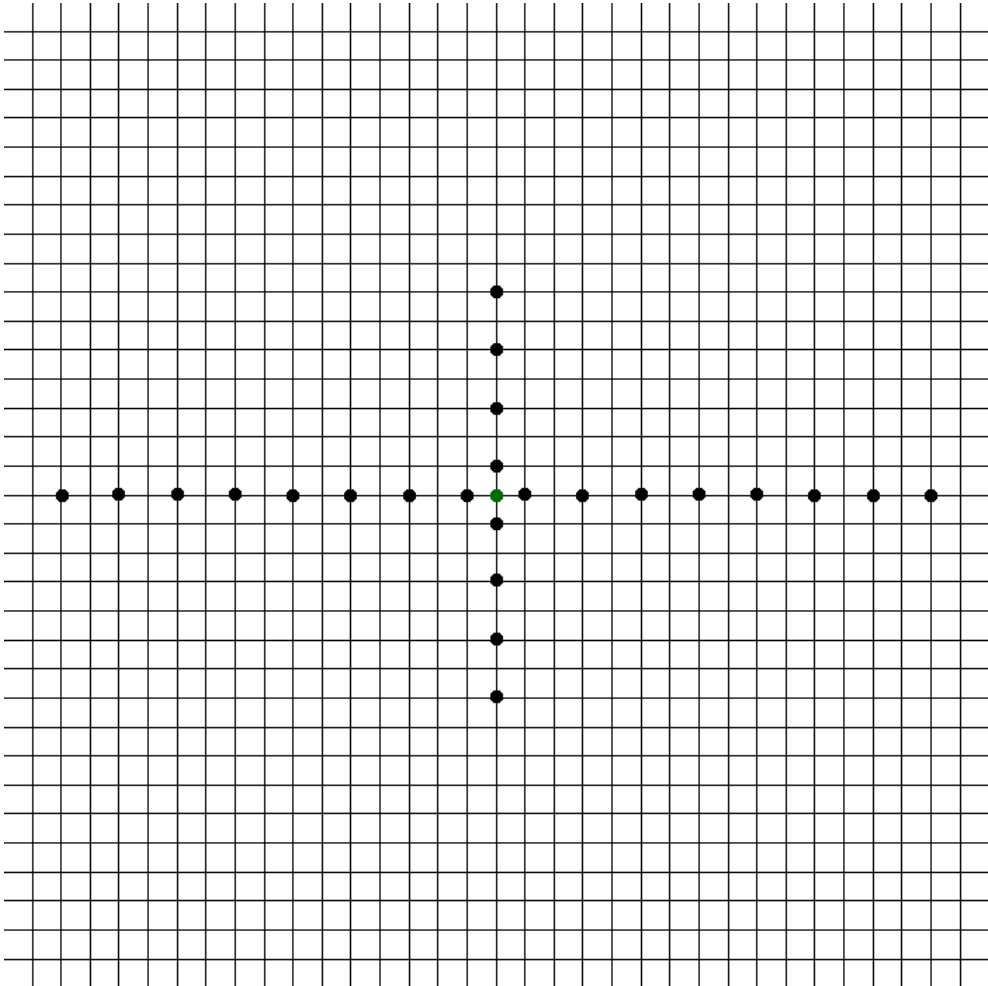


Figura 4.3: 1º passo do algoritmo de estimação de movimentos *uneven multi-hexagon*: busca *unsymmetrical cross*. O ponto verde ocorre quando o melhor casamento se encontra no ponto inicial do passo e os pretos representam os deslocamentos sendo analisados.

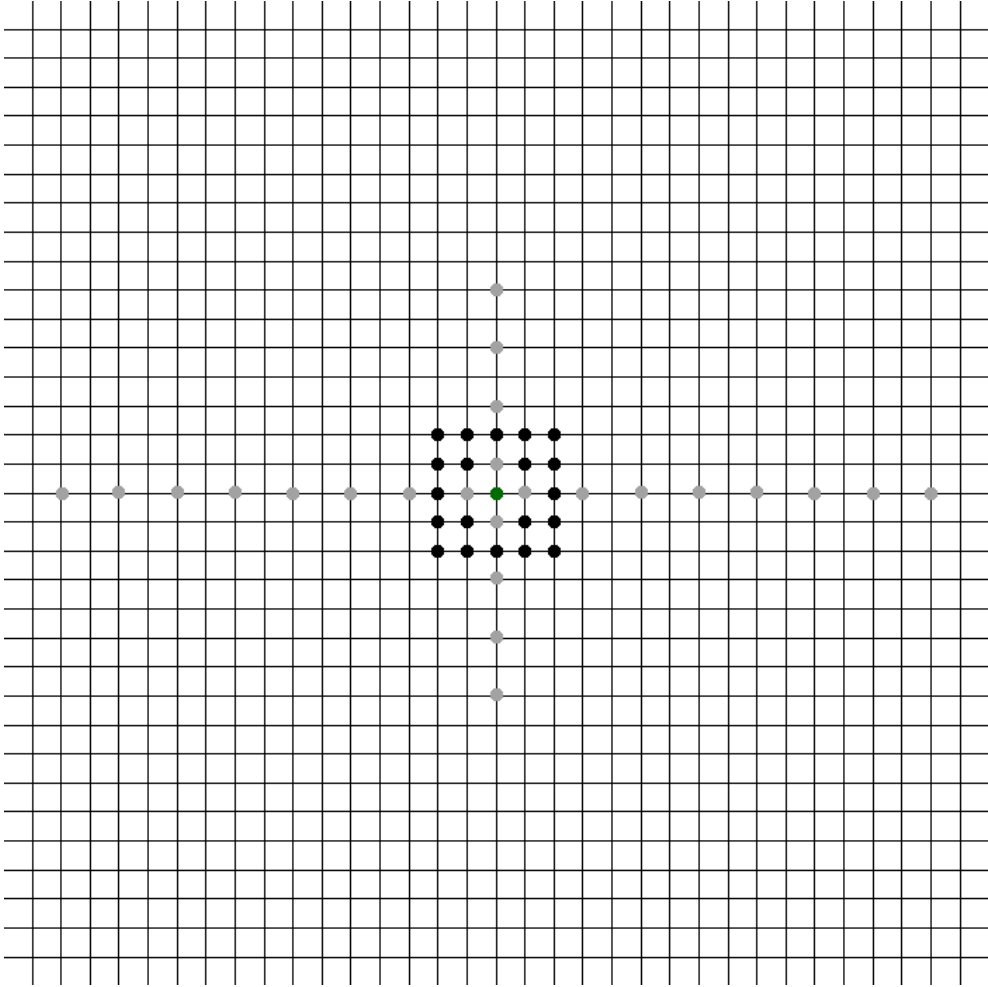


Figura 4.4: 2º passo do algoritmo de estimação de movimentos *uneven multi-hexagon*: busca exaustiva em um retângulo de 5 x 5 píxeis. O ponto verde ocorre quando o melhor casamento se encontra no ponto inicial do passo, os pretos representam os deslocamentos sendo analisados, e os pontos cinza representam os deslocamentos já analisados.

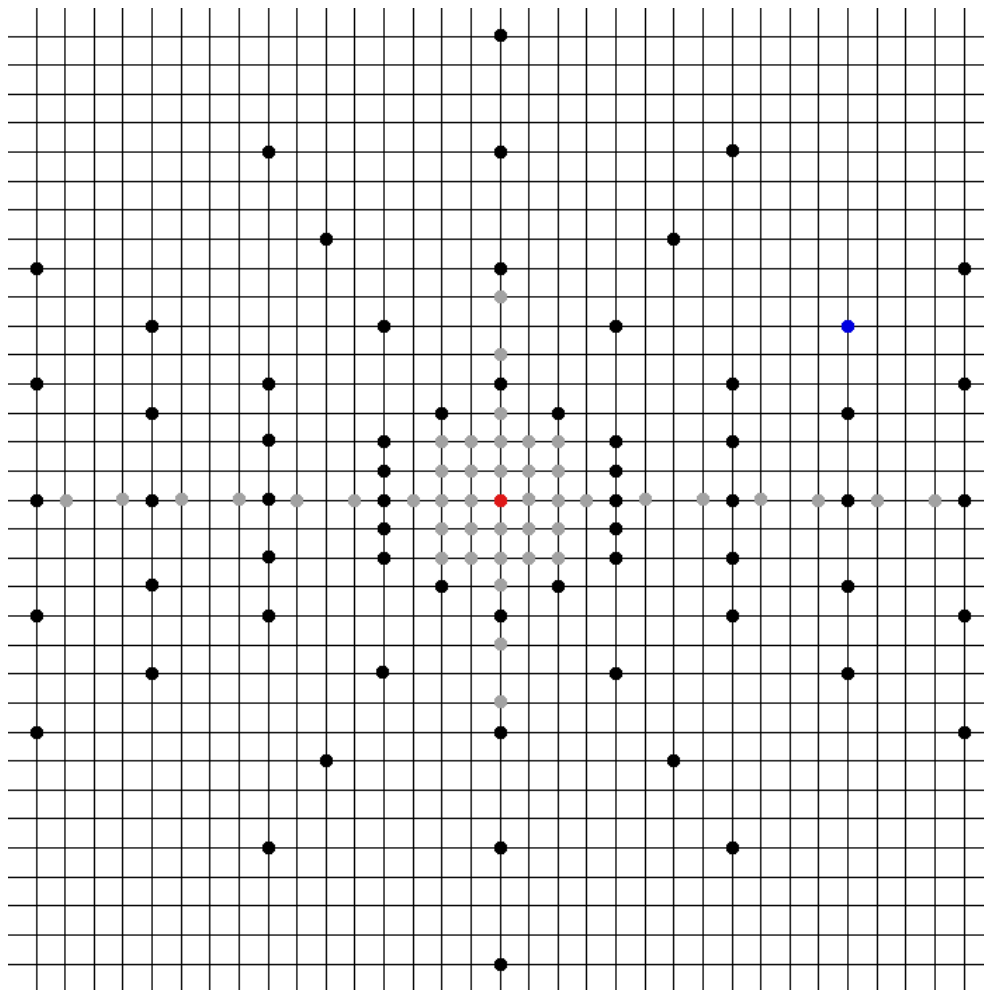


Figura 4.5: 3º passo do algoritmo de estimação de movimentos *uneven multi-hexagon*: busca *uneven multi-hexagon*. O ponto vermelho representa o ponto inicial do passo, os pretos representam os deslocamentos sendo analisados, o azul representa o deslocamento de melhor casamento, e os pontos cinza representam os deslocamentos já analisados.

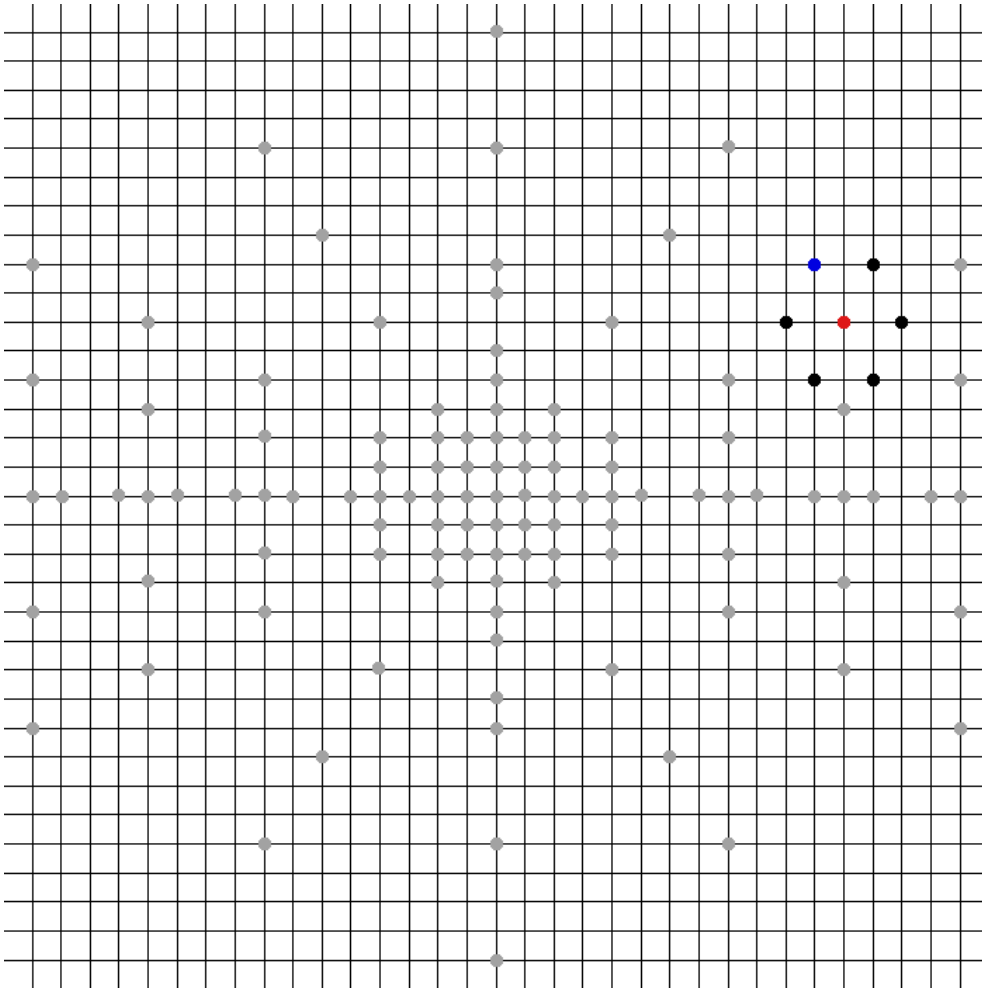


Figura 4.6: 4º passo do algoritmo de estimação de movimentos *uneven multi-hexagon*: busca *hexagon*. O ponto vermelho representa o ponto inicial do passo, os pretos representam os deslocamentos sendo analisados, o azul representa o deslocamento de melhor casamento, e os pontos cinza representam os deslocamentos já analisados.

