5
Métodos rápidos utilizados para estimação de movimento

A estimação de movimento para CODEC H.264/AVC é computacionalmente custosa se o método Full Search (que será apresentado neste capítulo) é usado. A fim de reduzir o tempo de codificação, o atual software de referência JM (aqui chamado de JM 98, pois encontra-se atualmente em sua versão 9.8) adota um método rápido de estimação de movimento para pixel inteiro chamado UMHexagonS e um método para pixel fracionário chamado CBFPS. Um método proposto por Xiaoquan Yi, Jun Zhang, Nam Ling e Weijia Shang [2] aqui chamado, por simplicidade, de P021 (uma vez que a denominação do documento no Joint Video Team (JVT) é JVT-P021), apresenta, em comparação com o software de referência JM, uma forma melhor e simplificada de estimação de movimento para aumentar a velocidade do processo de codificação e manter o desempenho em termos da taxa-distorção. A partir do método P021, propõe-se um método de estimação de movimento que não realize o processo de estimação de movimento de meio-pixel para certos blocos baseado num modelo de seleção linear. A idéia principal consiste em "escapar" de blocos que não se beneficiam da pesquisa de meio-pixel. Assim, obtém-se uma forma de estimação de movimento com velocidade de codificação superior e desempenho em termos da taxa-distorção semelhante, se comparado aos três métodos acima. (Full Search, JM98 e P021).

Neste capítulo serão apresentados e discutidos os métodos rápidos utilizados para estimação de movimento e citados acima: Full Search, JM98, P021 e o método proposto.

5.1
Full Search

Segundo Iain Richardson (2003), este método de estimação de movimento consiste em verificar em todos os pontos da janela de pesquisa, qual apresenta o

PUC-Rio - Certificação Digital Nº 0421021/CA

menor SAD, ou seja, o menor resultado da equação 2.3, descrita anteriormente no item 2.5. Portanto, demanda um grande esforço computacional.

Um exemplo da estratégia Full Search é apresentado na Figura 35. O ponto de pesquisa inicial é na extremidade superior esquerda da janela e a pesquisa continua em ordem de "varredura" até todas as posições serem avaliadas. Este método pode ser simplificado iniciando a pesquisa em (0,0) e realizando os procedimentos de teste em uma trajetória espiral ao redor desta localização, como mostra a Figura 36. O esforço computacional pode ser reduzido, caso se utilize o procedimento de "terminação precoce". Neste procedimento, ao se atingir determinado valor de SAD, a pesquisa é interrompida.

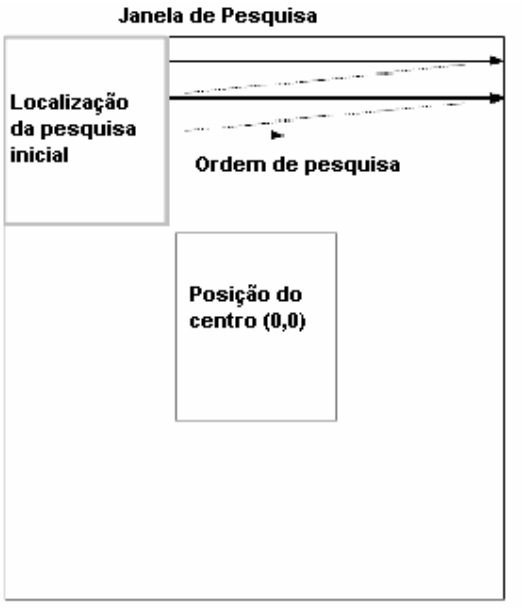

Figura 35- Método Full search com ordem de pesquisa em "varredura"
Fonte: Ref [1]

PUC-Rio - Certificação Digital Nº 0421021/CA

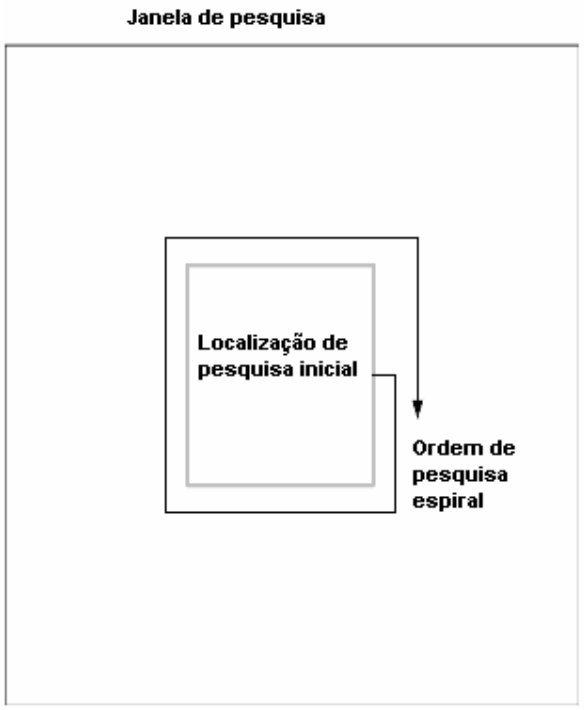

Figura 36- Método Full Search com ordem de pesquisa em espiral
Fonte: Ref [1]

Mesmo com o uso de terminação precoce, a estimação de movimento Full Search é ainda computacionalmente intensa para muitas aplicações práticas. Em aplicações com recursos computacionais e de energia limitados, os chamados algoritmos de pesquisa rápida são preferíveis.

