ALGORITMO DE CASAMENTO DE PADRÕES APLICADO NA ESTIMAÇÃO DE MOVIMENTO EM COMPRESSÃO DE VÍDEO

POR

RENÊ TOGNI DEL PIETRO

Bacharel em Engenharia Elétrica pela Unicamp (1990)

Mestre em Engenharia Elétrica pela Unicamp (1998)

TESE SUBMETIDA À FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO DA UNICAMP COMO PARTE DOS REQUISITOS EXIBIDOS PARA A OBTENÇÃO DO TÍTULO DE

DOUTORADO EM ENGENHARIA ELÉTRICA

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

DEPARTAMENTO DE COMUNICAÇÕES

Banca Examinadora:

Prof. Dr. Max Henrique Machado Costa (Orientador) – FEEC – Unicamp.
Prof. Dr. Abraham Alcaim – PUC – RJ.
Prof. Dr. Eduardo Antonio Barros da Silva – COPPE – UFRJ.
Prof. Dr. Dalton Soares Arantes – FEEC – Unicamp.
Prof. Dr. Yuzo Iano – FEEC – Unicamp.
Prof. Dr. João Marcos Romano – FEEC – Unicamp.

Campinas, 10 de Janeiro de 2003.

Este exemplar corresponde a redação final da tese defendida por Renê Togni Del Pietro e aprovada pela Comissão julgada em 10 / 01 / 2003

Orientador

UNICAMP
BIBLIOTECA CENTR

UNIDADE BC UNICAMP
Nº CHAMADA D37a
V ____ EX ____
TOMBO BC/ 55763
PROC. 16.124/03
C ☐ D ☒
PREÇO R$ 11,00
DATA 17/09/03
Nº CPD ____

CM00189061-1
bib id300930

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

~~P618a~~
D37a

Pietro, Renê Togni Del
Algoritmo de casamento de padrões aplicado na estimação de movimento em compressão de vídeo / Renê Togni Del Pietro. --Campinas, SP: [s.n.], 2003.

Orientador: Max Henrique Machado Costa.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Compressão de vídeo. 2. Estimação de movimento. 3. Videodigital. 4. Transmissão de imagem. 5. Compressão de imagens. I. Costa, Max Henrique Machado. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

RESUMO

Com a aplicação das técnicas de estimação e compensação de movimento em seqüências de vídeo, consegue-se reduzir o número de bits gerados na codificação e aumentar a qualidade da seqüência processada. Em contrapartida, o alto custo computacional do processo exige desempenho do codificador. O objetivo desta tese é propor um novo algoritmo de estimação que concilia qualidade de processamento e rapidez computacional. Realiza-se uma análise do problema de codificação e dos vários algoritmos de estimação utilizados, enfatizando-se a questão da qualidade e da velocidade de processamento. Apresenta-se um algoritmo de estimação inédito denominado MSEA (uma modificação do algoritmo SEA). O algoritmo SEA (*Successive Elimination Algorithm*) obtém o vetor de movimento ótimo fazendo uma pré-seleção de macroblocos na região de pesquisa. Modifica-se o algoritmo SEA introduzindo-se novos critérios de descarte executados em ordem crescente de complexidade e criando uma ordem de leitura dos macroblocos dentro da região de pesquisa. Tem-se um ganho expressivo na velocidade de processamento, conservando-se a solução ótima obtida pela busca exaustiva ou pelo algoritmo SEA. O algoritmo é implementado como parte de um codificador MPEG-2 e mostra-se que o MSEA atinge curvas de taxa × distorção adequadas, a um custo computacional inferior aos de seus concorrentes. Mostra-se, também, a possibilidade de se mesclar o MSEA com outros algoritmos de estimação, resultando na otimização das curvas de taxa × distorção × tempo de processamento frente aos métodos de estimação costumeiramente utilizados na prática (busca logarítmica e busca hierárquica).

Versão Final

ABSTRACT

With the application of motion estimation and compensation techniques in video sequences, we can reduce the number of bits needed to encode the video sequence and to increase the quality of the reconstructed video signal. On the other hand, the high computational cost of the process demands high performance of the encoder. This thesis proposes a new motion estimation algorithm that reconciles processing quality and computational speed. Coding problems and several estimation algorithms are analyzed with emphasis on the quality and processing speed issues. A new motion estimation algorithm, called MSEA (a modification of the SEA algorithm) is presented. The algorithm known as SEA (Successive Elimination Algorithm) obtains the optimum motion vector by performing a pre-selection of macroblocks in the search area. The SEA algorithm is modified by introducing new discard criteria applied in increasing order of complexity and by reordering the reading of macroblocks inside of the search area. An expressive improvement in processing speed is obtained and the optimum solution of the motion vector estimation, also achieved by exhaustive search and SEA, is preserved. The algorithm is implemented as part of an MPEG-2 encoder and it is shown that MSEA achieves adequate rate $\times$ distortion curves at a smaller computational cost than its competitors. Also, the possibility is shown of mixing MSEA with other estimation algorithms, improving the rate $\times$ distortion $\times$ processing time curves with respect to other estimation methods used in practice (two dimensional logarithm search and hierarchical search)

Aos meus pais, Adolpho e Margarida Togni Del Pietro,
a quem devo tudo o que sou.

AGRADECIMENTOS

Gostaria de agradecer a todos que, direta ou indiretamente, contribuíram para a realização deste trabalho. Agradeço, em especial, ao amigo e orientador Prof. Max H. M. Costa pelo estímulo, paciência e conhecimento que, ao longo destes anos, conduziram-me para a realização desta tese. Meu muito obrigado ao Prof. Yuzo Iano cujo incentivo e o debate sobre o tema originou a idéia do algoritmo. Agradeço ao Luis Rômulo, Fabbryycio, Alexandra e outros colegas de curso o suporte computacional na implementação dos algoritmos. Gostaria de agradecer também a secretária do departamento, Sra. Maria Lúcia de Costa Cardoso, o zelo com que conduziu os assuntos burocráticos relacionados ao trabalho. Gostaria, por fim, de agradecer ao CNpQ o apoio financeiro, parte de uma proposta arrojada de promover o desempenho científico e tecnológico do país.

ÍNDICE

1 INTRODUÇÃO 1

2 PRINCÍPIOS DA CODIFICAÇÃO DE VÍDEO 5

2.1 - Introdução 5
2.2 - Sistema de Comunicação de Sinais de Vídeo 5
2.3 - Codificador e Decodificador de Fonte 7
2.4 - Padrão MPEG-2 8
2.5 - Fundamentos do Algoritmo de Compressão MPEG 9
2.5.1 - Codificação Intraquadros 10
2.5.1.1 - Conversor RGB para YCrCb 11
2.5.1.2 - Segmentação em Blocos 12
2.5.1.3 - Transformada Discreta do Co-seno (DCT) 12
2.5.1.4 - Quantização 15
2.5.1.5 - Codificação VLC 15
2.5.2 - Codificação Interquadros 17
2.5.2.1 - Estimação de Movimento (ME) 17
2.5.2.2 - Compensação de Movimento (MC) 20
2.5.3 - Decodificação 21
2.6 - Algoritmos Utilizados na Estimação de Movimento 21
2.6.1 - Modelo de Estimação 21
2.6.2 - Critério de Casamento 22
2.6.3 - Algoritmos de Estimação 23
2.6.3.1 - Busca Exaustiva – ESA (*Exhaustive Search Algorithm*) 24
2.6.3.2 - Busca Logarítmica - TDLS (*Two Dimensional Logarithm Search*) 25
2.6.3.3 - Busca Hierárquica – HME (*Hierarchical Motion Estimation*) 26
2.6.3.4 - SEA (*Successive Elimination Algorithm*) 29

3 ALGORITMO SEA MODIFICADO 33

3.1 - Introdução 33
3.2 - Algoritmo MSEA 33
3.2.1 - Critérios de Descarte 34
3.2.2 - Pré-Ordenação dos Macroblocos 38
3.3 - Interpretação Espacial 42

4 RESULTADOS 45

4.1 - Introdução 45
4.2 - Parâmetros das Simulações 45
4.3 - Estimação com Quadros Estáticos 49
4.3.1 - Estimação com Região de Pesquisa Incluindo Todo o Quadro de Referência 49
4.3.2 - Estimação com Região de Pesquisa Limitada 55

4.3.3 - Incorporação do Algoritmo MSEA a Outros Algoritmos de Estimação 62
4.4 - Estimação Empregando Seqüências de Vídeo 66
4.4.1 - Análise do Codificador Implementado 67
4.4.2 - Resultados com Região de Pesquisa Compreendendo Todo o Quadro de Referência 69
4.4.3 - Resultados com Região de Pesquisa Limitada 70

5 CONCLUSÕES 79

5.1 - Sumário 79
5.2 - Contribuições 80
5.3 - Trabalhos Futuros 81

A INFORMAÇÕES AUXILIARES 83

A.1 - Matrizes de Quantização Utilizadas pelo Codificador MPEG-2 [34] 83
A.2 - Filtros Utilizados na Conversão RGB ⇔ YCrCb 84
A.3 - Tabelas de Codificação dos Coeficientes DC [34] 85
A.4 - Tabelas de Codificação dos Coeficientes AC [34] 86
A.5 - Códigos para os Vetores de Movimento (regiões de pesquisa de [-16, +16] pixels) [34]. 88
A.6 - Quadro 100 de Cada Seqüência de Vídeo Utilizado nos Diversos Algoritmos de Estimação. 89

REFERÊNCIAS 91

LISTA DE FIGURAS

Figura 2.1 – Diagrama em blocos de um sistema de comunicação de sinais de vídeo. 5

Figura 2.2 – Diagrama em blocos do codificador e decodificador híbrido DPCM/DCT.(a) Codificador DPCM/DCT destacando os blocos de codificação intraquadros e interquadros; (b) Decodificador DPCM/DCT. 10

Figura 2.3 – Efeito da mudança de base no número de bits necessários para a representação do sinal: (a) São necessários N_xN_y níveis para a representação do sinal; (b) São necessários $N_uN_v < N_xN_y$ níveis para representar o mesmo sinal. 13

Figura 2.4 – Processo de leitura dos coeficientes no bloco: (a) leitura na forma "zig-zag" do bloco; (b) posição do elemento do bloco lido segundo a ordem "zig-zag". 16

Figura 2.5 – Obtenção do vetor de movimento. (a) Quadro atual particionado em macroblocos; (b) Quadro anterior com o vetor de movimento encontrado dentro da região de pesquisa. 18

Figura 2.6 – Ordem de codificação dos quadros de uma seqüência de vídeo segundo um GOP dado por IBBPBBI. 19

Figura 2.7 – Macroblocos $\boldsymbol{C}$ e $\boldsymbol{R}$ representados como vetores no espaço multidimensional (256 dimensões), sendo $\boldsymbol{D}(\boldsymbol{C},\boldsymbol{R})$ a distância entre os blocos. 23

Figura 2.8 – Exemplo do algoritmo de busca logarítmica com três passos para um bloco de 8 × 8 pixels e área de pesquisa p = [-8,8] pixels, com o valor do MAE mínimo obtido para a posição 4. 25

Figura 2.9 – Algoritmo típico para a busca hierárquica. 26

Figura 2.10 – Procedimento para o cálculo das normas dos macroblocos. 30

Figura 3.1 – Exemplo do cálculo dos agrupamentos definidos pela Equação (3.8). (a) Bloco de 8 × 8 pixels; (b) Soma dos agrupamentos de 2 × 2 pixels; (c) Soma dos agrupamentos de 4 × 4 pixels; (d) Soma dos agrupamentos de 8 ×8 pixels. 38

Figura 3.2 – Determinação da região dos macroblocos pesquisáveis pela pré-ordenação dos macroblocos. 41

Figura 3.3 – Exemplo da aplicação do primeiro critério de descarte via pré-ordenação dos macroblocos. 41

Figura 3.4 – Interpretação espacial do algoritmo MSEA. 42

Figura 4.1 – Tempo de processamento para os algoritmos ESA, SEA, MSEA, HME e TDLS. 56

Figura 4.2 – Erro de predição para os algoritmos ESA, SEA, MSEA, HME e TDLS. 58

Figura 4.3 – Diferença entre o quadro de referência e o quadro atual para a seqüência "Flower Garden". (a) Sem compensação de movimento; (b) Com TDLS; (c) Com HME; (d) Com MSEA. 59

Figura 4.4 – Diferença entre o quadro de referência e o quadro atual para a seqüência "Mobile Calendar". (a) Sem compensação de movimento; (b) Com TDLS; (c) Com HME; (d) Com MSEA. 59

Figura 4.5 – Número de operações efetuadas pelos algoritmos ESA, SEA, MSEA, HME e TDLS. 61

Figura 4.6 – Comparação dos erros de predição do algoritmo HME1/MSEA com os outros algoritmos de estimação. 64

Figura 4.7 – Taxa de bits das seqüências de vídeo "Flower", "Kiel", "Mobile" e "Tennis" processadas pelo codificador MPEG-2 implementado. 68

Figura 4.8 – PSNR das seqüências "Flower", "Mobile", "Kiel"e "Tennis" codificadas. 68

Figura 4.9 – Curva de PSRN × taxa para a seqüência "Flower" utilizando os vários algoritmos de estimação com região de pesquisa com $p = 16$ pixels. 71

Figura 4.10 – Curva de PSNR × taxa para a seqüência "Kiel" utilizando os vários algoritmos de estimação com região de pesquisa com $p = 16$ pixels. 71

Figura 4.11 – Curva de taxa × distorção para a seqüência "Mobile" utilizando os vários algoritmos de estimação com região de pesquisa com $p = 16$ pixels. 72

Figura 4.12 – Curva de PSNR × taxa para a seqüência "Tennis" utilizando os vários algoritmos de estimação com região de pesquisa com $p = 16$ pixels. 72

Figura 4.13 – Curvas de PSNR × taxa para os algoritmos MSEA, HME e HME1/MSEA para a seqüência "Flower" com região de pesquisa com $p = 16$ pixels. 74

Figura 4.14 – Curvas de PSNR × taxa para os algoritmos MSEA, HME e HME1/MSEA para a seqüência "Kiel" com região de pesquisa com $p = 16$ pixels. 75

Figura 4.15 – Curvas de PSNR × taxa para os algoritmos MSEA, HME e HME1/MSEA para a seqüência "Mobile" com região de pesquisa com $p = 16$ pixels. 75

Figura 4.16 – Curvas de PSNR × taxa para os algoritmos MSEA, HME e HME1/MSEA para a seqüência "Tennis" com região de pesquisa com $p = 16$ pixels. 76

Figura 4.17 – PSNR da seqüência FLOWER processada pelos diversos algoritmos de estimação, para uma região de pesquisa limitada em [-16,+16] pixels e com taxa de compressão R=0,3bit/pixel. 77

Figura 4.18 – PSNR da seqüência KIEL processada pelos diversos algoritmos de estimação, para uma região de pesquisa limitada em [-16,+16] pixels e com taxa de compressão R=0,4bit/pixel. 77

Figura 4.19 – PSNR da seqüência MOBILE processada pelos diversos algoritmos de estimação, para uma região de pesquisa limitada em [-16,+16] pixel e com taxa de compressão R=0,4bits/pixel. 78

Figura 4.20 – PSNR da seqüência TENNIS processada pelos diversos algoritmos de estimação, para uma região de pesquisa limitada em [-16,+16] pixels e com taxa de compressão R=0,4bit/pixel. 78

LISTA DE TABELAS

Tabela 4.1 – Resultados da estimação de movimento com o algoritmo ESA. 50

Tabela 4.2 – Resultados da estimação de movimento com o algoritmo SEA. 50

Tabela 4.3 – Resultados da estimação de movimento com o algoritmo MSEA (sem ordenação). 50

Tabela 4.4 – Resultados da estimação de movimento com o algoritmo MSEA (com ordenação). 50

Tabela 4.5 – Ganhos no tempo de processamento dos algoritmos SEA e MSEA com relação ao ESA. 52

Tabela 4.6 – Redução no número de operações dos algoritmos SEA e MSEA com relação ao ESA. 52

Tabela 4.7 – Número de macroblocos testados em cada critério de descarte pelo algoritmo MSEA sem ordenação. 52

Tabela 4.8 – Número de macroblocos testados em cada critério de descarte para o algoritmo MSEA com ordenação. 53

Tabela 4.9– Percentual do número de macroblocos testados em cada critério de descarte para o algoritmo MSEA sem ordenação. 54

Tabela 4.10 – Percentual do número de macroblocos testados em cada critério de descarte para o algoritmo MSEA com ordenação. 54

Tabela 4.11 – Overhead de processamento para os algoritmos MSEA sem e com ordenação. 54

Tabela 4.12 – Tempos de processamento para os algoritmos ESA, SEA, MSEA, HME e TDLS com região de pesquisa variável. 55

Tabela 4.13 – Coeficientes da curva polinomial ajustada do tempo de processamento × largura da região de pesquisa. 57

Tabela 4.14 – Qualidade do quadro processado (RMSE) para os algoritmos ESA, SEA, MSEA, HMSE e TDLS com região de pesquisa variável. 57

Tabela 4.15 – Número de operações efetuadas pelos algoritmos ESA, SEA, MSEA, HME e TDLS para uma região de pesquisa variável. 60

Tabela 4.16 – Coeficientes da curva polinomial ajustada para o número de operações em função da largura da região de pesquisa 61

Tabela 4.17 – Comparação dos algoritmos HME e HME/ MSEA. 62

Tabela 4.18 – Resultados das simulações do HME com estimação no primeiro nível utilizando o algoritmo MSEA (HME1/MSEA). 63

Tabela 4.19 – Memória extra necessária a execução dos algoritmos de estimação. 66

Tabela 4.20 – Tempo gasto pelo codificador no processamento das seqüências de vídeo para os algoritmos de estimação ESA, SEA e MSEA. 69

Tabela 4.21 – Ganho no tempo de processamento das seqüências de vídeo normalizado pelo tempo gasto pelo algoritmo ESA. 70

Tabela 4.22 – Tempo de processamento médio das seqüências de vídeo pelo codificador MPEG-2 utilizando vários algoritmos de estimação. 73

Tabela 4.23 – Comparação entre o tempo de processamento médio das seqüências de vídeo utilizando os algoritmo MSEA, HME e HME1/MSEA. 76

GLOSSÁRIO

DCT ..... *Discrete Cosine Transform.*

DPCM ..... *Differential Pulse Code Modulation.*

GOP ..... *Group Of Picture.*

HME ..... *Hierarchical Motion Estimation.*

MPEG ..... *Motion Picture Experts Group.*

MSEA ..... *Modified Successive Elimination Algorithm.*

NOP ..... Número de **OP**erações.

PSNR ..... *Peak Signal to Reconstruction Noise.*

RMSE ..... *Root Mean Square Error.*

SEA ..... *Successive Elimination Algorithm.*

TB ..... Taxa de Bits.

TDLS ..... *Two Dimensional Logarithm Search.*

TP ..... Tempo de **P**rocessamento.

VLC ..... *Variable Length Code.*

Capítulo 1

INTRODUÇÃO

Em um mundo cada vez mais globalizado, o sucesso das empresas, atrelado à satisfação dos clientes, está vinculado à obtenção de informação, nos seus mais diferentes meios, e de sua manipulação, transmissão e armazenamento eficientes. Neste sentido, a tecnologia tem promovido mudanças fundamentais. Destacam-se:

- Crescente digitalização dos sistemas, unificando as várias mídias em unidades de informação (bits). Em bits, as várias mídias são facilmente manipuláveis e armazenadas a um custo relativamente baixo.
- Facilidade de implementação de projetos, devido à evolução da microeletrônica que possibilita o desenvolvimento de sistemas complexos, rápidos, programáveis e de baixos custos para a manipulação e o armazenamento da informação.
- Proliferação das redes de dados e de computadores, criando canais de comunicação para o tráfego das informações entre clientes e fornecedores nas suas mais diferentes formas.

Nas últimas décadas, todas estas inovações propiciaram um avanço nas formas visuais de comunicação. As mídias visuais (imagens, vídeo, etc.) sempre foram um meio eficiente de comunicação, já que conseguem condensar uma grande quantidade de informação em uma forma que é facilmente interpretada pelo ser humano. A grande quantidade de informação presente nas mídias digitais constitui um dos grandes problemas da sua utilização (transmissão e armazenamento). Tomemos, por exemplo, um sinal de vídeo de televisão. Supondo cada quadro contendo cerca de 300.000 elementos de imagem (pixels) formado de três componentes RGB, com 256 níveis de brilho igualmente prováveis, sendo transmitidos a uma taxa de 30 quadros por segundo, tem-se um conteúdo de informação de, aproximadamente:

- Informação por elemento do quadro = $3 \times \log_2(256) = 24$ bits/elemento;
- Informação por quadro = $300.000 \times 24 = 7{,}2 \times 10^6$ bits/quadro;
- Informação por segundo = $30 \times 7{,}2 \times 10^6 = 216$ Mbits/segundo.

Valores de taxa de transmissão altos, como o valor acima (216Mbit*s*/segundo), são comumente encontrados em transmissão de sinais de vídeo. De modo a compatibilizá-los às capacidades dos sistemas de comunicação e armazenamento de informação, necessita-se de circuitos chamados codificadores de vídeo. Os circuitos codificadores atuam no sinal de vídeo reduzindo a sua entropia, permitindo, assim, uma transmissão e um armazenamento eficientes. Com uma taxa de transmissão de 216Mbit*s*/segundo, um codificador ajustando a taxa aos valores usuais de transmissão do sinal de televisão, da ordem de 4 a 6Mbit*s*/segundo, exige fatores de compressão da ordem de 54:1 a 36:1, respectivamente.

Os circuitos codificadores de vídeo atuam extraindo as redundâncias presentes no sinal de vídeo. A extração destas redundâncias ocorre, preferencialmente, em dois níveis: o espacial, onde se explora a correlação entre os pixel*s* de cada quadro da seqüência de vídeo, e o temporal, onde se explora a correlação entre os pixel*s* nos diversos quadros da seqüência. A extração da redundância espacial é comumente conseguida utilizando a técnica da transformada, tendo a DCT (***Discrete Cosine Transform***) como núcleo da transformação, seguido de uma quantização. Utilizando a quantização, ocorre perda de informação no processo de extração da redundância espacial. Assim, a taxa de compressão pode ser superior às taxas conseguidas com processos sem perda de informação. Com o conjunto DCT e quantização conseguem-se taxas de compressão da ordem de 12:1 [1]. O restante da compressão deve ser obtido às custas da extração da redundância temporal que, pelo primeiro exemplo acima, seria de 4,5:1. Fatores de compressão sem perdas da ordem de 4:1 ou superiores na extração da redundância temporal não são triviais, necessitando de circuitos codificadores com um certo grau de elaboração.

Neste trabalho, aborda-se o problema da compressão dos sinais de vídeo, e em especial, o processo de extração da redundância temporal, tomando-se como base o padrão MPEG-2. Neste padrão, a extração da redundância temporal é efetuada através do processo de estimação e compensação de movimento. O trabalho é focado em melhorar o processo de estimação de movimento, através de um novo algoritmo denominado SEA modificado (MSEA), já que ele representa uma evolução do algoritmo SEA (***Successive Elimination Algorithm***) proposto originalmente por Li e Salari [2]. O algoritmo SEA modificado apresenta como diferenciais a extrema velocidade de processamento, reduzindo o custo

computacional do processo de estimação, ao mesmo tempo em que mantém a solução ótima do vetor de movimento para uma dada região de pesquisa.

O algoritmo SEA tem sido referenciado por uma grande quantidade de artigos (quarenta e seis segundo a *Web of Science*). Muitos destes artigos tratam de estender o SEA de modo a reduzir o espaço de busca ou tornar mais rápido o cálculo do MAE. O MSEA apresenta as seguintes inovações:

- Busca em múltiplos níveis (espaços de dimensionalidade crescente);
- Pré-ordenação dos macroblocos do quadro de referência para descarte de macroblocos e seleção do ponto inicial da pesquisa.

Uma pesquisa de literatura revelou que vários artigos têm contribuições semelhantes às propostas do MSEA. Em particular, os artigos [3-12] se utilizam da busca exaustiva em níveis piramidais de forma semelhante à utilizada pelo MSEA.

Não se identificam artigos que tenham proposto a pré-seleção de macroblocos de modo como é utilizada pelo MSEA.

As contribuições deste trabalho estão parcialmente descritas nos artigos [13] e [14].

O trabalho está organizado da seguinte forma:

Capítulo 2: Descreve-se o sistema de comunicação para sinais de vídeo, dando-se destaque ao codificador de vídeo padrão MPEG-2, e, em particular, ao procedimento de extração das redundâncias espaciais e temporais através das técnicas de estimação e compensação de movimento. Vários algoritmos de estimação comumente utilizados são apresentados para comparação posterior com o algoritmo MSEA.

- **Capítulo 3**: O algoritmo MSEA é apresentado, onde são expostas suas características e limitações e os pormenores de sua implementação.
- **Capítulo 4**: Apresentam-se as simulações utilizando os diversos algoritmos de estimação de movimentos. Os resultados, expressos em termos da velocidade de processamento, número de operações efetuadas, qualidade dos quadros processados, curvas de taxa × distorção, etc., fornecem as bases para a comparação do algoritmo MSEA com os outros algoritmos de estimação, revelando as qualidades e limitações do MSEA.

- **Capítulo 5**: Aqui são comentados os resultados das simulações, as contribuições e as conclusões deste trabalho. São sugeridas modificações para se melhorar o desempenho do algoritmo.

Capítulo 2

PRINCÍPIOS DA CODIFICAÇÃO DE VÍDEO

2.1 - INTRODUÇÃO

Neste capítulo, discute-se o modelo de comunicação utilizado para sinais de vídeo. Destaca-se a necessidade do codificador de sinais de vídeo como peça fundamental no processo de comunicação. Sendo o MPEG-2[1] [15-18] um padrão largamente aceito para os codificadores de vídeo, analisam-se, resumidamente, a sua funcionalidade e os procedimentos utilizados para a extração das redundâncias espaciais e temporais da seqüência de quadros. Fixa-se a atenção sobre o processo de extração das redundâncias temporais, através de técnicas de estimação e compensação de movimento, onde são apresentados vários algoritmos que executam esta função.

2.2 - SISTEMA DE COMUNICAÇÃO DE SINAIS DE VÍDEO

Um diagrama em blocos do processo de comunicação digital de sinais de vídeo [19] pode ser visualizado na Figura 2.1

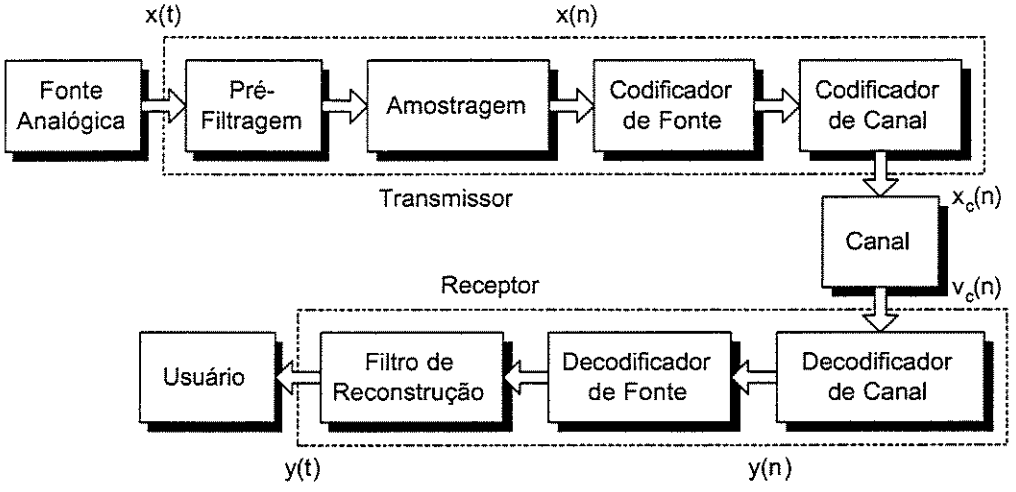


Figura 2.1 - Diagrama em blocos de um sistema de comunicação de sinais de vídeo.

[1] MPEG é a sigla para *Motion Picture Experts Group*.

Circuitos transmissores e receptores estabelecem uma comunicação eficiente através de um canal de transmissão entre a fonte emissora da informação e o usuário correspondente. O circuito transmissor ajusta as características da fonte emissora de informação às características do canal de transmissão, tornando o processo de transmissão eficiente e protegendo a qualidade do sinal transmitido. O circuito receptor extrai a informação de vídeo do sinal recebido, de modo a reconstruir convenientemente o sinal para o usuário da informação. Num sistema de comunicação digital, o sinal de uma fonte analógica (vídeo analógico) deve ser convertido e preparado para a transmissão por um canal digital. O processo de conversão se dá pelos circuitos de amostragem e quantização que processam o sinal analógico (contínuo no tempo e em amplitude) e o transforma em um sinal digital (discreto no tempo e em amplitude). A discretização no tempo, executada pela amostragem do sinal, baseia-se no teorema de Nyquist, que garante a reconstrução do sinal original a partir das suas amostras, dada à condição da freqüência da amostragem ser, no mínimo, superior a duas vezes a largura de banda do sinal analógico a ser amostrado. De modo a limitar a largura de banda do sinal analógico e evitar o efeito de *aliasing* (superposição das componentes espectrais no espectro do sinal amostrado) efetua-se uma pré-filtragem do sinal. O filtro utilizado deve apresentar características de amplitude e fase de modo a permitir a reconstrução do sinal no circuito do receptor. A discretização em amplitude, efetuada por um quantizador, atribui um único valor representativo de uma faixa de valores do sinal original a ser quantizado. Em decorrência desta atribuição, a quantização é um processo irreversível, ou seja, não se consegue obter o sinal original a partir do sinal amostrado, levando a uma perda de informação.