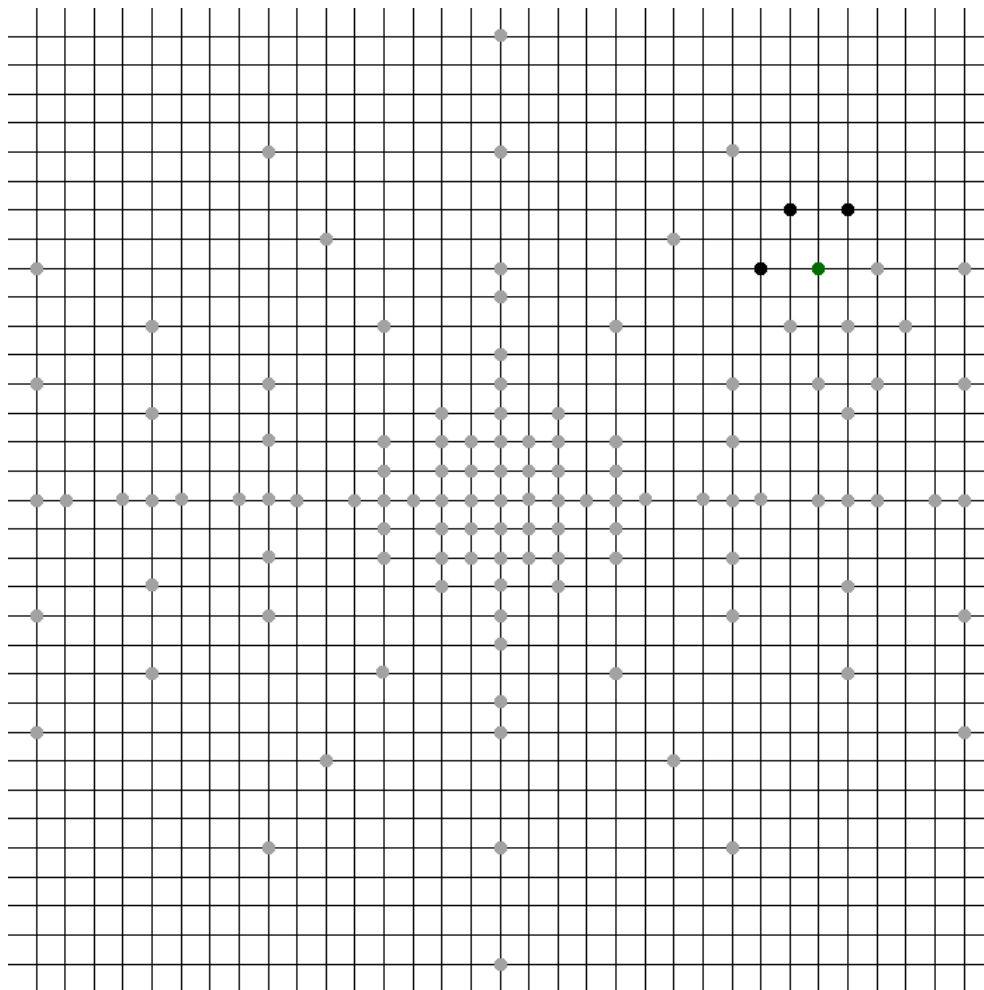


Figura 4.7: 5º passo do algoritmo de estimação de movimentos *uneven multi-hexagon*: busca *hexagon*. O ponto verde ocorre quando o melhor casamento se encontra no ponto inicial do passo, os pretos representam os deslocamentos sendo analisados, e os pontos cinza representam os deslocamentos já analisados.

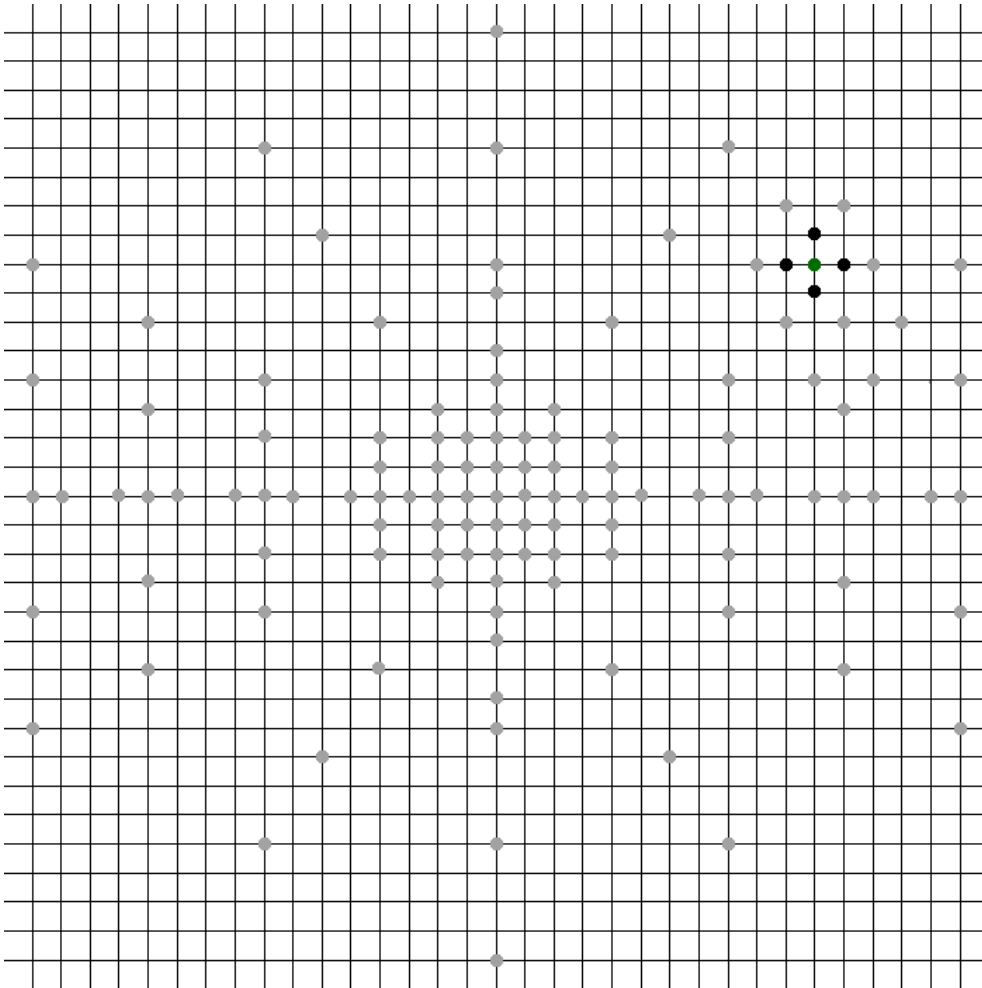


Figura 4.8: 6º passo do algoritmo de estimação de movimentos *uneven multi-hexagon*: busca *diamond*. O ponto verde ocorre quando o melhor casamento se encontra no ponto inicial do passo, os pretos representam os deslocamentos sendo analisados, e os pontos cinza representam os deslocamentos já analisados.

4.3.3 Estimação a Nível de Subpíxel

O refinamento a nível de subpíxel para a estimação de movimentos, como previsto na Seção 3.3.2, é definido pelo parâmetro **subq**. São implementados diversos modos, gradualmente mais complexos:

- **0**: estimação de píxel inteiro;
- **1**: estimação de píxel inteiro para escolher um tipo de macrobloco, com refinamento de 1/4 de píxel no tipo escolhido, utilizando como método de comparação a soma das diferenças absolutas (SAD);
- **2**: estimação de 1/2 píxel para escolher um tipo de macrobloco, com refinamento de 1/4 de píxel no tipo escolhido, utilizando como método de comparação a soma das diferenças absolutas das transformadas (SATD);
- **3**: estimação de 1/2 píxel para escolher um tipo de macrobloco, com refinamento mais lento de 1/4 de píxel no tipo escolhido, utilizando como método de comparação a soma das diferenças absolutas das transformadas (SATD);
- **4**: estimação de 1/4 de píxel para escolher um tipo de macrobloco, com refinamento de 1/4 de píxel no tipo escolhido;
- **5**: estimação mais lenta de 1/4 de píxel em todos os tipos de macrobloco, escolhendo o melhor deles;
- **6**: otimização de taxa-distorção em quadros **I** e **P**;
- **7**: otimização de taxa-distorção em todos os quadros;
- **8**: otimização de taxa-distorção com refinamento em quadros **I** e **P**;
- **9**: otimização de taxa-distorção com refinamento em todos os quadros;
- **10**: otimização de taxa-distorção com otimização de taxa-distorção em quarto de píxel;
- **11**: otimização de taxa-distorção completa, desabilitando as heurísticas de parada antecipada.

Modo de uso: **subq**=<modo>, para escolher algum dos modos descritos acima.

4.3.4 Uso da Informação de Crominância

O x264 implementa um algoritmo, habilitado pelo parâmetro **chroma_me**, que utiliza a informação de crominância dos quadros em conjunto com a de luminância no processo de busca da estimação de movimentos.

Modo de uso: **chroma_me**=0 para desativar ou **1** para ativar. Alternativamente, também é possível escrever somente **chroma_me** para manter ativo, e **nochroma_me** para desativar.

4.3.5 Número de Quadros de Referência

O número máximo de quadros de referência que podem ser utilizados como referência para quadros **P** é determinado por **ref**. Este parâmetro deve ser alterado caso se deseje impor alguma restrição referente a níveis, como visto na Tabela 3.1. Para quadros **B**, o número de quadros de referência é 1 ou 2 a menos do que o especificado por este parâmetro.

Modo de uso: **ref**=<quantidade>, que pode ir de **1** a **16**.

4.3.6 Referências Múltiplas

Para permitir que cada bloco de dimensões até 8×8 pixels escolha seu próprio quadro de referência, como foi mostrado nas Figuras 3.9 e 3.10, utiliza-se o parâmetro **mixed_refs**.

Modo de uso: **mixed_refs**=0 para desativar ou **1** para ativar. Alternativamente, também é possível utilizar somente **mixed_refs** para manter ativo, e **nomixed_refs** para desativar.

4.3.7 Predição Ponderada para Quadros P

Uma das possibilidades do padrão H.264 é permitir que uma estimação de um bloco **P** seja ponderada por um peso, e somada a um determinado valor, o que pode ser útil em efeitos de transição de cena. Para tanto, foram implementados

dois modos de diferente complexidade. No modo **1**, utiliza-se somente a soma de um determinado valor aos píxeis, sem realizar uma análise prévia do quadro. Já no modo **2**, executa-se um algoritmo de detecção de transição de cena, e aplicam-se as duas operações, o que vai ocasionar maior eficiência para esses casos. No x264, esta opção é controlada pelo parâmetro **weightp**.