Esses algoritmos calculam o valor de SAD em um subconjunto de localizações dentro da janela de pesquisa. Uma vez que o número de pontos de pesquisa é menor, esses algoritmos são mais simples do que o método Full Search, mas não apresentam a mesma precisão, uma vez que este sempre identifica o valor mínimo global de SAD, enquanto o algoritmo de pesquisa rápida pode identificar um mínimo local, fornecendo um pior resultado.

5.2
O software de referência JM 9.8

O atual software de referência JM na sua versão 9.8 adota um método rápido de estimação de movimento para pixel inteiro chamado UMHexagonS (Hybrid Unsymmetrical-cross Multi-Hexagon-grid Search) e um método para

PUC-Rio - Certificação Digital Nº 0421021/CA

pixel fracionário chamado CBFPS (Center Biased Fractional Pel Search). Estes métodos são descritos nos itens seguintes.

5.2.1 Algoritmo UMHexagonS para estimação de movimento de pixel inteiro

Este método utiliza uma estratégia de pesquisa que inclui 04 passos, apresentados a seguir e indicados na Figura 37 (Chen, Zhou e He, 2002). Maiores detalhes deste algoritmo podem ser obtidos em [3].

Passo 1: Predição do ponto de pesquisa inicial: utiliza preditor mediano espacial, preditor de nível superior, predição através de quadro de referência vizinho e predição temporal para estimar o vetor de movimento do bloco atual (MV). O preditor com o mínimo custo entre estes candidatos será escolhido como a posição de pesquisa inicial para o próximo passo de pesquisa. Detalhes sobre cada uma destas predições podem ser obtidos em [3];

Passo 2: Pesquisa com cruzamento não simétrico (Unsymmetrical-cross search): baseia-se no fato do movimento na direção horizontal ser muito maior do que na direção vertical. É seguido por um método de terminação precoce (early termination scheme).

Passo 3: Pesquisa com grade multi-hexagonal desigual (Uneven multi-hexagon-grid search): pontos de pesquisa na forma de hexágonos concêntricos de tamanhos crescentes;

Passo 4: Pesquisa baseada em hexágono extendido (Extended hexagon based search):): pontos de pesquisa na forma de hexágonos não concêntricos de tamanhos iguais. É também seguido por um método de terminação precoce (early termination scheme);

A Figura 37 mostra o procedimento típico numa janela de pesquisa de tamanho 16 x 16 onde é assumido aqui que o ponto de pesquisa inicial é o vetor (0,0).

PUC-Rio - Certificação Digital Nº 0421021/CA

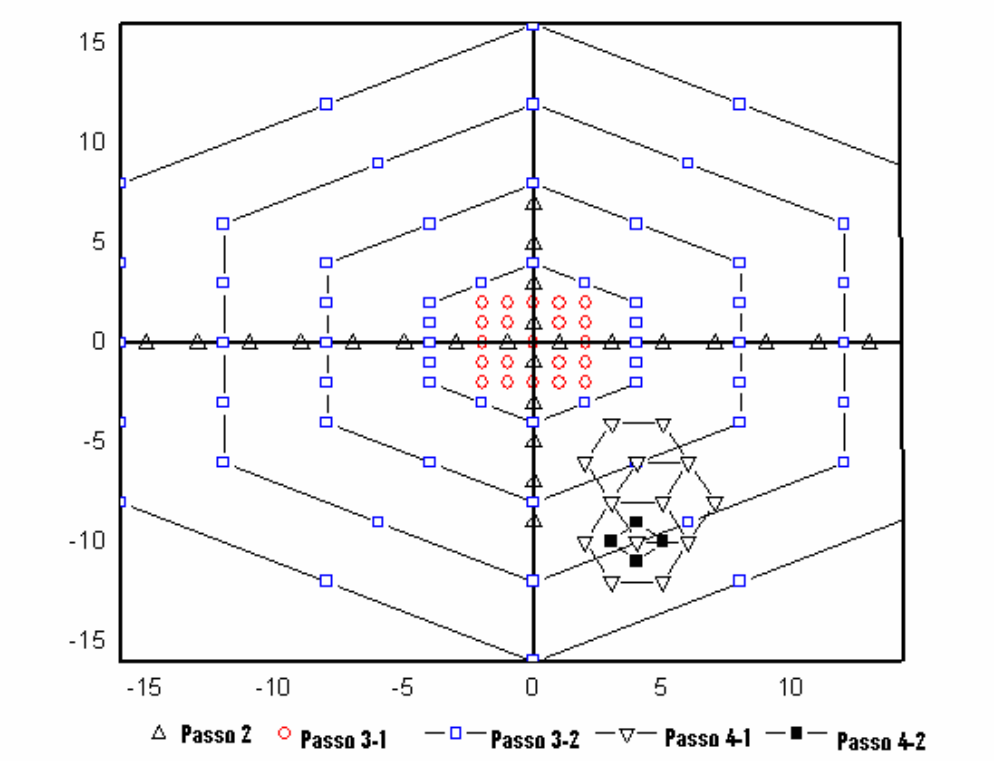

Figura 37- Pesquisa do algoritmo UMHexagonS
Fonte: Ref [3]

