Investigação Algorítmica sobre a Estimação de Movimento na Compressão de Vídeo Digital: Uma Análise Quantitativa

Leandro Zanetti P. da Rosa[1], Marcelo S. Porto[1], Fabiane K. Rediess[2], Altamiro A. Susin[1], Sergio Bampi[1], Luciano V. Agostini[2]

[1]Grupo de Microeletrônica (GME)
Universidade Federal do Rio Grande do Sul (UFRGS)

[2]Grupo de Arquiteturas e Circuitos Integrados (GACI)
Universidade Federal de Pelotas (UFPel)

{leandrozanetti.rosa, msporto, bampi}@inf.ufrgs.br,
altamiro.susin@ufrgs.br, {frediess_ifm, agostini}@ufpel.edu.br

Abstract. *This work presents an algorithmic investigation of motion estimation (ME) in digital video compression. This analysis is a solid evaluation of ME algorithms based on different criteria, targeting the best choice to be designed in software or hardware. This choice has a direct impact in the motion vector quality and in the motion estimator performance. Six algorithms and two subsamples techniques were investigated. All algorithms were developed in C and they use SAD as distortion criterion. For each algorithm, three block sizes and four different search area sizes were evaluated. The algorithms were applied to ten video sequences and their average results were considered in the presented evaluations.*

Resumo. *Este artigo apresenta uma investigação algorítmica da estimação de movimento (ME) na compressão de vídeo digital. Essa análise é uma sólida avaliação de algoritmos de ME baseada em diferentes critérios, com o objetivo de escolher o melhor algoritmo para ser implementado em software ou em hardware. Essa escolha tem impacto direto na qualidade do vetor de movimento e no desempenho da ME. Seis algoritmos e duas técnicas de subamostragem foram investigados. Todos os algoritmos foram desenvolvidos na linguagem C e usaram o SAD como critério de distorção. Para cada algoritmo, três tamanhos de bloco e quatro áreas de pesquisa foram avaliados quando aplicados a dez amostras de vídeo, sendo que as médias dos resultados são apresentadas nas avaliações.*

1. Introdução

O sucesso de aplicações que manipulam vídeos digitais depende da compressão de vídeo, pois um vídeo não comprimido utiliza uma quantidade de bits muito elevada para representar cada ponto da imagem. Sem a compressão, essas aplicações seriam inviáveis, em função do elevado custo em termos de armazenamento e transmissão desses vídeos.

Apesar das seqüências digitalizadas de vídeo precisarem de uma grande quantidade de bits para serem representadas, elas têm uma importante característica:

apresentam elevado grau de redundância. O objetivo da compressão de vídeo é a máxima eliminação possível desses dados redundantes, para conseguir representar o vídeo digital com um número de bits muito menor do que o original.

Existem diversos padrões de codificadores de vídeo, como o H.264/AVC [ITU-T 2005]. A Figura 1 apresenta os módulos de um codificador de vídeo atual.

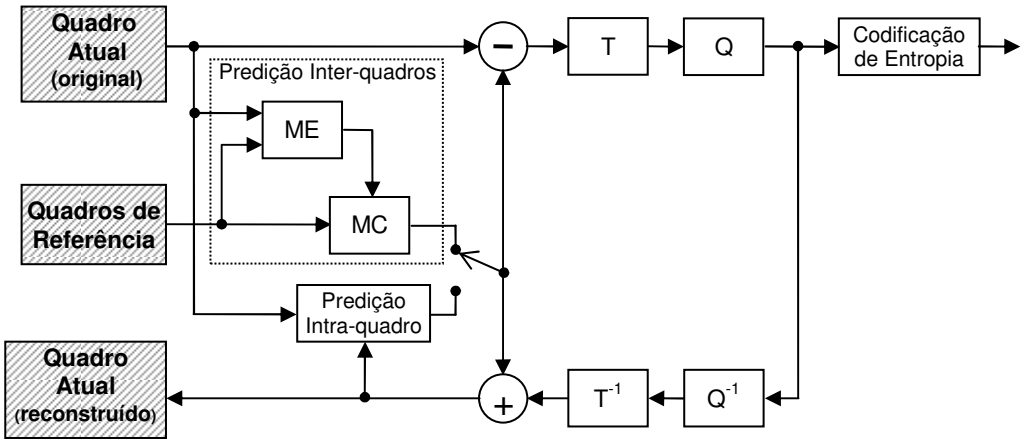

Figura 1. Módulos de um codificador de vídeo atual

Dois ou mais quadros do vídeo são utilizados simultaneamente pelo codificador, sendo um o quadro atual, que será comprimido, e os outros, os quadros de referência que foram anteriormente processados. Os blocos do quadro atual são codificados através da codificação inter-quadros ou através da codificação intra-quadro.

O módulo de codificação intra-quadro é responsável por reduzir a redundância presente dentro do próprio quadro (chamada de redundância espacial), em função da similaridade entre pixels vizinhos. O módulo de codificação inter-quadros é formado pela estimação de movimento (ME – *Motion Estimation*) e pela compensação de movimento (MC - *Motion Compensation*). Esses módulos, em conjunto, são responsáveis por reduzir a redundância temporal, através da comparação dos blocos do quadro atual com os blocos dos quadros de referência. Assim que o bloco mais similar é encontrado, a ME (foco deste trabalho) deve gerar um vetor indicando a posição deste bloco em relação ao bloco que está sendo codificado. Este vetor é chamado de vetor de movimento.

A diferença residual após a codificação intra-quadro ou a inter-quadros é obtida através de uma subtração entre os valores dos blocos do quadro atual e dos valores gerados por essas codificações. Esse resíduo é enviado para os módulos responsáveis por reduzir ainda mais a redundância espacial. A primeira operação nessa direção é a transformada (módulo T), cujo objetivo é transformar a informação do domínio espacial para o domínio das freqüências. Neste domínio, a quantização pode ser aplicada, reduzindo a redundância espacial presente nos resíduos.

A quantização (módulo Q) é uma divisão inteira dos coeficientes gerados pela transformada o que reduz grande parte dos coeficientes a zero. Por fim, a codificação de entropia reduz a redundância entrópica, que está relacionada à forma como os dados são

codificados e com a probabilidade de ocorrência dos símbolos. O codificador descarta o quadro original depois de ser processado e armazena o quadro reconstruído.

No módulo da estimação de movimento, para determinar a semelhança entre os blocos, é usado um critério de similaridade. Neste trabalho, o critério utilizado foi a Soma das Diferenças Absolutas (SAD – *Sum of Absolute Differences*). A equação (1) apresenta esse critério, onde **SAD(x,y)** é o valor do SAD para a posição **(x,y)**, **R** representa a amostra de referência, **P** é a amostra do quadro de referência e **N** é o tamanho do bloco.

$$SAD(x, y) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left| R_{i,j} - P_{i+x, j+y} \right| \qquad (1)$$

Na estimação de movimento, existem diversos algoritmos e heurísticas usados para localizar os blocos mais similares entre os quadros de referência e atual. São esses algoritmos que determinam a forma utilizada para localizar o bloco e gerar o vetor de movimento correspondente a essa escolha. Esses algoritmos podem procurar blocos em todo quadro ou em uma área limitada ao redor do bloco que está sendo codificado. Essa área limitada é conhecida como área de pesquisa.