Após a conversão do sinal analógico em um sinal digital, o mesmo é codificado para que possa ser transmitido eficientemente pelo canal. Esta função é desempenhada pelos codificadores de fonte e de canal. O canal limita a eficiência da comunicação, já que impõem restrições à largura de banda do sinal transmitido e introduz ruído no processo de transmissão. O codificador de fonte tem a função de reduzir a quantidade de informação do sinal (e, conseqüentemente, a sua largura de banda) de modo a adequar o sinal ao canal. O codificador de fonte é classificado como sendo do tipo "muitos para um", pois deve ser capaz de codificar uma ampla variedade de sinais e convertê-los em um único padrão para transmissão, função que o torna, geralmente, de difícil implementação. O codificador de

canal é responsável por proteger o sinal quanto ao ruído de canal. Ele exerce esta função, na maioria das vezes, acrescentando redundância ao sinal através de códigos corretores de erro. Por ser um mapeamento "um para um", o codificador de canal é, em geral, de fácil implementação. A conjunção entre o codificador de fonte (o quanto de redundância extrair do sinal) e o codificador de canal (o quanto de redundância acrescentar ao sinal) constitui um dos grandes problemas de projeto destes codificadores.

Depois de codificado, o sinal é transmitido e coletado no circuito do receptor. Nele, o sinal é decodificado através do decodificador de canal, que estima o sinal original a partir do sinal corrompido pelo ruído de canal. O decodificador de canal constitui um mapeamento "um para muitos", pois das infinitas possibilidades do sinal de entrada (causadas pelo ruído), escolhe-se uma única saída, que constitui a estimativa para o sinal original. O sinal resultante sofre uma decodificação de fonte (mapeamento "um para um"), onde o sinal amostrado e quantizado é reconstruído, seguido de uma filtragem (filtro de reconstrução) que compensa a pré-filtragem do circuito transmissor, ficando assim, o sinal disponível para o usuário final.

2.3 - CODIFICADOR E DECODIFICADOR DE FONTE

Como comentado, o codificador de fonte é responsável pela extração das redundâncias do sinal de vídeo. Através da quantização, ele reduz também a sua entropia (e conseqüentemente a sua largura de banda) de modo a adequar o sinal ao canal de transmissão. Um sinal de vídeo pode se modelado como uma seqüência sucessiva de quadros de imagens exibidos a uma certa taxa. Deste modo, a extração das redundâncias do sinal opera em dois níveis:

- **Intraquadros:** explorando a correlação espacial entre pixels pertencentes a um mesmo quadro.
- **Interquadros:** explorando a correlação temporal entre pixels pertencentes a quadros distintos.

As técnicas de codificação intraquadros se dividem em dois grandes grupos:

A. Técnicas de Compressão sem Perdas

As técnicas de extração das redundâncias espaciais pertencentes a este grupo não ocasionam perda de informação (são processos reversíveis). São geralmente utilizadas

quando a perda de informação no processo de codificação não é admissível (como, por exemplo, em imagens médicas), e apresentam uma baixa eficiência de compressão, quando comparadas com as técnicas de compressão com perdas.

B. Técnicas de Compressão com Perdas

Nas técnicas de compressão com perdas, é admitida perda de informação no sinal, tornando o processo de compressão irreversível. Devido à perda de informação, têm-se altas taxas de compressão. Uma das técnicas mais utilizadas é a codificação por transformada seguida de quantização, por apresentar bom desempenho e simplicidade de implementação [1]. Naturalmente, é importante que a perda de informação seja tolerada pela aplicação à qual o sinal se destina.

2.4 - PADRÃO MPEG-2

Técnicas modernas de compressão possibilitam uma maneira eficiente e robusta de transmitir e armazenar grandes quantidades de dados, quantidades estas necessárias à representação dos sinais digitais de imagens e vídeo. Novas aplicações audiovisuais, aplicações multimídias, transmissões em banda larga, etc. tornaram-se possíveis com o advento da tecnologia de vídeo digital. Com o desenvolvimento da microeletrônica e das tecnologias VLSI, obteve-se a integração tão desejada dos sistemas desencadeando a queda dos custos e o incremento da demanda. Comercialmente, a necessidade de um padrão para as comunicações audiovisuais tornou-se importante por dois aspectos básicos: economia de escala e compatibilidade dos sistemas. Em 1988, iniciaram-se os trabalhos do *Joint ISO/IEC Technical Committee (JCT1) on Information Technology* que demandava estudos sobre um padrão de codificação para imagens em movimento, áudio e suas combinações para armazenamento e transmissão digital a taxas inferiores a 1,5Mbps. O padrão desenvolvido denominou-se MPEG-1 [18]. O escopo do grupo foi estendido, mais tarde, de modo a englobar uma maior variedade de aplicações audiovisuais e assim operar a taxas mais altas, fora do escopo do padrão MPEG-1. Este novo padrão, denominado MPEG-2 [15-21], atendia a uma nova gama de aplicações, como televisão digital, serviços em rede operando com tecnologia ATM, aplicações digitais de transmissão e recepção via satélite, etc., ao mesmo tempo em que garantia, operando com taxas de 2 a 10Mbps, uma qualidade

do processamento não inferior aos antigos padrões em vigor (como o NTSC/PAL e CCIR 61) [18]. A importância do padrão MPEG-2 vem com a padronização da estrutura do decodificador e do formato da seqüência de bits utilizado. A padronização permitiu aos fabricantes otimizar o codificador com respeito à qualidade do sinal codificado frente à taxa de bits necessária a sua representação, tornou o codificador flexível quanto aos vários tipos de sinais codificáveis e permitiu sua evolução, com o desenvolvimento e a incorporação de novos algoritmos de codificação.

2.5 - FUNDAMENTOS DO ALGORITMO DE COMPRESSÃO MPEG

Genericamente, uma seqüência de vídeo apresenta uma quantidade significativa de redundâncias presente na forma estatística (correlação entre pixels) e subjetiva (informação relevante para o espectador da seqüência). O objetivo da compressão é diminuir a entropia do sinal explorando estas redundância objetivando o armazenamento e a transmissão da informação de modo eficiente. As técnicas utilizadas no MPEG são estatísticas por natureza, utilizando a correlação entre pixels tanto na dimensão espacial (intraquadros) quanto na dimensão temporal (interquadros). Na dimensão espacial, a codificação é fundamentada na aplicação da transformada DCT (*Discrete Cosine Transform*) seguida de uma quantização. Na dimensão temporal (codificação interquadros), utiliza-se um modelo de correlação baseado na movimentação translacional de objetos de cena de um quadro para outro. Quando a correlação entre os objetos de cenas (e seus pixels relacionados) é alta (como geralmente ocorre nas seqüências de vídeo), o uso da técnica DPCM (*Differential Pulse Code Modulation*) explorando a correlação temporal (estimação e compensação de movimento) produz bons resultados. Os resultados das codificações nas dimensões espaciais e temporais são processados por um codificador de entropia, reduzindo a taxa de bits do sinal através de um código VLC (*Variable Length Code*).

Na codificação MPEG, a combinação entre estimação e compensação de movimento e a codificação por transformada, através da DCT caracteriza o codificador como um circuito híbrido, genericamente denominado codificador DPCM/DCT [19]. O diagrama em blocos de um codificador MPEG pode ser visto na Figura 2.2.

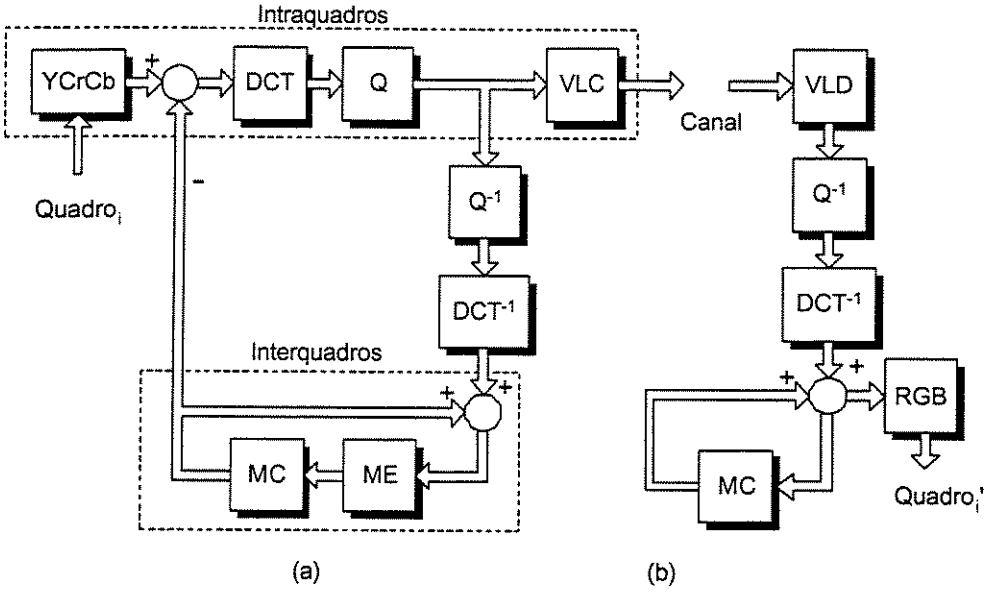


Figura 2.2 - Diagrama em blocos do codificador e decodificador híbrido DPCM/DCT. (a) Codificador DPCM/DCT destacando os blocos de codificação intraquadros e interquadros; (b) Decodificador DPCM/DCT.

No diagrama, separam-se os blocos referentes à codificação intraquadros e os blocos referentes à codificação interquadros. A seguir, segue uma explicação sucinta de cada bloco.

2.5.1 - CODIFICAÇÃO INTRAQUADROS

Os blocos que executam a codificação intraquadros extraem a redundância espacial presente nos quadros da seqüência de vídeo. A extração da redundância, que levará à diminuição da entropia do sinal, se processa em dois níveis. O primeiro nível, responsável pela redução da entropia do sinal, refere-se à extração das redundâncias com perda de informação, através da conversão dos planos de cores RGB para os planos YCrCb com posterior subamostragem e aplicação da DCT seguida de quantização. O segundo nível trata das redundâncias sem perda de informação, através da aplicação de um código de comprimento variável ao sinal (código VLC). As características de cada processo são descritas a seguir.

2.5.1.1 - CONVERSOR RGB PARA YCRCB

Tipicamente, os pixels de um quadro em cores são representados pelas componentes RGB (***Red***, ***Green*** e ***Blue***) [22] do sinal de vídeo. Codificar todas as componentes em separado exigiria uma alta capacidade de processamento devido à grande quantidade informação envolvida. Aproveita-se, então, uma característica física do olho humano de apresentar uma maior sensibilidade às mudanças de luminosidade em relação às mudanças de cromaticidade dos quadros, para se efetuar a compressão. As componentes RGB sofrem uma transformação linear de planos de cores RGB para planos YCrCb (sendo Y a componente de luminância e Cr e Cb as componentes de crominância), seguidas de uma amostragem das componentes de crominância, nas razões 4:2:0 ou 4:2:2 (a cada quatro componentes de luminância tomam-se uma ou duas componentes de crominância, respectivamente). Reduz-se, assim, a quantidade de pixels por um fator de 2 (4:2:0) ou 1,5 (4:2:2), obtendo-se compressão.

As matrizes de transformação entre planos são mostradas abaixo:

- **Transformação RGB → YCrCb**

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = \begin{bmatrix} 0.2990 & 0.5870 & 0.1440 \\ 0.5000 & -0.4187 & -0.0183 \\ -0.1678 & 0.3313 & 0.5000 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.1}$$

- **Transformação YCrCb → RGB**

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.0000 & 1.4020 & 0.0000 \\ 1.0000 & -0.7141 & -0.3141 \\ 1.0000 & 0.0000 & 1.7726 \end{bmatrix} \times \begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} \tag{2.2}$$

Após a amostragem das componentes Cr e Cb, filtra-se o sinal, resgatando somente as bandas básicas. Em nosso trabalho, são utilizados filtros obtidos a partir de critérios subjetivos adotados nos laboratórios de processamento de imagens do CPqD (Centro de Pesquisa e Desenvolvimento, Campinas - SP), apresentados no Apêndice A.

No processo de reconstrução, as componentes Cr e Cb são interpoladas, reconstruindo-se uma estimativa do quadro YCrCb original, de onde, através da transformação reversa, recuperam-se as componentes RGB.

2.5.1.2 - SEGMENTAÇÃO EM BLOCOS

Para um fator de amostragem 4:2:0, agrupam-se os pixels de cada componente Y, Cr e Cb em grupos de 8 x 8 pixels, denominados blocos, e em grupos formados por quatro blocos de pixels Y, um bloco de pixels Cr e um bloco de pixels Cb, denominado macrobloco. Os blocos constituem a unidade mínima de processamento no padrão MPEG-2, utilizados no processo de codificação intraquadros. Os macroblocos, unidades mínimas utilizadas no processo de codificação interquadros, terão sua função descrita mais adiante, quando serão abordados os processos de estimação e compensação de movimento.

O valor de 8 × 8 pixels para um bloco foi obtido experimentalmente de modo a otimizar a relação compressão × qualidade da imagem reconstruída. Testes demonstraram que um número maior de elementos nos blocos não é compensado pela melhoria de desempenho observada. Por outro lado, blocos menores não reduzem suficientemente a correlação entre os pixels, e aumenta o número de bits necessários para descrevê-los (menor taxa de compressão). O valor 8 × 8 é uma solução de compromisso entre o desempenho e complexidade [1].

2.5.1.3 - TRANSFORMADA DISCRETA DO CO-SENO (DCT)

Aplicar transformadas para descorrelacionar os pixels de um dado quadro corresponde a decompor estes pixels em uma nova base, onde são necessários menos bits para representá-los (Figura 2.3).

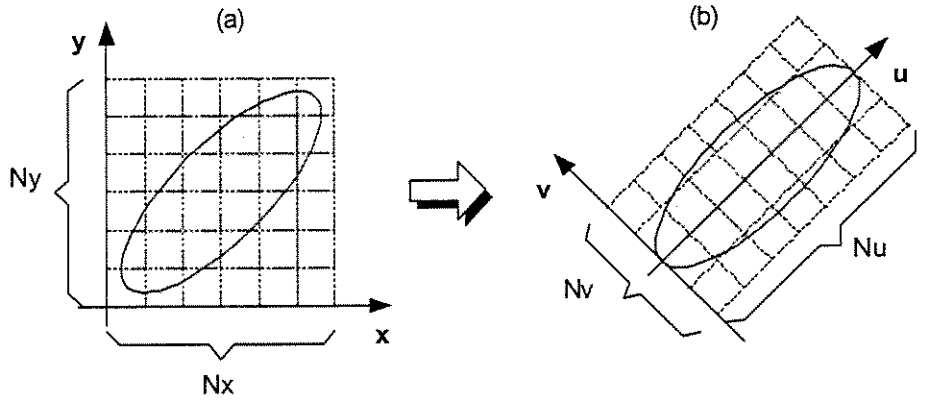


Figura 2.3 - Efeito da mudança de base no número de bits necessários para a representação do sinal: (a) São necessários N_xN_y níveis para a representação do sinal; (b) São necessários $N_uN_v < N_xN_y$ níveis para representar o mesmo sinal.

A operação de mudança de base é uma operação reversível e, portanto, não há perda de informação. O núcleo da transformada define as características dos novos pixels gerados pela transformação. O MPEG contempla a utilização da técnica das transformadas na codificação intraquadros, utilizando como núcleo a DCT. A mudança de base implementada pela DCT equivale a uma rotação no espaço do sinal. A definição da DCT (transformações direta e inversa) para uma dimensão de $N \times N$ pixels ($N = 8$ para o caso do MPEG) é dada abaixo:

- **Transformada Direta (DCT)**

$$F(u,v) = \frac{2C(u)C(v)}{N} \sum_{i=1}^{N-1} \sum_{j=0}^{N-1} f(i,j) \cos\left[\frac{(2i+1)u\pi}{2N}\right] \cos\left[\frac{(2j+1)v\pi}{2N}\right], \tag{2.3}$$

- **Transformada Inversa (IDCT)**

$$f(i,j) = \frac{2}{N} \sum_{u=1}^{N-1} \sum_{v=0}^{N-1} C(u)C(v)F(u,v) \cos\left[\frac{(2i+1)u\pi}{2N}\right] \cos\left[\frac{(2j+1)v\pi}{2N}\right], \tag{2.4}$$

onde

$$C(w) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{para w} = 0, \\ 0 & \text{para w} \neq 0. \end{cases} \tag{2.5}$$

A adoção da DCT se deve a uma série de características descritas a seguir:

- ***Alta capacidade em descorrelacionar os*** pixels ***dos blocos***

Para quadros que apresentam alta correlação entre os seus pixels, a DCT descorrelaciona os pixels de uma forma bastante eficiente, que se aproxima da transformação ótima, a transformada de Karhunen-Loève, calculada em função das estatísticas dos quadros. De fato, para quadros que seguem o modelo de Markov de primeira ordem, apresentando pixels adjacentes com correlação próxima do valor unitário, o núcleo da transformada de Karhunen-Loève coincide com o núcleo da DCT. Como vantagem, o núcleo da DCT é fixo (independente do quadro), o que a torna atrativa em termos computacionais.

- ***Geração de coeficientes propícios à quantização***

Em decorrência da transformação, obtém-se uma concentração da energia do bloco em apenas alguns poucos coeficientes. Os coeficientes restantes, com valores pequenos em amplitude, podem ser descartados pelo processo de quantização sem afetar de maneira significativa a qualidade do bloco. A concentração de energia tende a ocorrer para os coeficientes de menor freqüência do sinal (o primeiro coeficiente, que geralmente é o maior de todos, corresponde à média dos pixels do bloco). Sendo o olho humano mais sensível às baixas freqüências, a DCT tende a preservar a qualidade visual dos blocos.

- ***Rapidez na computação***

A DCT é uma transformada que apresenta uma série de propriedades que aceleram o processo de computação: apresenta núcleo separável, de modo que a computação da transformada em duas dimensões pode ser desmembrada em duas computações mais simples, uma para cada dimensão; gera coeficientes reais, não necessitando de aritmética complexa ou de espaço de memória extra para armazenamento de coeficientes complexos e dispõe de algoritmos rápidos de computação (como o dado no Apêndice B) que reduzem a complexidade original do algoritmo de $O(N^2)$ para $O(Nlog(N))$, acelerando o processo de obtenção dos coeficientes transformados.

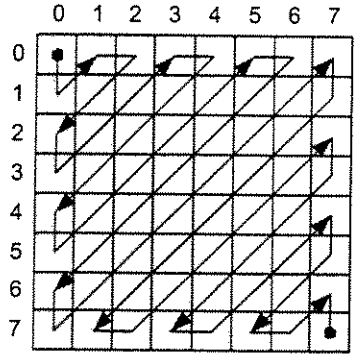
2.5.1.4 - QUANTIZAÇÃO

A função do quantizador é reduzir a precisão com que os coeficientes da DCT são representados, de modo a permitir uma redução na taxa de bits correspondente. Os coeficientes do quantizador são escolhidos com base nas características de sensibilidade do olho humano, de modo que as degradações visuais sofridas sejam minimizadas. Valores pequenos de coeficientes na matriz de quantização são adotados para a quantização dos coeficientes mais importantes dos blocos (coeficientes correspondentes às baixas freqüências) enquanto valores altos são adotados para os coeficientes de altas freqüências nos blocos. Este critério tende a fazer com que muitos coeficientes de alta freqüência sejam zerados. As matrizes de quantização típicas utilizadas pelo MPEG são mostradas no Apêndice A deste trabalho. Estas matrizes diferem para quadros tipo Intra e quadros tipo Inter, com todos os coeficientes de quantização constantes para estes últimos, já que o sistema visual humano tende a responder igualmente aos erros nos coeficientes DC da diferença entre os coeficientes.

2.5.1.5 - CODIFICAÇÃO VLC

Os dados quantizados são transformados em bits, sendo posteriormente enviados pelo canal. A codificação VLC associa códigos de comprimento variável aos coeficientes quantizados de modo que coeficientes mais freqüentes sejam associados a códigos mais curtos e coeficientes menos freqüentes sejam associados a códigos mais longos. Reduz-se, assim, o comprimento médio da palavra transmitida. A DCT tem a propriedade de compactar a energia do bloco nos seus coeficientes transformados. O coeficiente mais importante é aquele que ocupa a posição relativa (0,0) do bloco ($u = v = 0$ em (2.3)), retendo o valor médio dos elementos do bloco. Devido a esta propriedade, este coeficiente é comumente denominado coeficiente DC, sendo os restantes, em decorrência, denominados coeficientes AC. O coeficiente DC, sendo o mais importante do bloco, é codificado de forma distinta do restante dos coeficientes AC. Sofre, inicialmente, uma codificação diferencial efetuada através da diferença entre os valores dos coeficientes DC de um dado bloco com relação ao bloco anterior (justificado pela alta correlação existente entre os coeficientes DC de blocos adjacentes) seguido de uma codificação VLC, onde são utilizadas tabelas distintas das utilizadas para os coeficientes AC (vide Apêndice A).

Na codificação VLC ocorre um mapeamento do espaço bidimensional dos coeficientes dos blocos para um vetor unidimensional, onde os coeficientes, depois de convertidos em bits, são enviados pelo canal. Devido à DCT e à quantização, muitos coeficientes são nulos. Assim, na terminologia do MPEG, codificando o tamanho da corrida de elementos nulos (*run*) e o próximo elemento não nulo (*level*) em um par (*run,level*) obtém-se uma codificação eficiente. Cada par (*run,level*) é codificado segundo tabelas VLC, baseadas em códigos de Huffman. Para se maximizar a ocorrência de elementos nulos consecutivos, utiliza-se a ordem de leitura "zig-zag", definida na Figura 2.4.



(a)

0	2	3	9	10	20	21	35
1	4	8	11	19	22	34	36
5	7	12	18	23	33	37	48
6	13	17	24	32	38	47	49
14	16	25	31	39	46	50	57
15	26	30	40	45	51	56	58
27	29	41	44	52	55	59	62
28	42	43	53	54	60	61	63

(b)

Figura 2.4 - Processo de leitura dos coeficientes no bloco: (a) leitura na forma "zig-zag" do bloco; (b) posição do elemento do bloco lido segundo a ordem "zig-zag".

Existem várias combinações entre o número de zeros obtidos na leitura "zig-zag" e o valor do próximo elemento não nulo encontrado. Codificar todas estas combinações exigiria um código (*run,level*) tão extenso que o tornaria impraticável. O MPEG resolve este problema, atribuindo palavras código de comprimento variável às combinações (*run,level*) mais prováveis e um código de ESCAPE às combinações menos prováveis. Ocorrendo um código ESCAPE, a combinação (*run, level*) é codificada separadamente, recebendo cada uma das variáveis: *run* e *level*, códigos de comprimento fixo. (vide tabelas dos códigos no Apêndice A). Para seqüências de zeros que se prolongam até o último elemento do bloco é atribuído um código END OF BLOCK, evitando as codificações parciais.

2.5.2 - CODIFICAÇÃO INTERQUADROS

A função da codificação interquadros é extrair a redundância temporal presente entre os quadros consecutivos da seqüência de vídeo. Esta redundância provém, em grande parte, dos *backgrounds* (fundos de cena) que sofrem uma movimentação lenta de um quadro para outro na seqüência de vídeo exibida. Baseada nas técnicas de estimação e compensação de movimento (codificação DPCM do sinal temporal), a codificação interquadros, segundo a classificação apresentada na Figura 2.2, não introduz perda de informação no processo de codificação. São descritos, a seguir, seus blocos constitutivos.

2.5.2.1 - ESTIMAÇÃO DE MOVIMENTO (ME)

Para seqüências de vídeo onde os elementos de cena sofrem pequenos deslocamentos de um quadro para outro, o movimento destes elementos pode ser descrito por um número limitado de parâmetros. Um deles, denominado vetor de movimento [23-25], descreve o deslocamento sofrido por uma dada região de um quadro com relação à posição original que ela ocupava em outro quadro tomado como referência. No MPEG, as regiões são formadas repartindo os quadros em agrupamentos de 16 × 16 pixels denominados macroblocos. Assim, o processo de estimação de movimento consiste em se determinar o vetor de movimento formado pela diferença entre as coordenadas do macrobloco no quadro atual e as coordenadas do macrobloco mais semelhante a este (segundo um dado critério) no quadro de referência. No processo de estimação, a região de pesquisa do macrobloco no quadro de referência pode abranger todo o quadro ou apenas uma parte dele. A estimativa recai somente sobre a componente de luminância Y, pois se admite que as componentes Cr e Cb associadas seguem o mesmo deslocamento (Figura 2.5).

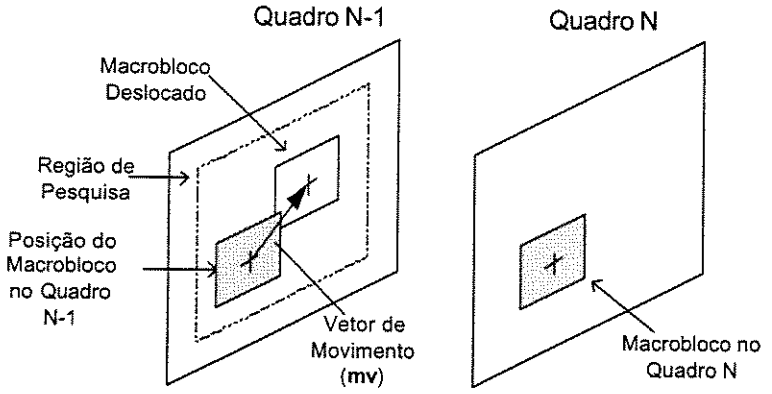


Figura 2.5 – Obtenção do vetor de movimento. (a) Quadro atual particionado em macroblocos; (b) Quadro anterior com o vetor de movimento encontrado dentro da região de pesquisa.

A estimação de movimento pode ser efetuada tendo-se como referência o quadro anterior, o posterior ou ambos. Utilizando-se somente um quadro como referência (predição), tem-se apenas um vetor de movimento por macrobloco. Utilizando-se o quadro anterior e o posterior como referências (interpolação), têm-se dois vetores de movimento por macrobloco (um para cada quadro). Como o decodificador tem apenas acesso aos quadros reconstruídos para recompor os quadros originais estimados, é necessário que o codificador e o decodificador estejam casados, ou seja, que se utilize o quadro reconstruído no decodificador como referência para o codificador no processo de estimação de movimento. Sendo a quantização a principal responsável pela degradação do sinal (supondo que não ocorram erros de canal), consegue-se gerar uma cópia do quadro reconstruído pelo decodificador aplicando-se quantização inversa (Q^{-1}) e a DCT inversa à saída o quantizador (Q) a todos os blocos do quadro de referência utilizados no processo de estimação (Figura 2.2)[2]. A utilização de quadros processados no processo de estimação altera a ordem natural de codificação dos quadros na seqüência de vídeo, já que a interpolação necessita que o quadro anterior e o posterior já tenham sido codificados. A ordem de codificação dos quadros segue uma seqüência definida pelo MPEG como GOP (***Group Of Picture***). Esta seqüência especifica o tipo de processamento que cada quadro deverá sofrer ao longo do processo de codificação. Os quadros podem ser classificados em três tipos básicos:

[2] Em rigor, a quantização não pode ser invertida, mas costuma-se denotar por quantização inversa (Q^{-1}) o processo de escalonamento que caracteriza a estimação do sinal a partir do sinal quantizado.

- **Tipo I**: O quadro não deverá sofrer estimação e compensação de movimento, somente codificação intraquadro.
- **Tipo P**: O quadro será codificado utilizando estimação e compensação de movimento empregando apenas um quadro como referência.
- **Tipo B**: O quadro será codificado utilizando estimação e compensação de movimento empregando blocos do quadro anterior, do posterior ou de ambos como referências.

Exemplificando com um GOP definido por IBBPBBI, tem-se a seguinte ordem de codificação mostrada na Figura 2.6:

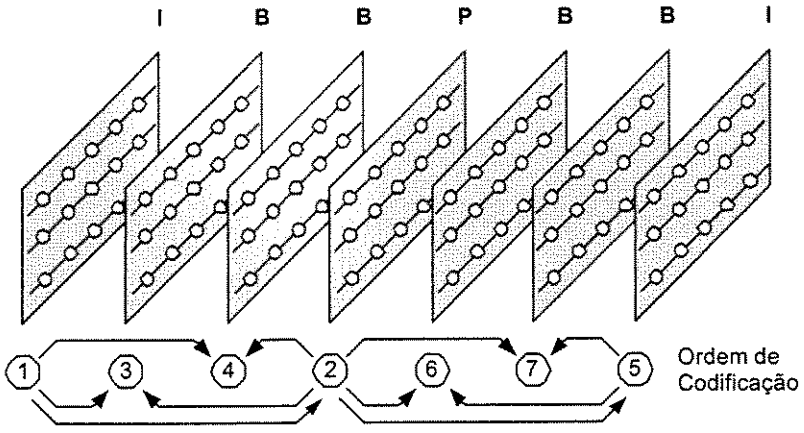


Figura 2.6 – Ordem de codificação dos quadros de uma seqüência de vídeo segundo um GOP dado por IBBPBBI.

Na regra de composição de um GOP, o processo de codificação deve, necessariamente se iniciar com um quadro do tipo I (quadro de referência). Quadros do tipo P utilizam quadros do tipo I ou outros quadros do tipo P como referência para a predição. Quadros do tipo B, que exigem duas referências, devem aparecer entre quadros do tipo I ou P. Quadros do tipo B não são referências para quadros do tipo P, já que estes (tipo B) não foram ainda processados. Também não se utilizam quadros do tipo B como referências para quadros do tipo B.