5.2.2
Algoritmo CBFPS para estimação de movimento de pixel fracionário

Segundo Chen, Zhou e He (2002) a implementação do algoritmo CBFPS é descrita pelos 04 passos a seguir especificados e ilustrados na Figura 38:

Passo 1: O vetor de movimento estimado do bloco atual (pred_mv) é calculado como a mediana dos vetores de movimento localizados à esquerda, imediatamente acima e acima à direita (ou acima à esquerda) do bloco atual, conforme apresentado na equação (5.1)

$$pred\_mv = median(Mv\_A, Mv\_B, Mv\_C) \qquad (5.1)$$

O vetor de movimento estimado do bloco atual é definido como uma unidade de pixel fracionário, de tal forma que ele inclui a informação de vetor de movimento de pixel inteiro estimado e vetor de movimento de pixel fracionário estimado. Conseqüentemente, pode-se extrair o vetor de movimento de pixel fracionário estimado usando a fórmula:

PUC-Rio - Certificação Digital Nº 0421021/CA

$$frac\_pred\_mv = (pred\_mv - mv).\alpha.\beta \qquad (5.2)$$

onde mv é o vetor de movimento de pixel inteiro do bloco atual, e aqui mv está também em unidade de pixel fracionária, α é o modo de operação, $\beta = 4$ no caso de ¼ pixel e $\beta = 8$ no caso de 1/8 pixel.

Portanto, estima-se o vetor de movimento do bloco atual pelas equações (5.1) e (5.2), sendo o vetor de movimento estimado dado por (Pred_x, Pred_y).

Passo 2: O custo no centro de pesquisa original (0,0) e em (Pred_x, Pred_y) são comparados. O ponto com o menor SAD é escolhido como o centro de pesquisa;

Passo 3: Se o ponto com mínima distorção de bloco (Minimum Block Distortion-MBD), ou seja, com menor SAD, é localizado no centro, vá para o passo 4. Caso contrário, escolha o ponto MBD nesse passo como o centro do próximo passo. Então repita a partir do passo 1;

Passo 4: Escolha o ponto MBD como o vetor de movimento.

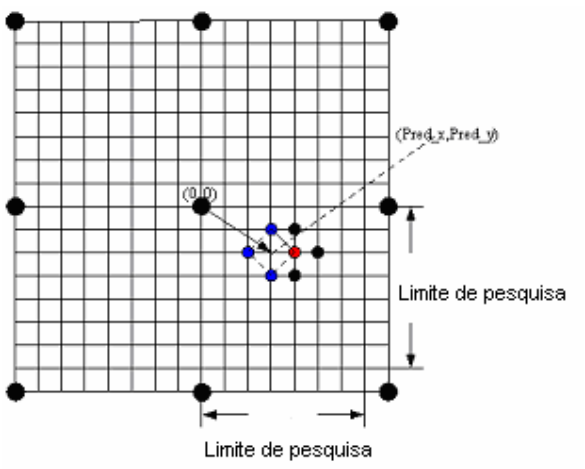

Figura 38 - Implementação do algoritmo CBFPS
Fonte: Ref [3]

5.3
P021 - Método de Xiaoquan Yi, Jun Zhang, Nam Ling e Weijia Shang

O software de referência é mais ajustado em termos de eficiência de codificação do que em termos de velocidade de processamento, pois o

PUC-Rio - Certificação Digital Nº 0421021/CA

desempenho em termos de taxa-distorção (rate-distortion, R-D) é o parâmetro mais preocupante durante o processo de padronização. Este software é geralmente usado como um "benchmark" ou referência para os pesquisadores devido à sua disponibilidade pública. Conforme visto no item 5.2, ele adota um método de estimação de movimento rápido incluindo UMHexagonS para pesquisa de pixel inteiro e CBFPS para pesquisa de pixel fracionária, apresentando um desempenho semelhante de R-D em relação ao método Full Search. Para manter este desempenho de R-D, é empregado muitos passos de full search durante os processos de pesquisa de pixel inteiro e subpixel. Mesmo com o uso do mecanismo de terminação precoce, UMHexagonS apresenta ainda uma velocidade de processamento limitada em relação ao full search. Portanto, o método proposto por Xiaoquan Yi, Jun Zhang, Nam Ling e Weijia Shang, objetivou desenvolver um método simples, porém eficiente, de estimação de movimento rápido híbrido baseado no método UMHexagonS existente.

PUC-Rio - Certificação Digital Nº 0421021/CA

5.3.1
Descrição do método P021

Nos itens a seguir apresentaremos uma descrição do método P021, conforme mostrado em [2].