Neste trabalho, diversos algoritmos publicados na literatura foram analisados com o objetivo de melhor entender suas características e especificidades e de gerar uma comparação entre estes algoritmos de acordo com diversas métricas. A motivação principal para o desenvolvimento desta investigação foi a não existência de uma avaliação completa na literatura sobre algoritmos para a estimação de movimento, pois cada novo algoritmo publicado utiliza parâmetros de comparação que tendem a destacar suas qualidades e não permitem uma avaliação confiável dos resultados. Além disso, a manipulação de grandes volumes de dados multimídia, como é o caso de vídeos digitais, e a respectiva gestão da informação por eles representada, é desafio extremamente atual na academia, sendo, inclusive, destacado como um dos Grandes Desafios da Pesquisa em Computação do Brasil, definidos pela SBC.

Para viabilizar a investigação apresentada neste trabalho, os algoritmos selecionados foram implementados na linguagem C e os resultados gerados a partir das execuções destas implementações em software foram avaliados através de diversas métricas. No total, foram vinte e seis combinações algorítmicas investigadas, com três tamanhos de bloco, quatro diferentes áreas de pesquisa e dez vídeos de entrada. Foram 3.120 execuções, com um tempo total aproximado de 30 mil horas de execução. No total, foram gerados mais de quinze mil arquivos com resultados, em um total aproximado de 50GB de dados. Apenas um extrato destes resultados é apresentado neste artigo devido à limitação de espaço.

Este trabalho está organizado como segue: a Seção 2 apresenta os algoritmos investigados; a Seção 3 mostra os resultados alcançados e a Seção 4 apresenta as conclusões e os trabalhos futuros.

2. Algoritmos de Busca Investigados

A seguir, os seis algoritmos selecionados para esta investigação e as duas técnicas de subamostragem aplicadas nesses algoritmos serão descritos resumidamente.

2.1. Algoritmo *Full Search*

O algoritmo *Full Search* (FS) procura a maior semelhança para o bloco que está sendo processado (bloco atual) comparando-o com todos os blocos candidatos existentes dentro da área de pesquisa do quadro de referência [Bhaskaran e Konstantinides 1997], [Lin e Leou 2005]. Quando todos os blocos candidatos tiverem sido avaliados, o bloco que apresentar o menor valor de SAD será o escolhido. Após a escolha, é gerado um vetor de movimento referente ao deslocamento do bloco atual em relação ao bloco de maior similaridade na área de busca do quadro de referência. Como o FS testa todas as possibilidades, ele é considerado um algoritmo ótimo.

2.2. Algoritmo *Three Step Search*

O principal objetivo do *Three Step Search* (TSS) é reduzir o número de comparações realizadas, em relação ao FS, para se encontrar o bloco de maior similaridade. Para isso, um número finito de comparações é determinado e o algoritmo é dividido em três passos. Inicialmente o SAD é calculado para o centro da área de pesquisa e, em seguida, mais oito valores em torno do centro da área de pesquisa são calculados. Então, esses valores de SAD são comparados com o valor do centro e a posição de menor erro se torna a nova posição de origem. O TSS é mais difundido com o nome de busca em três passos, mas pode ser utilizado para n passos [Jing e Chau 2004].

2.3. Algoritmo *One at a Time Search*

Este algoritmo é considerado simples e visa reduzir ainda mais o número de comparações, baseando-se na idéia de encontrar, primeiro, um mínimo horizontal e, em seguida, um mínimo vertical [Richardson 2002]. O *One at a Time Search* (OT) começa calculando o SAD para a posição central da área de pesquisa e de duas posições horizontais, uma imediatamente à direita e outra imediatamente à esquerda da posição central. A posição de menor SAD é definida como o novo centro e esse passo se repete até que o menor erro seja encontrado na posição central. Depois se inicia a busca vertical. Quando o menor SAD é encontrado no centro, o algoritmo finaliza a busca.

2.4. Algoritmo *Diamond Search*

O algoritmo *Diamond Search* (DS) possui esse nome devido ao formato das posições dos blocos utilizados para as comparações em relação ao centro. O DS possui dois padrões em formato diamante, o *Large Diamond Search Pattern* (LDSP) e o *Small Diamond Search Pattern* (SDSP), que são usados na etapa inicial e final do algoritmo, respectivamente [Yi e Ling 2005]. O padrão LDSP consiste em nove comparações, enquanto o padrão SDSP é formado por quatro comparações [Kuhn 1999]. A etapa inicial é repetida até que o menor SAD seja encontrado no centro do LDSP. Quando isso ocorre, o formato SDSP é aplicado para refinar o resultado.

2.5. Algoritmo *Hexagon Search*

O algoritmo *Hexagon Search* (HS) [Zhu, Lin e Chau 2002] pode ser considerado uma evolução do algoritmo DS. O HS possui dois padrões para realizar as comparações, só que agora em formato de hexágono. O formato *Large Hexagon Pattern* (LHP) é usado na etapa inicial e o *Small Hexagon Pattern* (SHP) para o final. Novamente, a etapa

inicial é repetida até que o centro tenha o menor SAD em relação aos outros 8 blocos comparados, e depois é realizado o refinamento com o formato SHP.

2.6. Algoritmo *Dual Cross Search*

O *Dual Cross Search* (DCS) segue a mesma linha do DS e HS, tendo dois padrões de pesquisa, só que em formato de cruz, o 2x2 *Cross Search Pattern* e o 4x4 *Cross Search Pattern* [Banh e Tan 2004]. Em ambos os padrões, cinco pontos são comparados e a diferença entre os dois é a distância do centro até a extremidade da cruz.

2.7. Técnicas de Subamostragem

Neste trabalho serão apresentadas e utilizadas duas técnicas de subamostragem que foram associadas aos algoritmos de ME. A técnica de subamostragem de pixel (Pel - *Pel Subsampling*), também conhecida como *Pel Decimation*, foi incorporada a todos os algoritmos de ME investigados neste trabalho. O objetivo dessa técnica é reduzir o número de comparações, pois uma parcela dos pixels não é usada nos cálculos [Kuhn 1999]. A cada comparação, o algoritmo calcula o erro do bloco usando apenas alguns pixels do bloco e simplesmente descartando os outros, reduzindo o tempo de processamento. Essa técnica pode ser aplicada em qualquer relação, como 2:1 e 4:1.

A técnica de *Block Subsampling* (Bck) [Korah 2005] pode ser aplicada apenas ao algoritmo *Full Search* e visa acelerar ainda mais o processo de busca. Da mesma forma que a técnica Pel, uma subamostragem é realizada, mas não em pixels e sim em blocos. Nesta técnica, não são comparados todos os blocos candidatos da área de pesquisa, portanto os blocos são subamostrados, sendo que o erro é calculado para alguns blocos e outros são simplesmente descartados. Da mesma forma que para a técnica PS, o Bck pode ser aplicado em diferentes níveis, 2:1 e 4:1, por exemplo.

3. Resultados Obtidos

Para todos os seis algoritmos, foram desenvolvidas versões usando a técnica Pel. Para o algoritmos FS, a técnica foi utilizada nas relações de 2:1 (Pel2:1), 4:1 (Pel4:1), 8:1 em dois padrões diferentes (Pel8:1 e Pel8:1_2pl) e 16:1 (Pel16:1). Os algoritmos rápidos (TSS, OT, DS, HS e DCS), também foram desenvolvidos usando essa técnica, porém apenas utilizando a proporção 2:1.