As maiores compressões são conseguidas utilizando os quadros do tipo B e P, nesta ordem, motivo pelo qual predominam os quadros do tipo B nos GOP's comumente utilizados. Os quadros tipo I, por sofrerem uma codificação independente dos outros

quadros, servem como quadros de sincronismo do GOP, caso ocorram erros de transmissão do sinal. Estes quadros evitam a propagação de erros.

Embora aos quadros da seqüência de vídeo seja atribuído um tipo de codificação I, P ou B, os macroblocos internos ao quadro podem sofrer uma codificação diferenciada, resultando em uma economia de bits na codificação. Podem ocorrer várias situações:

- Quando o erro de predição obtido pelo processo de estimação de movimento for semelhante (10% de diferença [1]) ao erro obtido utilizando o macrobloco no quadro de referência na mesma posição do macrobloco que se deseja estimar no quadro atual (vetor de movimento nulo), não se utiliza a referência produzida pela estimação de movimento. Economizam-se assim os bits relativos aos vetores de movimento.
- Para erros de predição grandes, pode ser mais vantajoso codificar o macrobloco como sendo do tipo I (intra) ao invés dos tipos P ou B (inter), economizando também na codificação dos vetores de movimento.
- Para erros de predição nulos (blocos P e B), não há necessidade de se codificar o macrobloco. Codificam-se apenas os vetores de movimento, já que estes carregam toda a informação para se restaurar o macrobloco na sua forma original.
- Para quadros do tipo B, efetua-se a estimação com relação ao quadro anterior, ao posterior e a ambos (utilizando a combinação linear entre as estimativas do bloco anterior e do posterior), na busca do menor erro de predição. O tipo do macrobloco codificado é função do processo de estimação que resulta no menor erro de predição.

Combinando as situações descritas acima, obtém-se maior eficiência de codificação. Nos quadros do tipo I, os macroblocos são sempre codificados como do tipo I (intra).

2.5.2.2 - COMPENSAÇÃO DE MOVIMENTO (MC)

A compensação de movimento se faz subtraindo de todos os macroblocos do quadro atual as suas referências obtidas pelo processo de estimação de movimento. O quadro resultante desta operação apresenta uma distribuição dos valores dos pixels mais centrada em torno do zero. Como o sinal a ser codificado possui uma variância menor, a variância do erro de quantização é proporcionalmente menor, para uma mesma taxa de bits. Para os macroblocos do tipo Inter, utiliza-se uma matriz de quantização uniforme. Os vetores de movimento obtidos são codificados utilizando um código de comprimento variável (VLC) e

são conjuntamente enviados com os dados do quadro e outros sinais de sincronismo e controle através do canal de transmissão.

2.5.3 - DECODIFICAÇÃO

Recebendo os dados codificados, o decodificador MPEG efetua os passos inversos aos da codificação para restaurar o quadro original (Figura 2.2). O quadro recebido é decodificado através de uma decodificação do código VLC (VLD) seguida da aplicação da transformada discreta inversa do co-seno (IDCT), da quantização inversa (i.e., escalonamento) (Q^{-1}) e da conversão YCrCb para RGB. O quadro tomado como referência no processo de estimação e compensação de movimento, depois de decodificado, é armazenado em uma memória interna do decodificador, sendo utilizado, juntamente com os vetores de movimento, para restaurar os quadros tipo P e B.

2.6 - ALGORITMOS UTILIZADOS NA ESTIMAÇÃO DE MOVIMENTO

Dos vários blocos que compõem o MPEG, a estimação de movimento é o bloco que apresenta o maior custo computacional no processo de codificação. O alto custo computacional provém do processo de minimização das distâncias entre os macroblocos de 256 dimensões (256 pixels) sobre todas as possíveis combinações de posições na região de pesquisa. Melhorando o processo de estimação, seja pela redução no tempo de processamento, seja pelo aumento da qualidade da estimação, aumenta-se o desempenho do codificador. O processo de estimação de movimento pode ser visto como composto por três partes: o modelo de estimação, o critério de casamento (função de custo) e os algoritmos de estimação utilizados no processo de minimização. A seguir, descreve-se cada uma das partes, detalhando-se suas características.

2.6.1 - MODELO DE ESTIMAÇÃO

A câmara de vídeo captura uma seqüência de quadros bidimensionais (2D) que são convenientemente processados pelo codificador. Os quadros 2D têm coordenadas $x = (i, j)^T \in \mathbf{\Phi}$, onde $\mathbf{\Phi}$ representa uma superfície amostrada. Deste modo, a movimentação de um elemento de imagem pode ser expressa como o deslocamento da sua projeção nesta

superfície nos vários instantes de tempo [23]. O deslocamento dos elementos de imagem pode ser avaliado pela diferença entre os valores dos pixels individuais ou de regiões de pixels. Considerar regiões de pixels diminui a complexidade do modelo de estimação, aumentando a velocidade de processamento. O tipo de agrupamento dos pixels em regiões pode variar desde a forma mais simples, que consiste em agrupar os pixels em regiões retangulares, até formas mais complexas, onde o agrupamento depende das características dos objetos da cena a serem codificados (segmentação). No padrão MPEG, adota-se o modelo mais simples de agrupamento, o seja, agrupam-se os pixels em regiões quadradas de 16 × 16 pixels (macrobloco). Novamente, a escolha é motivada pela velocidade de processamento e pela facilidade de implementação do codificador.

Um modelo geral de deslocamento dos macroblocos, entre os instantes t e τ pode ser descrito em termos de uma transformação afim [23-24], definida por

$$\boldsymbol{d}_{t,\tau}(\boldsymbol{x}) = (\boldsymbol{A} - \boldsymbol{I})\boldsymbol{x} + \boldsymbol{B}, \text{ onde} \tag{2.6}$$

$$\boldsymbol{A} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} di \\ dj \end{bmatrix}. \tag{2.7}$$

No modelo, a matriz $\boldsymbol{A}$ define uma rotação seguida de uma deformação da região retangular e o vetor $\boldsymbol{B}$ a translação da região. No modelo mais simplificado (adotado pelo MPEG), os macroblocos não sofrem rotação e deformação, de modo que o seu deslocamento estimado consiste apenas em uma translação definida pelo vetor $\boldsymbol{B}$ da transformação afim. Este deslocamento, $\boldsymbol{d}_{t,\tau}(\boldsymbol{x})$, na metodologia do MPEG, é denominado vetor de movimento ($\boldsymbol{mv}$), sendo o único parâmetro necessário na definição do deslocamento dos macroblocos entre os quadros da seqüência de vídeo.

2.6.2 - CRITÉRIO DE CASAMENTO

Pode-se definir um grau de semelhança (casamento) entre dois elementos de imagem, $\boldsymbol{C}$ e $\boldsymbol{R}$, baseado na distância entre estes elementos $\boldsymbol{D}(\boldsymbol{C},\boldsymbol{R})$ em relação a um sistema de coordenadas (Figura 2.7). Quanto menor a distância entre os elementos, mais semelhantes serão eles.

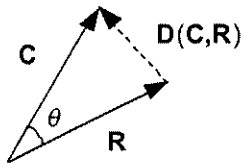


Figura 2.7 – Macroblocos C e R representados como vetores no espaço multidimensional (256 dimensões), sendo $D(C,R)$ a distância entre os blocos.

Considerando como elemento de imagem a região retangular definida pelo MPEG, pode-se representar cada macrobloco como um ponto em um espaço de 256 dimensões ortogonais entre si, sendo os valores dos pixels que compõem o macrobloco os valores das componentes nas diversas dimensões. A distância entre os macroblocos fica definida por $D(C,R) = C - R$, ou seja, o vetor diferença entre os vetores C e R. A raiz quadrada das somas dos valores quadráticos das componentes do vetor diferença representa a distância euclidiana entre C e R. Não se utiliza a distância euclidiana em vista do alto custo computacional exigido em seu cálculo (operações de potenciação). Como alternativa, utiliza-se a distância de Manhatam, dada pela soma dos módulos das diferenças entre os pixels nas várias coordenadas do macrobloco, dada por

$$D(C,R) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left|C(i,j) - R(i,j)\right|, \tag{2.8}$$

onde $N = 16$ (blocos com 16×16 pixels).

2.6.3 - ALGORITMOS DE ESTIMAÇÃO

Seja $C(x+k, y+l)$ a localização dos pixels do macrobloco C no quadro atual e $R(x+k+i, y+l+j)$ a localização dos pixels do macrobloco R no quadro de referência. Baseados em (2.8), define-se o grau de casamento entre os blocos C e R através de um coeficiente denominado MAE (*Mean Absolute Error*) [3], dado por

$$MAE(i,j) = \sum_{k=1}^{N}\sum_{l=1}^{N}\left|C(x+k, y+l) - R(x+k+i, y+l+j)\right|, \tag{2.9}$$

[3] Em rigor, a Equação (2.9) define o erro absoluto médio cumulativo, conquanto não se divide a soma pelo número de elementos dos blocos. Esta divisão é omitida por não ser relevante para o processo de busca.

com $N = 16$ e (x, y) representando a coordenada inicial dos macroblocos $\boldsymbol{C}$ e $\boldsymbol{R}$ nos respectivos quadros. As coordenadas (i, j) identificam a posição de todos os possíveis macroblocos candidatos ao vetor de movimento dentro da região de pesquisa, limitada ao intervalo p = [-16,+16] pixels no padrão MPEG. A coordenada (i, j) onde o menor valor de MAE é encontrado constitui o vetor de movimento.

Vários algoritmos [23-25] são utilizados de forma a se minimizar o MAE. Podem-se dividi-los em duas grandes classes: os algoritmos exatos, onde, a um custo computacional relativamente alto, se garante a solução ótima para o vetor de movimento dentro da região de pesquisa, e os algoritmos aproximados, onde se abre mão da solução ótima em troca de um menor custo computacional. A relação entre a qualidade da estimação (MAE do macrobloco selecionado) e o custo computacional de sua obtenção define o algoritmo utilizado em cada aplicação. A seguir, descrevem-se alguns dos algoritmos de estimação comumente utilizados, destacando-se suas principais características e limitações.

2.6.3.1 - BUSCA EXAUSTIVA – ESA (*EXHAUSTIVE SEARCH ALGORITHM*)

A busca exaustiva [1] procura, entre todas as possíveis combinações de coordenadas (i, j) dentro da região de pesquisa, a coordenada (i, j) que minimiza o MAE. Expandindo a procura por toda a região de pesquisa, garante-se que o valor mínimo do MAE obtido corresponde ao mínimo global da função de custo, de modo que o vetor de movimento obtido é o vetor ótimo (algoritmo exato). A procura por toda a região de pesquisa acarreta um custo computacional alto. Sendo p o intervalo que define a região de pesquisa, têm-se $(2p+1)^2$ localizações pesquisáveis. Para cada macrobloco de $N \times N$ pixels, efetuam-se três tipos de operações: subtração, módulo e adição. Supondo o mesmo peso computacional para as três operações, necessita-se de $(2p+1)^2 \times N^2 \times 3$ operações[4] por macrobloco. Para uma seqüência de vídeo com quadros de $I \times J$ dimensões, a uma taxa de F quadros/s, tem-se um total de $I \times J \times F / N^2$ macroblocos. Efetuamos, então, um total de operações dado por $I \times J \times F \times (2p+1)^2 \times 3$. Para os valores típicos $I = 720$, $J = 480$, $F = 30$ quadros/s, $p =$

[4] Como se deseja apenas estimar o número de operações, desconsidera-se a redução das regiões pesquisáveis nas bordas do quadro e o fato de necessitarmos de N^2-1 operações de soma (ao invés de N^2) no cálculo do MAE. Adotas-se as mesmas considerações na estimativa do número de operações dos outros métodos de estimação.

16 pixels, o número de operações necessárias para o processo de estimação de movimento utilizando busca exaustiva é de ≅ 33.87Gops (**G**iga **o**perações **p**or **s**egundo).

2.6.3.2 - BUSCA LOGARÍTMICA - TDLS (*TWO DIMENSIONAL LOGARITHM SEARCH*)

Uma das formas de reduzir o custo computacional do processo de estimação é diminuir o número dos macroblocos pesquisados na região de pesquisa. A busca logarítmica [1] efetua esta redução computando inicialmente o MAE em nove localizações dentro da região de pesquisa: para o ponto de coordenada $(i, j) = (0,0)$, centro da região de pesquisa, e para os oito pontos distantes de d_1 pixels, nas localizações: $(0,-d_1)$, $(-d_1,0)$, $(d_1,0)$, (d_1,d_1), $(d_1,-d_1)$, $(-d_1,d_1)$ e $(-d_1,-d_1)$, sendo a distância d_1 definida por $d_1 = 2^{k-1}$, onde $k = \lceil log_2 p \rceil$. Exemplificando com p = 16, tem-se k = 4 e d_1 = 8. Uma vez obtido o ponto onde ocorre o valor mínimo do MAE das nove possibilidades anteriores, efetua-se novamente a pesquisa em outros oito pontos com distâncias definidas por $d_2 = d_1/2$. Continua-se o processo iterativo até o valor de d tornar-se unitário. Tem-se, então, uma estimativa do valor do menor MAE dentro da região de pesquisa e a localização do vetor de movimento correspondente. A Figura 2.8 exemplifica o exposto, considerando blocos de 8 × 8 pixels com região de pesquisa com p = 8 pixels.

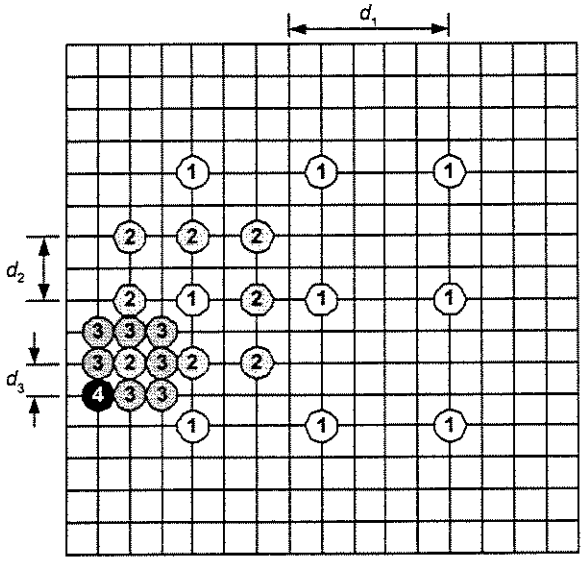


Figura 2.8 – Exemplo do algoritmo de busca logarítmica com três passos para um bloco de 8 × 8 pixels e área de pesquisa p=[-8,8] pixels, com o valor do MAE mínimo obtido para a posição 4.

Ao todo, a busca logarítmica examina $(8k+1)$ posições, computando o MAE para cada uma delas, com um total de $3\times(8k+1)\times N^2$ operações aritméticas ou de módulo. Sendo o número de macroblocos dado por $I\times J\times F/N^2$ e com valores típicos de I = 720, J = 480, F = 30 quadros/s e k = 4, tem-se um total de 1,67Gops operações, aproximadamente. Este valor corresponde a 4,9% do número de operações efetuadas pela busca exaustiva, levando a uma redução clara do custo computacional.

2.6.3.3 - BUSCA HIERÁRQUICA – HME (*HIERARCHICAL MOTION ESTIMATION*)

Algoritmos de busca hierárquica [1] combinam estratégias de redução, tanto no número de macroblocos dentro da região de pesquisa como no número de pixels utilizados no cálculo da função de casamento (MAE), para diminuir o custo computacional da estimação de movimento. Embora existam muitas variações destes algoritmos, descreve-se um algoritmo típico para análise de seu funcionamento (Figura 2.9).

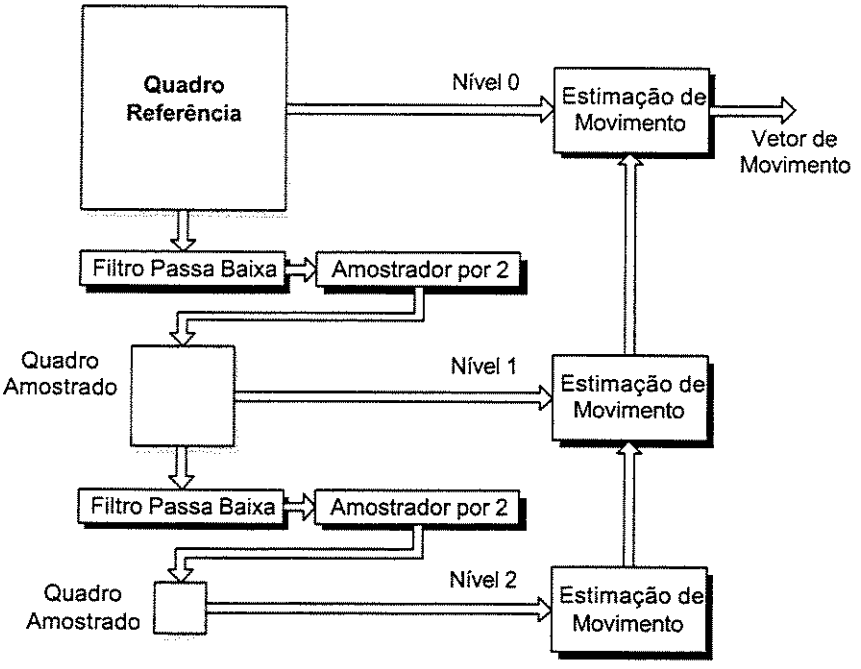


Figura 2.9 – Algoritmo típico para a busca hierárquica.

A procura do vetor de movimento se processa criando-se duas versões em baixa resolução do quadro de referência, utilizando as operações de filtragem e dizimação

aplicadas consecutivamente. A filtragem é necessária de modo a limitar a largura de banda do sinal, reduzindo o efeito do *aliasing* do sinal dizimado [26]. Ao final das operações, têm-se três níveis de quadros: o nível zero, composto pelo quadro de referência, e os níveis um e dois, formados, respectivamente, pela metade e por um quarto dos pixels do quadro de referência. Qualquer referência ao quadro do nível zero, tem um correspondente nos quadros de nível um e dois, com uma mudança conveniente de escala. Assim, um macrobloco ocupando a posição (x, y) no quadro de referência, com x e y múltiplos de quadro, tem os seus blocos correspondentes ocupando as posições $(x/2, y/2)$ e $(x/4, y/4)$ nos níveis um e dois, respectivamente.Inicia-se o processo de estimação através da procura do vetor de movimento no nível dois, utilizando qualquer algoritmo de estimação (busca exaustiva, busca logarítmica, etc.). Neste nível, os macroblocos são representados por blocos 4×4 pixels, com a região de pesquisa reduzida para um quarto do seu valor original. Diminuindo-se a região de pesquisa e o tamanho dos macroblocos, reduz-se o número de cálculos efetuados, aumentando a velocidade de processamento do algoritmo. Assumindo que o MAE é minimizado para o vetor de movimento (u_2, v_2), utiliza-se este valor como referência para a procura do vetor de movimento nos quadros de maior resolução, melhorando a estimativa do vetor em torno daquela posição. Assim, utilizando um algoritmo de estimação arbitrário, efetua-se a estimação de movimento no nível um, considerando blocos com 8×8 pixels e uma área de pesquisa compreendendo o intervalo de [-1,1] pixels ao redor da localização onde o vetor de movimento do nível dois foi encontrado. Esta localização corresponde, em termos de coordenadas no nível um, ao ponto $(x/2 + 2u_2, y/2 + 2v_2)$. Supondo que o MAE seja minimizado para o vetor de movimento (u_1, v_1), repete-se o mesmo processo de estimação para o nível zero, considerando agora o blocos de 16×16 pixels (macroblocos) e uma região de pesquisa de [-1,1] ao redor do ponto de referência $(x + 2u_1, y + 2v_1)$. O vetor de movimento obtido corresponde ao vetor de movimento do processo de estimação.

Uma estimativa do número de operações efetuadas no método da busca hierárquica pode ser realizada considerando como parâmetros quadros de 720×480 pixels para uma taxa de 30 quadros/s. Os blocos têm de 16×16 pixels (macrobloco) no nível zero e região

de pesquisa com p de 16 pixels neste mesmo nível. O método de estimação comumente utilizado em cada nível é a busca exaustiva. Assim, tem-se:

1) **Nível 2**
   - Tamanho do quadro = 180 × 120, com blocos de 4 × 4 pixels.
   - Número de blocos $= \frac{\text{Tamanho do quadro}}{\text{Dimensão do bloco}} = 1350$. Para uma taxa de 30 quadros por segundo, têm-se 40.500 blocos.
   - Dimensão da região de pesquisa = $16/4 = 4$.
   - Número de blocos na região de pesquisa = $(2\times 4+1)^2 = 81$.
   - Número de operações por bloco = número de pixels × 3 = 48 (operações de subtração, módulo e soma).
   - Número de operações no nível 2 = 40.500 × 81 × 48 = 157,46Mops.
2) **Nível 1**
   - Tamanho do quadro = 360 × 240, com blocos de 8 × 8 pixels.
   - Número de blocos = 1350. Para uma taxa de 30 quadros por segundo, têm-se 40.500 blocos.
   - Dimensão da região de pesquisa = 1.
   - Número de blocos na região de pesquisa = $(2\times 1+1)^2 = 9$.
   - Número de operações por bloco = número de pixels × 3 = 192.
   - Número de operações no nível 1 = 40.500 × 9 × 192 = 69,98Mops.
3) **Nível 0**
   - Tamanho do quadro = 720 × 480, com blocos de 16 × 16 pixels.
   - Número de macroblocos = 1350. Para uma taxa de 30 quadros por segundo, têm-se 40.500 macroblocos.
   - Dimensão da região de pesquisa = 1.
   - Número de macroblocos na região de pesquisa = $(2\times 1+1)^2 = 9$.
   - Número de operações por macrobloco = número de pixels × 3 = 768
   - Número de operações no nível 1 = 40.500 × 9 × 768 = 279.94Mops.

No total, tem-se um número de operações de 507,38Mops, apresentando uma redução significativa, com relação aos 33.87Gops utilizadas pelo método da busca

exaustiva. Do ponto de vista computacional, o método da busca hierárquica é muito eficiente, embora necessite de uma quantidade extra de memória para armazenar as imagens filtradas e dizimadas. O método pode apresentar erros significativos para a estimação de regiões contendo pequenos objetos de cena, que podem ser bastante distorcidos.

2.6.3.4 - SEA (*SUCCESSIVE ELIMINATION ALGORITHM*)

O algoritmo de Eliminação Sucessiva (SEA), proposto originalmente por Li e Salari [2], reduz o custo computacional do processo de estimação, eliminando sucessivamente posições dentro da região de pesquisa, diminuindo assim o número de operações. Além do conseqüente aumento de velocidade, o algoritmo converge para a mesma solução encontrada pelo algoritmo ESA, ou seja, o vetor de movimento que minimiza globalmente a função de casamento na região de pesquisa.

Seja $\boldsymbol{C}$ um macrobloco do quadro atual cujo movimento se deseja estimar e $\boldsymbol{R}(i,j)$ o macrobloco na região de pesquisa do quadro de referência ocupando as coordenadas[5] (i,j). Deseja-se minimizar o MAE entre estes blocos, definido por

$$MAE(\boldsymbol{C},\boldsymbol{R}) = \left\|\boldsymbol{C} - \boldsymbol{R}(i,j)\right\|, \tag{2.10}$$

onde $\|\boldsymbol{C}\| = \sum_k |c_k|$, com i e j variando na região de pesquisa (intervalo delimitado por [-16,16]), para macroblocos (k = 1,..,256). Aplicando a desigualdade modular dada por $\| \boldsymbol{A} - \boldsymbol{B} \| \geq \| \boldsymbol{A} \| - \| \boldsymbol{B} \|$ com $\boldsymbol{A} = \boldsymbol{C}$ e $\boldsymbol{B} = \boldsymbol{R}(i,j)$, tem-se

$$MAE[\boldsymbol{C}, \boldsymbol{R}(i,j)] = \left\|\boldsymbol{C} - \boldsymbol{R}(i,j)\right\| \geq \left\|\boldsymbol{C}\right\| - \left\|\boldsymbol{R}(i,j)\right\|. \tag{2.11}$$

A desigualdade (2.11) afirma que o valor do MAE entre os blocos $\boldsymbol{C}$ e $\boldsymbol{R}(i,j)$ é sempre superior ao módulo da diferença entre as normas dos blocos. No cálculo do vetor de movimento ótimo, pode-se descartar a posição (i,j) dentro da região de pesquisa se o valor do MAE destes pontos for superior a um MAE de referência (MAE_{ref}), calculado

[5] Considera-se, para efeito de análise do algoritmo, as coordenadas relativas do macrobloco dentro da região de pesquisa.

inicialmente em algum ponto da região de pesquisa. Assim, descarta-se a posição (i, j) quando:

$$MAE[\boldsymbol{C}, \boldsymbol{R}(i,j)] > MAE_{ref} \tag{2.12}$$

Combinando (2.11) e (2.12), tem-se:

$$MAE[\boldsymbol{C}, \boldsymbol{R}(i,j)] \geq \left\| \|\boldsymbol{C}\| - \|\boldsymbol{R}(i,j)\| \right\| > MAE_{ref} \tag{2.13}$$

Assim, descarta-se a posição (i, j) da região de pesquisa sempre que o módulo da diferença entre as normas dos blocos $\boldsymbol{C}$ e $\boldsymbol{R}(i, j)$ for superior ao valor do MAE de referência. Como o cálculo do módulo da diferença entre as normas consome menos operações que o cálculo do MAE, obtém-se um ganho computacional no processo. Quando o teste dado em (2.13) falha, calcula-se o MAE entre os blocos $\boldsymbol{C}$ e $\boldsymbol{R}(i, j)$. Este valor é comparado com o valor de MAE_{ref}, tomando-se o menor valor como a nova referência (MAE_{ref}). Esta atribuição torna o teste (2.13) mais eficaz para avaliações futuras e aumenta a velocidade do processo de estimação. O algoritmo descrito sempre converge para o mínimo global, pois se baseia na busca exaustiva para a obtenção do valor mínimo do MAE na região de pesquisa.

Para a execução do algoritmo, há a necessidade do cálculo prévio das normas definidas em (2.13). Uma maneira eficaz de calculá-las, de acordo com [2], é descrita a seguir (Figura 2.10).

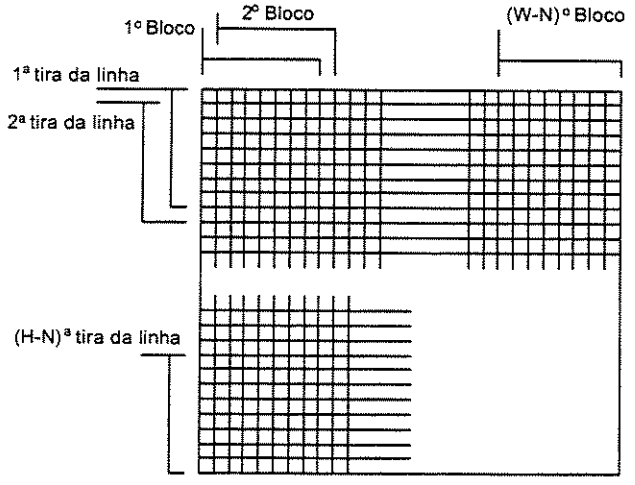


Figura 2.10 – Procedimento para o cálculo das normas dos macroblocos.

Sejam blocos de $N \times N$ pixels, pertencentes a uma região de $W \times H$ pixels, e $f(i,j)$ o valor do pixel na posição (i,j). Efetuam-se dois passos seguintes:

Primeiro Passo: Somam-se, na direção vertical, os valores dos N pixels consecutivos (uma tira de pixels) para todas as W colunas da região de pesquisa, totalizando os coeficientes $K_{1,1}, K_{1,2}, \cdots K_{1,W}$. Este cálculo requer $W \times (N-1)$ operações. A seguir, efetua-se a soma dos elementos da segunda tira, com o deslocamento de um pixel na vertical para todas as colunas. Nota-se que não há necessidade de efetuar novamente todas as operações de soma com os elementos das tira, pois se aproveitam muitas somas realizadas no cálculo anterior. Assim, os coeficientes para a segunda tira são dados por $K_{2,j} = K_{1,j} - f(1,j) + f(N+1,j)$. O cálculo da segunda tira para todas as W colunas exige $2 \times W$ operações. Em procedimento análogo, calcula-se o valor de uma tira a partir do valor da tira anterior pela seguinte fórmula de recorrência $K_{i,j} = K_{i-1,j} - f(i-1,j) + f(N+i-1,j)$, válida para o intervalo $2 \le i \le H - N + 1$ e $1 \le j \le W$. O número total de operações efetuadas no primeiro passo é de $W \times (N-1) + 2 \times W \times (H - N - 1)$.

Segundo Passo: Efetuada a soma das tiras na vertical, somam-se as tiras na horizontal, resultando no valor da norma dos blocos. Primeiramente, somam-se os N primeiros elementos na horizontal para as $(H - N)$ posições na vertical, gerando-se os coeficientes $SN_{i,1}$ (valor das normas). De modo análogo ao primeiro passo, a soma dos coeficientes de uma tira na horizontal pode ser dada em função da soma dos coeficientes da tira horizontal anterior, ou seja, $S_{i,j} = S_{i,j-1} - K_{i,j-1} + K_{i,N+j-1}$, para $1 \le i \le H - N$ e $2 \le j \le W - N + 1$. Assim, tem-se $(H - N) \times (N-1) + 2 \times (H - N) \times (W - N - 1)$ operações para o segundo passo.