5.3.1.1
Utilização de pouca memória

Como descrito na seção 5.2.1, o método UMHexagonS usa vetores de movimento espaciais, temporais e de nível superior e estimações de SAD que requerem bastante memória. Conforme Xiaoquan Yi, Jun Zhang, Nam Ling e Weijia Shang (2005), a predição temporal requer maior quantidade de memória devido aos tamanhos de blocos variáveis do H.264, quadros de referência múltiplos e compensação de movimento de precisão de ¼ de pixel. Isto é particularmente indesejável para aplicações com restrição de memória. P021 apresenta uma forma de estimação de movimento de baixo consumo de memória que usa somente informação dentro do mesmo quadro. Sendo assim, o espaço de

memória é significativamente reduzido. Para estimação do vetor de movimento usa-se somente preditores de média espacial e de nível superior [3]. Para estimação de SAD, usa-se somente predição de nível superior denominada de pred_SAD_uplayer em oposição às quatro predições usadas pelo UMHexagonS.

Este preditor simplificado apresenta o mesmo desempenho em termos de taxa-distorção que o outro preditor não simplificado.

5.3.1.2
Pesquisa rápida de estimação de movimento de pixel inteiro

Como descrito na seção 5.2.1, durante o processo de pesquisa de pixel inteiro, UMHexagonS FME usa o método de pesquisa local full search e vários outros métodos tais como pesquisa cruzada, pesquisa baseada em diamante, baseada em hexágono e baseada em hexágono estendido para evitar incorrer em erro na determinação do ponto com mínimo SAD. Segundo Xiaoquan Yi, Jun Zhang, Nam Ling e Weijia Shang (2005), embora seja utilizada a terminação precoce, que se baseia no parâmetro de quantização, a pesquisa inteira geralmente leva bastante tempo. Além disso, limites em ponto flutuantes devem ser calculados e testados para todo bloco de pesquisa em UMHexagonS. Em P021, é evitado local full search e são usadas técnicas do tipo terminação precoce mais simples.

Ainda segundo Xiaoquan Yi, Jun Zhang, Nam Ling e Weijia Shang (2005), um padrão típico de pesquisa de pixel inteiro rápida do método P021 é apresentado no fluxograma da Figura 39 e descrito a seguir.

Após a pesquisa dos preditores, se a condição de convergência é satisfeita, segue-se para o passo de convergência e a pesquisa é então interrompida.

Se a condição de pesquisa intensiva é satisfeita, realiza-se a pesquisa intensiva para evitar incorrer em erro na determinação do ponto ótimo local. Caso esta condição não seja satisfeita, verifica-se o preditor de nível superior (up layer predictor)[3] e seus pontos vizinhos e verifica-se a condição de convergência novamente. As condições de convergência e de pesquisa intensiva são apresentadas em [2]. É dada especial consideração para blocos do tipo 1 (16x16)

PUC-Rio - Certificação Digital Nº 0421021/CA

pois estes têm uma quantidade mínima de informação de predição e é usado como preditor de nível superior (up layer predictor) [3].

No algoritmo de referência, o limite de terminação precoce envolve multiplicação de ponto flutuante. Contudo, no algoritmo P021, estas são simplificadas e substituídas por operações de deslocamento de bits e comparação.