A técnica Bck foi empregada apenas em duas relações diferentes: 2:1 e 4:1. Essas relações foram implementadas com as cinco relações de Pel aplicadas ao FS e, portanto, as relações de Bck estão sempre associadas às relações de Pel.

Todas as versões foram executadas para quatro diferentes áreas de pesquisa: 46x46, 80x80, 144x144 e 208x208 amostras, totalizando 104 implementações. Essas versões foram executadas com três tamanhos de blocos diferentes: 16x16, 8x8 e 4x4, gerando um total de 312 implementações. Todas as implementações foram executadas para os 100 primeiros quadros de 10 seqüências de vídeos não comprimidos na resolução SDTV (720 x 480 pixels) [VQEG 2007]. A Figura 2 apresenta o primeiro quadro de cada uma das amostras utilizadas, bem como o nome dos vídeos usados neste trabalho e os nomes originais dos mesmos.

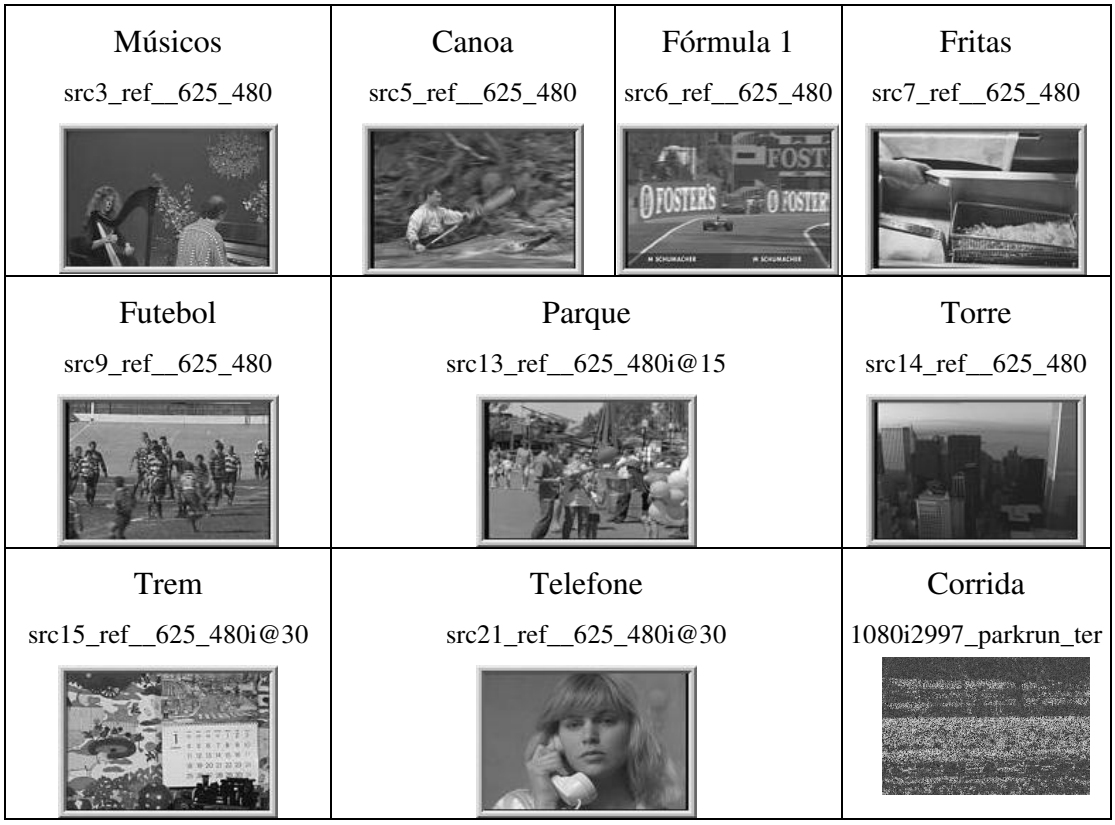

Figura 2. Primeiro quadro das dez amostras de vídeo utilizadas nos testes

Nas próximas seções, em função da limitação de espaço do artigo, apenas alguns dos resultados obtidos serão apresentados para cada métrica de avaliação. Os resultados completos podem ser encontrados em [Rosa 2007].

3.1. Análise do Impacto do Tamanho da Área de Pesquisa

Nesta seção será apresentada uma avaliação do impacto da área de pesquisa no desempenho dos algoritmos. A primeira investigação buscou avaliar a influência do tamanho da área de pesquisa na qualidade dos vetores gerados. Para tanto, alguns experimentos foram realizados utilizando o algoritmo *Full Search*. Nestes experimentos, a área de pesquisa foi definida como o quadro de referência inteiro. Desta forma, todos os vetores encontrados serão os vetores ótimos para esse quadro. Então os vetores gerados foram classificados de acordo com o módulo máximo de suas componentes. Foram definidas 11 diferentes faixas de tamanhos de componentes: 32, 64, 96, 128, 160, 192, 224, 256, 288, 320 e maior que 320. A estimação foi realizada para os 100 primeiros quadros de 5 das 10 seqüências de vídeo apresentadas anteriormente. Dentre as seqüências avaliadas, quatro possuem uma grande quantidade de movimento, o que implica em uma maior probabilidade de vetores com tamanhos grandes de componentes. A última amostra é de um vídeo com pouco movimento, onde a tendência é de que os vetores possuam componentes de módulo menor.

A Figura 3 ilustra a curva percentual do módulo máximo das componentes dos vetores de movimento (MMCVM) para cada uma das amostras de vídeo avaliadas, considerando o tamanho de bloco de 16x16 amostras. Analisando as curvas do gráfico apresentado na Figura 3 é possível perceber que a maior parte dos vetores ótimos

encontra-se nas primeiras faixas. Nos vídeos de maior movimento ("Canoa", "F1", "Fritas" e "Futebol") existe uma oscilação de 60% a 80% dos vetores ótimos na faixa com tamanho de componente de até 32 pixels. Já para o vídeo "Telefone", que possui menor quantidade de movimento, cerca de 95% dos vetores ótimos estão dentro desta primeira faixa. As curvas ainda apresentam um crescimento considerável, para todas as seqüências, até a faixa com tamanho de componente de até 96 pixels. Após esta faixa, todas as curvas começam a saturar, apresentando um crescimento mínimo no percentual de vetores, sendo que para a amostra "Telefone", o crescimento é de menos de 1% a partir da faixa 96. Para esta investigação, mais de 85% dos vetores ótimos foram encontrados em uma área de pesquisa com vetores com componente de até 96 pixels, para a média obtida com as cinco amostras avaliadas.