Modo de uso: **weightp**=<modo>, para ativar os modos citados anteriormente. O modo **0** ou **noweightp** equivale ao modo que desabilita esta função.

4.3.8 Predição Ponderada para Quadros **B**

Habilita-se com o parâmetro **weight_b** o modo de predição ponderada para quadros **B**, que permite que as várias referências de cada macrobloco sejam combinadas ponderadamente.

Modo de uso: **weight_b**=**0** para desativar ou **1** para ativar. Alternativamente, também é possível utilizar somente **weight_b** para manter ativo, e **noweight_b** para desativar.

4.3.9 Número de Quadros **B**

Pelo parâmetro **bframes**, define-se o número máximo de quadros **B** consecutivos que podem ser inseridos entre quadros **P** ou **I**.

Modo de uso: **bframes**=<quantidade>, que pode ir de **0** a **16**.

4.3.10 Uso Adaptativo de Quadros **B**

Foi implementado um algoritmo de decisão do melhor número de quadros **B** consecutivos que devem ser utilizados, até o máximo definido por **bframes**. O básico deste algoritmo consiste de, primeiramente, uma decisão de usar ou não quadros **B**. Dessa forma, executa-se, para os dois próximos quadros, a estimação de movimentos considerando que ambos são quadros **P** e considerando que o primeiro é quadro **B**. O resultado de ambas as estimações é comparado, com base também no parâmetro **b.bias**, que controla o limiar de decisão inserção de quadros **B**.

Caso seja decidido pelo uso de quadros **B**, o codificador realiza a estimação para valores gradativamente maiores de quadros **B**, passando para **BBP**, **BBBP**,

etc. Conforme se aumenta o número de quadros **B**, a tendência é que o quadro **P** final busque uma referência cada vez mais distante deste, de forma que os macroblocos presentes no mesmo vão ter maior inclinação a utilizar de predição intra. Quando o número de macroblocos com predição intra ultrapassar um certo limiar, ou quando o número de quadros **B** atingir o valor de **bframes**, o algoritmo é encerrado. A ativação deste módulo é dada por **b_adapt**.

Modo de uso: **b_adapt=1** para o algoritmo rápido, **2** para o algoritmo ótimo e **0** ou **nob_adapt** para utilizar sempre o número máximo de quadros **B**.

4.3.11 Uso de Quadros **B** como Referência

Como visto na Seção 3.3.3, o padrão possibilita o uso de quadros **B** para servirem de referência para outros quadros. Esta opção é habilitada por **b_pyramid**.

Modo de uso: **b_pyramid=0**, **none** ou **nob_pyramid** para desativar, **1** ou **strict** para utilizar 1 quadro **B** por minigop como referência, e **2** ou **normal** para permitir que vários quadros **B** sejam usados como referência.

4.3.12 Predição Direta

Utilizando o parâmetro **direct_pred**, permite-se que os vetores de movimento sejam preditos a partir de alguma vizinhança, de maneira que não seja necessário codificar o vetor de movimento de todos os blocos, num modo denominado estimação direta. As vizinhanças previstas são:

- **spatial**: utiliza a predição, para cada bloco, a partir dos blocos vizinhos em um mesmo quadro;
- **temporal**: utiliza a predição a partir dos quadros vizinhos;
- **auto**: escolha automática entre os modos acima;
- **none**: desabilita a estimação direta.

Modo de uso: **direct_pred=<modo>**, escolhendo entre um dos modos descritos acima.

4.3.13 Transformadas 8×8

Habilita-se, com **8x8dct**, o uso adaptativo de DCT de tamanho 8×8 em quadros **I**.

Modo de uso: **8x8dct=0** para desativar ou **1** para ativar. Alternativamente, também é possível utilizar somente **8x8dct** para manter ativo, e **no8x8dct** para desativar.

4.3.14 Codificação Aritmética

O parametro **cabac** ativa o modo de codificação aritmética adaptativa, descrito na seção 3.5, ou desabilita o mesmo, utilizando o CAVLC.

Modo de uso: **cabac=0** para a ativar a codificação CAVLC e **1** para CABAC. Para o modo 0, também é possível utilizar a expressão **nocabac**.

4.3.15 Árvore de Macroblocos

Foi desenvolvido um modo de predição da propagação de informação por uma árvore de macroblocos. Neste modo, o codificador utiliza um conjunto de quadros futuros em relação ao quadro atual, e estima os custos de se utilizar predição intra e inter para cada quadro, baseado na SATD. A partir destes valores, ele estima uma quantidade de informação que é propagada entre os quadros, e quanto cada macrobloco contribui para a qualidade do bloco seguinte. Com base nos resultados, altera-se o passo de quantização de cada macrobloco, diminuindo a qualidade dos blocos que menos contribuem para os quadros seguintes e aumentando a qualidade nos outros quadros. Este modo, dado por **mbtree**, opera de maneira semelhante ao determinado pelo parâmetro **qcomp**, que diminui a qualidade de quadros com detalhes mais complexos, que são menos perceptíveis.

Modo de uso: **mbtree=0** para desativar ou **1** para ativar. Alternativamente, também é possível utilizar somente **mbtree** para manter ativo, e **nombtree** para desativar.

4.3.16 Número de Quadros para a Árvore de Macroblocos

Atuando em conjunto com o algoritmo de árvore de macroblocos, descrito no item acima, está o parâmetro **rc_lookahead**, que determina a quantidade de quadros futuros que o algoritmo considera ao fazer a estimação.

Modo de uso: **rc_lookahead**=<quantidade>, até um máximo de **250**.

4.3.17 Filtro de Desblocagem

O filtro de desblocagem, descrito na Seção 3.6, é controlado por **deblock**.

Modo de uso: **deblock**=**0** para desativar ou **1** para ativar. Alternativamente, também é possível utilizar somente **deblock** para manter ativo, e **nodeblock** para desativar.

4.3.18 Quantização Adaptativa

O codificador implementa uma quantização adaptativa, que permite que cada macrobloco de um quadro possua um passo de quantização diferente. Dessa forma, os *bits* que seriam alocados em macroblocos com pouca informação são redistribuídos entre os outros macroblocos. No modo **1**, o quantizador pode distribuir os *bits* entre os diversos quadros do vídeo. No modo **2**, o quantizador também adequa a quantização adaptativa entre os blocos de cada quadro. Este algoritmo é habilitado pelo parâmetro **aq_mode**.

Modo de uso: **aq_mode**=**1**, **2**, ou **0** para desabilitar (além de **noaq_mode**).

4.3.19 Codificação por Treliça

O x264 realiza uma codificação por treliça para os coeficientes obtidos da DCT. Neste modo, dado por **trellis**, alguns coeficientes são quantizados com maior passo de quantização e outros com menor passo, redistribuindo alguns *bits* de maneira a achar uma quantização mais próxima da ótima em termos de PSNR para uma determinada taxa.

Modo de uso: **trellis**=**1**, para habilitar a treliça somente na codificação final de um macrobloco, **2** para habilitar em todas as codificações realizadas durante o processo de decisão dos diversos modos de decisão e **0** para desabilitar.

4.3.20 Perfis

Para adequar os parâmetros do codificador aos perfis apresentados na Seção 3.7, utiliza-se o comando **profile**.

Modo de uso: **profile**=<perfil>, que podem ser **baseline**, **main**, **high**, **high10**, **high422** ou **high444**.

4.3.21 Opções Pré-definidas

Também foi definido outro conjunto de configurações, de maneira semelhante aos perfis, mas que permitem trocar eficiência por velocidade de compressão. Estes conjuntos são escolhidos a partir do valor do parâmetro **preset**.

Modo de uso: **preset**=<modo>, que pode ser, em ordem crescente de complexidade: **ultrafast**, **superfast**, **veryfast**, **faster**, **fast**, **medium**, **slow**, **slower**, **veryslow** ou **placebo**.

4.3.22 Multiprocessamento

O x264 permite o uso de mais de 1 *thread* durante a codificação, utilizando o parametro **threads**.

Modo de uso: **threads**=<quantidade>, até um máximo de **128**, ou **auto**, para o codificador definir a quantidade de *threads*.

4.3.23 PSNR

O comando **psnr** habilita um cálculo de *Peak Signal to Noise Ratio*, ou PSNR, entre os quadros do vídeo comprimido e do vídeo original, para quantificar a diferença entre ambos. O cálculo pode ser realizado a partir do *Mean Square Error* (MSE) entre os quadros, que é da seguinte forma, dados os quadros A e B de dimensões m x n:

$$\text{MSE} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [A(i, j) - B(i, j)]^2. \quad (4.1)$$

Definindo então uma constante K, como sendo o valor máximo que um píxel pode assumir, que neste caso é 255, a PSNR assume a seguinte expressão:

$$\mathbf{PSNR} = 10 \log_{10} \left(\frac{K^2}{MSE} \right). \quad (4.2)$$

Modo de uso: **psnr**, para habilitar.

4.3.24 SSIM

De maneira alternativa ao cálculo da PSNR, o comando **ssim** introduz o cálculo da métrica *Structural Similarity*, também chamada de SSIM, entre os quadros original e comprimido. Para o cálculo, os quadros são divididos em blocos e, para cada bloco \mathbf{x} do quadro original e \mathbf{y} do quadro comprimido, computam-se as médias μ_x e μ_y , as variâncias σ_x^2 e σ_y^2 e a covariância σ_{xy} . Segue-se então o seguinte cálculo, sendo k_1 e k_2 duas constantes para impedir a divisão por zero:

$$\mathbf{SSIM} = \frac{(2\mu_x\mu_y + k_1)(2\sigma_{xy} + k_2)}{(\mu_x^2 + \mu_y^2 + k_1)(\sigma_x^2 + \sigma_y^2 + k_2)}. \quad (4.3)$$

Modo de uso: **ssim**, para habilitar.

4.4 Trabalhos Correlatos

O fato de ser uma biblioteca atual e em constante desenvolvimento motiva o seu estudo em diversos trabalhos, alguns deles propondo melhorias nos algoritmos implementados. No trabalho [14], é proposta uma otimização do tempo gasto pelo x264 com o acesso à memória do computador. Já em [15] é proposta uma implementação da métrica SSIM na otimização taxa-distorção para predição intra, ao passo que [16] tem uma proposta semelhante, para predição inter.

A tese desenvolvida em [17] possui uma temática diferente, mais próxima do trabalho apresentado neste documento. Nela, são apresentadas curvas ótimas de configuração a partir de parâmetros selecionados *a priori*, visando a maximizar a eficiência energética do codificador. O presente trabalho se diferencia do apresentado em [17] por realizar também uma análise dos parâmetros, de forma a determinar, para os parâmetros relevantes, os melhores modos de atuação. Além disso, o enfoque deste trabalho é gerar um codificador, a partir da biblioteca x264, que realize a codificação de vídeos *full HD* de forma ótima.

Capítulo 5

Resultados Experimentais

Neste capítulo serão mostrados os resultados dos testes realizados.

5.1 Descrição do Experimento

Conforme descrito na Seção 1.5, em um primeiro momento foram realizadas simulações da codificação de um conjunto de vídeos, utilizando o x264, para determinar uma limitação para a análise combinacional dos parâmetros. Foram escolhidos os parâmetros presentes nas Seções 4.3.1 a 4.3.19, realizando uma análise particular a cada um destes, com a possibilidade de realização de algum refinamento a partir da combinação de um subconjunto de parâmetros que possuam alguma dependência direta. Como controle de taxa, optou-se pelo controle por taxa média descrito na Seção 4.2.2, de forma a gerar um sinal de vídeo com taxa de *bits* compatível com o utilizado na prática.

Para cada parâmetro, foi feita a medida de PSNR presente na Seção 4.3.23, e também de velocidade de codificação, a partir da informação exibida pelo programa. Estes testes foram analisados em relação ao desempenho obtido por uma configuração de parâmetros onde cada um deles se manteve em seu modo de atuação menos complexo. de modo que as velocidades obtidas foram comparadas à velocidade que teoricamente seria a máxima obtida pelo codificador. Pôde-se então determinar os parâmetros mais propícios a serem mantidos em um determinado estado, e quais permitem maior troca entre qualidade e velocidade.