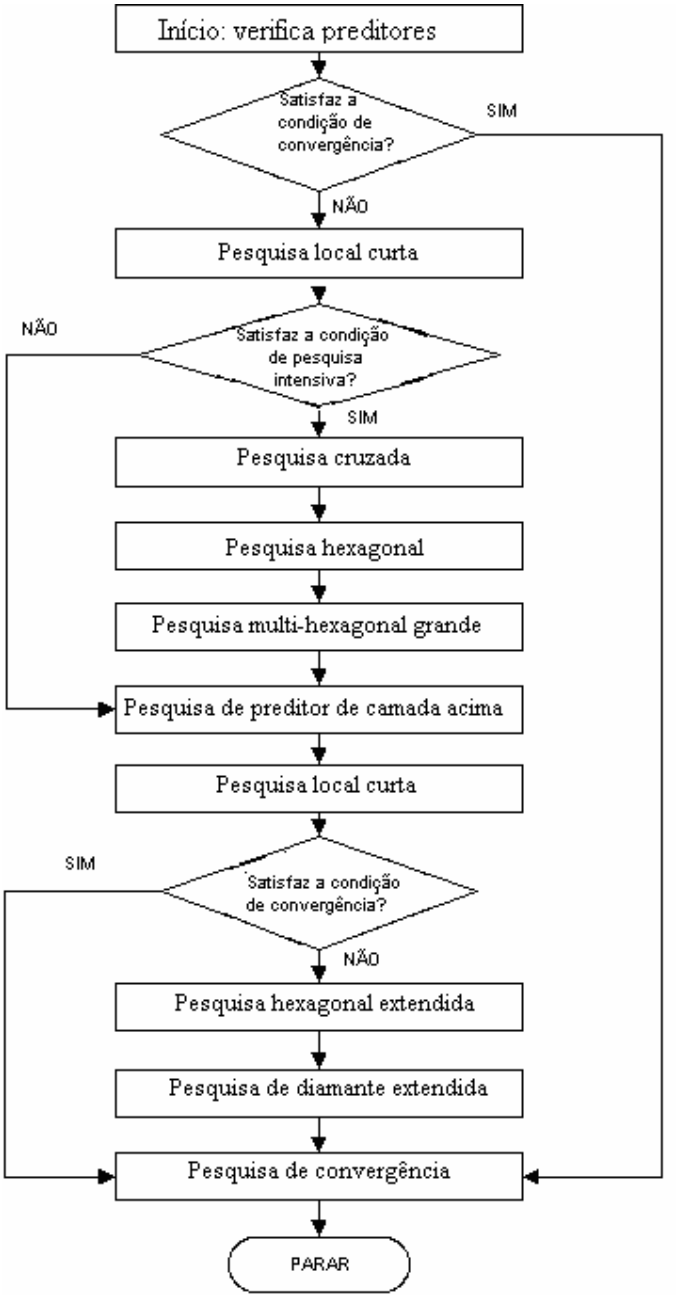

Figura 39- Fluxograma da pesquisa rápida de pixel inteiro do método P021.
Fonte: Ref [2]

PUC-Rio - Certificação Digital Nº 0421021/CA

5.3.1.3
Pesquisa rápida de estimação de movimento de pixel fracionário

A resolução em subpixel contribui significativamente para o aumento da eficiência de codificação com a adição de custos computacionais não triviais. A pesquisa rápida neste nível de resolução apresenta tempo de processamento não negligenciável quando comparada com a estimação de movimento de pixel inteiro. Para reduzir este custo computacional e satisfazer restrições de complexidade, um método simples não teria uma precisão de subpixel da compensação de movimento, o que comprometeria grandemente a eficiência de codificação. Muitos métodos fracionários rápidos foram propostos na comunidade científica [29] [30]. No método de pesquisa fracionária usado no software de referência, o algoritmo CBFPS é usado para partições de blocos pequenas como mostrado na Figura 40. Em outras palavras, blocos grandes, por exemplo 16 × 16, 16 × 8 e 8 × 16 ainda usam full search. Portanto, a velocidade usando CBFPS é seriamente comprometida. Para superar esta desvantagem, P021 apresenta dois métodos de pesquisa de subpixel rápidos [2]:

- um método simples e eficiente de escapar da estimação de subpixel baseado em análise estatísticas; e

- uma técnica de parada imediata baseada em custo mínimo.

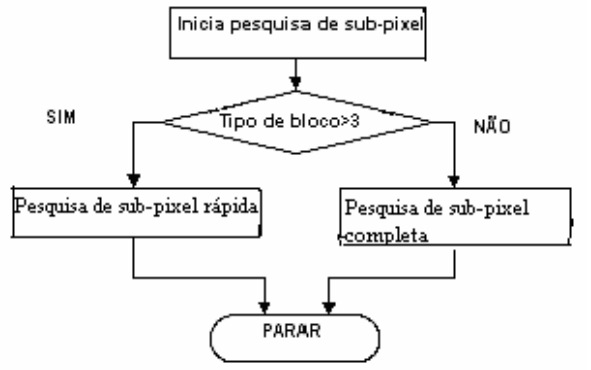

Figura 40- Fluxograma do algoritmo de referência de pesquisa de subpixel.
Fonte: Ref [2]

PUC-Rio - Certificação Digital Nº 0421021/CA

Na Figura 41, para partições de blocos 16 x 16, uma pesquisa de subpixel do tipo rápida completa é usada, pois blocos 16x16 são usados como preditores para blocos de partição menores. Para todos as outras partições de bloco, uma pesquisa de subpixel rápida é aplicada. Estes dois métodos de pesquisa são mostrados na Figura 42 e 43 respectivamente. Portanto, o método P021 acomoda tanto a pesquisa de subpixel completa (full sub-pel search) quanto o CBFPS. Maiores detalhes destes dois tipos de pesquisa, da condição de parada imediata de subpixel e das condições de escape de um quarto de pixel são apresentados em [2]

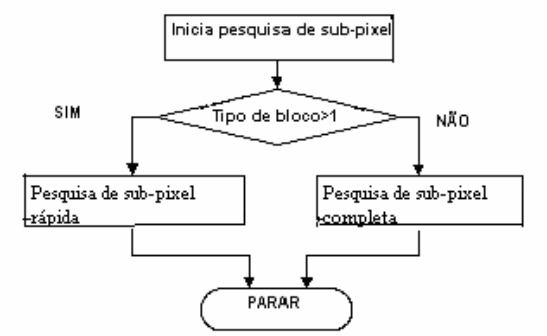

Figura 41- Fluxograma do algoritmo P021 de pesquisa de subpixel.
Fonte: Ref [2]