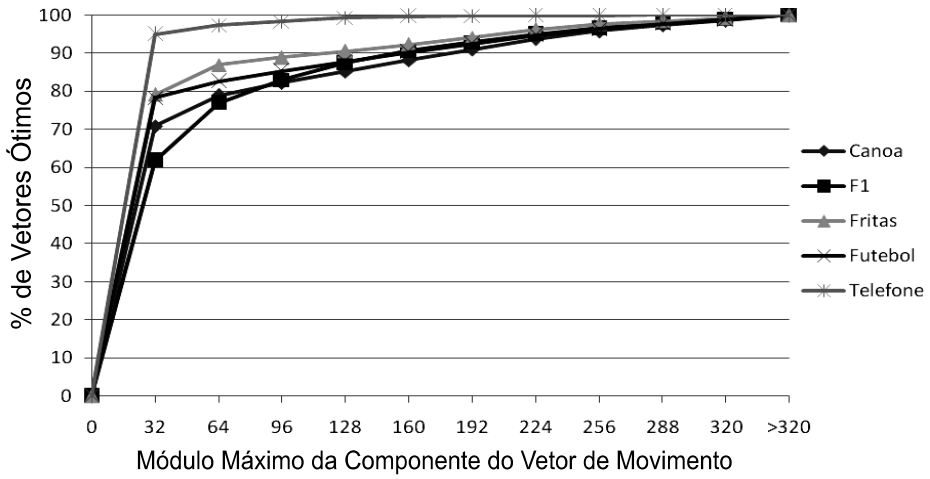

Figura 3. Gráfico com percentual de vetores ótimos por faixa de tamanho de vetor

Uma segunda análise foi realizada no intuito de avaliar a qualidade dos resultados quando a área de pesquisa é restrita a cada uma das faixas apresentadas no gráfico da Figura 3. Esta análise é importante, pois a escolha de um vetor de movimento sub-ótimo pode ter um impacto mínimo no erro total gerado, após a compensação de movimento. Por exemplo, se um vetor em uma área maior foi escolhido por possuir um SAD de 812, a escolha de um vetor em uma área mais restrita pode conduzir a um SAD de 813, com um impacto mínimo na qualidade final da estimação.

Para este experimento, os tamanhos de componente de vetores foram restritos para as mesmas faixas do primeiro experimento, ou seja: 32, 64, 96, 128, 160, 192, 224, 256, 288, 320 e maior que 320. Para definir qual é a área de pesquisa a partir do tamanho do vetor, basta considerar que, para um vetor de tamanho máximo de componente igual a n, existem + ou – n amostras na vertical e na horizontal em torno do bloco atual. Como o bloco possui 16x16 amostras, então a área de pesquisa será definida como ($2n$+16) x ($2n$+16) amostras. Considerando vetores com componentes de até 32 amostras, isto é, com n=32, a área de pesquisa será de 80x80 amostras.

Para analisar a qualidade do processo de estimação, foi utilizado o erro gerado no processo de estimação, conforme gráfico apresentado na Figura 4. O valor de erro é a soma de todos os SADs resultantes do processo de estimação e, quanto menor o valor

do erro, melhor será a estimação obtida. Os valores apresentados no gráfico da Figura 4 para o erro estão em milhões de unidades. Pode-se perceber que existe uma diminuição mais perceptível no erro entre as faixas de tamanhos de componentes dos vetores de 32, 64 e 96. A partir da faixa de 96, a diminuição do erro passa a ser muito pequena, e sua variação em relação ao aumento da faixa de vetores passa a ser imperceptível. Esta pequena redução do erro indica um pequeno impacto na qualidade do processo de estimação de movimento quando a área de pesquisa é reduzida.

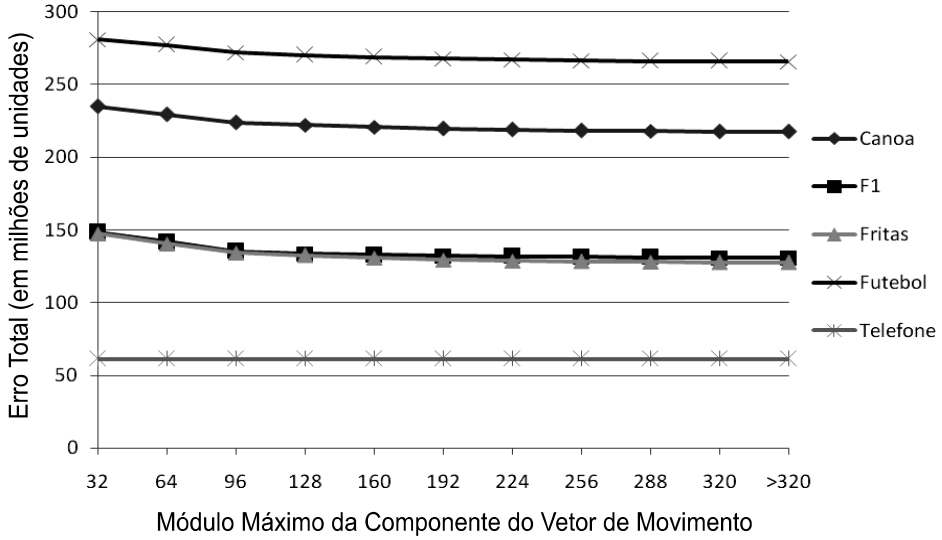

Figura 4. Gráfico com curvas de erro para as amostras avaliadas (em milhões de unidades)

Utilizando o algoritmo *Full Search* para realizar a busca é possível determinar quantos cálculos de SAD são necessários para cada área de pesquisa considerada. Conhecendo as dimensões da área de pesquisa, é possível determinar a quantidade de blocos candidatos e, com o número de blocos candidatos, é possível determinar o número de cálculos de SAD que serão realizados. O número de cálculos de SAD é uma boa métrica para avaliar a complexidade computacional de cada solução. Considerando blocos de 16x16 amostras o número de blocos candidatos em uma área de k x k amostras é definido por $(k - 16 + 1)^2$. Tomando como exemplo a área de pesquisa de 80x80, que abrange os vetores com módulo de componentes de até 32 amostras, são 4.225 blocos candidatos por área de pesquisa. Cada bloco candidato possui 16x16 amostras, o que resulta em 256 cálculos de SAD por bloco avaliado. Desta forma, 1.081.600 cálculos de SAD devem ser realizados para avaliar todos os blocos candidatos.

O gráfico da Figura 5 mostra as curvas de crescimento do número de cálculos de SAD e a diminuição do erro, para cada uma das faixas de vetores. O número de operações está representado em bilhões de operações. Os valores da curva de erro representam a média dos valores de erro obtidos para as cinco amostras avaliadas. Os resultados do erro total foram multiplicados por 100 para que pudessem ser observados mais claramente no gráfico. O crescimento da curva de operações é mais acentuado até a faixa de 224. Isto ocorre, pois para as faixas de vetores maiores que 256, não é possível

aumentar as duas dimensões da área de pesquisa na mesma proporção. Para a faixa de vetores de 256, por exemplo, a área de pesquisa resultante é de 528x528, no entanto, este tamanho extrapola o tamanho vertical do quadro, que é de 480. Desta forma, a área de pesquisa efetiva para a faixa de vetores de 256 é de 528x480 pixels. Esta restrição se repete para as faixas de 288, 320 e também para a faixa >320, para a qual a área de pesquisa é o quadro inteiro.