De posse destes resultados, foram realizados os testes com as combinações

relevantes, com as mesmas medidas mostradas anteriormente. Neste caso, foram geradas nuvens de pontos para 4 valores de taxas de *bits*, a partir das quais foi obtido um conjunto de configurações ótimas para cada taxa de *bits*, maximizando a qualidade obtida para uma dada complexidade. Para validação, realizaram-se os mesmos testes para o conjunto padronizado de configurações contido no parâmetro **preset**, presente na Seção 4.3.21, de modo a se permitir a comparação com os resultados obtidos.

Em todas as simulações, foram efetuadas 30 execuções do programa para cada configuração. Os testes foram realizados em um computador com processador Intel® Core™ i7 950 de 3,07 GHz, 8GB de memória RAM e 2 discos rígidos de 1 TB e 7200 rpm.

5.2 Base de Dados

Foi utilizada para os testes uma base de dados composta de vídeos em resolução *full HD* e sem compressão, gravados no Laboratório de Processamento de Sinais (LPS) [18] e em uma biblioteca da TV Futura [19], e também algumas sequências de vídeo utilizadas na literatura [20]. Foram gerados 6 vídeos a partir de combinações das sequências de vídeo mencionadas. As sequências *crowd run*, *ducks take off*, *into tree*, *old town*, *park joy*, *blue sky*, *pedestrian area*, *riverbed*, *rush hour*, *station 2*, *sunflower* e *tractor* foram combinadas, gerando 4 vídeos, respeitando-se a restrição de não se utilizar mais que duas vezes cada sequência. Já as sequências gravadas no LPS e na TV Futura foram combinadas gerando 1 vídeo cada. Todos os vídeos gerados possuem duração de 1 a 2 minutos.

5.3 Problemas Encontrados

Nas primeiras execuções do programa, foi detectada uma característica da codificação, exemplificada pela Figura 5.1, que mostra a taxa de codificação atingida conforme os quadros de um vídeo eram codificados. O gráfico mostra que para pelo menos os primeiros 800 quadros, a taxa de codificação obtida chega a ser 50 % maior que a taxa em que o sistema se estabiliza, além de a mesma apresentar uma variação maior para as várias realizações. Isto pode ocorrer devido ao fato de os algoritmos de

controle de taxa ainda estarem coletando informação sobre o conteúdo dos quadros e o processo de compressão. Esse efeito mostra que a velocidade obtida somente se estabiliza após aproximadamente 800 quadros. Assim, os vídeos utilizados devem ter duração superior a 30 s, o que gerou a necessidade de concatenar as sequências, usualmente com duração de 10 s cada.

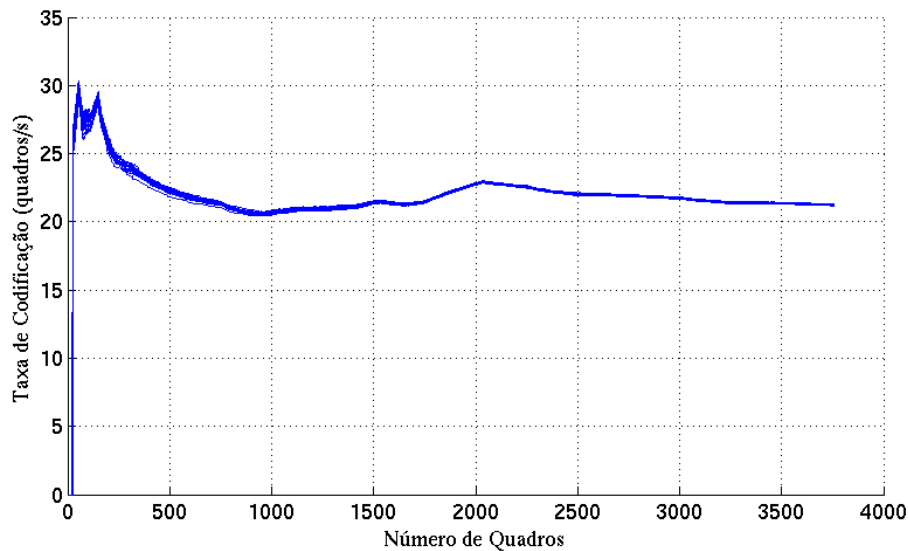


Figura 5.1: Taxa de quadros atingida pelo codificador em função do número de quadros codificados, para diversas realizações.

Também foi constatado que o codificador não mantém o uso do *hardware* em um nível constante. Ao utilizar o programa especificando em 4 o número de *threads*, o mesmo chegou a utilizar, para algumas configurações, somente 1/3 do especificado de CPU, como mostra a Figura 5.2. Esta variação é um problema porque, ao considerar a velocidade de codificação como medida de complexidade de um algoritmo, o ideal seria que todas as outras variáveis fossem iguais, para que a comparação entre as velocidades possa ser realizada de forma quantitativa. Assim, optou-se por utilizar somente 1 *thread*, onde a discrepância entre os percentuais de uso da CPU não é tão grande, mas que acaba aumentando ainda mais o tempo de simulação.


```

top - 17:46:44 up 7 days, 21:32, 4 users, load average: 3.16, 1.47, 1.01
Tasks: 234 total, 2 running, 232 sleeping, 0 stopped, 0 zombie
%Cpu(s): 4.6 us, 1.1 sy, 10.8 ni, 72.8 id, 9.2 wa, 0.6 hi, 0.9 si, 0.0 st
KiB Mem: 8165720 total, 7988308 used, 177412 free, 152212 buffers
KiB Swap: 16948536 total, 948 used, 16947588 free, 5266208 cached

```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
7795	allan.fr	20	0	602m	127m	6096	D	126.6	1.6	1:03.93	mencoder
65	root	20	0	0	0	0	R	1.3	0.0	0:02.28	kswapd0
29914	allan.fr	20	0	4134m	515m	66m	S	1.3	6.5	43:49.92	MATLAB
28794	allan.fr	20	0	1253m	179m	56m	S	1.0	2.3	8:28.67	chrome
7139	allan.fr	20	0	1222m	167m	26m	S	0.7	2.1	0:39.13	chrome
7725	allan.fr	20	0	115m	1552	1048	R	0.7	0.0	0:00.21	top
21363	root	20	0	267m	118m	77m	S	0.7	1.5	13:47.64	X
28436	allan.fr	20	0	3038m	116m	42m	S	0.7	1.5	7:42.37	kwin
29343	allan.fr	20	0	792m	64m	48m	S	0.7	0.8	0:32.67	dolphin
90	root	20	0	0	0	0	S	0.3	0.0	0:41.53	kworker/u:5
5874	root	20	0	0	0	0	S	0.3	0.0	0:00.52	kworker/6:0
6913	root	20	0	0	0	0	S	0.3	0.0	0:00.91	kworker/3:0
6996	root	20	0	0	0	0	S	0.3	0.0	0:00.83	kworker/2:0
28484	allan.fr	20	0	1230m	25m	4228	S	0.3	0.3	0:25.72	mysqld
28521	allan.fr	20	0	555m	24m	18m	S	0.3	0.3	0:10.61	konsole
1	root	20	0	51428	7856	2348	S	0.0	0.1	0:05.98	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.26	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:55.39	ksoftirqd/0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:0H
6	root	20	0	0	0	0	S	0.0	0.0	0:39.30	kworker/u:0
7	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/u:0H
8	root	rt	0	0	0	0	S	0.0	0.0	0:03.83	migration/0
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
10	root	20	0	0	0	0	S	0.0	0.0	2:46.05	rcu_sched
11	root	rt	0	0	0	0	S	0.0	0.0	0:01.54	watchdog/0
12	root	rt	0	0	0	0	S	0.0	0.0	0:02.33	watchdog/1
13	root	rt	0	0	0	0	S	0.0	0.0	0:07.32	migration/1
14	root	20	0	0	0	0	S	0.0	0.0	0:24.47	ksoftirqd/1
16	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/1:0H
17	root	rt	0	0	0	0	S	0.0	0.0	0:02.74	watchdog/2
18	root	rt	0	0	0	0	S	0.0	0.0	0:06.94	migration/2
19	root	20	0	0	0	0	S	0.0	0.0	0:03.14	ksoftirqd/2

Figura 5.2: Quantidade de CPU consumida pelo MEncoder, ao utilizar 4 threads.

5.4 Análise Individual de Parâmetros

5.4.1 Estimação de Movimentos

Está presente nas Tabelas 5.1 e 5.2 o resultado atingido pelos algoritmos de estimação de movimentos *diamond*, *hexagon* e *uneven multi-hexagon*, descritos na Seção 4.3.2. Os valores obtidos mostram que as técnicas mais complexas apresentaram uma enorme perda na eficiência, já que a qualidade obtida pelas mesmas não foi significativamente superior à obtida pelo algoritmo *diamond*, que possui velocidade de codificação notavelmente inferior.

Tabela 5.1: Valores obtidos de qualidade e velocidade de compressão para o algoritmo de estimação de movimentos, com o vídeo 1 e taxa de 10Mb/s

Algoritmo	dia	hex	umh
Qualidade (dB)	39,41	39,42	39,41
Taxa de quadros (quadros/s)	19,51	19,27	17,66
Desvio-padrão (quadros/s)	0,28	0,13	0,15

Tabela 5.2: Valores obtidos de qualidade e velocidade de compressão para o algoritmo de estimação de movimentos, com o vídeo 2 e taxa de 10Mb/s

Algoritmo	dia	hex	umh
Qualidade (dB)	28,33	28,36	28,40
Taxa de quadros (quadros/s)	33,12	30,62	22,65
Desvio-padrão (quadros/s)	1,27	1,27	0,40

5.4.2 Filtro de Desblocagem

O ganho de qualidade do filtro de desblocagem pode ser visto na Tabela 5.3. Percebe-se que mesmo para uma alta taxa de *bits*, o que diminui os defeitos gerados pela compressão, o filtro é capaz de proporcionar alguma melhoria na qualidade obtida, ao passo que não adiciona grande complexidade.

Tabela 5.3: Valores obtidos de qualidade e velocidade de compressão para o filtro de desblocagem, com o vídeo 2 e taxa de 10Mb/s

Desblocagem	Ausente	Presente
Qualidade (dB)	38,40	38,52
Taxa de quadros (quadros/s)	21,25	20,51
Desvio-padrão (quadros/s)	0,27	0,12

5.4.3 Estimação a Nível de Subpíxel

Outro teste realizado foi a relação entre qualidade e complexidade para os vários modos de interpolação a nível de subpíxel presentes na Seção 4.3.3. A partir

dos resultados presentes na Tabela 5.4, não foi possível definir o melhor modo de interpolação, pois eles permitem uma troca entre qualidade e complexidade.

Tabela 5.4: Valores obtidos de qualidade e velocidade de compressão para diferentes modos de predição a nível de subpíxel, com o vídeo 3 e taxa de 10Mb/s

Modo Subpíxel	0	1	2	3	4
Qualidade (dB)	38,40	38,88	39,06	39,15	39,16
Taxa de quadros (quadros/s)	21,25	19,25	16,52	15,95	14,99
Desvio-padrão (quadros/s)	0,27	0,38	0,23	0,47	1,56

5.4.4 Número de Quadros de Referência

Um resultado semelhante está presente nas Tabelas 5.5 e 5.6, que mostram o efeito do aumento do número máximo de quadros de referência considerados ao realizar a estimação. Para as quantidades consideradas, também não foi possível determinar o valor ótimo, de modo que este parâmetro deve ser analisado posteriormente.