PUC-Rio - Certificação Digital Nº 0421021/CA

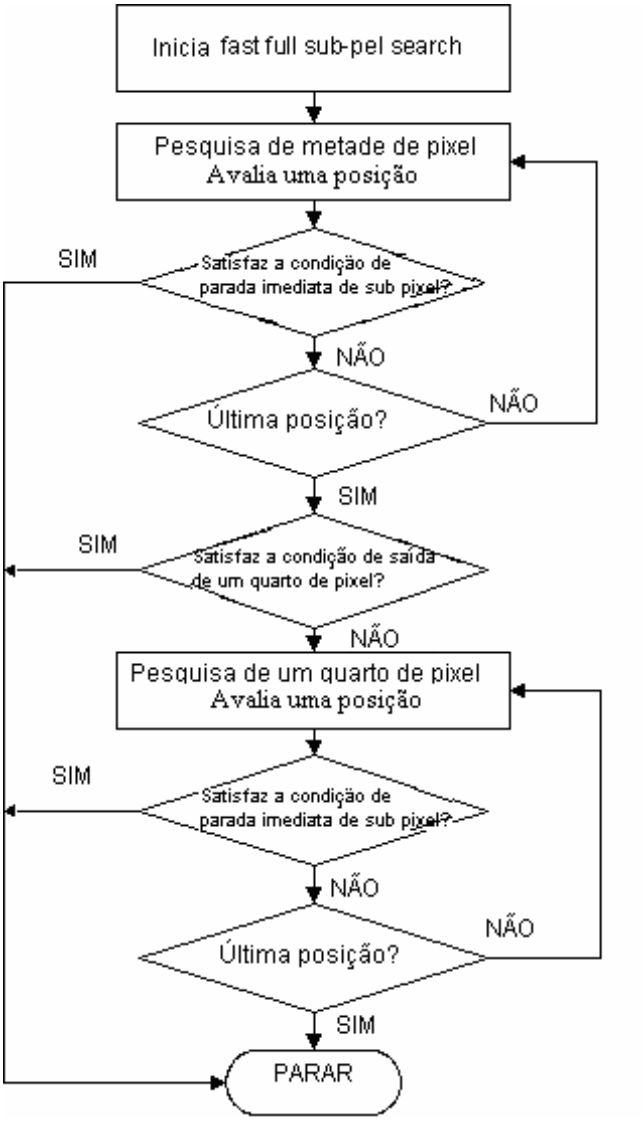

Figura 42- Fluxograma da pesquisa de subpixel rápida completa (fast full sub-pel search) usada no método P021.

Fonte: Ref [2]

PUC-Rio - Certificação Digital Nº 0421021/CA

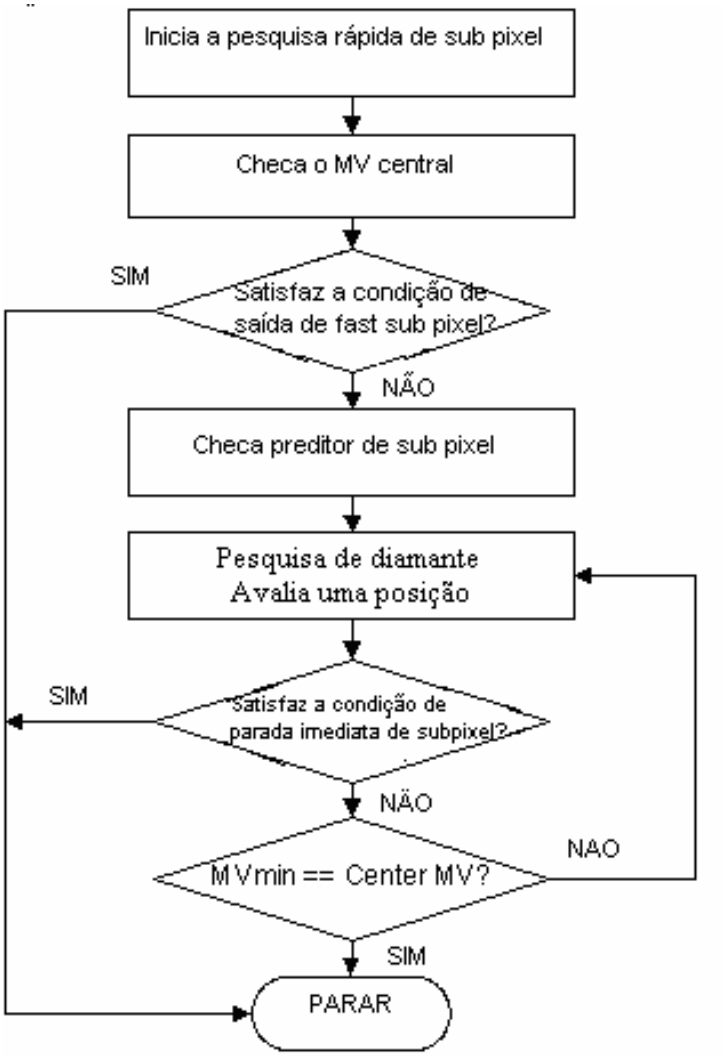

Figura 43- Fluxograma da pesquisa rápida de subpixel usada no método P021. MVmin: MV tem o custo minímo.
Fonte: Ref [2]

No software de referência JM, como foi apresentado no item 3.2.3, o SAD é usado quando se calcula a estimação de movimento de pixel inteiro enquanto SATD é usado para estimação de movimento de subpixel. SA(T)D refere-se tanto ao SAD quanto ao SATD dependendo do status da flag da transformada de Hadamard. Segundo Xiaoquan Yi, Jun Zhang, Nam Ling e Weijia Shang (2005), no modo de baixa complexidade, Hadamard ON apresenta um aumento de 0.3-0.5 dB no desempenho medido pelo PSNR do que Hadamard OFF com complexidade maior. SATD envolve subtração, adição, deslocamento e operação modular. Entre