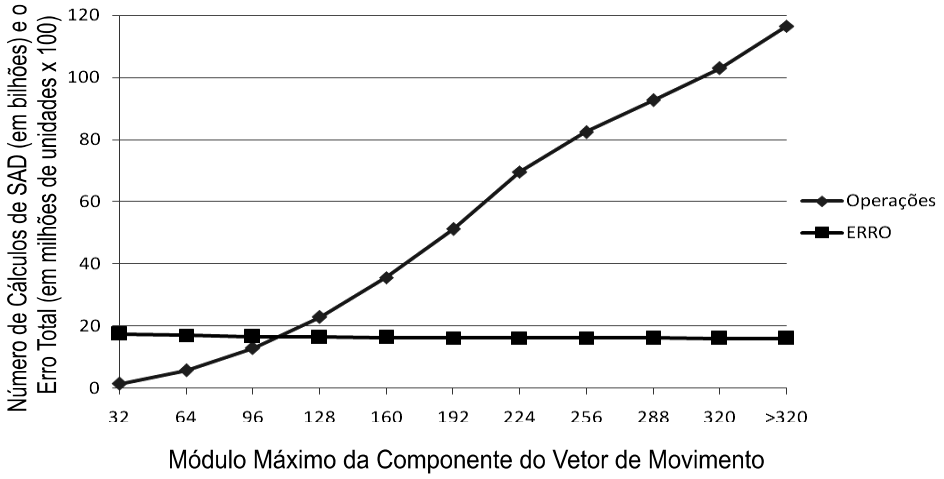

Figura 5. Gráfico com número de cálculos de SAD (em bilhões) versus o erro absoluto (em milhões de unidades e multiplicado por 100)

A curva do erro médio mostra a mesma tendência das curvas de erro apresentadas no gráfico da Figura 4, sendo que o erro cai entre as faixas de 32, 64 e 96 e, a partir disso, a curva fica praticamente estável. O mais interessante no gráfico da Figura 5 é avaliar o número de operações necessárias, que cresce muito rapidamente com o acréscimo das áreas de pesquisa, em contraste a diminuição do erro, que é praticamente imperceptível. O número de operações passa de cerca de 12,8 bilhões de operações, para a faixa de vetores de 96, para cerca de 116,5 bilhões de operações para a faixa >320. Este aumento de aproximadamente nove vezes no número de operações resulta em um ganho em diminuição de erro de apenas 2,9%. Estes resultados mostram que não é uma estratégia interessante ampliar a área de pesquisa para vetores maiores que 96 quando o tamanho de bloco é 16x16 e a resolução dos vídeos é 740x480 pixels, pois a relação entre diminuição do erro e número de operações necessárias passa a ser muito desfavorável.

Com estes resultados genéricos, uma investigação mais detalhada foi realizada considerando apenas as áreas de pesquisa menores. Nesta nova análise, diversos algoritmos foram avaliados e todos os 10 vídeos de entrada foram considerados. Além disso, foram considerados diversos tamanhos de bloco.

A Tabela 1 apresenta o resultado de redução de erro de alguns algoritmos, incluindo diferentes tamanhos de blocos, para todas as áreas de pesquisas investigadas.

Os algoritmos baseados no FS obtiveram os melhores resultados e pode-se perceber que os resultados melhoram quando a área de pesquisa é maior. Já os algoritmos rápidos (DS e HS) não alcançam um aumento significativo na qualidade com o aumento da área de pesquisa.

Tabela 1. Diminuição do erro em relação ao erro total (%)

Área de Pesquisa	FS 16x16	FS 4x4	FS+ Pel2:1 16x16	FS+ Pel2:1 4x4	DS 16x16	DS 4x4	DS+ Pel2:1 16x16	DS+ Pel2:1 4x4	HS 16x16	HS 4x4
46x46	51,80	70,53	51,52	65,91	47,15	56,31	46,74	53,03	45,72	54,60
80x80	55,70	72,96	55,40	67,70	48,97	56,47	48,52	53,13	47,38	54,75
144x144	56,69	74,51	56,38	68,61	49,15	56,48	48,69	53,14	47,55	54,76
208x208	57,04	75,29	56,73	69,00	49,17	56,48	48,71	53,14	47,57	54,76

Uma informação muito importante para algoritmos de ME é o número de operações de SAD, principalmente visando implementação em hardware. Esse número é importante para avaliar a complexidade computacional do algoritmo. A Tabela 2 apresenta o resultado de número de cálculos de SAD para diferentes tamanhos de bloco e para as quatro áreas de pesquisa dos mesmos algoritmos apresentados na Tabela 1.

Tabela 2. Número de cálculos de SAD (em bilhões de operações)

Área de Pesquisa	FS 16x16	FS 4x4	FS+ Pel2:1 16x16	FS+ Pel2:1 4x4	DS 16x16	DS 4x4	DS+ Pel2:1 16x16	DS+ Pel2:1 4x4	HS 16x16	HS 4x4
46x46	33,21	63,90	16,60	31,95	0,82	0,74	0,41	0,37	0,60	0,57
80x80	146,02	204,91	73,01	102,46	0,86	0,74	0,43	0,37	0,63	0,57
144x144	575,11	687,09	287,55	343,55	0,87	0,74	0,43	0,37	0,63	0,57
208x208	1287,32	1452,38	643,66	726,19	0,87	0,74	0,43	0,37	0,63	0,57

Os algoritmos baseados no FS aumentam significativamente o número de operações conforme vai aumentando o tamanho da área de pesquisa, enquanto que os algoritmos rápidos não sofrem esse aumento tão significativo. Comparando as Tabelas 1 e 2, pode-se perceber que os algoritmos baseados no FS conseguem melhores resultados com o aumento da área de pesquisa e diminuição do tamanho do bloco. Porém o custo computacional, em termos de número de operações de SAD, é muito elevado.

O algoritmo FS 16x16 obteve uma redução do erro 4,65% superior ao resultado obtido pelo algoritmo DS 16x16, porém o FS 16x16 utiliza 40 vezes mais cálculos de SAD para a área de 46x46 amostras. Essa comparação deixa claro que os algoritmos rápidos aceleram a estimação de movimento, sem comprometer a qualidade do resultado.

Nas próximas seções, para poder aumentar o número de combinações algorítmicas apresentadas e poder destacar a variação do resultado conforme o tamanho do bloco, serão apresentados apenas os resultados para a área de 208x208 amostras.

3.2. Resultados de Redução do Erro Total

Na seção anterior foi mostrada uma análise dos resultados da redução de erro para algumas combinações algorítmicas. Nesta seção, esse resultado será apresentado para mais combinações, incluindo os três tamanhos de blocos implementados para cada

algoritmo. A Figura 6 apresenta um gráfico com o resultado de redução de erro para a área de 208x208 amostras de 17 das combinações algorítmicas avaliadas.