Tabela 5.5: Valores obtidos de qualidade e velocidade de compressão para várias quantidades de quadros a serem usados como referência, com o vídeo 3 e taxa de 10Mb/s

Quadros de referência	1	2	3	4
Qualidade (dB)	28,33	28,41	28,44	28,47
Taxa de quadros (quadros/s)	33,12	28,63	25,15	22,43
Desvio-padrão (quadros/s)	1,25	0,44	0,25	0,17

5.4.5 Partições

As Tabelas 5.7, 5.8 e 5.9 apresentam as simulações realizadas utilizando os modos de partição dos macroblocos descritos na Seção 4.3.1. De acordo com os dados obtidos, na maioria dos casos, somente para quadros **I** existe vantagem em particionar os blocos em tamanhos menores que 16×16 píxeis, além de este ter um

Tabela 5.6: Valores obtidos de qualidade e velocidade de compressão para várias quantidades de quadros a serem usados como referência, com o vídeo 4 e taxa de 10Mb/s

Quadros de referência	1	2	3	4
Qualidade (dB)	34,55	34,70	34,74	34,77
Taxa de quadros (quadros/s)	30,88	26,62	23,02	20,34
Desvio-padrão (quadros/s)	0,67	0,34	0,20	0,12

impacto menor na complexidade do codificador. Uma possível justificativa para este fato é a resolução dos vídeos ser de 1920 x 1080, o que pode ser considerado alto em relação a outros vídeos utilizados comercialmente. Por essa razão, a resolução pode ser alta o bastante para que o movimento presente entre os quadros não proporcione um nível de detalhes suficiente para haver vantagem em utilizar blocos de dimensão menor na estimação. Já para quadros **I**, a predição intra com blocos menores apresenta mais modos, o que possibilita uma estimação mais eficaz.

Tabela 5.7: Valores obtidos de qualidade e velocidade de compressão para os modos de partição dos quadros, com o vídeo 2 e taxa de 10Mb/s

Partições	none	i8x8,i4x4	p8x8,p4x4	b8x8	all
Qualidade (dB)	28,33	28,36	28,44	28,43	28,45
Taxa de quadros (quadros/s)	33,12	32,27	25,24	30,72	24,39
Desvio-padrão (quadros/s)	1,27	0,85	0,28	0,46	0,29

Tabela 5.8: Valores obtidos de qualidade e velocidade de compressão para os modos de partição dos quadros, com o vídeo 3 e taxa de 10Mb/s

Partições	none	i8x8,i4x4	p8x8,p4x4	b8x8	all
Qualidade (dB)	38,40	38,49	38,32	38,10	38,37
Taxa de quadros (quadros/s)	21,25	19,40	16,52	20,23	16,02
Desvio-padrão (quadros/s)	0,27	0,59	0,38	0,51	0,35

Tabela 5.9: Valores obtidos de qualidade e velocidade de compressão para os modos de partição dos quadros, com o vídeo 4 e taxa de 10Mb/s

Partições	none	i8x8,i4x4	p8x8,p4x4	b8x8	all
Qualidade (dB)	34,55	34,59	34,54	34,55	34,57
Taxa de quadros (quadros/s)	30,88	29,86	22,47	27,58	21,66
Desvio-padrão (quadros/s)	0,67	0,55	0,19	0,28	0,15

5.4.6 Transformadas 8×8

Para a DCT, também ocorre um efeito semelhante. Ao possibilitar o uso da DCT com tamanho 8×8 , e deixar o codificador escolher entre este tamanho ou o 4×4 , a qualidade obtida foi superior ao que aconteceria se somente a DCT de tamanho 4×4 fosse utilizada, como está presente na Tabela 5.10. Isto mostra que, para os vídeos utilizados, a alta resolução faz com que os blocos de tamanho 8×8 possuam píxeis com valores suficientemente similares para a DCT ter um bom desempenho, ao passo que, em blocos 4×4 , a DCT também funciona corretamente, mas vai haver um gasto de *bits* com as informações de controle referentes aos blocos. Entretanto, entre os parâmetros considerados, este é o único que, uma vez habilitado, implica em um vídeo pertencente ao perfil *high*, perdendo compatibilidade com todos os decodificadores do perfil *main*. Assim, habilitar a DCT 8×8 pode nem sempre ser uma boa opção.

Tabela 5.10: Valores obtidos de qualidade e velocidade de compressão para o uso de DCT com tamanho 8×8 , com o vídeo 3 e taxa de 10Mb/s

DCT 8×8	Ausente	Presente
Qualidade (dB)	38,40	38,60
Taxa de quadros (quadros/s)	21,25	20,66
Desvio-padrão (quadros/s)	0,27	0,46

5.4.7 Codificação Aritmética

Os modos de codificação adaptativa CABAC e CAVLC tiveram seu desempenho comparado, como visto na Tabela 5.11. Percebe-se que o uso do CABAC acarreta em um aumento significativo na qualidade obtida, enquanto ocasiona uma pequena queda na velocidade obtida.

Tabela 5.11: Valores obtidos de qualidade e velocidade de compressão para do CABAC, com o vídeo 3 e taxa de 10Mb/s

CABAC	ausente	presente
Qualidade (dB)	38,40	39,25
Taxa de quadros (quadros/s)	21,25	19,88
Desvio-padrão (quadros/s)	0,27	0,42

5.4.8 Quadros B

Foram analisados também os efeitos do uso de quadros **B** na codificação. Inicialmente, a Tabela 5.12 mostra os valores obtidos para três quantidades distintas de quadros **B**, sem utilizar nenhum algoritmo de decisão do número ótimo de quadros, ou seja, o codificador considera que entre quadros **P** e **I** existem sempre a mesma quantidade de quadros **B**, dados pelo parâmetro **bframes**. Como pode ser visto na tabela, a falta de um modo de decisão que escolha a quantidade ótima de quadros **B** faz com que o codificador perca eficiência ao permitir um maior número de quadros deste tipo.

Tabela 5.12: Valores obtidos de qualidade e velocidade de compressão para a máxima quantidade de quadros **B** consecutivos sem utilizar algoritmo adaptativo de decisão, com o vídeo 3 e taxa de 10Mb/s

Quadros B	0	8	16
Qualidade (dB)	38,40	38,06	37,79
Taxa de quadros (quadros/s)	21,25	20,07	20,12
Desvio-padrão (quadros/s)	0,27	0,45	0,34

Por consequência, foram averiguados os algoritmos que atuam em conjunto com os quadros **B**, ou controlando os mesmos. As Tabelas 5.13 e 5.14 expõem os dados obtidos ao codificar um sinal de vídeo utilizando, respectivamente, os algoritmos adaptativos **1** e **2** de controle da quantidade utilizada de quadros **B**, vistos na Seção 4.3.10. Nesta simulação, o algoritmo **2** apresentou um ganho de qualidade em relação ao uso da quantidade máxima de quadros **B** da Tabela 5.12, mas aumentando drasticamente a complexidade. Já o algoritmo **1** apresentou desempenho também superior em qualidade, porém com um aumento na complexidade mais aceitável.

Tabela 5.13: Valores obtidos de qualidade e velocidade de compressão para a máxima quantidade de quadros **B** consecutivos utilizando o algoritmo adaptativo de decisão **1**, com o vídeo 3 e taxa de 10Mb/s

Quadros B	0	8	16
Qualidade (dB)	38,40	38,68	38,68
Taxa de quadros (quadros/s)	21,25	19,55	19,58
Desvio-padrão (quadros/s)	0,27	0,37	0,38

Tabela 5.14: Valores obtidos de qualidade e velocidade de compressão para a máxima quantidade de quadros **B** consecutivos utilizando o algoritmo adaptativo de decisão **2**, com o vídeo 3 e taxa de 10Mb/s

Quadros B	0	8	16
Qualidade (dB)	38,40	38,34	38,34
Taxa de quadros (quadros/s)	21,25	5,98	2,61
Desvio-padrão (quadros/s)	0,27	0,27	0,20

Já a Tabela 5.15 mostra o resultado do modo que possibilita a quadros **B** serem utilizados como referência para a estimação de movimentos. Comparando à Tabela 5.12, a presença deste modo demonstrou desempenho superior à sua ausência.

Finalmente, avaliou-se o desempenho conjunto dos melhores modos anteriores com várias quantidades de quadro **B**, para outros vídeos. Os resultados presentes nas Tabelas 5.16 e 5.17 expõem que, utilizando algoritmos adaptativos de decisão

Tabela 5.15: Valores obtidos de qualidade e velocidade de compressão para a máxima quantidade de quadros **B** consecutivos permitindo que quadros **B** sejam usados como referência. com o vídeo 3 e taxa de 10Mb/s

Quadros B	0	8
Qualidade (dB)	38,40	38,28
Taxa de quadros (quadros/s)	21,25	19,84
Desvio-padrão (quadros/s)	0,27	0,56

e permitindo o uso de quadros **B** como referência, não é vantajoso utilizar uma quantidade maior do que 3 quadros **B** consecutivos, pois a qualidade obtida não aumenta ao ultrapassar este limite.

Tabela 5.16: Valores obtidos de qualidade e velocidade de compressão para a máxima quantidade de quadros **B** consecutivos permitindo que quadros **B** sejam usados como referência e utilizando o algoritmo adaptativo de decisão **1**, com o vídeo 1 e taxa de 10Mb/s

Quadros B	0	1	2	3	16
Qualidade (dB)	39,41	39,92	39,98	40,04	40,04
Taxa de quadros (quadros/s)	19,51	18,44	18,23	18,07	18,03
Desvio-padrão (quadros/s)	0,28	0,16	0,20	0,13	0,04

Tabela 5.17: Valores obtidos de qualidade e velocidade de compressão para máxima a quantidade de quadros **B** consecutivos permitindo que quadros **B** sejam usados como referência e utilizando o algoritmo adaptativo de decisão **1**, com o vídeo 2 e taxa de 10Mb/s

Quadros B	0	1	2	3	16
Qualidade (dB)	28,33	29,03	29,06	29,09	29,09
Taxa de quadros (quadros/s)	33,12	30,68	29,50	29,38	29,05
Desvio-padrão (quadros/s)	1,27	1,27	1,17	1,07	0,60

5.4.9 Modos Ótimos

Após realizar a análise para todos os parâmetros considerados, foi gerada uma lista com os melhores modos de operação para cada um deles, como está contido na Tabela 5.18. Somente para dois parâmetros, **subq**, que controla os algoritmos de estimação a nível de subpíxel, e **ref**, que especifica o número máximo de quadros de referência, não foi possível determinar a melhor escolha. Assim sendo, para estes, será realizada uma análise posterior, descrita a seguir.

Tabela 5.18: Melhores modos de operação para os parâmetros testados.

Para âmetro	Configuração	Parâmetro	Configuração
partitions	i8x8,i4x4	direct_pred	spatial
me	dia	ref	não definido
subq	não definido	8x8dct	1
chroma_me	0	cabac	1
mixed_refs	0	mbtree	0
weightp	0	rc_lookahead	0
weight_b	0	deblock	1
bframes	3	aq_mode	0
b_adapt	1	trellis	1
b_pyramid	1		

5.5 Análise Combinacional de Parâmetros

Com base no conjunto de parâmetros mantidos fixos, foram realizados testes com as combinações dos possíveis valores assumidos pelos parâmetros que necessitam de maior investigação, presentes na Tabela 5.19. Os valores considerados foram aqueles que implementam algoritmos de maior velocidade pois, para os outros, o tempo de simulação aumentaria substancialmente e os resultados fugiriam do escopo de aplicação em tempo real. A partir dos resultados, gerou-se uma nuvem de pontos, no plano qualidade *versus* velocidade de codificação, de onde foi possível extrair uma curva que representa os pontos ótimos de operação para diversos valores de

complexidade.

Parametro	Valores Assumidos
subq	modos 0 a 4
ref	1 a 4

Tabela 5.19: Valores assumidos pelos parâmetros testados combinadamente.

As simulações foram realizadas para todos os vídeos gerados, restringindo a taxa de *bits* para os valores de 1, 2, 5 e 10 Mb/s. As Figuras 5.3, 5.4, 5.5 e 5.6 mostram a nuvem de pontos e a curva ótima para os vídeos **2** e **6**, com algumas taxas de *bits*.