O total de operações efetuadas no cálculo das normas dos blocos é o resultado da soma das operações do primeiro e do segundo passo, dado por

$$T = W \times (N-1) + 2 \times W \times (H - N - 1) + (H - N) \times \left[(N-1) + 2 \times (W - N - 1) \right]. \qquad (2.14)$$

Simplificando (2.14) ,tem-se

$$T = 4 \times W \times H - (N+3) \times (W + H - N) - 2 \times W \times N. \qquad (2.15)$$

Estas operações são vistas como um *overhead* de processamento. Comparando este valor com o número de operações necessárias para o cálculo dos vetores de movimento pelo método ESA ($(2p+1)^2$ localizações, com $3\times N^2$ operações para cada um dos $(W/N)\times(H/N)$ blocos do quadro de referência), tem-se a proporção δ dada por

$$\delta = \frac{4\times W\times H-(N+3)\times(W+H-N)-2\times W\times N}{(2p+1)^2\times 3\times N^2\times\left(\frac{W}{N}\times\frac{H}{N}\right)}, \tag{2.16}$$

que representa a relação do custo computacional de *overhead* em relação ao custo computacional da busca exaustiva.

Simplificando (2.16), onde considera-se que W e H são bem maiores que N, tem-se

$$\delta = \frac{4}{3\times(2p+1)^2}. \tag{2.17}$$

Para uma região típica de pesquisa, com p = 16 pixels, encontra-se um *overhead* relativo ao ESA de aproximadamente δ = 0,12%. Sendo este *overhead* pequeno, consegue-se um ganho computacional com este processo de estimação. O número de computações efetuadas pelo SEA é uma variável aleatória, já que depende dos valores de MAE_{ref} tomados ao longo da pesquisa (método adaptativo).

O próximo capítulo examina uma modificação deste algoritmo de estimação de movimento.

Capítulo 3

ALGORITMO SEA MODIFICADO

3.1 - INTRODUÇÃO

Neste capítulo apresenta-se um novo algoritmo de estimação de movimento denominado (MSEA – *Modified Successive Elimination Algorithm*). Este algoritmo é uma evolução no algoritmo SEA, descrito no capítulo anterior, que atua pré-selecionando os macroblocos na região de pesquisa antes do cálculo da função de casamento (MAE). Como resultado da aplicação do algoritmo MSEA, tem-se uma melhora significativa no custo computacional do processo de estimação conservando a solução ótima do vetor de movimento obtida pelo SEA (mínimo global da função de casamento). O melhor desempenho do algoritmo MSEA decorre de uma pré-seleção mais apurada dos macroblocos dentro da região de pesquisa, efetuada pela incorporação de novos critérios de descarte dos macroblocos e pela ordem em que os macroblocos são testados. Apresentamos, a seguir, o algoritmo MSEA.

3.2 - ALGORITMO MSEA

Similarmente ao exposto no Capítulo 2, dado um macrobloco $\boldsymbol{C}$ com coordenadas (x, y) no quadro atual, deseja-se encontrar as coordenadas (i, j) do macrobloco $\boldsymbol{R}(i, j)$ (dentro da região de pesquisa definida no intervalo $[-p, p]$ em relação as coordenadas (x, y)) no quadro de referência que minimiza a norma definida por

$$MAE[\boldsymbol{C}, \boldsymbol{R}(i, j)] = \|\boldsymbol{C} - \boldsymbol{R}(i, j)\|, \tag{3.1}$$

onde $\|\boldsymbol{C}\| \stackrel{\Delta}{=} \sum_k |c_k|$ e $k = 1,...,N \times N$, sendo $N = 16$ (lado do macrobloco).

Expandindo (3.1), tem-se

$$MAE[\boldsymbol{C}, \boldsymbol{R}(i, j)] = \sum_{k=0}^{15} \sum_{l=0}^{15} \left| c_{(x+k, y+l)} - r_{(x+k+i, y+l+j)} \right|, \tag{3.2}$$

onde c e r são os valores dos pixels dos macroblocos $\boldsymbol{C}$ e $\boldsymbol{R}$, respectivamente. Reduz-se a quantidade de macroblocos $\boldsymbol{R}(i,j)$ avaliados em (3.2) adotando-se critérios de descarte. Estes critérios reduzem o número de macroblocos submetidos ao teste do MAE dentro da região de pesquisa sem, contudo, comprometer a obtenção do vetor de movimento ótimo. Sendo o desempenho dos critérios de descarte sensível à ordem com que os macroblocos são testados dentro da região de pesquisa, consegue-se um acréscimo na velocidade de processamento ordenando-se os macroblocos na região de pesquisa antes dos critérios de descarte serem aplicados. São descritos, a seguir, os critérios de descarte e o processo de ordenação do algoritmo MSEA.

3.2.1 - CRITÉRIOS DE DESCARTE

Os critérios de descarte são derivados, utilizando a Equação (3.2) e as seguintes desigualdades modulares [27-28]:

$$|a|+|b| \geq |a+b| \tag{3.3}$$

$$|a-b| \geq |a|-|b|. \tag{3.4}$$

Utiliza-se também o lema a seguir.

***Lema* 1:** Sejam a, b, c e $d \in \Re$. Então se tem a seguinte desigualdade:

$$|a-b|+|c-d| \geq |a+c|-|b+d|. \tag{3.5}$$

Prova:

Aplicando as desigualdades (3.3) e (3.4), tem-se

$$|a-b|+|c-d| \geq |a-b+c-d| \Leftrightarrow$$

$$|a-b|+|c-d| \geq |a+c-(b+d)| \Leftrightarrow$$

$$|a-b|+|c-d| \geq |a+c|-|b+d| \qquad \text{c.q.d}$$

Obtém-se os critérios de descarte reordenando-se os termos do somatório da Equação (3.2) que define o MAE e aplicando recursivamente a desigualdade dada em (3.5). Tem-se, então

$$MAE[\boldsymbol{C},\boldsymbol{R}(i,j)] = \underbrace{\sum_{k=0}^{15}\sum_{l=0}^{15}\left|c_{(x+k,y+l)} - r_{(y+i+k,j+l+j)}\right|}_{\text{MAE}} \geq \underbrace{\sum_{k=0}^{7}\sum_{l=0}^{7}\left|a^{(2)}_{(x+k,y+l)} - b^{(2)}_{(x+k+i,y+l+j)}\right|}_{\text{MAE}_{2\times2}} \geq$$

$$\geq \underbrace{\sum_{k=0}^{3}\sum_{l=0}^{3}\left|a^{(4)}_{(x+k,y+l)} - b^{(4)}_{(x+k+i,y+l+j)}\right|}_{\text{MAE}_{4\times4}} \geq \underbrace{\sum_{k=0}^{1}\sum_{l=0}^{1}\left|a^{(8)}_{(x+k,y+l)} - b^{(8)}_{(x+k+i,y+l+j)}\right|}_{\text{MAE}_{8\times8}} \geq \underbrace{\left|a^{(16)}_{(x+l,y+l)} - b^{(16)}_{(x+k+i,y+l+j}\right|_{k=l=0}}_{\text{MAE}_{16\times16}} \tag{3.6}$$

para os coeficientes $a^{(P)}_{(x+k,y+l)}$ e $a^{(P)}_{(x+k+i,y+l+j)}$ definidos por

$$a^{(P)}_{(x+k,y+l)} = \sum_{m=0}^{P-1}\sum_{n=0}^{P-1} c_{(x+P\times k+m,y+P\times l+n)}\,, \tag{3.7}$$

$$b^{(P)}_{(x+k+i,y+l+j)} = \sum_{m=0}^{P-1}\sum_{n=0}^{P-1} r_{(x+P\times k+i+m,y+P\times l+j+n)}\,. \tag{3.8}$$

Os coeficientes definidos pelas Equações (3.7) e .(3.8) são obtidos somando-se os agrupamentos de $P \times P$ pixels adjacentes em cada macrobloco $\boldsymbol{C}$ e $\boldsymbol{R}(i,j)$, respectivamente. As desigualdades expressas em (3.6) estabelecem os critérios de descarte (um total de quatro critérios acrescido do cálculo do valor do MAE). Elas nos mostram que o valor do MAE para os agrupamentos dos pixels aumenta ou permanece constante à medida que os agrupamentos vão se tornando menores. Assim, o MAE calculado para os pixels individuais dos blocos $\boldsymbol{C}$ e $\boldsymbol{R}(i,j)$ (agrupamento de 1×1 pixels) é sempre igual ou superior ao MAE calculado entre estes mesmos pixels quando são agrupados e somados em grupos de 2×2 pixels ($MAE_{2\times2}$). Este, por sua vez, é igual ou superior ao MAE calculado para os pixels agrupados e somados em grupos de 4×4 pixels ($MAE_{4\times4}$), e assim por diante até o último agrupamento de 16×16 pixels ($MAE_{16\times16}$, que corresponde ao critério definido pelo SEA, baseado no módulo das diferenças das normas dadas por $\|\boldsymbol{C}\| - \|\boldsymbol{R}(i,j)\|$). O número de operações efetuadas para o cálculo dos MAE decresce, aproximadamente de um fator P^2 à medida que P aumenta, pois este é o fator de redução do número de elementos nos blocos onde o valor de $MAE_{P\times P}$ é calculado. Consegue-se, assim, uma redução no número de computações aumentando a velocidade de processamento do método. Na aplicação dos critérios de descarte, inicia-se com os testes mais simples ($MAE_{16\times16}$) progredindo para os testes mais complexos. O último teste corresponde ao

valor do MAE entre os macroblocos. Procedendo deste modo, minimiza-se o número de operações necessárias para o descarte dos macroblocos não relevantes na seleção do vetor de movimento ótimo dentro da região de pesquisa. Na aplicação dos critérios de descarte, segue-se o seguinte algoritmo:

1° Passo: Calcula-se um valor de MAE entre o macrobloco $\boldsymbol{C}$ do quadro atual que se deseja estimar e o macrobloco $\boldsymbol{R}(i,j)$ do quadro de referência localizado nas coordenadas (i,j) a serem escolhidas dentro da região de pesquisa. O valor deste MAE será denominado MAE_{ref}, pois será utilizado como referência na aplicação dos critérios de descarte. As coordenadas (i,j) são escolhidas convenientemente numa tentativa de minimizar o valor de MAE_{ref} inicial, acelerando o algoritmo.

2° Passo: Para todos os outros macroblocos $\boldsymbol{R}(u,v)$ com $u \neq i$ e $v \neq j$ localizados dentro da região de pesquisa, com coeficientes $a_{(.,.)}^{(P)}$ e $b_{(.,.)}^{(P)}$ definidos pelas Equações (3.7) e (3.8) calculados previamente numa etapa de pré-processamento, calculam-se os valores de $MAE_{16\times16}$ para cada valor de (u,v) e comparam-se-os com o valor de MAE_{ref}. Caso o valor de $MAE_{16\times16}$ seja superior ao valor de MAE_{ref}, o MAE entre os macroblocos $\boldsymbol{C}$ e $\boldsymbol{R}(u,v)$ apresenta valor superior ao valor do MAE entre os macroblocos $\boldsymbol{C}$ e $\boldsymbol{R}(i,j)$, podendo o macrobloco $\boldsymbol{R}(u,v)$ ser descartado. Por outro lado, caso o valor do $MAE_{16\times16}$ seja igual ou inferior ao valor de MAE_{ref}, não se pode afirmar nada sobre o valor do MAE entre os macroblocos $\boldsymbol{C}$ e $\boldsymbol{R}(u,v)$ (o teste falha), de modo que o teste subseqüente (obtido da comparação de $MAE_{8\times8}$ com MAE_{ref}) passa a ser aplicado. Procede-se, assim, sucessivamente com os testes envolvendo $MAE_{4\times4}$, $MAE_{2\times2}$ e MAE até o macrobloco ser eliminado por algum deles. Neste caso outro macrobloco $\boldsymbol{R}$ dentro da região de pesquisa passa a ser analisado, reiniciando-se o **2° Passo**. Alternativamente, caso o macrobloco passe por todos os testes, ele passa a ser o novo MAE_{ref}, e o processo é reinicializado (**2° Passo**).

Usar o menor valor do MAE em cada passo da estimação do macrobloco $\boldsymbol{R}$ ocasiona dois efeitos: um aumento da velocidade de processamento, pois torna os critérios de descartes mais eficazes (critérios baseados na comparação com valores de MAE cada vez menores) e a convergência do algoritmo de estimação para o mínimo da função de casamento, garantindo que o vetor de movimento ótimo é sempre encontrado.

Os cálculos dos coeficientes $a_{(.,.)}^{(P)}$ e $b_{(.,.)}^{(P)}$ definidos em (3.7) e .(3.8) podem ser efetuados em uma etapa de pré-processamento, já que seus valores não se alteram durante o processo de estimação. Os coeficientes são calculados considerando-se todo o quadro da seqüência, pois valores para uma região de pesquisa ou para um macrobloco particular serão sempre subconjuntos destes valores. No cálculo dos coeficientes, utiliza-se um algoritmo mais eficiente ao que foi apresentado pelo algoritmo SEA (Capítulo 3), devido às condições particulares do problema: cálculo dos coeficientes em quatro níveis de agrupamentos com lados iguais a potência de dois (P = 2, 4, 8 e 16). Tem-se, então o seguinte algoritmo de cálculo (exemplificado para o cálculo dos coeficientes $a_{(.,.)}^{(P)}$).

- Dada uma imagem de $W \times H$ pixels, calculam-se os coeficientes $a_{(.,.)}^{(P)}$ para os agrupamentos de 2 × 2 pixels (P = 2), somando-se dois a dois os pixels adjacentes em todas as $W-1$ direções na horizontal e em todas as $H-1$ direções na vertical. O cálculo de cada coeficiente consome três operações de soma, de modo que tem-se um total de $3\times(W-1)\times(H-1)$ operações.
- No cálculo dos coeficientes $a_{(.,.)}^{(P)}$ para o próximo agrupamento (P = 4) utilizam-se os resultados dos coeficientes $a_{(.,.)}^{(P)}$ obtidos no agrupamento anterior (P = 2), como forma de reduzir o número de operações. Isto é possível, pois a soma dos elementos de agrupamento 4 × 4 pixels pode ser vista como a soma do agrupamento 2 × 2 elementos, onde cada elemento é o resultado da soma anterior dos agrupamentos de 2 × 2 destes pixels, (P = 2). Assim, cada elemento consome três operações de soma para $W-3$ elementos na direção horizontal e $H-3$ elementos na direção vertical, num total de $3\times(W-3)\times(H-3)$ operações.
- Para o cálculo dos agrupamentos superiores (P = 8 e 16), utiliza-se a mesma argumentação anterior, ou seja, para o cálculo dos coeficientes $a_{(.,.)}^{(P)}$ de ordem P, utilizam-se os coeficientes de ordem $P/2$, num total de $3\times(W-P+1)\times(H-P+1)$ operações.

Assim, o método de cálculo dos coeficientes $a_{(.,.)}^{(P)}$ para os quatro níveis de agrupamento consome um total de $12\times[W\times H-6{,}5\times(W+H)+71]$ operações.

Comparando-se com o número de operações executadas pelo algoritmo SEA (Equação (2.15) para o pré-processamento) e lembrando-se que o pré-processamento do algoritmo SEA deve ser executado quatro vezes para obter os coeficientes nos quatro níveis de agrupamentos, tem-se, aproximadamente

$$\delta = \frac{4\times[4\times W\times H+(N+3)\times(W+H-N)-2\times W\times N]}{12\times[W\times H-6{,}5\times(W+H)+71]}. \tag{3.9}$$

Como geralmente W e H são maiores que N (número de pixels de um macrobloco = 16), pode-se simplificar .(3.9), resultando em $\delta \cong 16/12 \cong 1{,}33$, ou seja, o pré-processamento baseado no algoritmo SEA necessita de cerca de 33% mais operações para efetuar o cálculo dos coeficientes $a^{P}_{(.,.)}$ para os quatro níveis quando comparado com o método aqui proposto.

A Figura 3.1 mostra um exemplo do cálculo dos coeficientes $a^{(P)}_{(.,.)}$, considerando um bloco de 8 × 8 elementos. Neste caso, consideram-se três níveis de agrupamento, com P = 2,4 e 8 pixels.

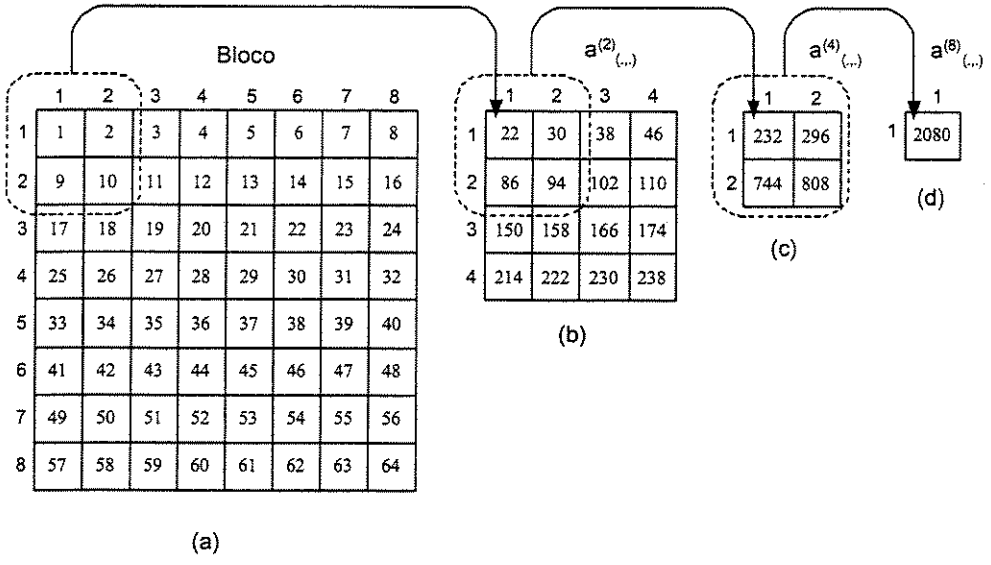


Figura 3.1 – Exemplo do cálculo dos agrupamentos definidos pela Equação (3.8). (a) Bloco de 8 × 8 pixels; (b) Soma dos agrupamentos de 2 × 2 pixels; (c) Soma dos agrupamentos de 4 × 4 pixels; (d) Soma dos agrupamentos de 8 × 8 pixels.

3.2.2 - PRÉ-ORDENAÇÃO DOS MACROBLOCOS

Naturalmente, quanto menor o valor do MAE de referência empregado na análise dos macroblocos dentro da região de pesquisa, mais seletivos se tornam os critérios de

descarte, o que permite eliminar muitos macroblocos candidatos ao vetor de movimento sem a necessidade de computações irrelevantes. A pré-ordenação dos macroblocos estabelece uma ordem eficiente de leitura e teste dos macroblocos dentro da região de pesquisa, aumentando a possibilidade de se encontrar os melhores casamentos (menores valores de MAE) logo no início do processo de estimação e causando um aumento da seletividade dos critérios de descarte e uma redução no número de operações efetuadas.

A ordenação dos macroblocos baseia-se na probabilidade de obter valores pequenos de MAE. Pela desigualdade (3.7), verifica-se que, de certa forma, os valores pequenos de MAE estão relacionados com valores pequenos de $\left\|\|\boldsymbol{C}\| - \|\boldsymbol{R}(i,j)\|\right\|$ ($\mathrm{MAE}_{16\times16}$), pois, sendo o MAE sempre superior a este valor (Equação 2.11), valores pequenos de MAE implicam em valores pequenos de $\left\|\|\boldsymbol{C}\| - \|\boldsymbol{R}(i,j)\|\right\|$, embora o inverso nem sempre ocorra[1]. Quando se estima um macrobloco $\boldsymbol{C}$ dentro da região de pesquisa, o valor de $\|\boldsymbol{C}\|$ é fixo, de modo que valores pequenos de MAE tendem a ocorrer em macroblocos cujos valores de $\|\boldsymbol{R}(i,j)\|$ são mais próximos de $\|\boldsymbol{C}\|$. Assim, adota-se o seguinte procedimento de teste dos macroblocos:

1. Ordenam-se os valores de $\|\boldsymbol{R}(i,j)\|$ na etapa de pré-processamento para todos os macroblocos do quadro que se deseja estimar. A ordenação, que é um processo relativamente custoso, é efetuada uma única vez na etapa de pré-processamento, pois os valores $\|\boldsymbol{R}(i,j)\|$ permanecem constantes para a estimação de todos os outros macroblocos do quadro.
2. Procura-se, dentro do conjunto $\|\boldsymbol{R}(i,j)\|$ ordenado, a posição (u,v) do macrobloco cuja norma $\|\boldsymbol{R}(u,v)\|$ mais se aproxima do valor de $\|\boldsymbol{C}\|$. Calcula-se o valor do MAE entre os macroblocos $\boldsymbol{R}(u,v)$ e $\boldsymbol{C}$, o qual é tomado como referência (MAE_{ref}).
3. Testam-se os outros macroblocos seguindo a ordenação estabelecida, ou seja, avançando ou retrocedendo posições a partir da posição (u,v) encontrada de modo a tornar as diferenças $\left\|\|\boldsymbol{C}\| - \|\boldsymbol{R}(.,.)\|\right\|$ em ordem crescente. Espera-se que o valor de MAE_{ref} obtido anteriormente, juntamente com os que porventura venham a ser calculados

[1] Observa-se que $\left|\|\boldsymbol{C}\| - \|\boldsymbol{R}(i,j)\|\right|$ representa o comprimento da projeção do "vetor" $\boldsymbol{C} - \boldsymbol{R}(i,j)$ em uma das coordenadas do espaço de sinal (cf. Seção 3.3)

resultem em valores pequenos, aumentando a eficiência dos critérios de descarte e do algoritmo de estimação.

Com base no conjunto ordenado de valores de $\|\boldsymbol{R}(i,j)\|$ e com o valor de MAE_{ref} calculado previamente, consegue-se delimitar a região de macroblocos pesquisáveis, ou seja, os macroblocos cujo MAE podem ser inferiores ao valor de MAE_{ref}. Um macrobloco $\boldsymbol{R}(i,j)$ é ainda pesquisável, quando

$$MAE_{ref} \geq MAE[\boldsymbol{C},\boldsymbol{R}(i,j)]. \tag{3.10}$$

Como

$$MAE[\boldsymbol{C},\boldsymbol{R}(i,j)] \geq \|\boldsymbol{C}\| - \|\boldsymbol{R}(i,j)\|, \tag{3.11}$$

testam-se, a partir de (3.10) e (3.11), apenas os macroblocos $\boldsymbol{R}(i,j)$ tais que

$$MAE_{ref} \geq \|\boldsymbol{C}\| - \|\boldsymbol{R}(i,j)\|. \tag{3.12}$$

Desenvolvendo a inequação modular, chegamos a seguinte desigualdade

$$(\|\boldsymbol{C}\| - MAE_{ref})^{+} \leq \|\boldsymbol{R}(i,j)\| \leq MAE_{ref} + \|\boldsymbol{C}\|, \tag{3.13}$$

onde a função $(x)^{+}$ é igual a x para $x \geq 0$ e a 0, caso contrário.

A desigualdade (3.13) delimita o intervalo dos macroblocos ainda pesquisáveis, correspondendo à aplicação do primeiro critério de descarte. Assim, a ordenação apresenta uma dupla função: permite uma ordem de leitura e teste dos macroblocos tornando os critérios de descarte mais seletivos e implementa o primeiro critério de descarte de modo mais eficiente, pois seleciona diretamente a região de macroblocos ainda válidos, sem ser necessário testá-los um a um, como sugere o primeiro critério de descarte. A Figura 3.2 ilustra o procedimento descrito acima.

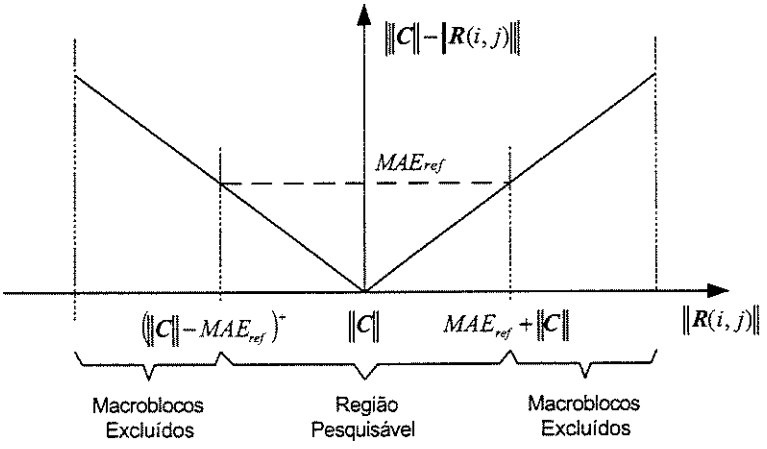


Figura 3.2– Determinação da região dos macroblocos pesquisáveis pela pré-ordenação dos macroblocos.

Como exemplo da aplicação do método, considera-se a Figura 3.3.

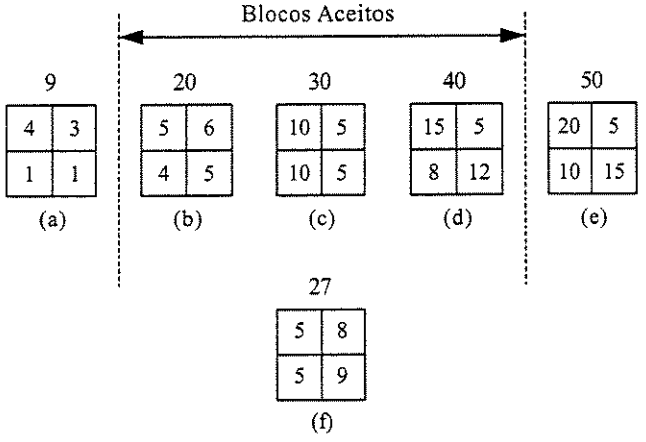


Figura 3.3 – Exemplo da aplicação do primeiro critério de descarte via pré-ordenação dos macroblocos.

Nesta figura, tem-se um conjunto $R(i,j)$ de 5 blocos de 4×4 indexados de (a) a (e), no qual será selecionado o bloco que mais se aproxima do bloco C definido por (f), segundo o critério definido pelo MAE. Os resultados dos cálculos de $\|C\|$ e $\|R(i,j)\|$, bem como sua ordenação, podem ser vistos na figura, onde o valor da norma aparece na parte superior de cada bloco. O bloco (f) apresenta norma 27, sendo mais "semelhante" ao bloco (c) com norma 30. Calculando-se o MAE entre os blocos (f) e (c) (MAE_{ref}), obtém-se um valor de 17. Portanto, os blocos como norma superior a $17+27=44$ ou inferior a $(27-17)^+=10$ podem ser rejeitados, pois apresentam um valor de MAE com relação ao

bloco (f) superior ao MAE_{ref}. Os blocos não rejeitados serão submetidos posteriormente aos outros critérios de descarte.

De passagem, faz-se referência a um critério alternativo de escolha do ponto inicial do algoritmo de busca, baseado no algoritmo ABM (*Adaptative Block Matching*) [29].

3.3 - INTERPRETAÇÃO ESPACIAL

Um macrobloco composto de 16 × 16 pixels pode ser interpretado como um ponto em um espaço vetorial de 16 × 16 = 256 dimensões. A distância entre dois pontos em um espaço vetorial é sempre maior que a distância entre estes mesmos pontos quando projetados em algum subespaço do espaço original. Assim, utilizando o MAE como a distância, pode-se descartar, para um dado MAE_{ref} obtido entre o bloco $\boldsymbol{C}$ (objeto de estimação de movimento) e um determinado bloco $\boldsymbol{R}(u,v)$, todos os macroblocos $\boldsymbol{R}(i,j)$ cujas distâncias a $\boldsymbol{C}$, projetadas sobre um subespaço previamente escolhido, sejam superiores ao valor de MAE_{ref} tomado como limitante. A Figura 3.4 ilustra a idéia.

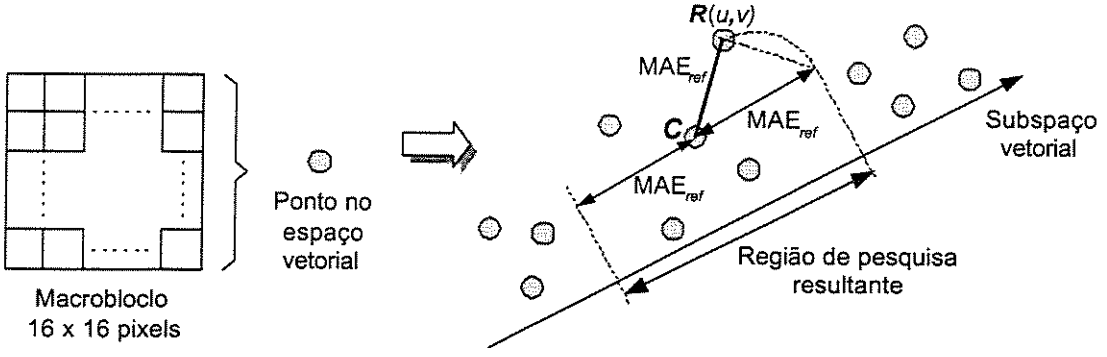


Figura 3.4 – Interpretação espacial do algoritmo MSEA.

Os subespaços vetoriais usados no algoritmo podem ser formados a partir de combinações lineares dos 256 vetores linearmente independentes dados pelos vetores dos pixels nas posições que compõem o macrobloco. Gerando-se o subespaço formado pela soma de todos os vetores de base (subespaço de dimensão unitária, $\mathbb{R}^1$), tem-se o primeiro critério de descarte definido pelo MSEA. Efetuando agrupamentos de 8 × 8 pixels com os elementos dos macroblocos, gera-se um subespaço de quatro dimensões ($\mathbb{R}^4$) e deriva-se daí o segundo critério de descarte definido pelo MSEA. Procedendo assim, sucessivamente, geram-se todos os critérios de descarte propostos pelo MSEA. Com a ordenação dos

vetores baseados nas suas projeções no subespaço $\mathbb{R}^1$, tenta-se, de modo simplificado, definir uma ordem adequada de leitura dos macroblocos para aumentar a probabilidade de se encontrar valores pequenos de MAE à medida que se prossegue no processo de estimação. Valores pequenos de distância entre os macroblocos do espaço vetorial original implicam em valores pequenos das projeções destas distâncias nos vários subespaços vetoriais, embora o inverso nem sempre ocorra.