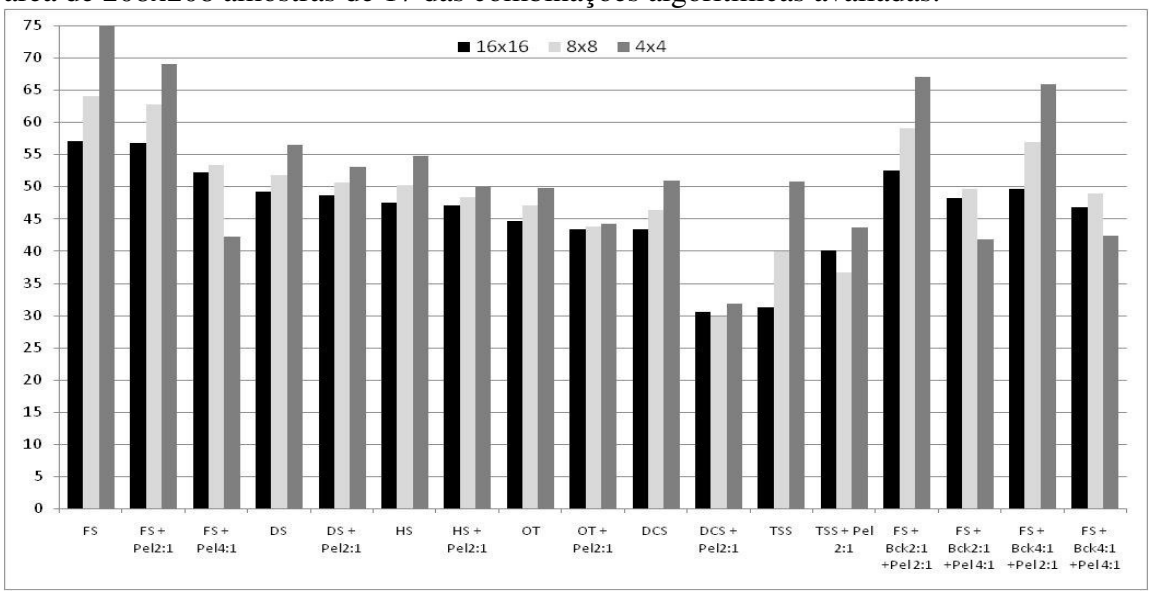

Figura 6. Gráfico com o percentual de redução de erro nos três tamanhos de blocos avaliados

Neste gráfico, pode-se perceber que 14 combinações apresentam um melhor desempenho para blocos menores e que os algoritmos com subamostragem de bloco apresentam um desempenho muito ruim. Os algoritmos rápidos com subamostragem de pixel apresentam um resultado muito parecido com as suas versões sem subamostragem, com exceção do DCS + Pel 2:1 que apresenta um resultado muito pior que o DCS. Outro resultado interessante é que para blocos de 4x4, os dois algoritmos com subamostragem de bloco mais Pel 2:1 melhoram muito seu desempenho.

Outra conclusão interessante sobre os resultados apresentados na Figura 6 é que a redução do tamanho de bloco gera um impacto enorme na qualidade dos algoritmos quando estes usam subamostragem em nível de pixel. O uso de PS faz com que o desempenho de todos os algoritmos caia muito mais do que quando tamanhos de blocos maiores são utilizados. Por outro lado, a subamostragem em nível de bloco para blocos de 4x4 amostras não possui um efeito tão nocivo quanto aquele apresentado para blocos 16x16 ou 8x8. Assim, o uso de subamostragem de blocos, para blocos de 4x4 amostras, é menos nocivo que o uso de subamostragem de pixel.

3.3. Resultados de PSNR

Um dos parâmetros de avaliação mais utilizados na literatura para comparar a qualidade objetiva de vídeos [Bhaskaran e Konstantinides 1997] é o *Peak Signal-to-Noise Ratio* (PSNR). A Figura 7 apresenta os resultados de PSNR para a área de 208x208 amostras para os três tamanhos de blocos de 17 das combinações algorítmicas investigadas.

Novamente o FS seguido do FS + Pel 2:1 apresentam os melhores resultados. Entre os algoritmos rápidos, as duas versões do DS se destacam. A Figura 7 mostra também que a subamostragem em nível de bloco possui um resultado de PSNR muito ruim, perdendo inclusive para os algoritmos rápidos. Outro algoritmo rápido que

apresenta um bom resultado é o HS, que chega a superar o DS + Pel 2:1, que até então apresentava o segundo melhor resultado entre algoritmos rápidos.

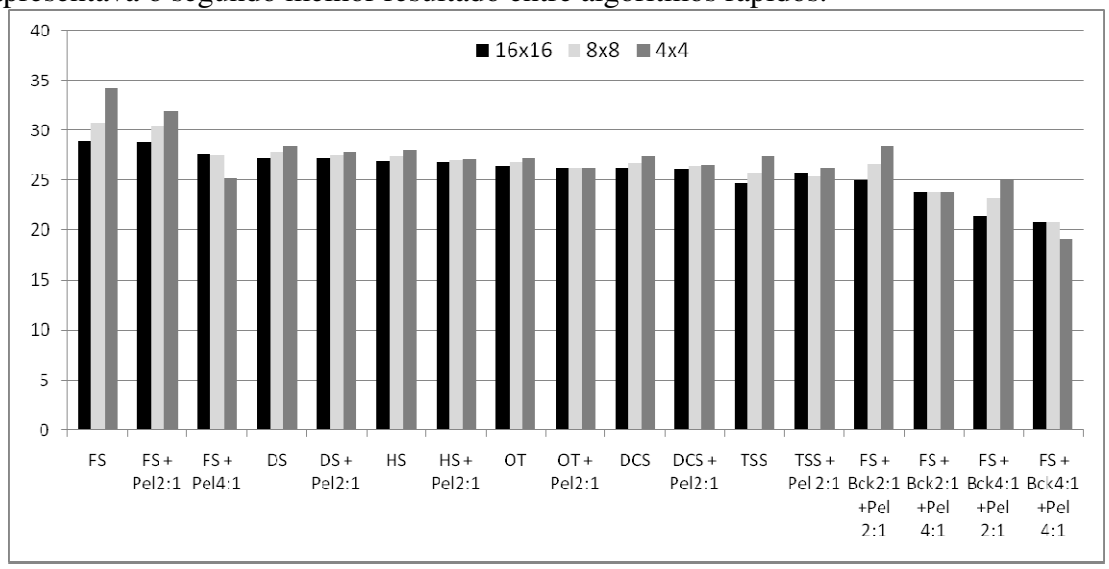

Figura 7. PSNR (em dB) para os três tamanhos de blocos e área de 208x208

3.4. Resultados do Número de Cálculos de SAD

Um critério de comparação muito importante é o número de cálculos realizados pelo algoritmo, sendo esta uma boa métrica de complexidade do algoritmo. Esse resultado foi mostrado na Tabela 2 e não é possível representá-lo em gráfico de maneira completa devido à enorme diferença nas grandezas entre os algoritmos baseados no FS e os algoritmos rápidos. Como exemplo, no pior caso, a diferença é de 1452,23 cálculos de SAD entre os algoritmos FS e OT + Pel 2:1. Esta diferença mostra a relevância do uso de algoritmos rápidos, principalmente para áreas de pesquisa maiores. Por outro lado, é possível traçar um gráfico com o número de cálculos de SAD dos algoritmos rápidos. O gráfico apresentado na Figura 8 mostra a evolução no número de cálculos de SAD para o tamanho de bloco 16x16, para as quatro áreas de pesquisa investigadas.