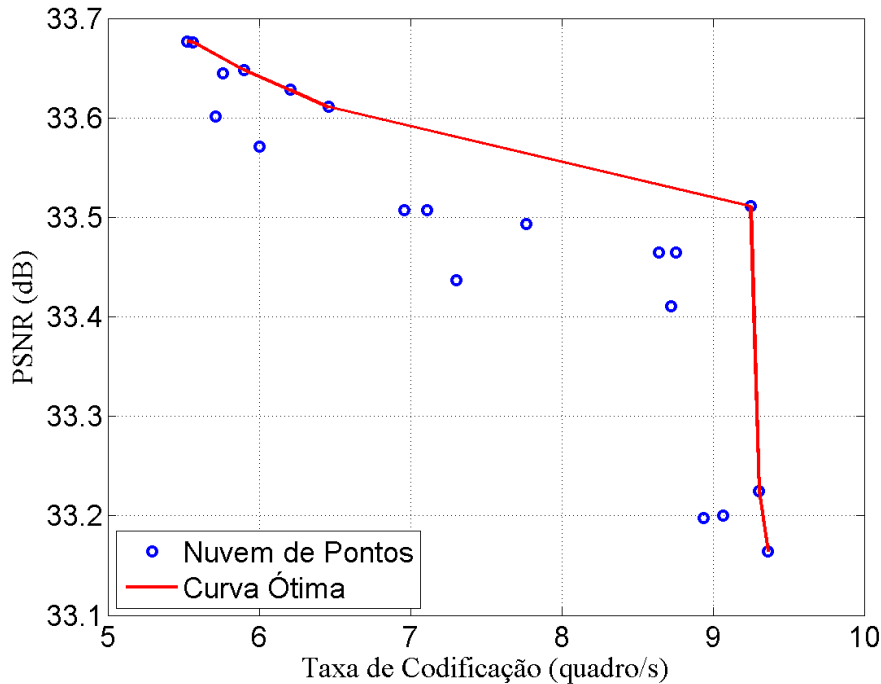


Figura 5.3: Nuvem de pontos e sua respectiva curva ótima para o vídeo 2, com taxa de 1 Mb/s.

Já para validação, estas curvas foram comparadas ao resultado obtido pelo parâmetro **preset**, devido ao fato de este parâmetro definir um conjunto de configurações que variam gradualmente a qualidade e a complexidade. As Figuras 5.7, 5.8, 5.9 e 5.10 exibem as curvas obtidas pela otimização dos pontos obtidos anterior-

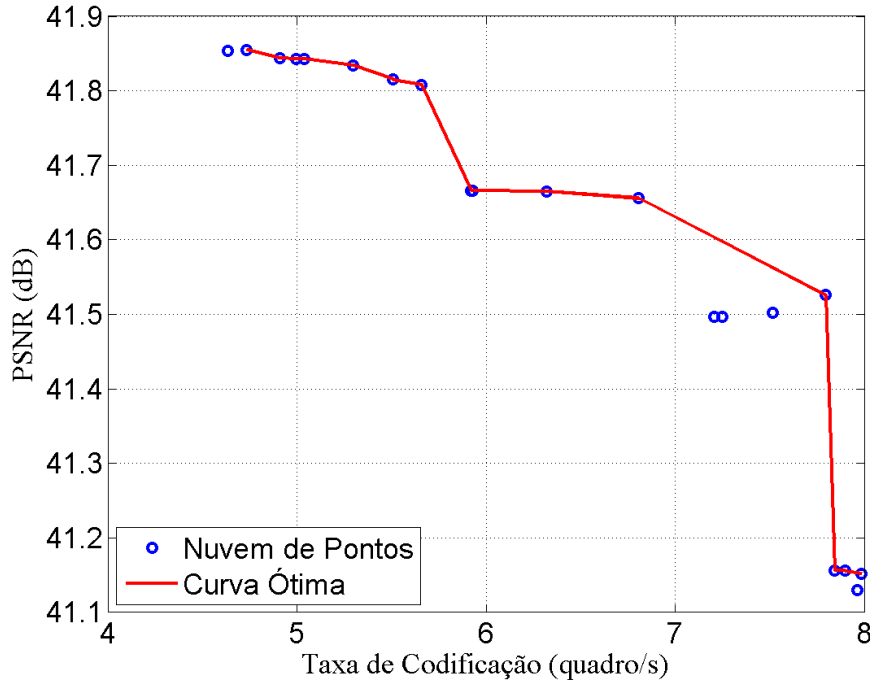


Figura 5.4: Nuvem de pontos e sua respectiva curva ótima para o vídeo 2, com taxa de 10 Mb/s.

mente, em comparação à curva gerada pelos valores **ultrafast**, **superfast**, **veryfast** e **faster** do parametro **preset**.

Como somente foram considerados dois parâmetros nesta etapa, e os mesmos assumiram somente alguns valores, a curva gerada corresponde a uma faixa estreita de velocidade. Porém, as curvas também demonstraram que as configurações obtidas possuem desempenho até 1,3 dB superior ao padrão com perfis de complexidade da biblioteca. Além disso, as mesmas curvas também possuem uma inclinação bem menor em relação ao eixo de velocidade, o que mostra que estas configurações permitem variar a complexidade culminando em um impacto menor na qualidade.

Outro efeito a ser observado é que para algumas curvas a qualidade se mantém mais ou menos linear com a velocidade até um determinado ponto a partir do qual ocorre uma grande descontinuidade, destacada na Figura 5.11. Este fato ocorre para alguns vídeos e taxas de *bits*, na transição entre o parâmetro **subq** passar do modo **1** para o modo **0**, o que mostra que mesmo no caso otimizado existe um impacto enorme em se utilizar ou não de estimação a nível de subpíxel.

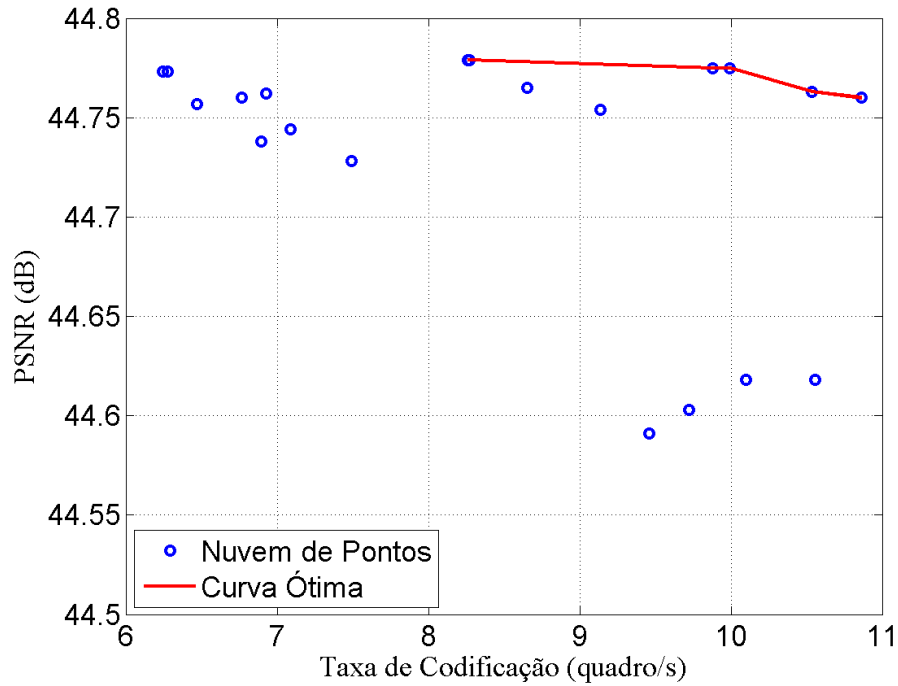


Figura 5.5: Nuvem de pontos e sua respectiva curva ótima para o vídeo 6, com taxa de 2 Mb/s.

Analisando para todos os vídeos, chegou-se ao conjunto de configurações presente na Tabela 5.20. Para cada taxa, foram selecionadas as configurações um bom desempenho para a maior parte dos vídeos testados. Com taxa de *bits* de 1 Mb/s, o conjunto gerado possui poucas configurações, devido ao fato de que, para esta taxa, o sistema executa a codificação com maior velocidade, e a mesma se torna mais imprecisa, porque o sistema pode não conseguir utilizar toda a CPU disponível, já que os algoritmos necessitam de pouco poder computacional, resultando num problema semelhante ao uso de vários *threads*.

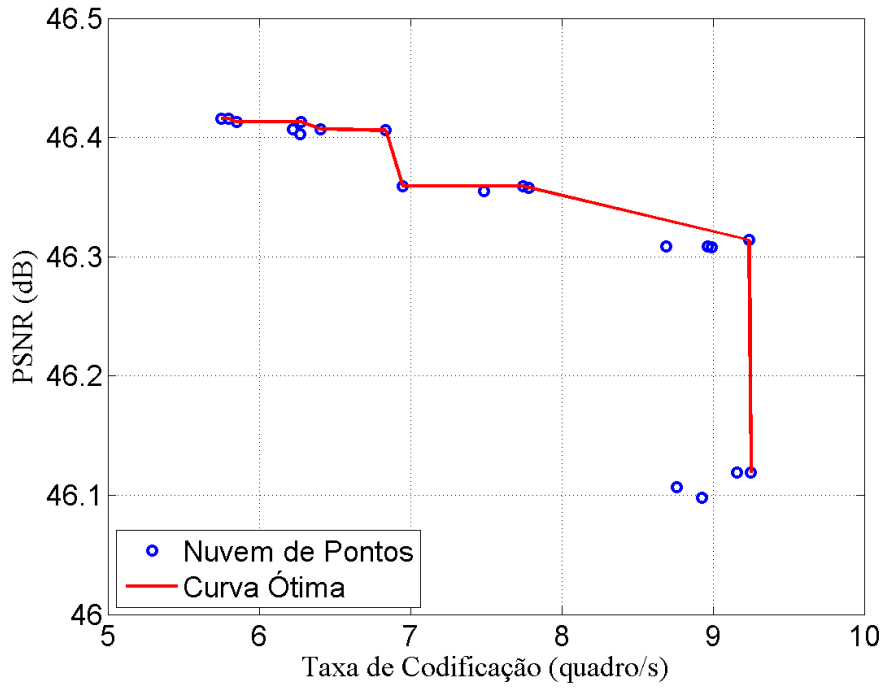


Figura 5.6: Nuvem de pontos e sua respectiva curva ótima para o vídeo 6, com taxa de 5 Mb/s.

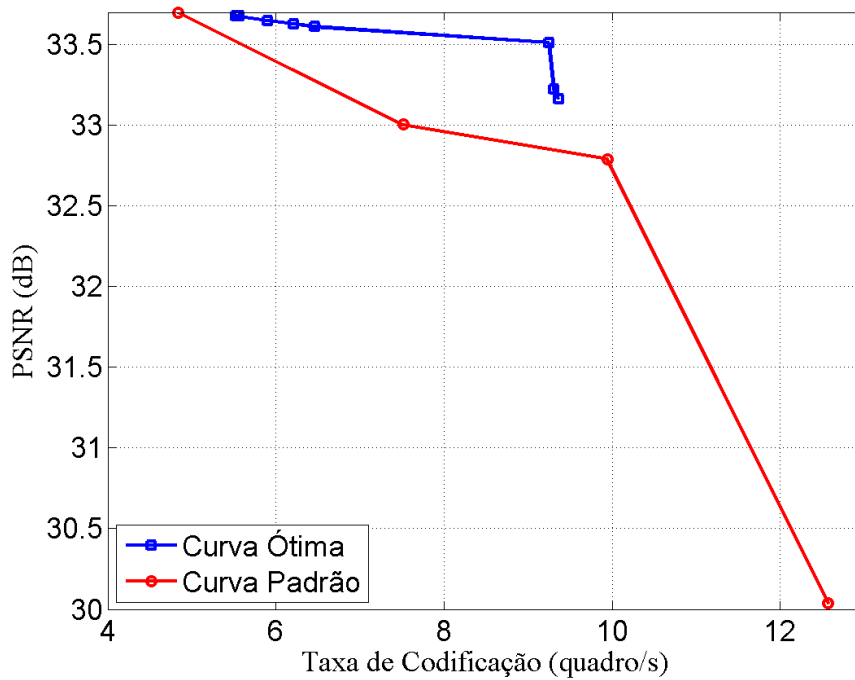


Figura 5.7: Nuvem de pontos e sua respectiva curva ótima para o vídeo 2, com taxa de 1 Mb/s.