PUC-Rio - Certificação Digital Nº 0421021/CA

estes, a operação modular apresenta maior tempo de processamento. Em P021 foi usado uma tabela para aumentar a velocidade desta operação. Para limitar os gastos de memória desta tabela, foi aplicado uma pesquisa na tabela somente para 15 coeficientes "AC" da transformada de Hadamard de blocos 4x4, mas não para coeficientes "DC". Essa técnica de pesquisa na tabela reduz o tempo total de codificação em aproximadamente 12%, com um aumento pequeno de memória requerida. O modo de decisão é também beneficiado com esta pesquisa na tabela uma vez que SATD é parte da computação do modo de decisão [2].

5.4
Método proposto

O esforço computacional requerido para estimação de movimento de meio-pixel é significativo e comparável com o esforço utilizado na estimação de movimento de pixel inteiro.

Ao longo dos últimos anos ocorreram avanços significativos nas técnicas de estimação de movimento rápidas de pixel inteiro, com o objetivo principal de diminuir o esforço computacional sem degradação da qualidade visual.

Quanto à estimação de movimento de sub-pixel, também foram desenvolvidos vários métodos visando a diminuição do tempo de processamento, sem degradação da qualidade visual, através da diminuição da quantidade de pontos de pesquisa de meio-pixel.

Segundo Yu, Lu e Li (2004), em um método que usa direções horizontais e verticais como referência apresentado em [13], o número de pontos de pesquisa de meio-pixel é reduzido para 5. Outro método de pesquisa rápido com precisão de meio-pixel apresentado em [14], reduz o esforço computacional em 50%. Na predição parabólica baseada na pesquisa de meio-pixel rápida [15], há uma economia de 59% no esforço computacional para a pesquisa de blocos. Um método rápido baseado na pesquisa direcional e um modelo linear [16] reduz o número de pontos de pesquisa para 2.2 em média, enquanto a qualidade da imagem das seqüências reconstruídas é similar à dos métodos tradicionais.

Visando diminuir ainda mais o tempo de processamento e a capacidade computacional apresentada pelos métodos expostos nas seções anteriores (Full Search, JM98 e P021), o que é necessário principalmente para aplicações de

PUC-Rio - Certificação Digital Nº 0421021/CA

codificação de vídeo móveis (onde o receptor geralmente apresenta reduzida capacidade computacional) e aplicações de tempo real (onde o retardo de processamento deve ser mínimo), propõe-se um algoritmo de estimação de movimento que não realize o processo de estimação de movimento de meio-pixel para certos blocos baseado num modelo de seleção linear. A idéia principal consiste em "escapar" de blocos que não se beneficiam da pesquisa de meio-pixel. Conseqüentemente, se reduz o número de pontos de pesquisa e o processo de interpolação.

Este algoritmo, que é baseado no método proposto por Keman Yu, Shan Lu, Jiang Li e Shipeng Li [31] com algumas modificações, foi adicionado ao método P021 apresentado anteriormente no item 5.3 (que por sua vez é uma adaptação do software de referência JM). O algoritmo mostrado em [31], que foi apresentado em 2003, apresenta apenas os resultados experimentais da aplicação deste método, implementado em um codificador H.263, para seis seqüências de vídeo. O método P021 foi apresentado em julho de 2005, implementado no software de referência H.264 apresentando melhorias em termos de tempo de processamento. Portanto, o que foi feito neste trabalho e que será apresentado a seguir, foi aliar as melhorias em termos de tempo de processamento do método P021 com as do método apresentado em [31], adicionando em seguida algumas modificações a este último.

Resultados experimentais mostram que uma redução computacional significativa é obtida quando uma pequena degradação na qualidade de vídeo é negligenciada.

Nas próximas seções, é descrito este algoritmo de estimação de movimento que foi implementado após o algoritmo de estimação de movimento de pixel inteiro e antes do algoritmo de estimação de movimento de pixel fracionário.

No algoritmo apresentado a seguir, assume-se que o quadro de vídeo é dividido em macroblocos com o tamanho de 16 x 16 pixel. A soma das diferenças absolutas (SAD), apresentada em 2.5, é usada como a medida do custo para selecionar o bloco com melhor casamento na estimação de movimento.

PUC-Rio - Certificação Digital Nº 0421021/CA

5.4.1
Taxa de pesquisa efetiva (ESR) e Limite ótimo de SAD

A motivação para o algoritmo apresentado em [31] vem da observação que, para a maioria das seqüências com cenas de pouco movimento, um número significativo de macroblocos, tipicamente 50% a 90% de todos os macroblocos, tem seu vetor de movimento final com precisão de pixel inteiro. No método de pesquisa convencional de meio-pixel, efetua-se para cada macrobloco a pesquisa de oito pontos de meio-pixel ao redor do vetor de movimento de pixel inteiro. Se o menor valor de SAD, obtido com precisão de meio-pixel é maior do que o obtido com a precisão de pixel inteiro, o vetor de movimento de pixel inteiro é selecionado como o resultado final e a pesquisa de meio-pixel para este macrobloco é desprezada. De outra forma, se o menor valor de SAD, obtido com precisão de meio-pixel é menor do que o obtido com a precisão de pixel inteiro, o vetor de movimento com precisão de meio-pixel é selecionado como o resultado final e a pesquisa de meio-pixel para este macrobloco é considerada como efetiva. Observa-se que, para cenas com relativamente baixa movimentação, a maioria das pesquisas de meio-pixel são desprezadas. Mesmo seqüências que apresentam grande movimentação, poucos macroblocos apresentam estimação de movimento de meio-pixel efetiva.