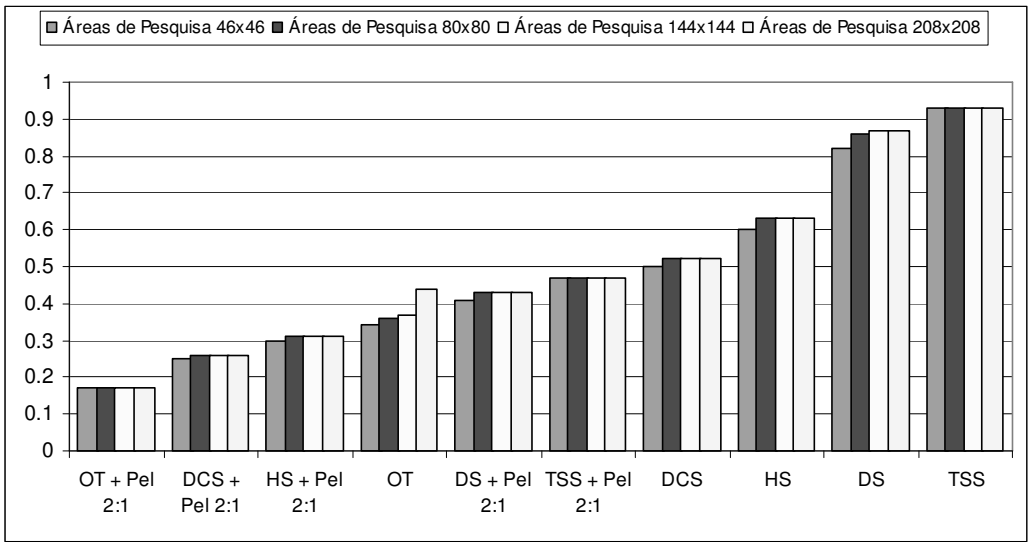

Figura 8. Cálculos de SAD (em bilhões) para as quatro áreas e bloco de 16x16

O gráfico da Figura 8 mostra claramente que o uso de subamostragem tem impacto significativo na redução dos cálculos de SAD para todos os algoritmos apresentados no gráfico. Além disso, neste critério, os algoritmos OT, DCS e HS foram os que apresentaram melhores resultados. Também é possível perceber que, para os algoritmos rápidos, o número de cálculos de SAD não aumenta de forma significativa com o aumento da área de pesquisa. Este comportamento, que é diferente dos algoritmos baseados em FS é em função da característica desses algoritmos de recair em mínimos locais, sem conseguir atingir o mínimo global. Deste modo, não interessa o acréscimo na área de pesquisa, pois, se o algoritmo rápido recaiu em um mínimo local na área menor, ele não vai buscar nas novas posições disponíveis em função do aumento da área.

3.5. Resultados de Saída no Primeiro Passo

Em geral, os algoritmos rápidos analisados neste trabalho realizam a busca em dois passos distintos. O primeiro avalia o centro da região de pesquisa, juntamente com alguns pontos ao redor, caso o melhor resultado seja obtido no centro a pesquisa é encerrada. Caso contrário, o segundo passo é aplicado, onde algum determinado padrão de busca é repetido até que uma condição de parada seja satisfeita.

A análise sobre o percentual de término da busca dos algoritmos rápidos no primeiro e segundo passos é importante, pois representa qual o percentual do melhor resultado possível seria alcançado se não fosse possível fazer mais do que uma iteração. Esse resultado é importante, pois na maioria dos vídeos existe uma grande quantidade de blocos estacionários. A Tabela 3 apresenta o percentual de término de busca no primeiro passo dos algoritmos, com subamostragem 2:1 ou sem subamostragem, para os três tamanhos de bloco e para a área de 208x208 amostras. O algoritmo TSS não é mostrado na Tabela 3 porque realiza a busca sempre em três passos.

Tabela 3. Porcentagem média de saída no primeiro passo dos algoritmos rápidos com e sem subamostragem

Algoritmos		Subamostragem	
		Nenhuma	2:1
16x16	DS (%)	22,76	22,57
	DCS (%)	19,70	20,71
	HS (%)	29,95	29,71
	OT (%)	19,38	20,34
8x8	DS (%)	20,79	20,68
	DCS (%)	19,06	19,74
	HS (%)	27,60	26,95
	OT (%)	18,51	18,98
4x4	DS (%)	19,30	19,14
	DCS (%)	18,99	19,98
	HS (%)	25,46	24,66
	OT (%)	17,75	18,25

Percebe-se que quanto menor o tamanho do bloco, menor é o número de vezes que o algoritmo finaliza no primeiro passo. Isso ocorre porque, com blocos pequenos, um SAD menor em apenas uma das posições comparadas pode conduzir a um SAD de bloco menor, gerando novas iterações. Para um bloco maior, os ganhos de posições individuais geram um impacto menor no SAD de bloco e, deste modo, a probabilidade do algoritmo convergir no primeiro passo é maior. Os resultados são bem similares entre os algoritmos, mas o HS apresenta resultados de 5% a 10% maiores que os demais. Todos os algoritmos diminuíram o percentual de saída com o uso da subamostragem.

3.6. Resultados de Média de Iterações no Segundo Passo

A Figura 9 apresenta o resultado da média de iterações dos algoritmos rápidos para o segundo passo, com exceção do algoritmo TSS, que possui o número de iterações fixo em três.

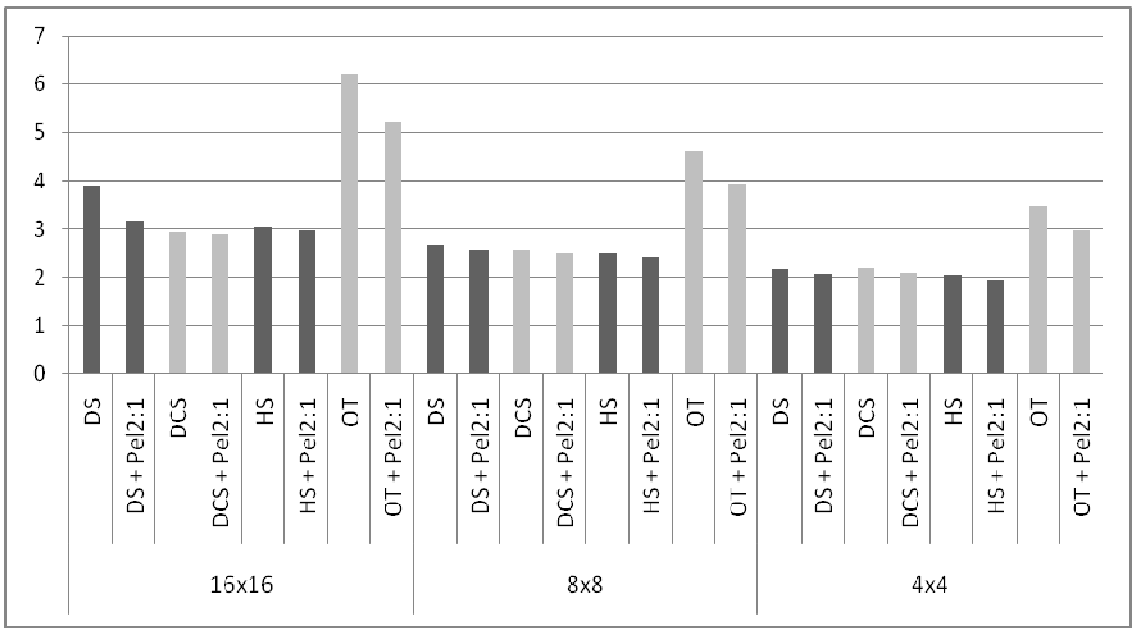

Figura 9. Média de iterações dos algoritmos rápidos com e sem subamostragem para os três tamanhos de bloco e a área de 208x208 amostras

O número de iterações é similar entre os algoritmos, ainda mais para blocos de tamanho 4x4. As versões com subamostragem apresentam uma pequena redução no número de iterações em relação às versões sem subamostragem. O resultado mais importante é o baixo número médio de iterações, que ficou sempre abaixo de sete.