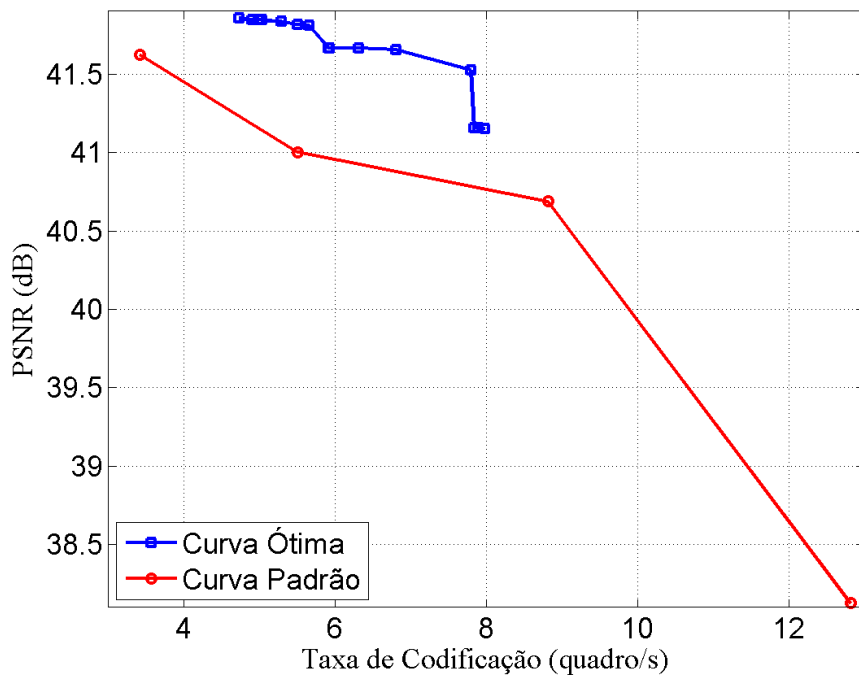


Figura 5.8: Nuvem de pontos e sua respectiva curva ótima para o vídeo 2, com taxa de 10 Mb/s.

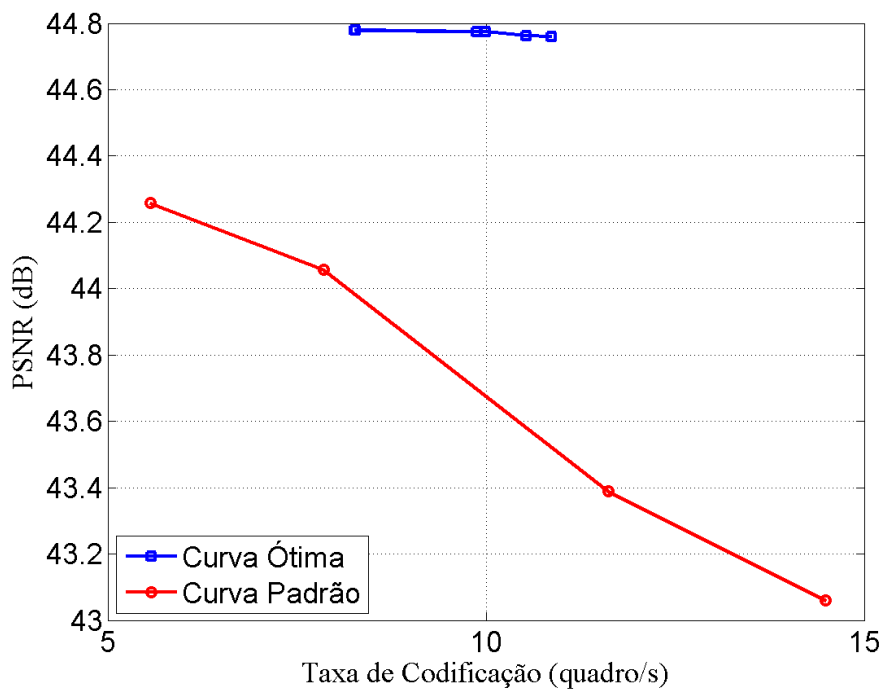


Figura 5.9: Nuvem de pontos e sua respectiva curva ótima para o vídeo 6, com taxa de 2 Mb/s.

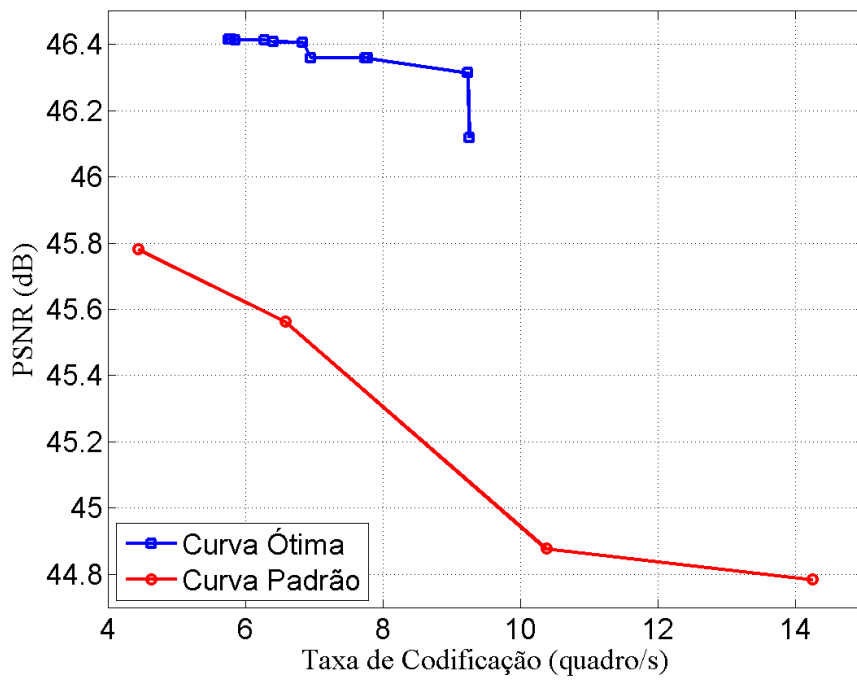


Figura 5.10: Nuvem de pontos e sua respectiva curva ótima para o vídeo 6, com taxa de 5 Mb/s.

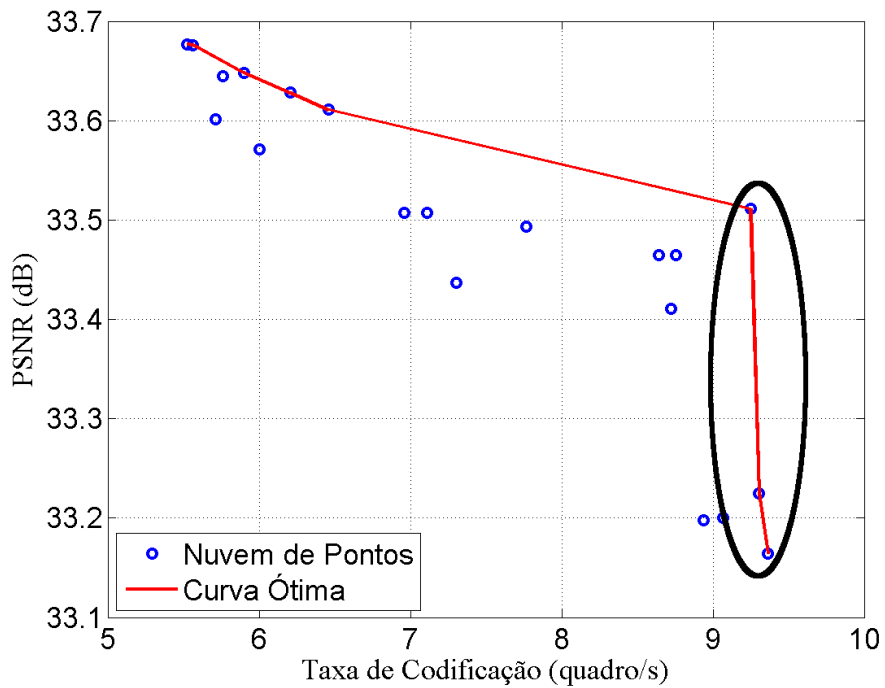


Figura 5.11: Exemplo da descontinuidade na qualidade obtida.

Tabela 5.20: Configurações ótimas para as taxas de *bits* testadas. Cada linha mostra as configurações obtidas para cada taxa em ordem decrescente de complexidade, da esquerda para a direita

Taxa	Configurações									
1000	ref=4:subq=1	subq=1								
2000	ref=4:subq=4	ref=3:subq=4	ref=4:subq=3	ref=2:subq=3	subq=4	subq=3	subq=3	subq=2	subq=2	subq=1
5000	ref=4:subq=4	ref=4:subq=3	ref=2:subq=3	subq=4	subq=3	ref=4:subq=2	ref=2:subq=2	subq=2	subq=2	subq=1
10000	ref=3:subq=4	ref=3:subq=3	ref=2:subq=3	subq=4	subq=3	ref=3:subq=2	subq=2	ref=3:subq=1	subq=1	

5.6 Codificação em Tempo Real

O método descrito foi executado em um sistema onde foi limitado o poder computacional do *hardware*, colocando somente um *thread*, para a obtenção de resultados mais coerentes. Entretanto, isto faz com que o codificador não atinja uma velocidade de codificação alta o bastante para aplicações em tempo real. Dessa forma, é preciso aproveitar os vários núcleos do processador utilizado.

As configurações ótimas presentes na Tabela 5.20 foram testadas para o conjunto de vídeos, utilizando desta vez um maior número de *threads*. As Figuras 5.12, 5.13, 5.14 e 5.15 apresentam resultados para 4 *threads*, com os mesmos vídeos e taxas anteriores. Os gráficos mostram que as curvas geradas permitem maximizar a qualidade com um certo compromisso de taxa de codificação entre 20 e 40 quadros/s, o que abrange as velocidades mais frequentes de exibição de quadros. Dessa forma, o sistema é capaz de operar em tempo real.

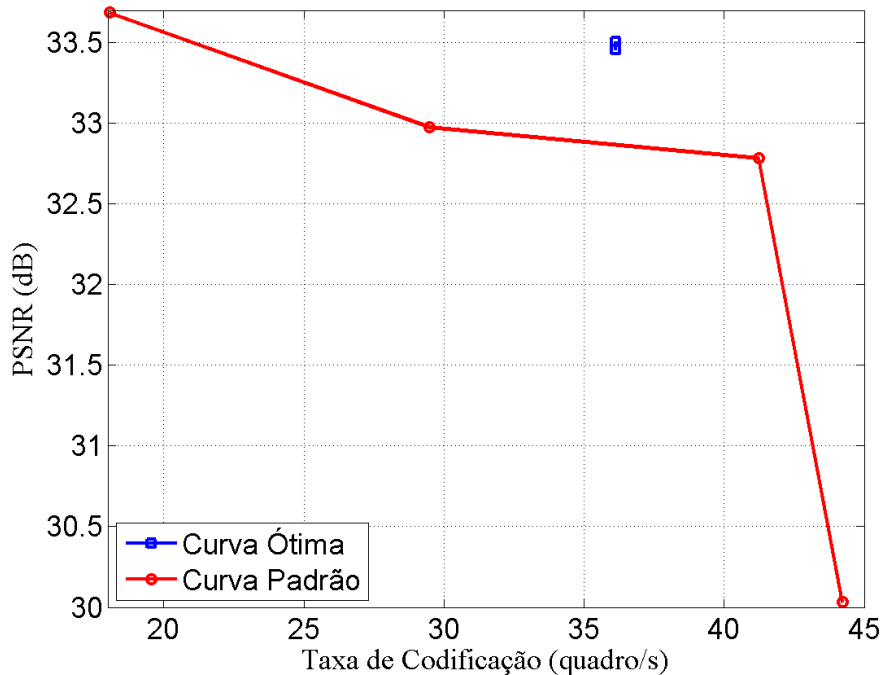


Figura 5.12: Configurações ótimas aplicadas no vídeo 2, com taxa de 1 Mb/s e 4 *threads*.