UNICAMP
BIBLIOTECA CENTRAL
SEÇÃO CIRCULANTE

Capítulo 4

RESULTADOS

4.1 - INTRODUÇÃO

Neste capítulo analisa-se o desempenho do algoritmo SEA modificado (MSEA), frente aos algoritmos ilustrados no Capítulo 3. A análise do desempenho é composta de dois conjuntos de simulações:

- No primeiro conjunto, são utilizados seis quadros consecutivos de várias seqüências de vídeo onde é efetuada a estimação e a compensação de movimento. Parâmetros como o tempo de processamento, o número de operações efetuadas, erros de predição, etc., são levantados e comparados com os resultados de vários métodos de estimação já analisados. Deseja-se estudar, em detalhes, o comportamento do algoritmo MSEA e a sua eficiência no processo de estimação.
- No segundo conjunto de simulações, os métodos de estimação são incorporados a um codificador MPEG-2 utilizado na codificação de várias seqüências de vídeo. Deseja-se analisar a influência do algoritmo MSEA no desempenho do codificador e, em especial, seu efeito sobre os seguintes parâmetros: taxa de bits, distorção e tempo de processamento.

4.2 - PARÂMETROS DAS SIMULAÇÕES

No primeiro conjunto de simulações (estimação com quadros estáticos) são utilizados quadros monocromáticos no formato CCIR 601 de 720 × 480 pixels, com profundidade de 8bpp (bits por pixel), compreendendo os quadros de 100 a 105 das seqüências de vídeo "Flower Garden", "Mobile Calendar", "Kiel", e "Tennis" (Apêndice A). Para cada seqüência escolhida, há um modo predominante de movimento dos objetos de cena, como uma simples translação linear para a seqüência "Flower Garden", movimentações horizontais e verticais lentas para a seqüência "Mobile Calendar", movimentações rápidas verticais para a seqüência "Tennis" e efeitos de *zoom* dos objetos na seqüência "Kiel". Esta variedade de movimentos permite uma melhor análise do

algoritmo MSEA no processo de estimação de movimento. As regiões de pesquisa compreendem todo o quadro ou regiões quadradas de lado $[-p,p]$, para o valor de $p \in \{16, 32, 64 \text{ e } 128\}$ pixels, contados a partir de cada coordenada do macrobloco a ser estimado. Dos quadros tomados de cada seqüência, o de menor ordem será a referência para a estimação dos vetores de movimento para o de maior ordem. Assim, o quadro 100 é a referência para o quadro 101, o quadro 101 é a referência para o quadro 102, e assim sucessivamente até o quadro 105. Os resultados são dados em termos da média dos resultados obtidos com o processamento dos quadros de cada seqüência.

Para o segundo conjunto de simulações (estimação com seqüências de vídeo), utilizaram-se versões coloridas das seqüências de vídeo definidas anteriormente. Os quadros são não entrelaçados, de 720 × 480 pixels, onde cada pixel é composto de três componentes (R,G e B), com profundidade de 8 bits, perfazendo um total de 24bpp. As regiões de pesquisa abrangem toda a área do quadro ou a região quadrada de lado [-16,16], padrão definido pelo MPEG. Utilizando-se um GOP dado pela seqüência IBBPBBPBB pode-se processar 55 quadros de cada seqüência de vídeo, abrangendo o modo intraquadro (I) e o modo interquadro, com predição (P) ou interpolação (B). Todos os outros parâmetros do codificador (tabelas de entropia, matrizes de quantização, etc.) adotados são mostrados no Apêndice A.

Adotam-se as seguintes medidas para a análise de desempenho dos vários métodos de estimação:

- RMSE (***Root Mean Squared Error***): O parâmetro RMSE mede a qualidade do processo de estimação, através do cálculo da raiz quadrada do valor médio da soma dos erros quadráticos dos pixels do quadro de erro, quadro este obtido da diferença entre o quadro atual (que originou a estimação) e o quadro estimado. Quanto menor o valor de RMSE, tanto melhor é o processo de estimação. A definição do RMSE é dada pela Equação (4.1)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} E_i^2}{N}}, \tag{4.1}$$

sendo E_i^2 o erro (diferença) entre o pixel i do quadro atual e o pixel i do quadro de referência. Para as simulações que envolvem imagens monocromáticas, o valor de E_i^2 é dado por

$$E_i^2 = \left(Y_i^E - Y_i^A\right)^2, \tag{4.2}$$

onde

Y_i^E = Componente de luminância de pixel i do quadro estimado.

Y_i^A = Componente de luminância de pixel i do quadro atual.

Para simulações com imagens coloridas, o valor de E_i^2 abrange as três componentes coloridas R, G e B, é dado por

$$E_i^2 = \frac{\left(R_i^E - R_i^A\right)^2 + \left(G_i^E - G_i^A\right)^2 + \left(B_i^E - B_i^A\right)^2}{3}, \tag{4.3}$$

onde

R_i^β = Componente R de pixel i no quadro atual ($\beta = A$) ou estimado ($\beta = E$),

G_i^β = Componente G de pixel i no quadro atual ($\beta = A$) ou estimado ($\beta = E$),

B_i^β = Componente B de pixel i no quadro atual ($\beta = A$) ou estimado ($\beta = E$).

- PSNR (***Peak Signal to Noise Ratio)***: O PSNR é uma medida similar ao RMSE, indicativa da qualidade do quadro reconstruído após o processamento. O valor do PSNR é dado pela seguinte equação (considerando uma profundidade de 8bpp):

$$PSNR = 20\log_{10}\left(\frac{255}{RMSE}\right)\text{dB}. \tag{4.4}$$

Evidentemente, quanto maior o PSNR, tanto melhor é a qualidade da imagem reconstruída.

- TP (**T**empo de **P**rocessamento): O tempo de processamento mede o tempo necessário para se completar a estimação de movimento para um dado algoritmo. Na sua medida,

utilizou-se um computador Pentium III, 800MHz, com 512Mbytes de memória RAM e um disco rígido de 20Gb, com velocidade de rotação de 7200 rpm. Como o tempo de processamento depende do computador utilizado, muitas vezes utiliza-se a comparação relativa entre os tempos de processamento (ganho de velocidade de processamento) como uma medida comparativa entre os vários algoritmos.

- R (Taxa de Bit): O parâmetro taxa de bit, medido em bits/pixel, informa a quantidade média de bits necessária para se representar cada pixel do quadro da seqüência de vídeo após a codificação. A partir da taxa de bit, obtém-se o fator de compressão do codificador (redução do número de bits necessários para se representar, em média, cada pixel do quadro), pois cada pixel é codificado, originalmente, com 24 bits nas imagens coloridas. O fator de compressão é, portanto

$$FC = \frac{24}{R}. \tag{4.5}$$

- *Overhead*: O termo *overhead* refere-se ao custo computacional extra (tempo de processamento ou número de operações) necessário à execução do algoritmo de estimação (fase de pré-processamento). Por representar um custo computacional fixo, o *overhead* impõe um limite na velocidade dos algoritmos, que necessitam de pré-processamento.
- NOP (**N**úmero de **OP**erações): O coeficiente NOP mede o número de operações efetuadas por cada método de estimação. Seguindo a orientação dada por [20], o número de operações é contabilizado considerando o número de operações aritméticas efetuadas (somas e subtrações)[1] e operações de módulo, atribuindo a cada operação o mesmo peso. Assim, por exemplo, o cálculo da expressão $|\ 2 - 1|$ resulta em duas operações: uma subtração e o cálculo do módulo do valor resultante. Por simplicidade, não se leva em consideração o custo de acessar os pixels nas suas diversas posições. Considera-se, também, o número de operações necessárias para as etapas de pré-processamento (quando necessário) dos algoritmos.

[1] Observa-se que os métodos não utilizam multiplicações ou divisões.

4.3 - ESTIMAÇÃO COM QUADROS ESTÁTICOS

A análise inicia-se comparando os diversos algoritmos de estimação utilizando quadros estáticos. Divide-se o conjunto de simulações em três grupos:

1) Simulações comparando o desempenho do algoritmo MSEA com os algoritmos ESA e SEA na obtenção dos vetores de movimento por todo o quadro de referência. Os algoritmos ESA e SEA aplicados em todo o quadro de referência garantem a obtenção do menor erro de predição para o quadro processado. Visa-se, assim, estudar o grau de eficiência do algoritmo MSEA na obtenção da solução ótima da estimação de movimento.
2) Simulações comparando o desempenho do algoritmo MSEA com os algoritmos ESA, SEA, HME e TDLS, com regiões de pesquisa limitadas ao lado de 32, 64, 128 e 256 pixels. Limita-se a região de pesquisa com o objetivo de diminuir o tempo de processamento e o custo computacional da estimação, tendo como resultado um vetor de movimento que minimiza localmente a função de custo. Utilizando o algoritmo MSEA, pode-se verificar o seu desempenho em regiões de pesquisa limitadas.
3) Simulações envolvendo estratégias híbridas de estimação como a fusão entre o algoritmo da busca hierárquica (HME) e o algoritmo MSEA. Sendo o MSEA um algoritmo de casamento de padrões, pode-se utilizá-lo como o próprio algoritmo de estimação ou como parte de outro algoritmo mais genérico. Assim, combinando os algoritmos de estimação, tenta-se melhorar a relação entre a velocidade de processamento e a qualidade da estimação.

4.3.1 - ESTIMAÇÃO COM REGIÃO DE PESQUISA INCLUINDO TODO O QUADRO DE REFERÊNCIA

Comparam-se os algoritmos ESA, SEA e MSEA com região de pesquisa englobando todo o quadro de referência. Sendo o ESA e o SEA algoritmos exaustivos, garante-se a obtenção do vetor de movimento ótimo para os macroblocos processados, minimizando-se o erro de predição. Com o MSEA, garante-se esta mesma convergência a um custo computacional inferior. Os resultados das simulações são mostrados abaixo:

Algoritmo ESA					
Seqüência	**TP(s)**	**RMSE**	**PSNR**	**MAE**	**NOP(Gop)**
Flower	2220,0	5,887	33,001	2,980	339,450
Kiel	2219,3	11,414	26,982	6,237	339,450
Mobile	2219,5	13,778	25,348	7,557	339,450
Tennis	2220,0	8,071	29,998	4,431	339,450

Tabela 4.1 – Resultados da estimação de movimento com o algoritmo ESA.

Algoritmo SEA					
Seqüência	**TP(s)**	**RMSE**	**PSNR**	**MAE**	**NOP(Gop)**
Flower	97,3	5,887	33,001	2,980	12,346
Kiel	430,3	11,414	26,982	6,237	62,267
Mobile	335,3	13,777	25,348	7,557	47,948
Tennis	387,0	8,071	29,998	4,431	55,787

Tabela 4.2– Resultados da estimação de movimento com o algoritmo SEA.

Algoritmo MSEA (sem ordenação)					
Seqüência	**TP(s)**	**RMSE**	**PSNR**	**MAE**	**NOP(Gop)**
Flower	26,5	5,887	33,001	2,980	1,591
Kiel	72,0	11,414	26,982	6,237	8,721
Mobile	50,3	13,777	25,348	7,557	5,171
Tennis	111,5	8,071	29,998	4,431	15,306

Tabela 4.3– Resultados da estimação de movimento com o algoritmo MSEA (sem ordenação).

Algoritmo MSEA (com ordenação)					
Seqüência	**TP(s)**	**RMSE**	**PSNR**	**MAE**	**NOP(Gop)**
Flower	11,0	5,887	33,001	2,980	0,625
Kiel	44,5	11,413	26,983	6,237	1,936
Mobile	31,8	13,777	25,348	7,557	1,229
Tennis	87,0	8,071	29,997	4,431	5,229

Tabela 4.4– Resultados da estimação de movimento com o algoritmo MSEA (com ordenação).

Nas Tabelas 4.3 e 4.4 são mostradas duas variações dos algoritmos MSEA. A Tabela 4.3 mostra os resultados da aplicação do algoritmo MSEA sem a ordenação inicial dos macroblocos na etapa de pré-processamento (somente os critérios de descarte são utilizados). Na Tabela 4.4 mostram-se os resultados quando a ordenação é efetuada. Com estes dados pode-se comparar separadamente os efeitos dos critérios de descarte e da ordenação inicial dos macroblocos no desempenho do algoritmo MSEA.

Comparando-se os resultados das simulações, verifica-se a superioridade do algoritmo MSEA na obtenção do vetor de movimento ótimo em relação aos algoritmos

ESA e SEA. Conseguem-se os mesmos valores da função de casamento (MAE) e dos parâmetros que medem a qualidade do quadro processado (RMSE e PSNR) a um custo computacional inferior. Para o algoritmo ESA, considerando os quadros da seqüência "Flower Garden", obtém-se tempos de processamento da ordem de 2220s e cerca de 340Gop (**G**iga **op**erações) para se completar a estimação. Com o algoritmo SEA, estes valores diminuem para 97s e cerca de 12Gop, ou seja, um ganho no tempo de processamento da ordem de 23 vezes e cerca de 28 vezes em termos do número de operações. Quando se utiliza o algoritmo MSEA (sem ordenação), o tempo de processamento atinge a marca de 26,5s com cerca de 1,6Gop, resultando num ganho em relação ao ESA de 84 vezes em tempo e de 213 vezes em operações, e um ganho em relação ao SEA de 3,6 vezes no tempo e de 7,5 vezes no número de operações. Observa-se que a redução no tempo de processamento não segue a mesma proporção da redução no número de operações (esta última é sempre superior), já que são desconsideradas as operações de acesso à memória no número de operações. Ao utilizarmos o algoritmo MSEA com a ordenação inicial dos macroblocos, reduz-se ainda mais o tempo de processamento e o número de operações, com valores de 11s e 0,625Gop, atingindo um ganho de 202 vezes em tempo e de 543 vezes em número de operações em relação ao ESA. Em relação ao SEA, os ganhos observados são de 8,9 vezes em tempo e de 19,8 vezes em número de operações. Portanto, a adoção do algoritmo MSEA leva a uma queda importante do tempo de processamento e do número de operações efetuadas, mantendo, ainda, a solução ótima do vetor de movimento. Os tempos de processamento e o número de operações do processo de estimação variam quanto ao tipo de quadro processado. No algoritmo ESA estes valores são constantes. Para os quadros da seqüência de vídeo onde a técnica de casamento de blocos se adapta bem à descrição do movimento dos objetos em cena, o algoritmo MSEA apresenta um melhor desempenho. Isto ocorre para a seqüência "Flower Garden", pois a movimentação dos objetos da cena se dá através de uma translação horizontal lenta. Para movimentos mais complexos, como o efeito de *zoom* da seqüência "Kiel", ou as movimentações horizontais e verticais das seqüências "Mobile Calendar" e "Tennis", observa-se uma redução no ganho de processamento. Esta perda de desempenho está associada ao valor de MAE_{ref} utilizado como referência nos critérios de descarte do algoritmo MSEA. Como se viu, quanto menor o valor de MAE_{ref} e quanto mais rápido ele é

encontrado, tanto mais restritivos se tornam os critérios de descarte, aumentando a eficiência da estimação. Para a seqüência "Flower Garden", conseguem-se valores pequenos de MAE_{ref} logo no início do processo de estimação, acelerando a velocidade do algoritmo. As tabelas abaixo descrevem os ganhos no tempo de processamento e as reduções no número de operações em relação ao algoritmo ESA, cujos valores são tomados como referência (normalizado para um valor unitário).

Ganho no Tempo de Processamento				
Seqüência	**ESA**	**SEA**	**MSEA (sem ordenação)**	**MSEA (com ordenação)**
Flower	1	22,8	83,8	201,8
Kiel	1	5,2	30,8	49,9
Mobile	1	6,6	44,2	69,9
Tennis	1	5,7	19,9	25,5

Tabela 4.5 – Ganhos no tempo de processamento dos algoritmos SEA e MSEA com relação ao ESA.

Redução do Número de Operações				
Seqüência	**ESA**	**SEA**	**MSEA (sem ordenação)**	**MSEA (com ordenação)**
Flower	1	27,5	213,3	542,8
Kiel	1	5,5	38,9	175,3
Mobile	1	7,1	65,6	276,3
Tennis	1	6,1	22,2	64,9

Tabela 4.6 – Redução no número de operações dos algoritmos SEA e MSEA com relação ao ESA.

Pelas tabelas acima, o algoritmo MSEA com ordenação é mais eficiente que a sua versão sem ordenação, pois ao se definir uma ordem adequada de leitura dos macroblocos dentro da região de pesquisa, aumenta-se a probabilidade de obtenção de valores pequenos de MAE_{ref}, o que eleva a eficiência dos critérios de descarte. Esta eficiência é mostrada nas tabelas abaixo, onde se contam o número de macroblocos testados em cada critério de descarte para os algoritmos MSEA sem e com ordenação.

Algoritmo MSEA sem ordenação					
Seqüência	**1° Critério**	**2° Critério**	**3° Critério**	**4° Critério**	**MAE**
Flower	442.563.750	14.942.132	1.990.257	894.611	361.496
Kiel	442.563.750	80.027.983	40.130.698	15.820.437	2.669.869
Mobile	442.563.750	61.360.245	24.355.674	8.063.329	1.207.309
Tennis	442.563.750	71.579.794	47.949.576	28.034.366	7.856.203

Tabela 4.7– Número de macroblocos testados em cada critério de descarte pelo algoritmo MSEA sem ordenação.

Algoritmo MSEA com ordenação					
Seqüência	**Ordenação**	**2° Critério**	**3° Critério**	**4° Critério**	**MAE**
Flower	442.563.750	14.743.462	1.822.041	754.759	304.226
Kiel	442.563.750	51.609.705	13.840.127	2.500.348	313.166
Mobile	442.563.750	39.749.134	8.358.109	1.299.915	195.987
Tennis	442.563.750	50.012.233	27.581.348	11.920.472	1.441.498

Tabela 4.8– Número de macroblocos testados em cada critério de descarte para o algoritmo MSEA com ordenação.

Para os quadros de 720 × 480 pixels, com macroblocos de 16 × 16 pixels e região de pesquisa equivalendo a todo o quadro (705 × 465), tem-se um total de 442.563.750 posições, que correspondem ao total de macroblocos a serem testados para a determinação de todos os vetores de movimento do quadro estimado. Para o algoritmo MSEA sem ordenação (Tabela 4.7) são observadas reduções sensíveis no número dos macroblocos a cada aplicação dos critérios de descartes. Toma-se, como exemplo, a seqüência "Flower Garden". Com o algoritmo ESA, necessita-se testar 442.563.750 macroblocos. Com algoritmo MSEA sem ordenação, o primeiro critério de descarte reduz este conjunto para 14.942.132 macroblocos, ou seja, apenas 3,38% dos macroblocos iniciais precisam ser testados. Esta é a redução obtida pelo algoritmo SEA, já que ele utiliza apenas o primeiro critério de descarte. Aplicando o segundo critério de descarte, este número cai para 1.990.257 macroblocos, ou seja, apenas 0,45% dos macroblocos originais necessitam ser testados. Continuando o processo, após a avaliação do último critério de descarte, chega-se a um percentual de 0,08% do número original. Para o algoritmo MSEA com ordenação, ordenam-se os macroblocos segundo a ordem crescente de suas normas e aplica-se a desigualdade (3.15) delimitando o número de macroblocos ainda pesquisáveis. A aplicação da ordenação tem o efeito de reduzir o número de macroblocos testados no segundo critério de descarte para 14.743.462 (inferior ao número obtido para o algoritmo MSEA sem ordenação), representando 3,3% dos macroblocos originais. Continuando o processo até o último critério de descarte, necessitam-se apenas de 304.226 cálculos de MAE para se determinar todos os vetores de movimento ótimos, o que equivale a 0,07% dos macroblocos iniciais. Nas tabelas abaixo, apresentam-se os resultados correspondentes para as outras seqüências.

Algoritmo MSEA sem ordenação					
Seqüência	**1° Critério**	**2° Critério**	**3° Critério**	**4° Critério**	**MAE**
Flower	100%	3,38%	0,45%	0,20%	0,08%
Kiel	100%	18,08%	9,07%	3,57%	0,60%
Mobile	100%	13,86%	5,50%	1,82%	0,27%
Tennis	100%	16,17%	10,83%	6,33%	1,78%

Tabela 4.9– Percentual do número de macroblocos testados em cada critério de descarte para o algoritmo MSEA sem ordenação.

Algoritmo MSEA com ordenação					
Seqüência	**Ordenação**	**2° Critério**	**3° Critério**	**4° Critério**	**MAE**
Flower	100%	3,33%	0,41%	0,17%	0,07%
Kiel	100%	11,66%	3,13%	0,56%	0,07%
Mobile	100%	8,98%	1,89%	0,29%	0,04%
Tennis	100%	11,30%	6,23%	2,69%	0,33%

Tabela 4.10 – Percentual do número de macroblocos testados em cada critério de descarte para o algoritmo MSEA com ordenação.

Observamos, pela Tabela 4.9 e pela Tabela 4.10, que a ordenação inicial dos macroblocos aumenta a eficiência dos critérios de descarte, tornando a triagem dos macroblocos pesquisáveis mais seletiva, o que eleva a eficiência global do algoritmo.

Necessita-se de um *overhead* de processamento para a execução do algoritmo MSEA com e sem ordenação. No algoritmo MSEA sem ordenação, o *overhead* é composto pelas operações de cálculo dos coeficientes $a_{(.,.)}^{(P)}$ e $b_{(.,.)}^{(P)}$ definidos pelas Equações (3.7) e (3.8). Para o algoritmo MSEA com ordenação, além destes cálculos, inclui-se a ordenação dos coeficientes $b_{(.,.)}^{(16)}$, conforme já visto (Capítulo 3). A tabela abaixo mostra o *overhead* de tempo necessário para ambas as versões do algoritmo e o porcentual em relação ao tempo total de processamento.

	MSEA sem ordenação		**MSEA com ordenação**	
Seqüência	***Overhead*(s)**	**%**	***Overhead*(s)**	**%**
Flower	0,233	0,88%	0,798	7,25%
Kiel	0,233	0,32%	0,800	1,80%
Mobile	0,233	0,46%	0,808	2,54%
Tennis	0,233	0,21%	0,785	0,90%

Tabela 4.11 – Overhead de processamento para os algoritmos MSEA sem e com ordenação.

Pela tabela acima, os tempos de processamento são pequenos, correspondendo, em média, a 0,223s para o algoritmo MSEA sem ordenação e a 0,798s para o algoritmo MSEA

com ordenação. Gastam-se 0,223s para o cálculo dos coeficientes $a_{(.,.)}^{(P)}$ e $b_{(.,.)}^{(P)}$ e 0,575s adicionais para a ordenação dos coeficientes $b_{(.,.)}^{(16)}$ com o algoritmo *QuickSort* (complexidade $O(Nlog(N))$ utilizado. Estes *overheads* impõem um limitante inferior ao tempo de processamento do algoritmo, embora este tempo ainda possa ser diminuído utilizando-se outras técnicas de soma dos coeficientes $a_{(.,.)}^{(P)}$ e $b_{(.,.)}^{(P)}$, novos algoritmos de ordenação ou em implementações utilizando *hardware* dedicado. Com relação ao tempo total de processamento dos algoritmos, os *overheads* têm pouca influência relativa, com exceção da seqüência "Flower Garden" (algoritmo MSEA com ordenação), que por apresentar um tempo de processamento pequeno, eleva o percentual de participação do *overhead*.

4.3.2 - Estimação com Região de Pesquisa Limitada

Nas simulações que se seguem, compara-se o desempenho (tempo de processamento) dos algoritmos ESA, SEA, MSEA, HME e TDLS, utilizando-se regiões quadradas de pesquisa, de lado $[-p,p]$, com p = 16, 32, 64 e 128 pixels. Os resultados são mostrados nas tabelas abaixo:

Seqüência	Região	TP_{ESA}(s)	TP_{SEA}(s)	TP_{MSEA}(s)	TP_{HME}(s)	TP_{TDLS}(s)
Flower	16	6,50	1,75	0,78	0,14	0,24
	32	25,75	4,25	1,25	0,20	---
	64	97,50	9,75	2,28	0,44	---
	128	345,75	24,00	5,53	1,23	---
Kiel	16	6,50	3,75	1,3750	0,1425	0,24
	32	25,50	11,00	2,9750	0,2050	---
	64	97,25	32,25	7,0250	0,4350	---
	128	345,50	91,25	18,1500	1,2350	---
Mobile	16	6,50	3,50	1,18	0,14	0,23
	32	26,00	10,00	2,53	0,20	---
	64	97,50	28,75	5,90	0,44	---
	128	345,25	78,00	14,20	1,23	---
Tennis	16	6,75	4,50	2,35	0,14	0,235
	32	25,50	14,00	5,90	0,21	---
	64	97,50	40,00	14,63	0,44	---
	128	345,50	101,50	32,70	1,24	---

Tabela 4.12 – Tempos de processamento para os algoritmos ESA, SEA, MSEA, HME e TDLS com região de pesquisa variável.

Sendo o algoritmo TDLS específico para regiões de pesquisa de 16 pixels, este algoritmo tem seu desempenho analisado somente para esta região. Nas simulações, utiliza-se o algoritmo MSEA sem ordenação, pois se a ordenação fosse aplicada a cada região de pesquisa, tornaria o algoritmo MSEA com ordenação mais lento que a sua versão sem ordenação. Pela Tabela 4.12, observa-se que o tempo computacional obtido pelo algoritmo ESA é progressivamente reduzido pela aplicação dos algoritmos SEA, MSEA, TDLS e HME, respectivamente, sendo que os algoritmos MSEA e SEA conservam a solução ótima do vetor de movimento obtida pelo ESA. O menor tempo de processamento obtido pelos algoritmos TDLS e HME decorrem de simplificações na procura do vetor de movimento, as quais resultam em soluções que não são necessariamente ótimas. Os gráficos dos tempos de processamento são dados abaixo

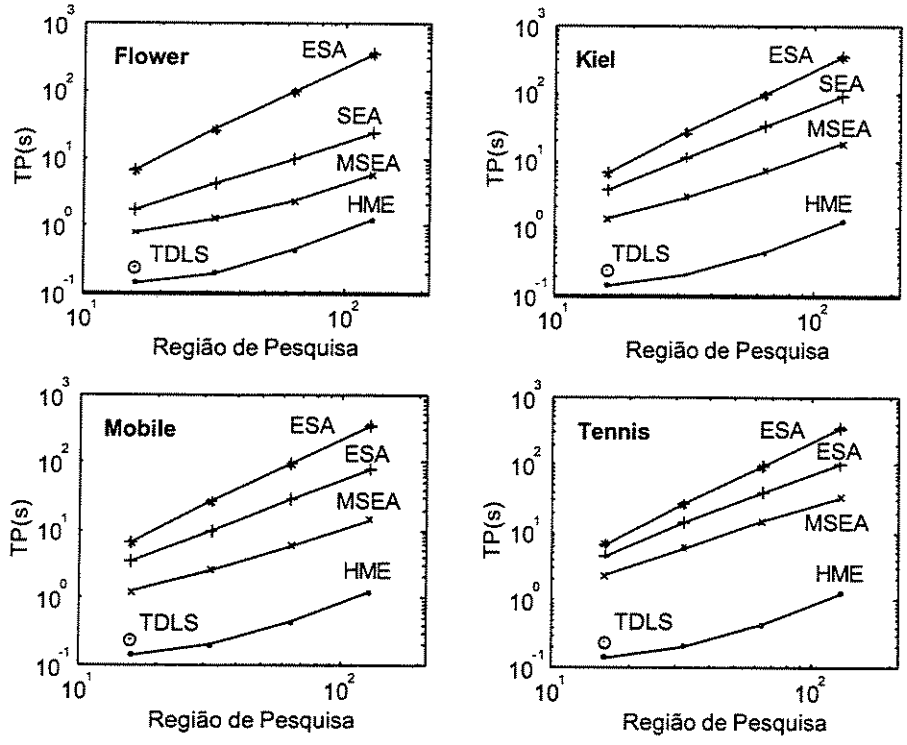


Figura 4.1 – Tempo de processamento para os algoritmos ESA, SEA, MSEA, HME e TDLS.

Ajustando uma curva polinomial do tipo $TP = \beta \times w^{\alpha}$ aos dados do gráfico, onde w é a largura da região de pesquisa e α e β são constantes, tem-se a seguinte tabela:

	Flower		Kiel		Mobile		Tennis	
Método	**α**	**β**	**α**	**β**	**α**	**β**	**α**	**β**
ESA	1,8735	0,0394	1,8801	0,0382	1,8655	0,0409	1,8801	0,0382
SEA	1,2487	0,0554	1,5262	0,0558	1,4817	0,0594	1,429	0,1009
MSEA	1,0727	0,029	1,3045	0,0319	1,2443	0,0337	1,2353	0,083
HME	1,3103	0,002	1,2954	0,0022	1,3103	0,002	1,2809	0,0024

Tabela 4.13– Coeficientes da curva polinomial ajustada do tempo de processamento x largura da região de pesquisa.

Pela Tabela 4.13, o algoritmo MSEA apresenta o menor crescimento (coeficiente α) do tempo de processamento com o aumento da região de pesquisa (exceção da seqüência Kiel). Assim, espera-se que para regiões grandes de pesquisa, o algoritmo MSEA seja mais eficiente com relação ao tempo de processamento frente aos seus concorrentes.