3.7. Resultados de Pior Caso no Número de Iterações no Segundo Passo

Os dados de média de iterações são interessantes, mas não são suficientes para determinar com precisão o desempenho de um algoritmo, é importante também determinar o pior caso.

O gráfico da Figura 10 apresenta os resultados de pior caso no número de iterações.

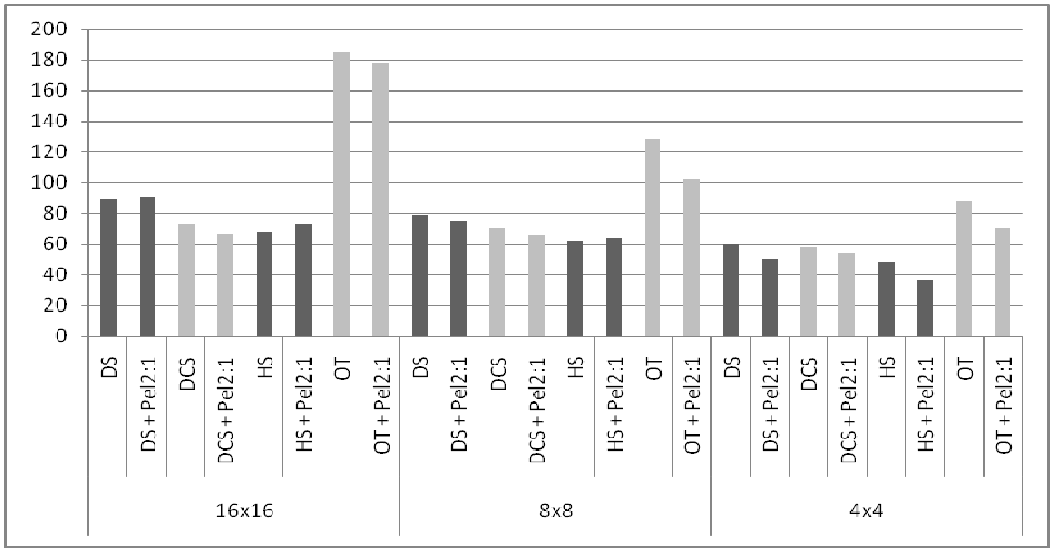

Figura 10. Pior caso no número de iterações dos algoritmos rápidos com e sem subamostragem para os três tamanhos de blocos e a área de 208x208

Comparando esses dados com os dados da Figura 9 pode-se perceber que existe uma enorme diferença entre o caso médio e o pior caso para todos os algoritmos apresentados. Esta diferença pode ser um problema, pois implica em uma variação muito grande nos tempos de geração de novos vetores e pode dificultar o sincronismo deste módulo com os demais módulos de um compressor de vídeo. Esta característica torna interessante o desenvolvimento de uma forma de limitar o número de iterações.

4. Conclusões

Dentre todos os algoritmos investigados, o FS foi o que apresentou a maior redução do erro em relação ao erro absoluto e ao PSNR. O inconveniente, neste caso, é o excessivo número de cálculos de SAD que este algoritmo realiza. As avaliações indicaram que os algoritmos rápidos podem aliar um alto desempenho com uma boa qualidade de vetores gerados. O algoritmo DS merece um destaque especial, pois obteve uma média de redução do erro 12,97% inferior em relação ao algoritmo FS, realizando em média 1729 vezes menos cálculos de SAD. Estes resultados foram gerados considerando os resultados dos 10 vídeos analisados, das 4 áreas de pesquisa e dos 3 tamanhos de bloco.

As investigações indicaram que o aumento da área de pesquisa conduz a uma redução de erro significativa quando os algoritmos baseados em FS são considerados, mas, para os algoritmos rápidos, o aumento da área de pesquisa causa uma redução do erro menos significativa. Esse comportamento ocorre porque as heurísticas utilizadas nos algoritmos rápidos fazem com que eles caiam em mínimos locais, acabando a pesquisa sem explorar bem a ampliação da área de pesquisa

Sobre o uso de diferentes tamanhos de blocos, foi possível concluir que tamanhos de blocos menores conduzem a resultados de erro significativamente menores, para todos os algoritmos. Por outro lado, o uso de blocos menores implica em um aumento no número de cálculos de SAD.

Se o foco for uma implementação em hardware, como o paralelismo pode ser explorado com muitos graus de liberdade, pode ser interessante optar por um algoritmo

com maior número de cálculos de SADs, como o FS, e que não possua dependências de dados entre estes cálculos. Mas, é claro que quanto maior for a exploração do paralelismo, maior será o consumo de recursos de hardware. Isso pode ser um problema, primeiro porque os recursos de hardware são limitados e, também, porque se houver mais hardware operando em paralelo, o consumo de energia será mais elevado.

Como trabalho futuro, será realizada uma avaliação de formas de limitação no número de iterações dos algoritmos rápidos, bem como do impacto dessa mudança na qualidade dos resultados, visando a facilitar o sincronismo da ME com outros módulos.

5. Referências

Banh, X. e Tan, Y. (2004) "Adaptive dual-cross Search algorithm for Block-matching motion estimation", IEEE Transactions on Consumer Electronics, [S.l.], v. 50, n. 2, p. 766-775, Mai.

Bhaskaran, V. e Konstantinides, K. (1997) "Image and Video Compression Standards: Algorithms and Architectures", Boston: Kluwer Academic Publishers, 2nd edition.

International Telecommunication Union. (2005) ITU-T Recommendation H.264/AVC (03/05): advanced video coding for generic audiovisual services.

Jing, X. e Chau, L. (2004) "An Efficient Three-Step Search Algorithm for Block Motion Estimation", IEEE Transactions on Multimedia. [S.l.], v. 6, n. 3, p. 435-438

Korah, R.; et al. (2005) "Motion Estimation with Candidate Block and Pixel Subsampling Algorithm", In: IST 2005 - IEEE International Workshop on Imaging Systems and Techniques. Proceedings… Niagara Falls: IEEE, 2005, p. 130-133.

Kuhn, P. (1999) "Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation", Boston: Kluwer Academic Publishers.

Lin, C. e Leou, J. (2005) "An Adaptative Fast Full Search Motion Estimation Algorithm for H.264/AVC", In: ISCAS 2005 - IEEE INTERNATIONAL SYMPOSIUM CIRCUITS AND SYSTEMS. Proceedings… Kobe: IEEE, 2005, p. 1493-1496.

Richardson, I. (2002) "Video Codec Design – Developing Image and Video Compression Systems", Chichester: John Wiley and Sons.

Rosa, L. (2007) "Investigação sobre Algoritmos para a Estimação de Movimento na Compressão de Vídeos Digitais de Alta Definição: Uma Análise Quantitativa", http://www.ufpel.tche.br/prg/sisbi/bibct/acervo/info/2007/mono_leandro_rosa.pdf.

VQEG (2007) The Video Quality Experts Group Web Site. http://www.its.bldrdoc.gov/vqeg/, Abril.

Yi, X. e Ling, N. (2005) "Rapid Block-Matching Motion Estimation Using Modified Diamond Search Algorithm", In: ISCAS 2005 - IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS. Proceedings… Kobe: IEEE, 2005, p. 5489 – 5492.

Zhu, C., Lin, X e Chau, L. (2002) "Hexagon-based Search pattern for fast Block motion estimation", In: IEEE Transactions on Circuits and Systems for Video Technology, Volume 12, Issue 5, p. 349 – 355.