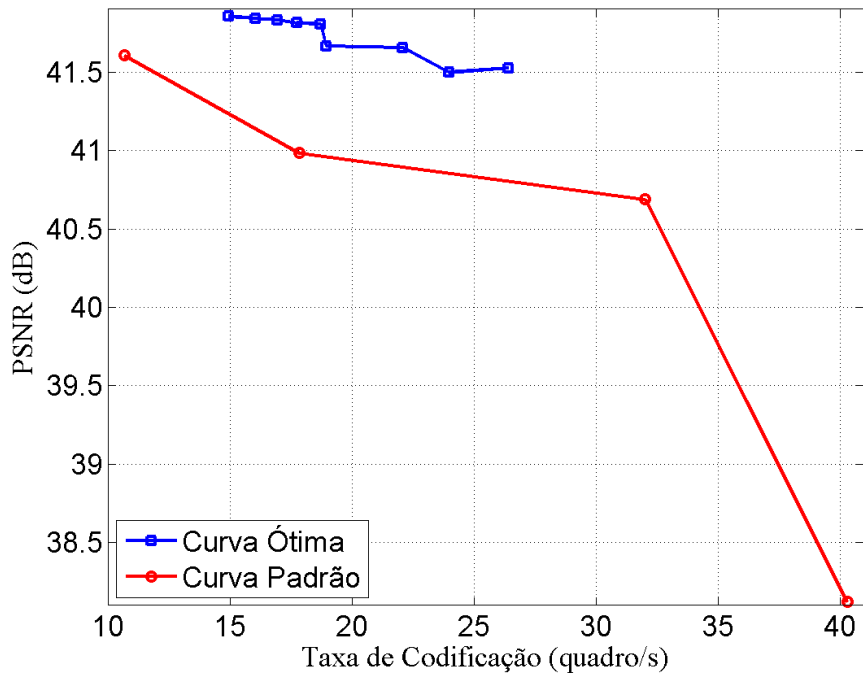


Figura 5.13: Configurações ótimas aplicadas no vídeo 2, com taxa de 10 Mb/s e 4 threads.

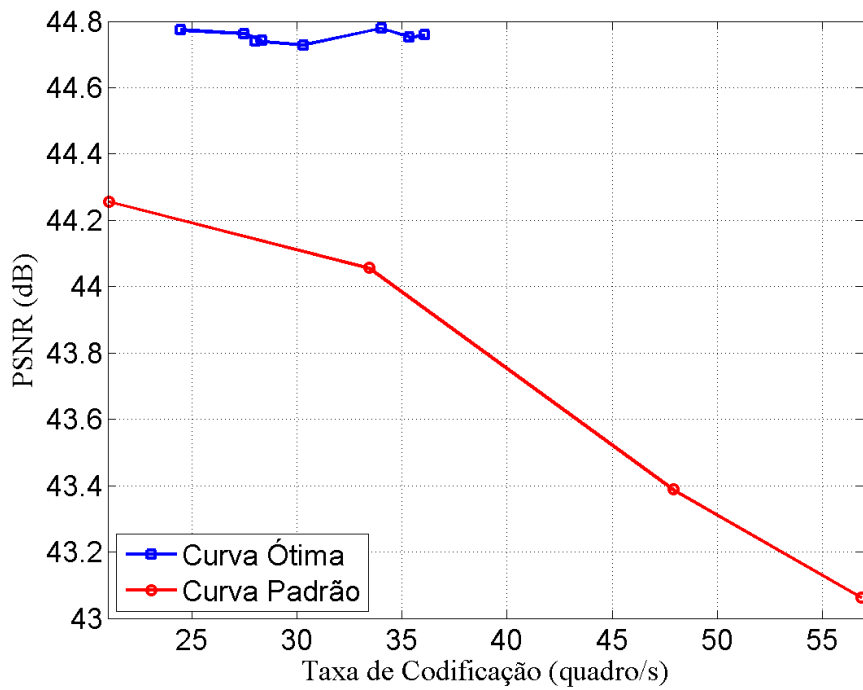


Figura 5.14: Configurações ótimas aplicadas no vídeo 6, com taxa de 2 Mb/s e 4 threads.

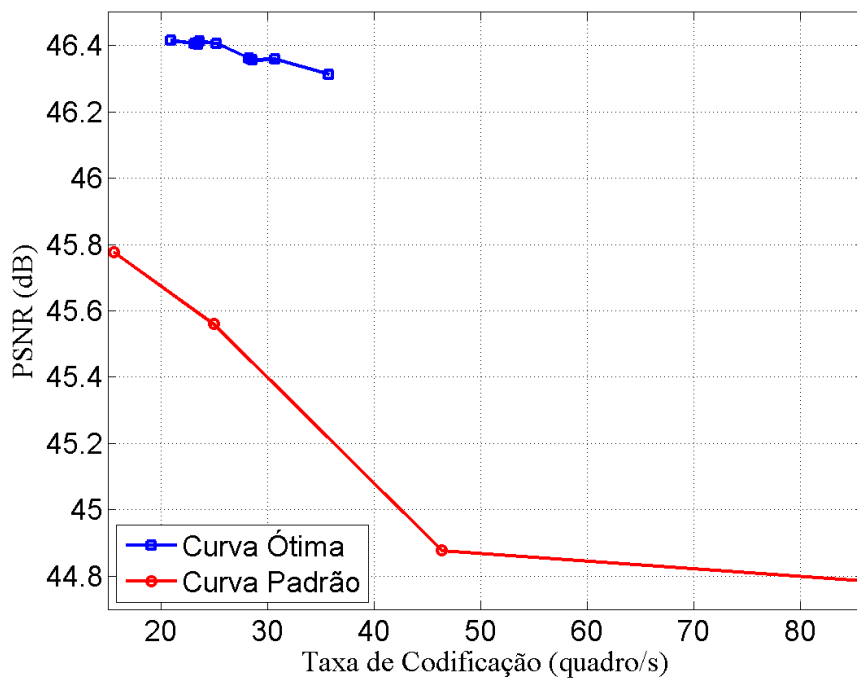


Figura 5.15: Configurações ótimas aplicadas no vídeo 6, com taxa de 5 Mb/s e 4 threads.

Capítulo 6

Conclusões

Neste capítulo serão feitas as considerações finais a respeito do trabalho realizado.

6.1 Considerações Finais

Neste trabalho, averigou-se a codificação em tempo real, via *software*, de vídeos em resolução *full HD* para o padrão H.264, utilizando a biblioteca de codificação x264 [2]. Para tanto, foi realizado um estudo dos algoritmos e técnicas relacionadas à codificação, seguido de uma análise do impacto na qualidade e na complexidade dos principais parâmetros da biblioteca que controlam os aspectos da codificação.

A análise consistiu de experimentos onde foi simulada a codificação, através da biblioteca x264, utilizando determinados conjuntos de configurações, de onde foram feitas medidas de qualidade, na forma da PSNR, e de complexidade, pela velocidade de codificação obtida pelo programa, em quadros por segundo. A partir dos resultados, foi possível determinar os melhores modos de atuação do codificador, que permitem a máxima qualidade obtida para uma dada complexidade.

Os resultados experimentais, se comparados aos obtidos pelos perfis presentes na biblioteca que definem diversos níveis de complexidade, apresentaram ganho de qualidade superior a 1 dB para uma mesma complexidade. Este ganho, apesar de pequeno, é compatível com a filosofia utilizada no desenvolvimento do padrão H.264, em que pequenas melhorias pontuais foram sendo acumuladas, visto que o uso das

configurações ótimas não traz nenhuma desvantagem aparente no codificador.

Por fim, foi verificada a velocidade obtida pelo sistema otimizado, que foi, em geral, entre 20 e 35 quadros/s, para um computador com processador Intel® Core™ i7 950 de 3,07 GHz, 8GB de memória RAM e 2 discos rígidos de 1 TB e 7200 rpm. Dessa forma, o sistema é capaz de codificar vídeos em tempo real compatíveis com um grande número de aplicações.

6.2 Trabalhos Futuros

O estudo realizado limitou-se a um conjunto de parâmetros, já que eles são muitos. Um estudo posterior poderia se focar em ampliar o conjunto de opções, obtendo mais resultados relevantes. Para a análise combinacional dos parâmetros, o conjunto foi ainda mais restrito, já que o número de combinações cresce exponencialmente com o número de parâmetros considerados nesta etapa. Assim, a curva ótima obtida poderia se estender por uma faixa de valores maior.

Outra melhoria ao sistema envolveria o uso de outros bancos de dados de vídeos com outras características, que possam ser agrupados em aplicações específicas, como vídeos de jornalismo, esporte, filmes, animações, etc. Dessa forma, o estudo apresentado poderia ser aprimorado se, ao invés de considerar os melhores modos somente para cada taxa de bits, o sistema também considerasse o tipo de vídeo utilizado, visto que cada tipo de vídeo se favorece mais com algumas técnicas (por exemplo, um vídeo de jornalismo possui vários trechos com fundo estático, enquanto algumas animações possuem vários trechos em que o fundo varia periodicamente).

Referências Bibliográficas

- [1] JVT of ISO/IECMPEGand ITU-T VCEG. **Advanced Video Coding for Generic Audiovisual Services**. [S.l.], March 2005.
- [2] Biblioteca x264. Disponível em: <http://www.videolan.org/developers/x264.html>, Acessado em Maio de 2013.
- [3] MEncoder, Codificador de Vídeo. Disponível em: <http://www.mplayerhq.hu/design7/news.html>, Acessado em Maio de 2013.
- [4] MERRITT, L.; VANAM, R. **Improved rate control and motion estimation for H.264 encoder**. In: Proc. IEEE International Conference on Image Processing. [S.l.: s.n.], 2007. v. 5.
- [5] JM, *Software* do Codificador de Referência H.264/AVC. Disponível em: <http://iphone.hhi.de/suehring/tml/>, Acessado em Maio de 2013.
- [6] ITU-T and ISO/IEC JTC 1. **Generic coding of moving pictures and associated audio information – Part 2: Video**. ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG-2), Nov. 1994 (with several subsequent amendments and corrigenda).
- [7] ITU-T. **Video codec for audiovisual services at px64 kbits/s**. ITU-T Rec. H.261 v1: Nov 1990, v2: Mar.1993.
- [8] RAO, K. R. ; YIP, P. C., **The Transform and Data Compression Handbook**, CRC Press, Inc., Boca Raton, FL, 2000.
- [9] AHMED, N.; NATARAJAN, T.; RAO, K. R. **On image processing and a discrete cosine transform**. In: IEEE Trans. on Computers, IEEE, C-23, n. 1, p. 90–93, 1974.

- [10] MALVAR, H. S. et al. **Low-Complexity transform and quantization in H.264/AVC**. In: IEEE Transactions on Circuits and Systems for Video Technology, v. 13, n. 13, p. 590–603, July 2003.
- [11] RICHARDSON, I. E. G., **The H.264 Advanced Video Compression Standard**. Wiley. pp. 208,221, 2010.
- [12] SULLIVAN, G. J.; TOPIWALA, P.; LUTHRA, A. **The H.264/AVC advanced video coding standard: Overview and introduction to the fidelity range extensions**. Proc. SPIE Conference on Applications of Digital Image Processing XXVII, v. 5558(1), p. 454–474, Aug. 2004.
- [13] **MSU Video Codec Comparison**. Disponível em: http://compression.ru/video/codec_comparison/index_en.html, Acessado em junho de 2013.
- [14] DUMA, Luiz Henrique., **Uma Proposta de Método para Melhoria de Desempenho do Codificador x264 Baseada na Análise do Acesso ao Barramento Externo de Memória**, 2011, 82 f, Dissertação de Mestrado em Engenharia Elétrica e Informática Industrial, Universidade Tecnológica Federal do Paraná, Curitiba, 2011
- [15] Ji ZHANG, Yu-bei LIN, Xingming ZHANG. **An improved intra mode decision method for x264 video encoding based on hybrid distortion discriminated criteria**. In: International Conference on Wavelet Analysis and Pattern Recognition, Xian, July, 2012.
- [16] Weilin WU, Xingming ZHANG. **Code performance improvement scheme for X264 based on SSIM**. In: International Conference on Wavelet Analysis and Pattern Recognition, Xian, July, 2012.
- [17] FONSECA, T. A., **Codificação de Vídeo Escalonável em Complexidade e em Energia**, Tese de Doutorado em Engenharia de Sistemas Eletrônicos e de Automação, Publicação PGEA.TD - 057/2012, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 139 pp.

- [18] LPS, Sequências *full HD* gravadas no Laboratório de Processamento de Sinais, Disponíveis em <http://www.lps.ufrj.br/~tvdigital/muque/data/lps/original>, Acessado em Junho de 2012.
- [19] FUTURA, Sequências *full HD* gravadas na biblioteca da TV Futura, Disponíveis em <http://www.lps.ufrj.br/~tvdigital/muque/data/futura/original>, Acessado em Junho de 2012.
- [20] XIPH, Sequências *full HD* da Xiph.org Video Test Media, Disponíveis em <http://media.xiph.org/video/derf>, Acessado em Maio de 2013.