A qualidade dos quadros processados, medida através do parâmetro RMSE para as várias seqüências e para os vários algoritmos é mostrada abaixo (Tabela 4.14).

Seqüência	Região	$RMSE_{ESA}$	$RMSE_{SEA}$	$RMSE_{MSEA}$	$RMSE_{HME}$	$RMSE_{TDLS}$
Flower	16	6,24	6,24	6,24	6,32	6,37
	32	6,17	6,17	6,17	6,31	---
	64	6,13	6,13	6,13	6,31	---
	128	6,08	6,08	6,08	6,32	---
Kiel	16	11,43	11,43	11,43	12,86	15,02
	32	11,42	11,42	11,42	13,62	---
	64	11,42	11,42	11,42	14,68	---
	128	11,42	11,42	11,42	15,55	---
Mobile	16	14,10	14,09	14,09	17,13	21,72
	32	14,03	14,03	14,03	18,67	---
	64	13,93	13,93	13,93	19,61	---
	128	13,85	13,85	13,85	20,67	---
Tennis	16	8,95	8,95	8,95	10,38	13,73
	32	8,73	8,73	8,73	10,61	---
	64	8,49	8,49	8,49	10,65	---
	128	8,18	8,18	8,18	10,61	---

Tabela 4.14 – Qualidade do quadro processado (RMSE) para os algoritmos ESA, SEA, MSEA, HMSE e TDLS com região de pesquisa variável.

Pela Tabela 4.14, verificam-se os mesmos valores de RMSE para os algoritmos MSEA, ESA e SEA, demonstrando a capacidade do SEA e do MSEA de obter o vetor de movimento que minimiza o erro de predição dentro da região de pesquisa. Salienta-se que a busca do vetor de movimento se baseia na minimização do MAE, e não do RMSE. O baixo custo computacional dos algoritmos HME e TDLS é contrabalançado pelo maior erro de predição e pela conseqüente deterioração da qualidade dos quadros reconstruídos. Para quadros com movimento translacional lento predominante, como os quadros da seqüência "Flower Garden", não há diferenças significativas nos erros de predição. Para os outros quadros, com movimentos diversos e mais complexos, o erro aumenta, sendo mais pronunciado para o algoritmo TDLS do que para o algoritmo HME, já que este último pesquisa sobre um conjunto maior de macroblocos na determinação do vetor de movimento. Observa-se um aumento do erro de processamento com o aumento da região de pesquisa para o algoritmo HME, pois com regiões de pesquisa maiores, há uma maior chance do algoritmo perder o caminho da convergência ótima. Os gráficos abaixo resumem os resultados da Tabela 4.14.

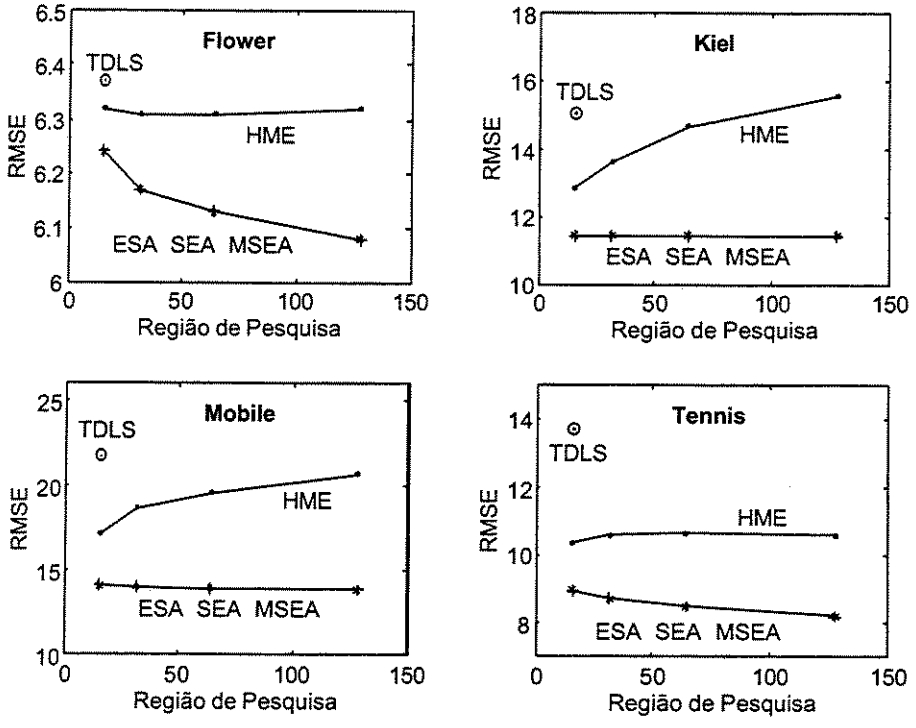


Figura 4.2 – Erro de predição para os algoritmos ESA, SEA, MSEA, HME e TDLS.

Nas figuras abaixo, tem-se um exemplo dos quadros processados para as seqüências "Flower" (menor erro) e "Mobile" (maior erro) com regiões de pesquisa com $p = 16$ pixels, evidenciando o erro produzido no processo de estimação.

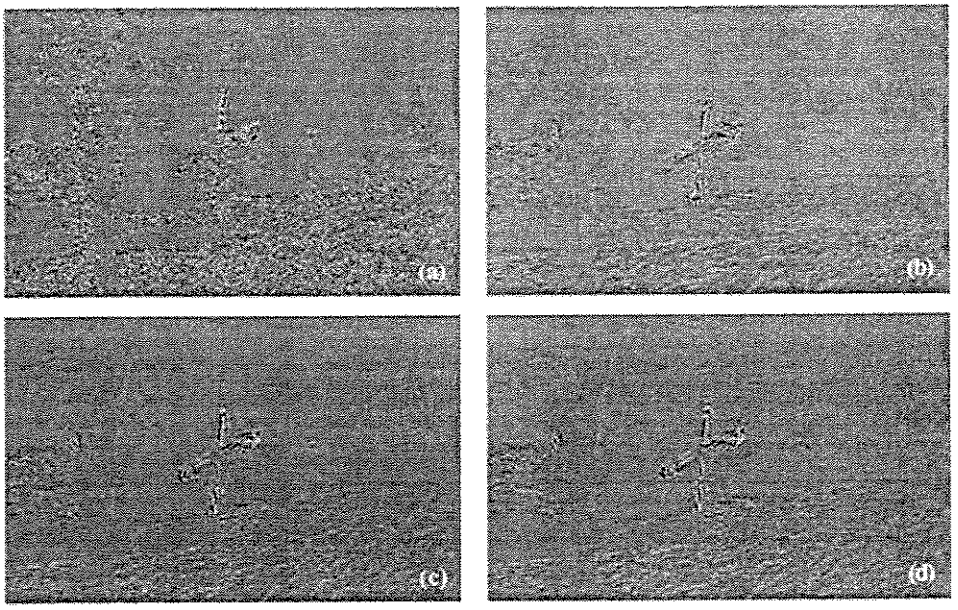


Figura 4.3 – Diferença entre o quadro de referência e o quadro atual para a seqüência "Flower Garden". (a) Sem compensação de movimento; (b) Com TDLS; (c) Com HME; (d) Com MSEA.

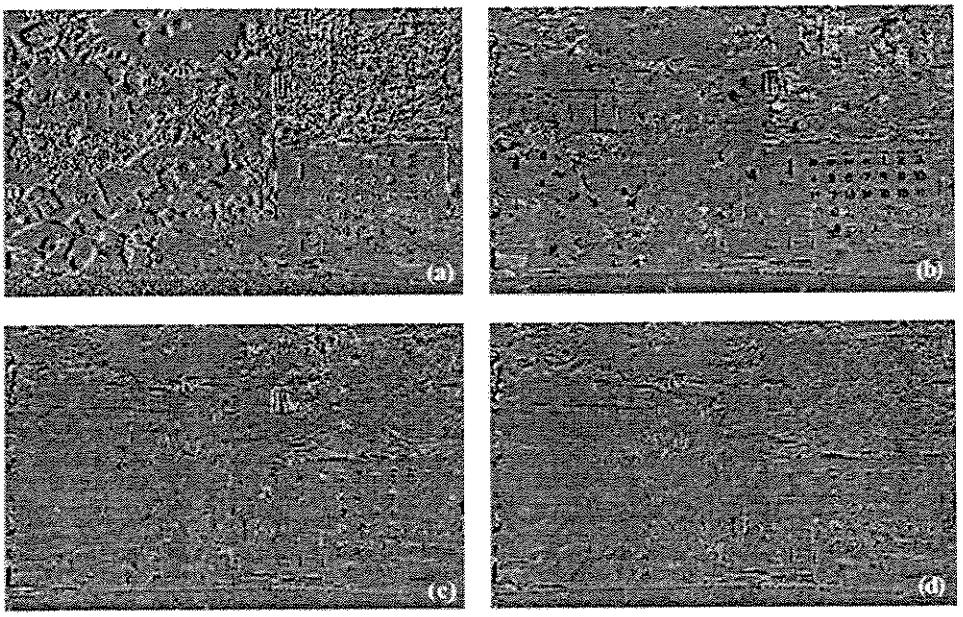


Figura 4.4 – Diferença entre o quadro de referência e o quadro atual para a seqüência "Mobile Calendar". (a) Sem compensação de movimento; (b) Com TDLS; (c) Com HME; (d) Com MSEA.

Em relação ao número de operações efetuadas, as comparações podem ser feitas a partir da seguinte tabela:

Seqüência	Região	NOP_{ESA} (Gop)	NOP_{SEA} (Gop)	NOP_{MSEA} (Gop)	NOP_{HME} (Gop)	NOP_{TDLS} (Gop)
Flower	16	1,004	0,237	0,082	0,015	0,032
	32	3,902	0,576	0,146	0,027	---
	64	14,715	1,394	0,254	0,068	---
	128	52,016	3,270	0,480	0,212	---
Kiel	16	1,004	0,562	0,180	0,016	0,032
	32	3,902	1,633	0,427	0,027	---
	64	14,715	4,733	1,028	0,068	---
	128	52,016	13,377	2,455	0,212	---
Mobile	16	1,004	0,490	0,148	0,015	0,032
	32	3,902	1,462	0,356	0,027	---
	64	14,715	4,259	0,841	0,068	---
	128	52,016	11,365	1,805	0,212	---
Tennis	16	1,004	0,658	0,336	0,016	0,032
	32	3,902	2,061	0,910	0,027	---
	64	14,715	5,936	2,254	0,068	---
	128	52,016	14,894	4,846	0,212	---

Tabela 4.15– Número de operações efetuadas pelos algoritmos ESA, SEA, MSEA, HME e TDLS para uma região de pesquisa variável.

Em concordância com os resultados obtidos nos tempos de processamento, o algoritmo MSEA apresentou uma redução considerável no número de operações efetuadas em comparação com o algoritmo ESA. Para os algoritmos HME e TDLS o número de operações foi menor, ma, como já se viu, estes algoritmos apresentam maiores erros de predição. Dos algoritmos testados, o MSEA é o algoritmo que apresenta o menor crescimento do número de operações com o aumento das larguras da região de pesquisa. O crescimento do número de operações pode ser avaliado pelos gráficos da Figura 4.5 e na Tabela 4.16, onde ajustou-se curvas da forma $NOP = \beta \times w^{\alpha}$ para se descrever a relação entre o número de operações (NOP) e as largura da região de pesquisa (w), com α e β constantes.

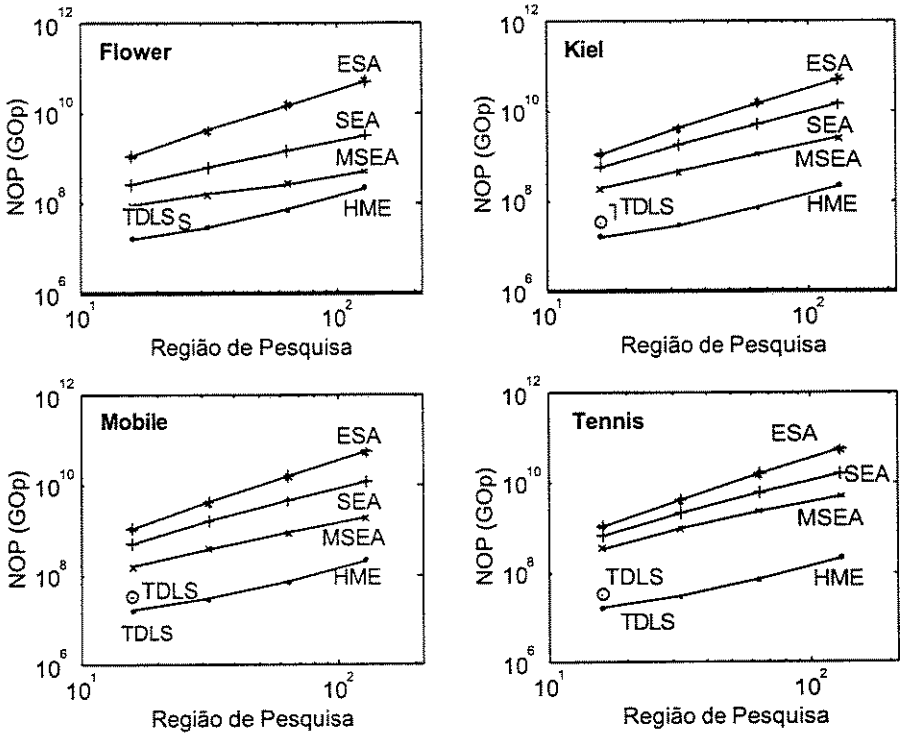


Figura 4.5 – Número de operações efetuadas pelos algoritmos ESA, SEA, MSEA, HME e TDLS.

	Flower		Kiel		Mobile		Tennis	
Método	**α**	**β (x10^6)**	**α**	**β (x10^6)**	**α**	**β (x10^6)**	**α**	**β (x10^6)**
ESA	1,8684	6,0787	1,8684	6,7866	1,8684	6,0787	1,8684	6,0786
SEA	1,2531	7,5207	1,5171	8,5364	1,4795	8,7989	1,4267	15,0220
MSEA	0,8601	7,2980	1,2618	5,3914	1,1715	6,2342	1,2067	14,2180
HME	1,4978	0,1433	1,4928	0,1469	1,4975	0,1435	1,4950	0,1453

Tabela 4.16– Coeficientes da curva polinomial ajustada para o número de operações em função da largura da região de pesquisa

Para o algoritmo ESA, o número de operações varia com o quadrado da largura da região de pesquisa (Seção 2.6.3.1), resultando em um coeficiente ajustado α próximo a dois. Como os algoritmos SEA e MSEA são algoritmos adaptativos, esperam-se coeficientes α menores em relação ao do ESA. Também, como o algoritmo HME utiliza o algoritmo ESA na imagem subamostrada, espera-se uma tendência de crescimento similar ao do ESA. Esta tendência não é visível em regiões de pesquisas pequenas, aparentemente devido à influência acentuada do número de operações exigido para a filtragem das

imagens subamostradas e *overhead*. Como observado na avaliação dos tempos de processamento, para grandes regiões de pesquisa, pode-se esperar que o algoritmo MSEA apresente o menor número de operações, quando comparado com os outros algoritmos.

4.3.3 - INCORPORAÇÃO DO ALGORITMO MSEA A OUTROS ALGORITMOS DE ESTIMAÇÃO

Nas análises com regiões de pesquisa variáveis, o algoritmo HME apresenta o menor tempo de processamento, mas a qualidade dos quadros processados não é sempre satisfatória. Nesta seção, incorpora-se o algoritmo MSEA a estrutura interna do algoritmo HME, melhorando-se o seu desempenho, tanto no decréscimo do tempo de processamento, quanto na melhoria da qualidade dos quadros processados. Como o algoritmo MSEA é basicamente um algoritmo de casamento de padrões, pode-se utilizá-lo na etapa de estimação dos vetores de movimento do algoritmo HME. Na Tabela 4.17 têm-se os resultados das simulações (repetidos para o algoritmo HME para facilitar a comparação).

Seqüência	Região	$TP_{HME}(s)$	$TP_{HME/MSEA}(s)$	$RMSE_{HME}$	$RMSE_{HME/MSEA}$
Flower	16	0,14	0,14	6,32	6,32
	32	0,20	0,17	6,31	6,31
	64	0,44	0,23	6,31	6,31
	128	1,23	0,40	6,32	6,32
Kiel	16	0,14	0,15	12,86	12,86
	32	0,21	0,18	13,62	13,62
	64	0,44	0,26	14,68	14,68
	128	1,24	0,49	15,55	15,55
Mobile	16	0,14	0,14	17,13	17,13
	32	0,20	0,17	18,67	18,67
	64	0,44	0,25	19,61	19,61
	128	1,23	0,43	20,67	20,67
Tennis	16	0,14	0,15	10,38	10,38
	32	0,21	0,18	10,61	10,61
	64	0,44	0,26	10,65	10,65
	128	1,24	0,46	10,61	10,61

Tabela 4.17– Comparação dos algoritmos HME e HME/ MSEA.

Como se observa, obtém-se o mesmo valor de erro com um tempo de processamento inferior. A diferença entre os tempos de processamento se acentua à medida que aumenta a largura da região de pesquisa, sendo os tempos praticamente iguais para

regiões de pesquisa com p = 16 pixels, reduzindo-se de um fator entre dois e três para regiões de pesquisa com p = 128 pixels.

Com a incorporação do algoritmo MSEA ao algoritmo HME, reduz-se o tempo de processamento e mantém-se o erro de processamento do HME. Pode-se aumentar a qualidade do quadro processado utilizando o algoritmo HME com apenas um nível de amostragem ao invés de se trabalhar com dois níveis, como é usual (cf. Figura 2.9). A estimação baseada na imagem menos dizimada (nível 1) melhora a qualidade da estimação, mas aumenta o número de operações pelo aumento das dimensões dos blocos estimados. O aumento no número de operações e do tempo de processamento pode ser parcialmente compensado com a incorporação do algoritmo MSEA ao algoritmo HME-nível1, melhorando-se a relação qualidade × tempo de processamento. A Tabela 4.18 mostra os resultados das simulações.

Seqüência	Região	RMSE MSEA	RMSE HME	RMSE HME1/SEA	TP (s) MSEA	TP(s) HME	TP(s) HME1/MSEA
Flower	16	6,24	6,32	6,25	0,78	0,14	0,36
	32	6,17	6,31	6,23	1,25	0,20	0,76
	64	6,13	6,31	6,14	2,28	0,44	1,77
	128	6,08	6,32	6,08	5,53	1,23	4,28
Kiel	16	11,43	12,86	11,69	1,38	0,14	0,43
	32	11,42	13,62	11,76	2,98	0,21	0,90
	64	11,42	14,68	11,82	7,03	0,44	2,16
	128	11,42	15,55	11,88	18,15	1,24	5,86
Mobile	16	14,09	17,13	14,33	1,18	0,14	0,39
	32	14,03	18,67	14,32	2,53	0,20	0,82
	64	13,93	19,61	14,25	5,90	0,44	1,90
	128	13,85	20,67	14,19	14,20	1,23	4,39
Tennis	16	8,95	10,38	9,30	2,35	0,14	0,44
	32	8,73	10,61	9,05	5,90	0,21	1,02
	64	8,49	10,65	8,79	14,63	0,44	2,50
	128	8,18	10,61	8,50	32,70	1,24	5,79

Tabela 4.18– Resultados das simulações do HME com estimação no primeiro nível utilizando o algoritmo MSEA (HME1/MSEA).

Comparando-se estes resultados com os obtidos na Tabela 4.12 e na Tabela 4.14, observa-se uma melhora no erro de predição (atingindo valores bem próximos dos valores ótimos para a região de pesquisa), com tempos de processamento inferiores aos obtidos com o algoritmo MSEA em separado. Assim a combinação dos algoritmos MSEA e HME-

nível1 (HME1/MSEA) pode se constituir em uma alternativa interessante para o problema de compensação de movimento. Nos gráficos abaixo (Figura 4.6), são apresentados os valores dos erros de predição em função das larguras das regiões de pesquisa, para os vários algoritmos analisados.

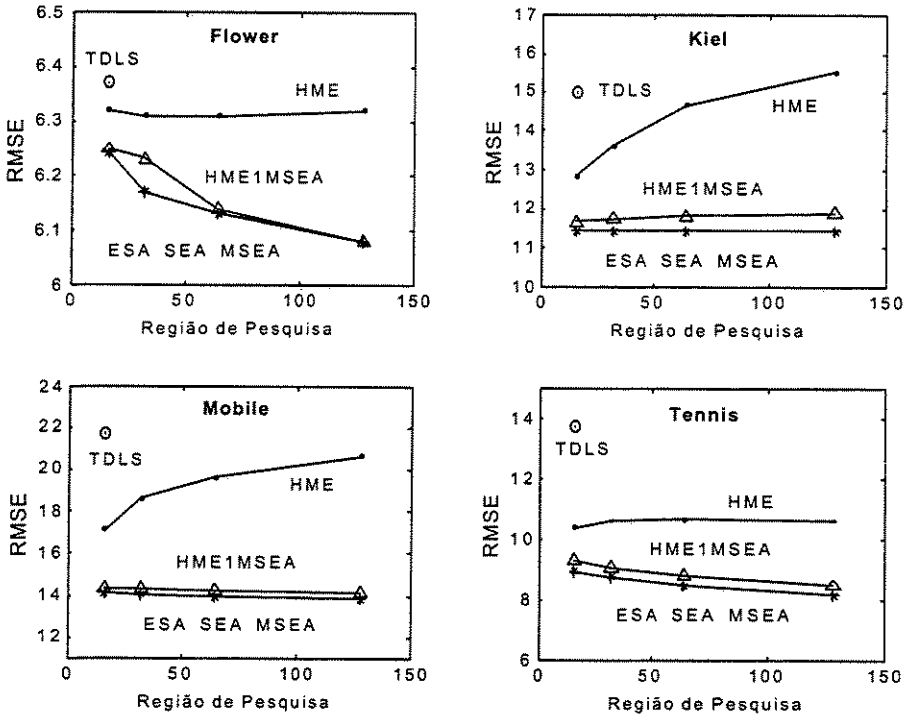


Figura 4.6 – Comparação dos erros de predição do algoritmo HME1/MSEA com os outros algoritmos de estimação.

Até o momento, verificamos que a eficácia do algoritmo MSEA é baseada na aplicação dos critérios de descarte e na pré-ordenação dos macroblocos que compõem a região de pesquisa. Supõe-se que tanto os coeficientes utilizados nos critérios de descarte quanto a pré-ordenação já estejam disponíveis quando se inicia a estimação. O cálculo e a pré-ordenação destes coeficientes são efetuados em uma etapa de pré-processamento, representando um custo computacional adicional (*overhead*) no tempo total de processamento (cf. Tabela 4.11). Analisa-se agora, a complexidade espacial dos algoritmos, ou seja, a quantidade de memória extra necessária para armazenar estes coeficientes pré-processados. Supõem-se os seguintes parâmetros:

- Quadro de referência composto de $W \times H$ pixels.

- Macroblocos de $N \times N$ pixels.
- Cada pixel representado por 8 bits = 1 byte.
- Cada inteiro representado por 16 bits = 2 bytes.
- Desconsidera-se a memória necessária para armazenar o quadro atual e do quadro de referência, pois são comuns a todos os métodos de estimação.

Tem-se, então:

1) Algoritmo ESA e TDLS

Não há armazenamento extra, pois os métodos são aplicados diretamente ao quadro de referência.

2) Algoritmo HME

- Armazenamentos dos quadros amostrados no primeiro e segundo nível
  - Número de pixels no quadro amostrado de primeiro nível $=(W \times H)/4$ pixels.
  - Número de pixels no quadro amostrado de segundo nível $=(W \times H)/16$ pixels.
- Total de memória utilizada $= [(W \times H)/4 + (W \times H)/16] = 5 \times (W \times H)/2$ bytes

3. Algoritmo SEA

- Cálculo do número de elementos $\|R(i,j)\|$ no quadro de referência = número de macroblocos no quadro de referência = $(W - N + 1) \times (H - N + 1)$ inteiros
- Número de elementos $\|C\|$ calculados no quadro atual = número de macroblocos estimados no quadro atual = $(W \times H)/(N \times N)$ inteiros
- Total de memória utilizada = $2 \times [(W - N + 1) \times (H - N + 1) + (W \times H)/(N \times N)]$ bytes

4. Algoritmo MSEA sem ordenação

- Cálculo dos elementos $b_{(\cdot,\cdot)}^{(P)}$ utilizados nos critérios de descarte
  - Primeiro critério (P=16) = $(W - N + 1) \times (H - N + 1)$ inteiros.
  - Segundo critério (P=8) = $(W - N + 1) \times (H - N + 1)$ inteiros.
  - Terceiro critério (P=4) = $(W - N + 1) \times (H - N + 1)$ inteiros.
  - Quarto critério (P=2)= $(W - N + 1) \times (H - N + 1)$ inteiros.
- Cálculo dos elementos $a_{(\cdot,\cdot)}^{(P)}$ utilizados nos critérios de descarte

- Primeiro critério (P=16) = $(W \times H)/(N \times N)$ inteiros.
- Segundo critério (P=8) = $(W \times H)/(N \times N)$ inteiros.
- Terceiro critério (P=4) = $(W \times H)/(N \times N)$ inteiros.
- Quarto critério (P=2)= $(W \times H)/(N \times N)$ inteiros.

o Total de memória gasta $=4 \times [(W-N+1) \times (H-N+1) + (W \times H)/(N \times N)]$ inteiros, com $8 \times [(W-N+1) \times (H-N+1) + (W \times H)/(N \times N)]$ bytes.

5. Algoritmo MSEA ordenação

o A ordenação dos coeficientes $b_{(.,.)}^{(16)}$ requer, além dos $(W-N+1) \times (H-N+1)$ inteiros, o armazenamento das coordenadas (x,y) de cada macrobloco do quadro de referência, num total de $2 \times (W-N+1) \times (H-N+1)$ elementos a mais que o requerido pelo algoritmo MSEA sem ordenação.

o Total de memória $=2 \times [6 \times (W-N+1) \times (H-N+1) + 4 \times (W \times H)/(N \times N)]$ bytes.

Particularizando os cálculos para W=720 pixels, H = 480 pixels e N=16 pixels, temos a seguinte tabela:

Memória Extra Utilizada nos Algoritmos de Estimação (Mbytes)					
ESA	**SEA**	**MSEA (sem ordenação)**	**MSEA (com ordenação)**	**TDLS**	**HME**
0	0,658	2,633	3,945	0	0,864

Tabela 4.19 –Memória extra necessária a execução dos algoritmos de estimação.

Pela Tabela 4.19, observa-se, para os algoritmos de solução ótima, que um aumento da velocidade de processamento causa um aumento da quantidade de memória utilizada.

4.4 - ESTIMAÇÃO EMPREGANDO SEQÜÊNCIAS DE VÍDEO

Analisa-se o desempenho do algoritmo MSEA quando o mesmo é utilizado no processamento de seqüências de vídeo. Utiliza-se o codificador MPEG descrito no Capítulo 2 com as condições de simulação (seqüências de vídeo, GOP, tabelas de codificação, matrizes de quantização, etc.) definidas na Seção 4.2. A análise segue os seguintes passos:

1) Análise do codificador proposto, onde se ajustam os parâmetros do codificador de modo a tornar a taxa de bits e a qualidade da seqüência de vídeo processada

compatíveis com os valores utilizados na prática. Os ajustes permitem a comparação mais realista entre os diversos algoritmos de estimação.

2) Simulações com região de pesquisa compreendendo todo o quadro de pesquisa, de modo a estudar o desempenho do algoritmo MSEA frente aos algoritmos ESA e SEA. Deste modo, analisa-se o algoritmo MSEA em termos da velocidade e da qualidade do processamento na busca do menor erro de estimação.
3) Simulações com região de pesquisa limitada, onde o desempenho do algoritmo MSEA é comparado ao desempenho de outros algoritmos (TDLS, HME, etc.) que primam pela velocidade em detrimento da qualidade da estimação. Analisa-se, assim, a relação entre a qualidade × taxa de bits × velocidade do algoritmo MSEA frente aos outros algoritmos e os parâmetros que tornam um algoritmo superior ao outro.

4.4.1 - ANÁLISE DO CODIFICADOR IMPLEMENTADO

Ajusta-se o codificador, de modo a tornar os resultados da codificação compatíveis com valores usuais. Um dos ajustes relevantes refere-se ao *limiar de erro*, parâmetro que é utilizado para decidir se um macrobloco pertencente a um quadro tipo B ou P tem a necessidade de ser codificado como um macrobloco do tipo I, caso o erro de predição na codificação do macrobloco tipo P ou B supere o valor deste limiar de erro. Um macrobloco que possui um erro de predição alto, apresenta uma codificação de baixa eficiência, aumentando o número de bits gerados. Este número de bits somado aos bits gastos na codificação dos vetores de movimento pode resultar em uma codificação menos eficiente da obtida caso se codificasse o macrobloco como do tipo I (sem o vetor de movimento associado).

No processo de ajuste dos parâmetros do codificador, para as seqüências utilizadas, obteve-se um valor de *limiar de erro* infinito, ou seja, foi sempre mais vantajoso em termos do número de bits gerados, codificar para os quadros tipo P e B os erros de predição dos macroblocos e seus vetores de movimento associados ao invés de codificá-los como do tipo I. Cabe ressaltar que o valor deste limiar é um resultado particular correspondentes às seqüências analisadas. Na Figura 4.7 e na Figura 4.8 pode-se avaliar o desempenho do codificador em termos da taxa de bit (R) e da qualidade dos quadros processados (PSNR) para as quatro seqüências processadas, onde utilizou-se o método ESA como o algoritmo de

estimação de movimento com todo o quadro como região de pesquisa. Como se observa, a estimação de movimento melhora o desempenho do codificador à medida que reduz a taxa de bit da codificação (quadros tipo P e B) e melhora a qualidade dos quadros estimados, já que a quantização, geradora dos erros de codificação, atua nos erros de predição ao invés de atuar diretamente nos pixels dos quadros, como o faz na codificação tipo I.

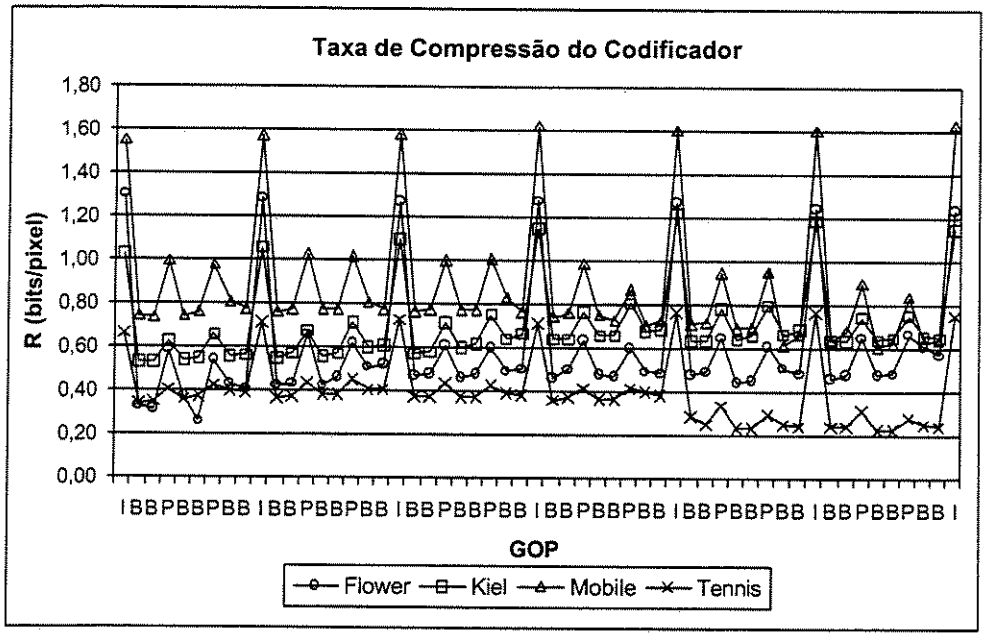


Figura 4.7 – Taxa de bits das seqüências de vídeo "Flower", "Kiel", "Mobile" e "Tennis" processadas pelo codificador MPEG-2 implementado.

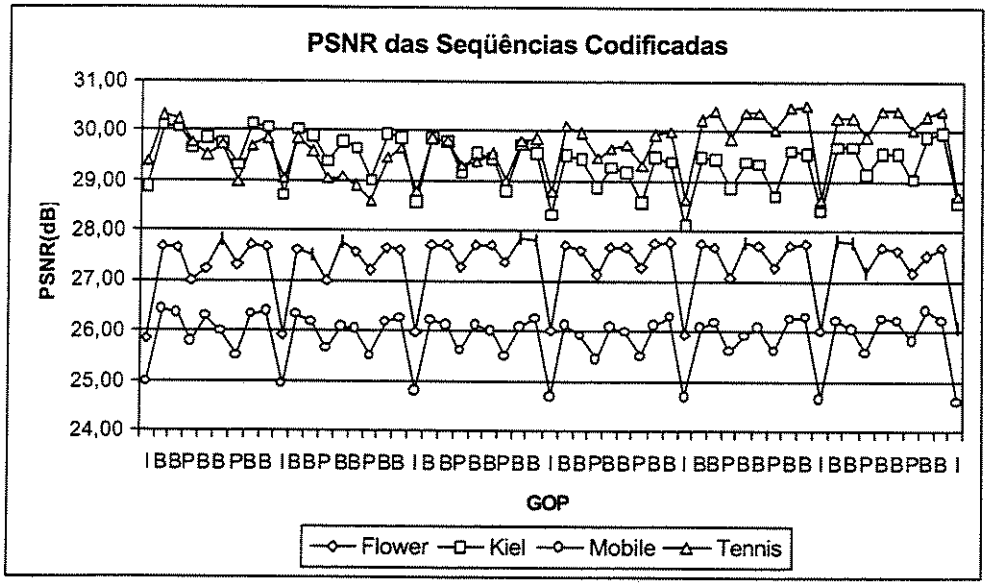


Figura 4.8 – PSNR das seqüências "Flower", "Mobile", "Kiel"e "Tennis" codificadas.

Nota-se, pelos gráficos, um desempenho compatível com os codificadores usualmente utilizados, sendo, portanto, adequada para os testes dos algoritmos de estimação de movimento.

4.4.2 - Resultados com Região de Pesquisa Compreendendo Todo o Quadro de Referência

Nestas simulações são comparados os desempenhos dos algoritmos MSEA, ESA e SEA. Optou-se por comparar estes algoritmos já que todos resultam na melhor estimativa do vetor de movimento global quando a região de pesquisa abrange todo o quadro de referência. Na Tabela 4.20, tem-se o tempo de processamento gasto na codificação de cada seqüência de vídeo para o primeiro GOP considerando os vários algoritmos de estimação. Como pode ser verificado, o algoritmo ESA possui o maior tempo de processamento, pois a busca exaustiva é o algoritmo que mais necessita de computações. Com a introdução do algoritmo SEA, o tempo de processamento se reduz por um fator de 5 a 12 vezes (o fator depende da seqüência e do tipo do quadro processado). Com o algoritmo MSEA chega-se a uma redução substancial no tempo de processamento dos quadros (de 17 a 102 vezes em relação ao ESA) o que demonstra a eficiência do algoritmo.

	Tempo de Processamento TP(s)											
	Flower			**Kiel**			**Mobile**			**Tennis**		
GOP	**ESA**	**SEA**	**MSEA**	**ESA**	**SEA**	**MSEA**	**ESA**	**SEA**	**MSEA**	**ESA**	**SEA**	**MSEA**
I	3	3	3	3	3	3	3	3	3	3	3	3
B	4471	385	49	4469	965	133	4467	798	98	4473	816	245
B	4471	379	47	4466	975	129	4462	785	97	4464	808	240
P	2242	238	39	2243	506	70	2235	432	79	2240	419	126
B	4468	384	54	4465	1001	138	4466	775	96	4470	792	248
B	4472	359	44	4467	1009	140	4464	788	102	4469	789	251
P	2235	221	27	2235	520	70	2232	422	57	2240	412	133
B	4476	502	62	4468	1055	149	4461	845	108	4514	795	236
B	4472	499	58	4465	1051	155	4472	832	111	4470	794	238
I	3	3	4	3	3	3	3	3	3	3	3	3
Total	**31313**	**2973**	**387**	**31284**	**7088**	**990**	**31265**	**5683**	**754**	**31346**	**5631**	**1723**

Tabela 4.20 - Tempo gasto pelo codificador no processamento das seqüências de vídeo para os algoritmos de estimação ESA, SEA e MSEA.

A Tabela 4.21 apresenta os ganhos no tempo de processamento normalizados pelo tempo gasto no algoritmo ESA. Observa-se uma redução do tempo de processamento do GOP da ordem de 18 vezes ("Tennis") e de cerca de 81 vezes para a seqüência "Flower"

em relação ao ESA. O erro de processamento para as três seqüências é o mesmo e pode ser avaliados a partir da Figura 4.8.

	Ganho no Tempo de Processamento											
	Flower			Kiel			Mobile			Tennis		
GOP	ESA	SEA	MSEA	ESA	SEA	MSEA	ESA	SEA	MSEA	ESA	SEA	MSEA
I	1	1	1	1	1	1	1	1	1	1	1	1
B	1	12	91	1	5	34	1	6	46	1	5	18
B	1	12	95	1	5	35	1	6	46	1	6	19
P	1	9	57	1	4	32	1	5	28	1	5	18
B	1	12	83	1	4	32	1	6	47	1	6	18
B	1	12	102	1	4	32	1	6	44	1	6	18
P	1	10	83	1	4	32	1	5	39	1	5	17
B	1	9	72	1	4	30	1	5	41	1	6	19
B	1	9	77	1	4	29	1	5	40	1	6	19
I	1	1	1	1	1	1	1	1	1	1	1	1
Média	**1**	**11**	**81**	**1**	**4**	**32**	**1**	**6**	**41**	**1**	**6**	**18**

Tabela 4.21 - Ganho no tempo de processamento das seqüências de vídeo normalizado pelo tempo gasto pelo algoritmo ESA.

4.4.3 - RESULTADOS COM REGIÃO DE PESQUISA LIMITADA

Analisam-se, agora, os desempenhos na codificação de seqüências de vídeo dos vários algoritmos de estimação, utilizando uma região de pesquisa quadrada de lado [-16, +16] pixels, padrão adotado pelo MPEG-2. Utilizam-se os algoritmos ESA, SEA, MSEA, TDLS e HME e verificam-se os seus desempenhos em termos das curvas de taxa × distorção e do tempo médio de processamento das seqüências. O algoritmo MSEA é utilizado na sua versão sem pré-ordenação, pois, como comentado, a necessidade de ordenação dos dados em cada região de pesquisa de lado igual a 32 ao longo do(s) quadro(s) de referência degradaria o desempenho do algoritmo.

Abaixo, são mostrados os resultados das simulações (curvas de PSNR × taxa) para as quatro seqüências e para os vários algoritmos de estimação analisados. As curvas de PSNR × taxa são obtidas tomando-se os valores do PSNR e da taxa de bit a cada incremento linear das matrizes de quantização "intra" e "inter" definidas para o codificador MPEG-2 (vide Apêndice A).

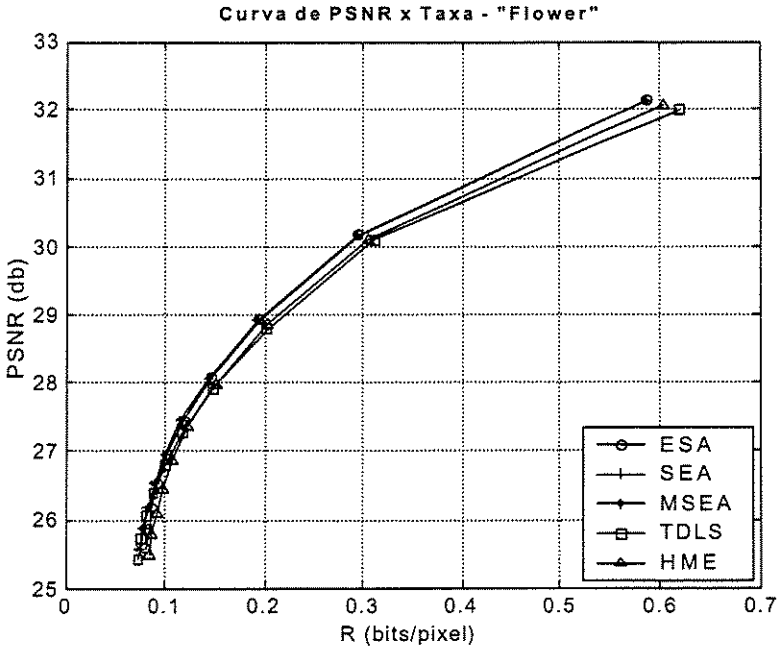


Figura 4.9 – Curva de PSRN × taxa para a seqüência "Flower" utilizando os vários algoritmos de estimação com região de pesquisa com $p = 16$ pixels.

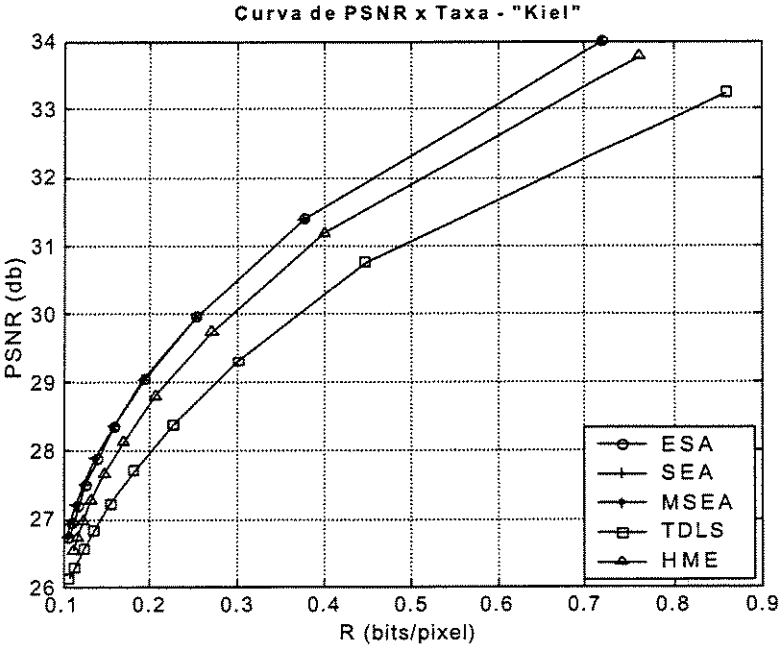


Figura 4.10 – Curva de PSNR × taxa para a seqüência "Kiel" utilizando os vários algoritmos de estimação com região de pesquisa com $p = 16$ pixels.

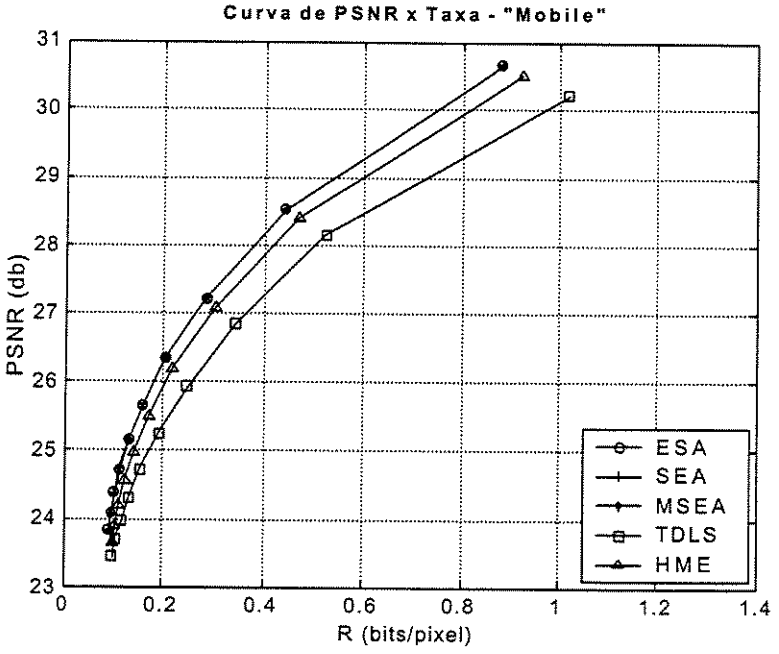


Figura 4.11 – Curva de taxa × distorção para a seqüência "Mobile" utilizando os vários algoritmos de estimação com região de pesquisa com $p = 16$ pixels.

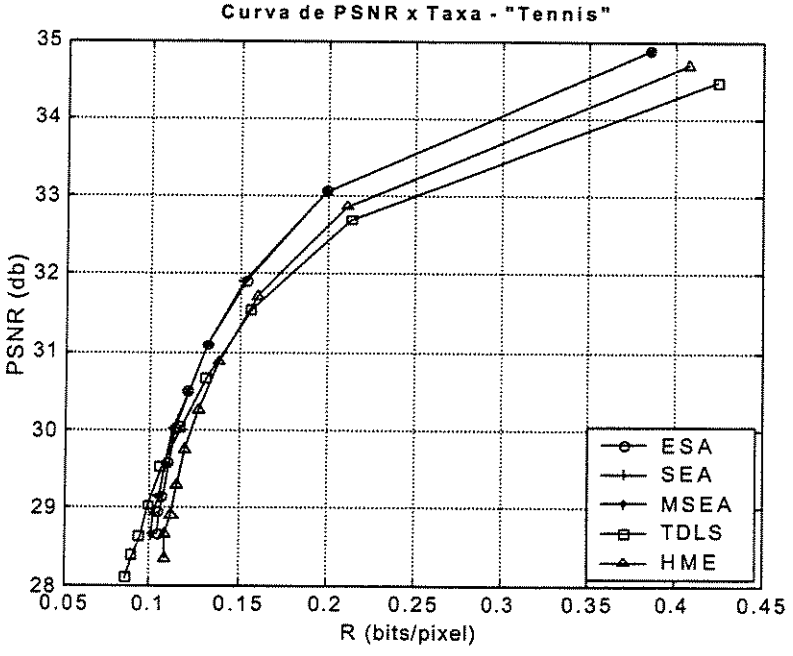


Figura 4.12 – Curva de PSNR × taxa para a seqüência "Tennis" utilizando os vários algoritmos de estimação com região de pesquisa com $p = 16$ pixels.

O melhor algoritmo é aquele que proporciona a menor distorção para uma dada taxa de bit ou alternativamente, uma menor taxa de bits para uma dada distorção. Pelas figuras acima, vê-se que os algoritmos ESA, SEA e MSEA proporcionam estas características, já que suas curvas de PSNR × taxa situam-se "acima" das curvas dos algoritmos HME e TDLS. Por serem algoritmos que resultam no menor erro de predição, têm suas curvas coincidentes em quase toda a faixa da taxa de bits analisada. O algoritmo HME mostrou-se superior ao algoritmo TDLS, resultado já esperado, pois o algoritmo HME varre toda a região de pesquisa (embora dizimada) na procura do vetor de movimento ótimo enquanto o algoritmo TDLS limita-se apenas a alguns macroblocos dentro da mesma região. Operando em taxas altas, os resultados dos algoritmos ESA, SEA e MSEA coincidem, com como é visto nos gráficos. Para baixas taxas, no entanto, os macroblocos sofrem uma forte quantização resultando em macroblocos distintos com erros de predição (MAE) iguais. Assim, os algoritmos de estimação podem selecionar macroblocos distintos com o mesmo MAE, resultando em diferentes codificações e diferentes distorções quadráticas (RMSE). Portanto, em baixas taxas as curvas divergem.

Na Tabela 4.22 têm-se os tempos médios de processamento, obtidos pela média aritmética dos tempos gastos na codificação de cada seqüência para cada incremento das matrizes "intra" e "inter" para os vários algoritmos de estimação estudados.

	Algoritmo				
Seqüência	**TP_{ESA}(s)**	**TP_{SEA}(s)**	**TP_{MSEA}(s)**	**TP_{HME}(s)**	**TP_{TDLS}(s)**
Flower	14,2	9,6	5,4	3,5	3,7
Kiel	13,8	11,0	6,6	3,5	3,7
Mobile	13,7	9,8	5,6	3,6	3,7
Tennis	13,7	10,4	8,4	3,6	3,7

Tabela 4.22– Tempo de processamento médio das seqüências de vídeo pelo codificador MPEG-2 utilizando vários algoritmos de estimação.

Embora o tempo de processamento do algoritmo MSEA seja menor que o tempo de processamento dos algoritmos ESA e SEA (para as mesmas taxas de bits e para o mesmo nível de qualidade), apresenta-se cerca de uma vez e meia a duas vezes e meia superior aos tempos obtidos pelos algoritmos HME e TDLS. Como as curvas de PSNR × taxa apontam para uma superioridade do algoritmo MSEA frente aos algoritmos HME e TDLS, uma solução de compromisso deve ser procurada para cada aplicação em particular. O algoritmo

TDLS pode ser descartado da análise, já que apresenta tempo de processamento compatível ao HME e curva de PSNR × taxa menos favorável.

Como alternativa ao maior tempo de processamento do algoritmo MSEA pode-se incorporar o algoritmo MSEA ao algoritmo HME, operando agora com um só nível de subamostragem (vide Seção 42.3). Os resultados, em termos das curvas de PSNR × taxa e em termos do tempo de processamento são vistos nas figuras abaixo e na Tabela 4.23.

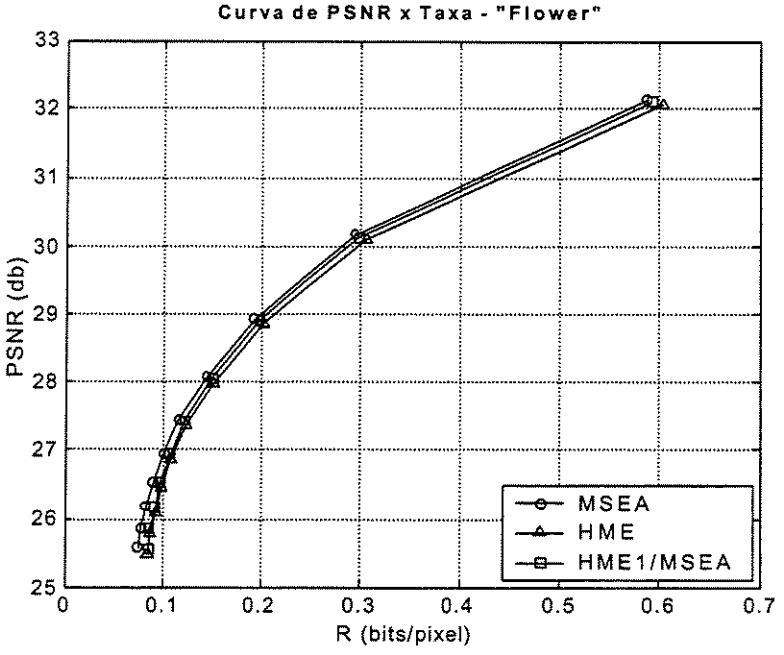


Figura 4.13 – Curvas de PSNR × taxa para os algoritmos MSEA, HME e HME1/MSEA para a seqüência "Flower" com região de pesquisa com $p = 16$ pixels.

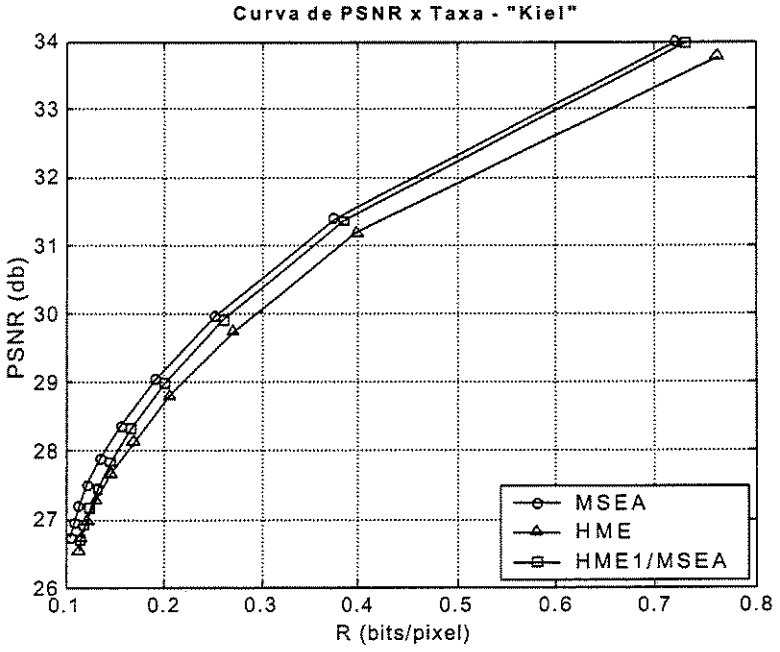


Figura 4.14 – Curvas de PSNR × taxa para os algoritmos MSEA, HME e HME1/MSEA para a seqüência "Kiel" com região de pesquisa com $p = 16$ pixels.

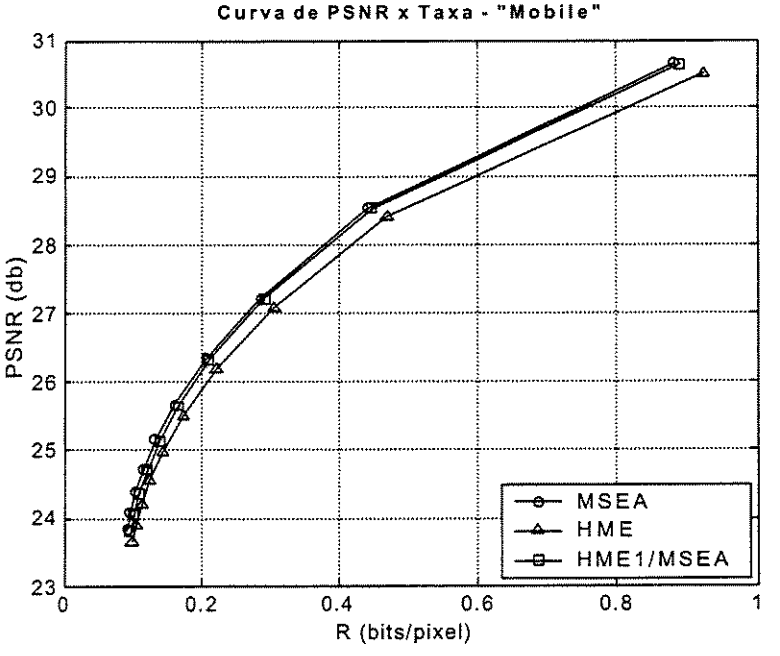


Figura 4.15 – Curvas de PSNR × taxa para os algoritmos MSEA, HME e HME1/MSEA para a seqüência "Mobile" com região de pesquisa com $p = 16$ pixels.

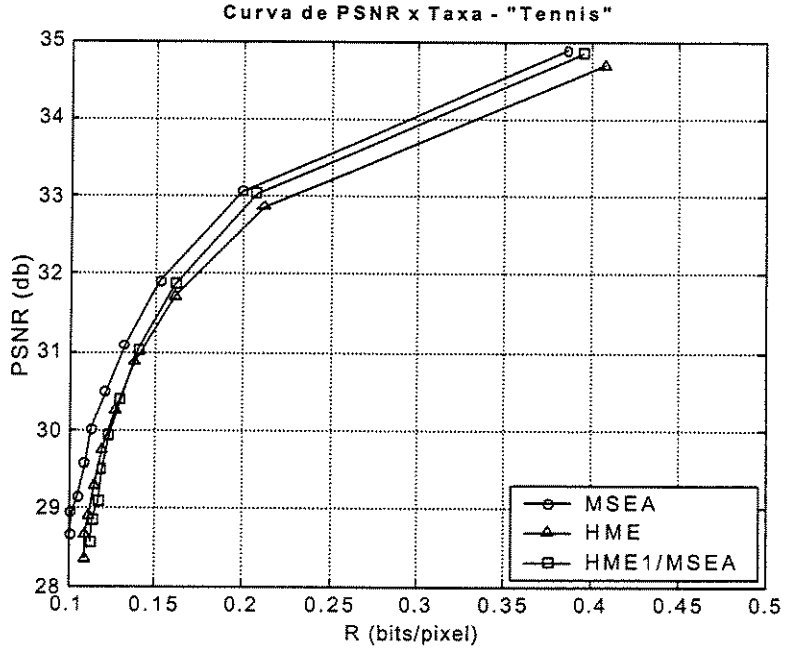


Figura 4.16 – Curvas de PSNR × taxa para os algoritmos MSEA, HME e HME1/MSEA para a seqüência "Tennis" com região de pesquisa com $p = 16$ pixels.

	Algoritmos		
Seqüência	**TP_{MSEA}(s)**	**TP_{HME}(s)**	**$TP_{HME1/MSEA}$(s)**
Flower	5,4	3,5	3,9
Kiel	6,6	3,5	3,9
Mobile	5,6	3,6	3,9
Tennis	8,4	3,6	4,1

Tabela 4.23 – Comparação entre o tempo de processamento médio das seqüências de vídeo utilizando os algoritmo MSEA, HME e HME1/MSEA.

Como se observa das simulações, a utilização conjunta do algoritmo HME (com um nível de subamostragem) e do algoritmo MSEA (HME1/MSEA) resultam em um aumento da velocidade de processamento com relação ao algoritmo MSEA (tornando-o praticamente tão rápido como o algoritmo HME) e em uma melhora na relação sinal/ruído comparada com os valores obtidos pelo algoritmo HME, corrigindo as deficiências de cada um dos algoritmos.

Abaixo são mostradas curvas da PSNR obtidas para as seqüências de vídeo processadas segundo os vários algoritmos de estimação.

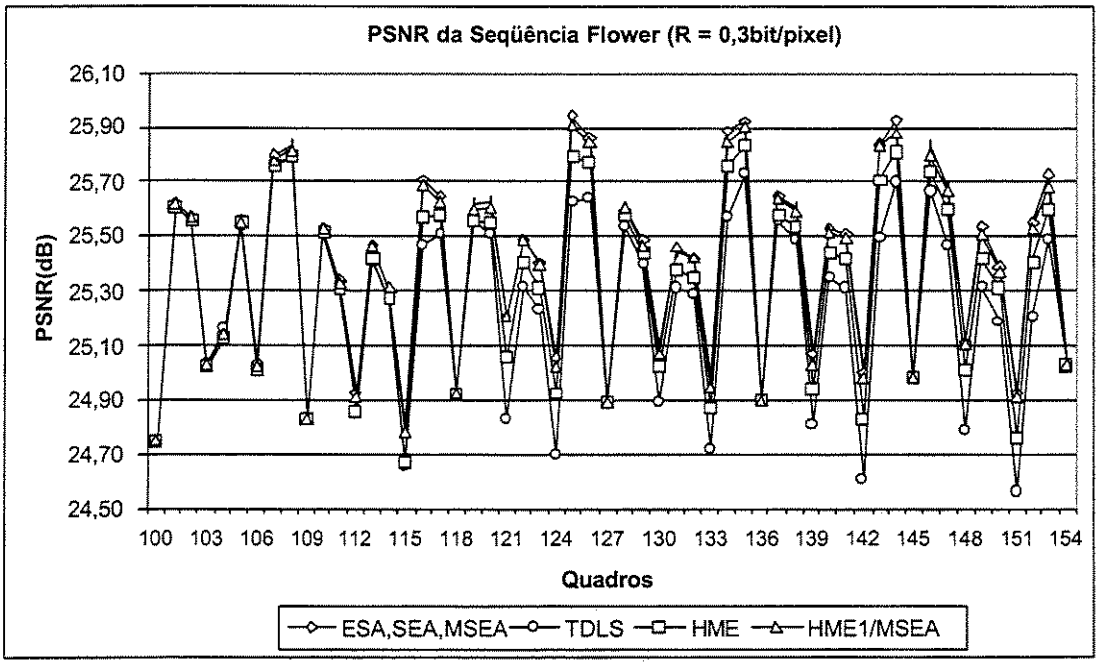


Figura 4.17 – PSNR da seqüência FLOWER processada pelos diversos algoritmos de estimação, para uma região de pesquisa limitada em [-16,+16] pixels e com taxa de compressão R=0,3bit/pixel.

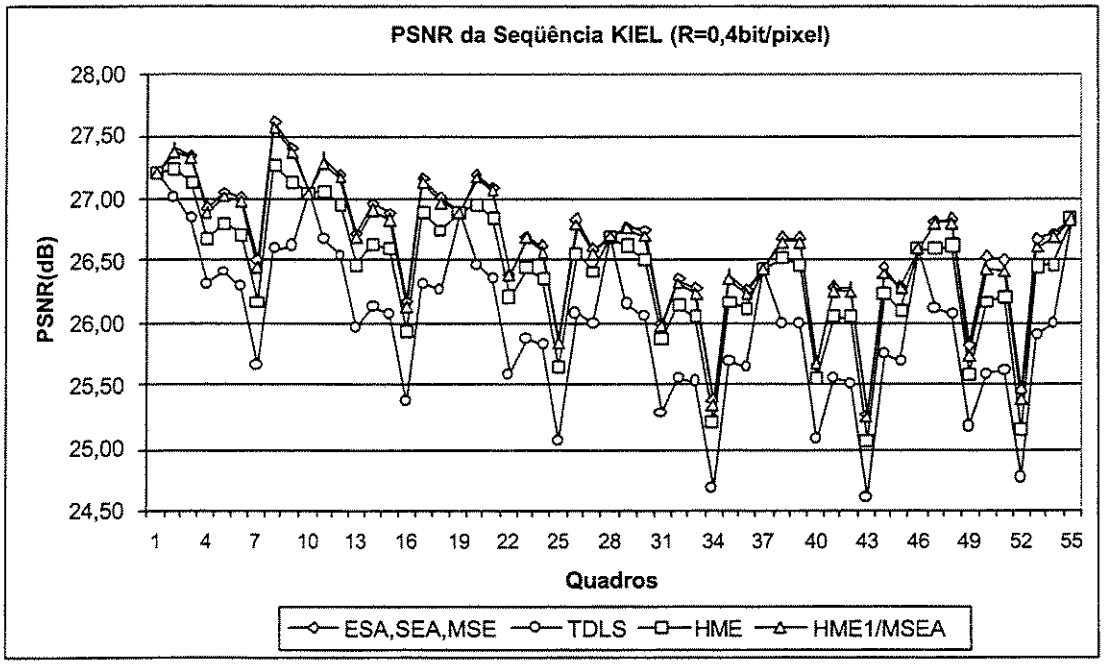


Figura 4.18 – PSNR da seqüência KIEL processada pelos diversos algoritmos de estimação, para uma região de pesquisa limitada em [-16,+16] pixels e com taxa de compressão R=0,4bit/pixel.

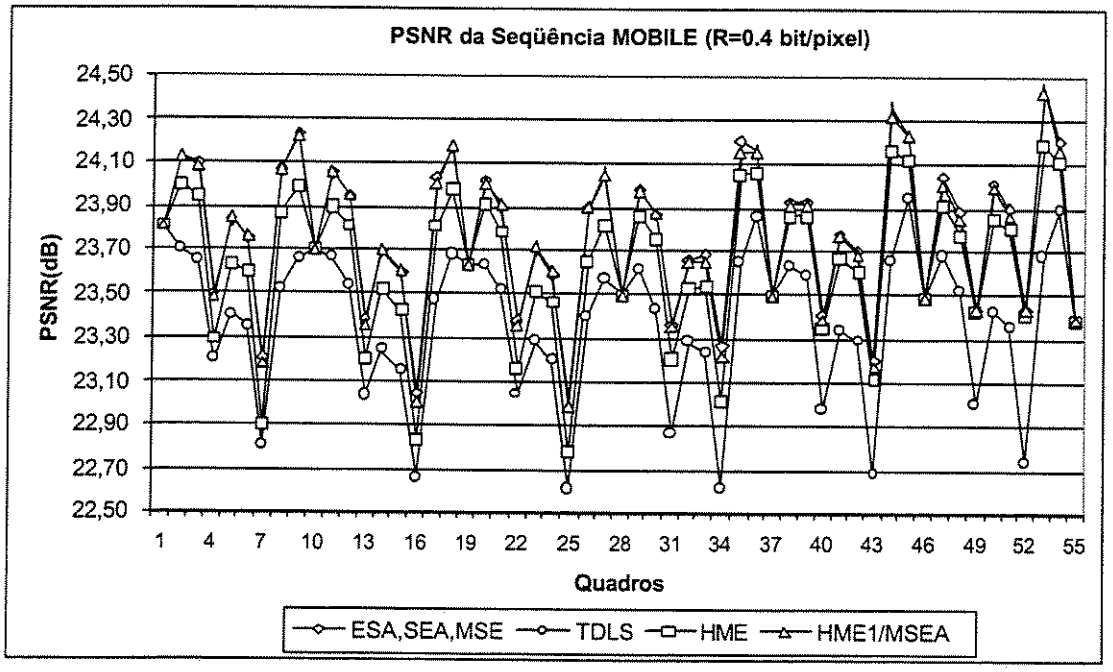


Figura 4.19 – PSNR da seqüência MOBILE processada pelos diversos algoritmos de estimação, para uma região de pesquisa limitada em [-16,+16] pixel e com taxa de compressão R=0,4bit/pixel.

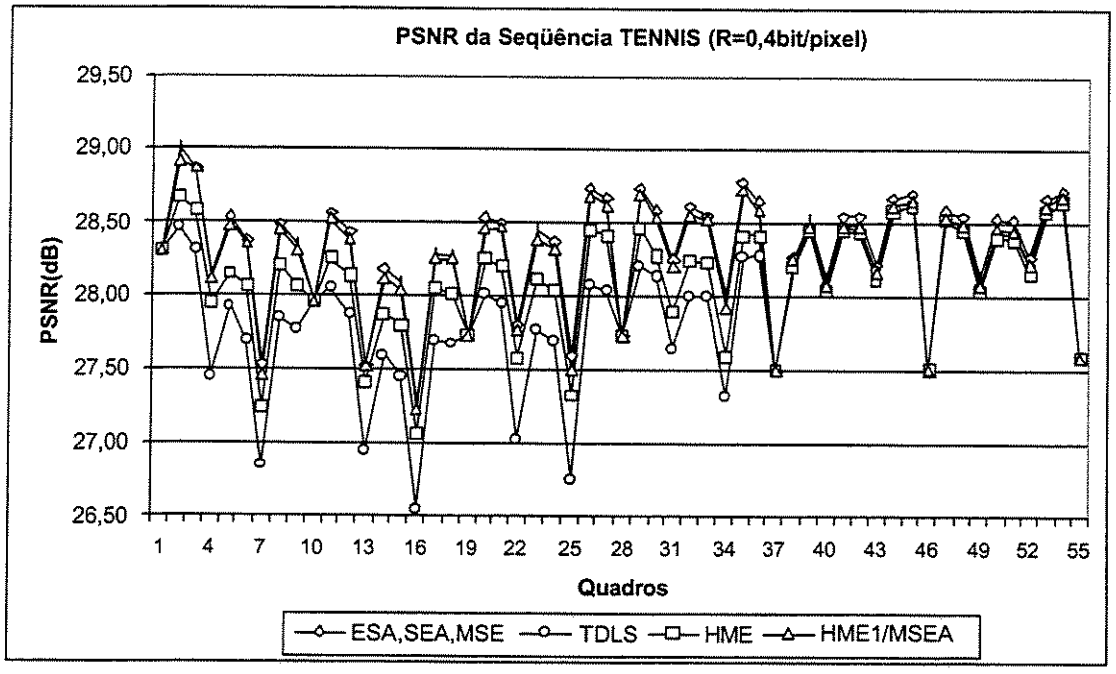


Figura 4.20 – PSNR da seqüência TENNIS processada pelos diversos algoritmos de estimação, para uma região de pesquisa limitada em [-16,+16] pixels e com taxa de compressão R=0,4bit/pixel.

Capítulo 5

CONCLUSÕES

5.1 - SUMÁRIO

O processo de estimação e compensação de movimento é uma etapa de grande importância no processo de codificação de seqüências de vídeo, pois reduz o número de *bits* necessário ao armazenamento ou transmissão da informação para uma dada qualidade do sinal processado. Em contrapartida, exige um alto custo computacional [30-31], tornando o processo de codificação lento. Motivados por este problema, estudam-se vários algoritmos de estimação [2-12][23-25][29-33] compreendendo aqueles que produzem a melhor qualidade de estimação possível (ESA e SEA) e aqueles que são mais rápidos em termos computacionais (HME e TSLS). Propõe-se um novo algoritmo, denominado *Modified* SEA, ou MSEA, que concilia essas duas características, na tentativa de obter o "melhor dos dois mundos". Sendo derivado do algoritmo SEA, o MSEA melhora seu desempenho através de novos critérios de descarte e de uma pré-ordenação e leitura ordenada dos macroblocos dentro da região de pesquisa de uma forma conveniente, provendo rapidez ao processo de estimação ao mesmo tempo em que conserva a solução ótima. Os critérios de descarte são baseados em desigualdades derivadas da função de casamento MAE. Sendo alinhados em ordem crescente de complexidade computacional, os critérios de descarte atuam pré-selecionando os macroblocos candidatos à estimação ótima de movimento, dispensando o cálculo do MAE nesta etapa de pré-seleção. Como o cálculo do MAE apresenta um alto custo computacional, consegue-se uma redução significativa do tempo de processamento. Definindo uma ordem adequada pela qual os macroblocos são pesquisados, pode-se aumentar a probabilidade da obtenção de macroblocos com valores de MAE pequenos, já que o menor valor de MAE obtido no processamento serve como ponto de comparação para a eliminação, pelos critérios de descarte, de macroblocos irrelevantes.

Resultados das simulações, comparando os diversos algoritmos, regiões de pesquisa variáveis, estimação com quadros estáticos e seqüências de vídeo mostram uma sensível redução do tempo de processamento e do número de operações do algoritmo MSEA frente aos algoritmos ESA e SEA, mantendo-se ainda a solução ótima dos vetores de movimento

(mínimo MAE). Com relação aos algoritmos HME e TDLS, o algoritmo MSEA melhora a qualidade do processamento, embora apresente um custo computacional superior decorrente do maior conjunto de macroblocos analisados na região de pesquisa. A vantagem computacional do algoritmo HME decresce com o aumento da região de pesquisa, como se verifica pelo crescimento do tempo de processamento e do número de operações com relação ao aumento das regiões de pesquisa. Assim, para regiões de pesquisa maiores, o MSEA pode apresentar vantagens em relação ao HME, podendo ser indicado para a pesquisa de vetores de movimento em seqüências com rápida movimentação dos objetos de cena, onde se utilizam grandes regiões de pesquisa no processo de estimação.

Por ser um algoritmo de casamento de padrões, o algoritmo MSEA pode ser fundido ao algoritmo HME, provendo qualidade e rapidez de processamento. Simulações demonstram que este novo algoritmo apresenta uma curva de PSNR × taxa próxima da curva ótima e um tempo de processamento compatível com os obtidos pelos algoritmos HME e TDLS. Tem-se, então, no MSEA e em suas combinações com outros algoritmos, uma alternativa interessante para o compromisso entre a qualidade de processamento e o custo computacional no processo de estimação de movimento em compressão de vídeo.

5.2 - CONTRIBUIÇÕES

As principias contribuições deste trabalho de doutoramento se situam na área de estimação e compensação de movimento, e são detalhadas a seguir:

- Um novo algoritmo de estimação denominado MSEA (SEA modificado). A idéia principal foi alterar o algoritmo SEA, introduzindo novos critérios de descarte, uma pré-ordenação e leitura ordenada dos macroblocos no processo de estimação. Este novo algoritmo apresenta as seguintes características:
  - Mantém a solução ótima do vetor de movimento na região de pesquisa.
  - Reduz o custo computacional do processamento:
    - Em até 202 vezes no tempo de processamento, em comparação com a busca exaustiva (ESA), para quadros estáticos;
    - Em até 543 vezes no número de operações aritméticas, em comparação com a busca exaustiva (ESA) para quadros estáticos;

- Em até 81 vezes no tempo de processamento de seqüências de vídeo.

- Possibilidade de combinação com outros algoritmos de estimação, melhorando o desempenho. A incorporação do algoritmo MSEA ao algoritmo HME resulta em um algoritmo com tempo de processamento próximo ao HME (menor tempo dos algoritmos analisados) e em uma curva de PSNR × taxa próxima à obtida pelo algoritmo ESA (melhor curva dos algoritmos analisados).

Observa-se que o algoritmo MSEA tem o seu desempenho melhorado para grandes regiões de pesquisa. Assim, o algoritmo é uma boa alternativa para estimação de movimento em seqüências de vídeo com movimentação rápida dos objetos de cena.

o Estudo e comparação de várias técnicas de estimação e compensação de movimento, levantando suas principais características e limitações. O estudo forneceu embasamento para a proposição do algoritmo de estimação MSE.

Estas contribuições estão parcialmente documentadas nos artigos [13-14].

5.3 - TRABALHOS FUTUROS

Algumas idéias que podem servir para um aperfeiçoamento do método proposto são as seguintes.

- Pesquisar outros critérios de descarte para a eliminação dos macroblocos antes de efetuar o cálculo do MAE. Como os critérios de descarte afetam a velocidade do processamento do algoritmo de estimação e o cálculo do *overhead*, uma escolha mais eficiente destes critérios poderia aumentar a eficiência do algoritmo.
- Para regiões limitadas de pesquisa, a pré-ordenação dos macroblocos causa muito processamento, embora localmente acelere a busca do vetor de movimento ótimo. Um método que efetuasse esta ordenação na etapa de pré-processamento aumentaria o desempenho do algoritmo para pequenas regiões de pesquisa.
- Utilizar a qualidade subjetiva como critério para a estimação e compensação de movimento.
- Utilizar deslocamento de fração dos intervalos de pixels, assim como rotações dos macroblocos do quadro de referência, na busca do melhor casamento para estimação de movimento.

Em conclusão, menciona-se a possibilidade de utilizar o algoritmo MSEA em aplicações de quantização vetorial [34], visto que se trata do mesmo problema computacional de encontrar o vetor mais próximo, dentre os vetores de uma lista (código), segundo um dado critério de distância.

Apêndice A

INFORMAÇÕES AUXILIARES

A.1 - MATRIZES DE QUANTIZAÇÃO DO CODIFICADOR MPEG-2 IMPLEMENTADO [16]

Mpeg_Intra

$$MPEG_INTRA = \begin{bmatrix} 8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 28 & 29 & 34 & 37 \\ 19 & 22 & 16 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{bmatrix}$$

Mpeg_Inter

$$MPEG_INTER = \begin{bmatrix} 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \end{bmatrix}$$

A.2 - FILTROS UTILIZADOS NA CONVERSÃO RGB ↔ YCRCB

- **Filtro Horizontal**

$$FH = [1 \quad 2 \quad 1]$$

- **Filtro Vertical**

$$FV = [1 \quad 2 \quad 1]$$

- **Filtro de Reconstrução Horizontal**

$$FRH = [5 \quad 11 \quad -21 \quad -37 \quad 70 \quad 228 \quad 228 \quad 70 \quad -37 \quad -21 \quad 11 \quad 5]$$

- **Filtro de Reconstrução Vertical**

$$FRV = [3 \quad 7 \quad -16 \quad -67 \quad 227 \quad 227 \quad 67 \quad -32 \quad -16 \quad -21 \quad 7 \quad 3]$$

A.3 - TABELAS DE CODIFICAÇÃO DOS COEFICIENTES DC [16]

- **Dct_Dc_Size_Luminance**

Size	Length	Code
0	3	100
1	2	00
2	2	01
3	3	101
4	3	110
5	4	1110
6	5	11110
7	6	111110
8	7	1111110
9	8	11111110
10	9	111111110
11	9	111111111

- **Dct_Dc_Size_Chrominance**

Size	Length	Code
0	2	00
1	2	01
2	2	10
3	3	110
4	4	1110
5	5	11110
6	6	111110
7	7	1111110
8	8	11111110
9	9	111111110
10	10	1111111110
11	10	1111111111

A.4 - TABELAS DE CODIFICAÇÃO DOS COEFICIENTES AC [16]

Run	Level	Length	Code
EOB	EOB	2	10
0	1	1	1
0	1	2	11
0	2	4	0100
0	3	5	00101
0	4	7	0000110
0	5	8	00100110
0	6	8	00100001
0	7	10	0000001010
0	8	12	000000011101
0	9	12	000000011000
0	10	12	000000010011
0	11	12	000000010000
0	12	13	0000000011010
0	13	13	0000000011001
0	14	13	0000000011000
0	15	13	0000000010111
0	16	14	00000000011111
0	17	14	00000000011110
0	18	14	00000000011101
0	19	14	00000000011100
0	20	14	00000000011011
0	21	14	00000000011010
0	22	14	00000000011001
0	23	14	00000000011000
0	24	14	00000000010111
0	25	14	00000000010110
0	26	14	00000000010101
0	27	14	00000000010100
0	28	14	00000000010011
0	29	14	00000000010010
0	30	14	00000000010001
0	31	14	00000000010000
0	32	15	000000000011000
0	33	15	000000000010111
0	34	15	000000000010110
0	35	15	000000000010101
0	36	15	000000000010100
0	37	15	000000000010011
0	38	15	000000000010010
0	39	15	000000000010001
0	40	15	000000000010000
1	1	3	011
1	2	6	000110
1	3	8	00100101
1	4	10	0000001100
1	5	12	000000011011
1	6	13	0000000010110
1	7	13	0000000010101
1	8	15	000000000011111
1	9	15	000000000011110
1	10	15	000000000011101
1	11	15	000000000011100
1	12	15	000000000011011
1	13	15	000000000011010
1	14	15	000000000011001
1	15	15	000000000001001
1	16	16	0000000000010010
1	17	16	0000000000010001
1	18	16	0000000000001000
2	1	4	0101
2	2	7	0000100
2	3	10	0000001011
2	4	12	000000010100
2	5	13	0000000010100
3	1	5	00111
3	2	8	00100100
3	3	12	000000011100
3	4	13	0000000010011
4	1	5	00110
4	2	10	0000001111
4	3	12	000000010010
5	1	6	000111
5	2	10	0000001001
5	3	13	0000000010010

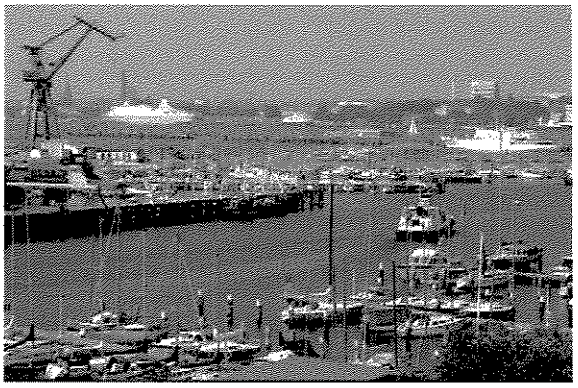
6	1	6	000101
6	2	12	000000011110
6	3	16	0000000000010100
7	1	6	000100
7	2	12	000000010101
8	1	7	0000111
8	2	12	000000010001
9	1	7	0000101
9	2	13	0000000010001
10	1	8	00100111
10	2	13	0000000010000
11	1	8	00100011
11	2	16	0000000000011010
12	1	8	00100010
12	2	16	0000000000011001
13	1	8	00100000
13	2	16	0000000000011000
14	1	10	0000001110
14	2	16	0000000000010111
15	1	10	0000001101
15	2	16	0000000000010110
16	1	10	0000001000
16	2	16	0000000000010101
17	1	12	000000011111
18	1	12	000000011010
19	1	12	000000011001
20	1	12	000000010111
21	1	12	000000010110
22	1	13	0000000011111
23	1	13	0000000011110
24	1	13	0000000011101
25	1	13	0000000011100
26	1	13	0000000011011
27	1	16	0000000000011111
28	1	16	0000000000011110
29	1	16	0000000000011101
30	1	16	0000000000011100
31	1	16	0000000000011011
SKIP	SKIP	6	000001

A.5 - CÓDIGOS PARA OS VETORES DE MOVIMENTO (REGIÕES DE PESQUISA DE [-16, +16] PIXELS) [16]

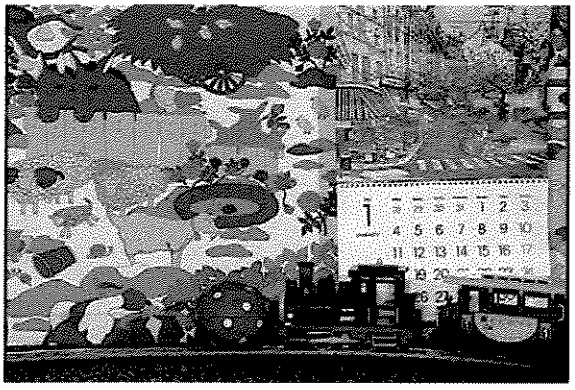
Position	Length	Code
-16	11	00000011001
-15	11	00000011011
-14	11	00000011101
-13	11	00000011111
-12	11	00000100001
-11	11	00000100011
-10	10	0000010011
-9	10	0000010101
-8	10	0000010111
-7	8	00000111
-6	8	00001001
-5	8	00001011
-4	7	0000111
-3	5	00001
-2	4	0011
-1	3	011
0	1	1
1	3	010
2	4	0010
3	5	00010
4	7	0000110
5	8	00001010
6	8	00001000
7	8	00000110
8	10	0000010110
9	10	0000010100
10	10	0000010010
11	11	00000100010
12	11	00000100000
13	11	00000011110
14	11	00000011100
15	11	00000011010
16	11	00000011000

6.6 - Quadro 100 de Cada Seqüência de Vídeo Utilizado pelos Diversos Algoritmos de Estimação

Flower Garden



Kiel



Mobile Calendar

Tennis

REFERÊNCIAS

[1] M. Rabani e P. W. Jones, *Digital Image Compression Techniques*, Bellingham, Washington: SPIE Optical Engineering Press, 1991.

[2] W. Li e E. Salari, "Successive Elimination Algorithm for Motion Estimation", *IEEE Trans. Image Processing*, vol. 4, no.1, pp. 105-107, January 1995.

[3] J.-N. Kim, S.-C. Byun, Y.-H. Kim e B.-H. Ahn, "Fast Full Search Motion Estimation Algorithm Using Early Detection of Impossible Candidate Vectors" *IEEE Trans. of Signal Processing*, vol. 50, no 9, pp. 2355 – 2365, September 2002.

[4] S.-M. Jung, S.-C. Shin, H. Baik e M.-S. Park, "Efficient Multilevel Successive Elimination Algorithms for Block Matching Motion Estimation", *IEE Proc.-Vis. Image Signal Process.*, vol. 149, no. 2, pp. 73 – 84, April 2002.

[5] J.-N. Kim, D.-K. Kang, S.-C. Byun, I.-L. Lee e B.-H. Ahn, "A Fast Full Search Motion Estimation Using Sequential Rejection of Candidates from Hierarchical Decision Structure", *IEEE Trans. of Broadcasting*, vol. 48, no. 1, pp. 43 – 46, March 2002.

[6] J.-N. Kim e B.-H. Ahn, "A Fast Full Search Motion Estimation Algorithm Using Sequential Rejection of Candidates from Multilevel Decision Boundary", *IEICE Trans. Commun.*, vol. E85-B, no. 1, pp. 355 – 358, January 2002.

[7] W.-G. Hong, T.-M. Oh e S.-J. Ko, "Fast Motion Estimation Algorithm Based on Progressive Partial Norm Matching", *Electronics Letters*, vol. 37, no. 15, pp. 890 – 892, 5th July 2001.

[8] B. C. Song, M. J. Kim e J. B. Ra, "A Fast Multiresolution Feature Matching Algorithm for Exhaustive Search in Large Image Databases", *IEEE Trans. on Circ. and Syst. for Video Tech.*, vol. 11, no. 5, pp. 673 – 678, May 2001.

[9] T.-M. Oh, Y.-R. Kim, E.-G. Hong e S.-J. Ko, "Partial Norm Based Search Algorithm for Fast Motion Estimation", *Electronics Letters*, vol. 36, no. 14, pp. 1195 – 1196, 6th July 2000.

[10] X. Q. Gao, C. J. Duanmu e C. R. Zou, "A Multilevel Successive Elimination Algorithm for Block Matching Motion Estimation", *IEEE Trans. on Image Process.*, vol. 9, no. 3, pp. 501 – 504, March 2000.

[11] J.-Y. Lu, K.-S. Wu e J.-C. Lin, "Fast Full Search in Motion Estimation by Hierarchical Use of Minkowski's Inequality (HUMI)", *Pattern Recognitions*, vol. 31, no. 7, pp. 945 - 952, July 1998.

[12] L.-C. Chang, K.-L. Chung e T.-C. Yang, "An Improved Search Algorithm for Motion Estimation Using Adaptive Search Order," *IEEE Signal Process. Letters*, vol. 8, no. 5, pp. 129 –130, May 2001.

[13] R. T. D. Pietro e M. H. M. Costa, "Algoritmo Rápido para a Estimação de Vetores de Movimento", *XIX Simpósio Brasileiro de Telecomunicações – SBrT2001*, 03 a 06 de Setembro de 2001, Fortaleza – CE.

[14] R. T. D. Pietro e M. H. M. Costa, "A Pattern Matching Algorithm Applied to the Estimation of Motion Vectors in Video Compression", *ITS2002 – International Telecommunications Symposium*, 08 a 12 de Setembro de 2002, Natal – RN.

[15] *Preliminary Text for MPEG Video Sandard Coding*, ISO-IEC JTC1/SC2/WG11 ISO, August 1990.

[16] *Information Technology – Generic Coding of Moving Pictures and Associated Audio Information*, ISO/IEC JTC1 IS 13819-2 (MPEG – 2), 1996.

[17] P. Prancha, M. El Zarki, "MPEG Coding for Variable Bit Rate Video Transmission", *IEEE Commun. Mag.*, pp. 54 – 66, May 1994.

[18] T. Sikora, "MPEG Digital Video-Coding Standards", *IEEE Signal Processing Magazine*, pp. 82 – 100, September 1997.

[19] R. T. D. Pietro, "Transmissão Robusta de Vídeo em Pacotes," Dissertação submetida à Faculdade de Engenharia Elétrica e de Computação para a obtenção do título de Mestre em Engenharia elétrica, Agosto 1998.

[20] V. Bhaskaran e K. Konstantinides, *Image and Video Compression Standards*, Boston, MA: Kluwer Academic Publishers, 1997,

[21] L. G. Didier, "MPEG: A Video Compression Standards for Multimedia Applications", *Communications of the ACM*, vol. 34, no. 4, pp. 47-058, April 1991.

[22] R. W. G. Hunt, *The Reproduction of color*, England: Fountain Press, 1987.

[23] C. Stiller e J. Konrad, "Estimating Motion in Image Sequences", *IEEE Signal Processing Magazine*, pp. 70 – 91, July 1999.

[24] F. Dufax e F. Moscheni, "Motion Estimation Techniques for Digital TV: A Review and a New Contribution", *Proceedings of the IEEE*, vol. 83, no. 6, pp. 858 – 876, June 1995.

[25] K. H. K. Chow e M. L. Liou, "Generic Motion Search Algorithm for Video Compression", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, pp. 440-446, December 1993.

[26] A. V. Oppenhein, R. W. Shafer, *Digital Signal Processing*, Englewood Cliffs, NJ: Prentice Hall, 1975.

[27] T. M. Apostol, *Mathematical Analysis*, Reading, MA: Addison-Wesley, 1975.

[28] J. L. Boldrini, S. I. Rodrigues, V. L. Figueiredo e H. G. Wetzler, *Álgebra Linear*, São Paulo, SP: Harper & Row do Brasil, 1980.

[29] J. Feng, K.-T. Lo, H. Mehpour e A.E. Karbowiak, "*Adaptive Block Matching Motion Estimation Algorithm for Video Coding*", *Electronics Letters*, vol. 31, no. 18, pp. 1542 – 1543, August 1995.

[30] P. Kuhn., *Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*, Norwell, MA: Kluwer, 1999.

[31] S.-C. Hsia, "VLSI Implementation for Low-Complexity Full-Search Motion Estimation", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 7, pp. 613 – 619, July 2002.

[32] M. Brünig e W. Niehsen, "Fast Full-Search Block Matching", *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 2, pp. 241 – 246, February, 2001.

[33] H. S. Hang e R. M. Merserau, "Fast Algorithms for the Estimation of Motion Vectors", *IEEE Trans. Image Processing*, vol. 8, no. 3, pp. 435-438, March 1999.

[34] A. Gersho e R. M. Gray, *Vector Quantization and Signal Compression*, Boston, MA: Kluwer, 1991.