Segundo Yu, Lu e Li (2004), se for possível estimar os macroblocos que não se beneficiam da pesquisa de meio-pixel, pode-se eliminar o esforço computacional da interpolação e pesquisa associada com esse macroblocos. Conseqüentemente, pode-se poupar bastante esforço computacional. Se o valor mínimo de SAD de um vetor de movimento de pixel inteiro é suficientemente pequeno, é altamente provável que a pesquisa de meio-pixel seja desnecessária. Isto inspira a encontrar um limiar. Somente os macroblocos que apresentarem um SAD mínimo maior do que o limiar precisam sofrer uma pesquisa de meio-pixel.

Como um limite de SAD menor implica em um número maior de pesquisas de meio-pixel e vetores de movimento mais precisos, porém com um esforço computacional maior, precisa-se encontrar um limite de SAD ótimo que forneça uma boa relação entre eficiência de compressão e complexidade.

Para obter um limite ótimo para a seqüência, o que foi apresentado em [31], uma seqüência de vídeo inteira foi codificada com vários valores de SAD. A

PUC-Rio - Certificação Digital Nº 0421021/CA

eficiência de codificação, que é medida em PSNR, diminui quando o limite aumenta. O objetivo é encontrar o ponto onde PSNR começa a diminuir bruscamente. Como mostrado na Figura 44, esse tipo de ponto é também o ponto que apresenta a maior curvatura.

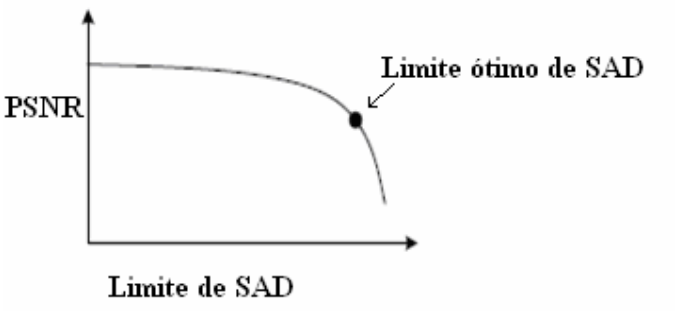

Figura 44-Diminuição do PSNR para um limite de SAD maior.
Fonte: Ref [31]

PUC-Rio - Certificação Digital Nº 0421021/CA

Como estes limites ótimos são selecionados através de pesquisa para todos os quadros de uma seqüência, eles não podem ser aplicados para codificação em tempo real. Este problema é extinto pois, conforme mostrado em [31], partindo de um valor inicial, recalcula-se este limite ótimo para cada quadro.

Observou-se que não existe uma equação que pode diretamente determinar o limite ótimo usando PSNR, ESR e a média de SAD. Contudo pode-se saber se o limite atual é apropriado pelo exame do resultado de compressão a cada quadro em tempo real. Desta forma, pode-se aumentar ou diminuir o limite para o próximo quadro para alcançar um valor apropriado.

5.4.2
Relação entre ESR e OSR para seqüências de vídeo e a primeira modificação do algoritmo proposto

Define-se a taxa de pesquisa ótima de meio-pixel (Optimal half-pixel Search Ratio - OSR) como a taxa de pesquisa de meio-pixel que é calculada quando o limite de SAD é ajustado como um limite ótimo, conforme descrito anteriormente. A Tabela 4, obtida de [31], lista o OSR e ESR respectivamente para seis seqüências.

Seqüência	ESR	OSR
Akiyo	2,90%	23,88%
Salesman	4,19%	20,85%
Miss America	12,52%	40,97%
Suzie	34,10%	61,19%
Carphone	34,49%	65,66%
Foreman	40,22%	70,73%

Tabela 4- ESR e OSR

Fonte: Ref [31]

Após plotar estes valores de OSR e ESR na Figura 45, foi observado que eles basicamente obedecem a uma relação linear (linha sólida na Figura 45). Dado o limite de SAD, ao final da pesquisa para todos os macroblocos de um quadro, pode-se calcular a taxa de pesquisa de meio-pixel atual (Actual half-pixel Search Ratio-ASR) e a ESR. Usando a relação linear obtida anteriormente, pode-se calcular OSR a partir de ESR. Depois disso, examina-se se o limite de SAD atual é apropriado pela comparação do OSR calculado com o atual ASR. Por exemplo, se o ASR é maior que o OSR, o limite de SAD atual é considerado pequeno e é, portanto, aumentado para a pesquisa dos macroblocos do próximo quadro.

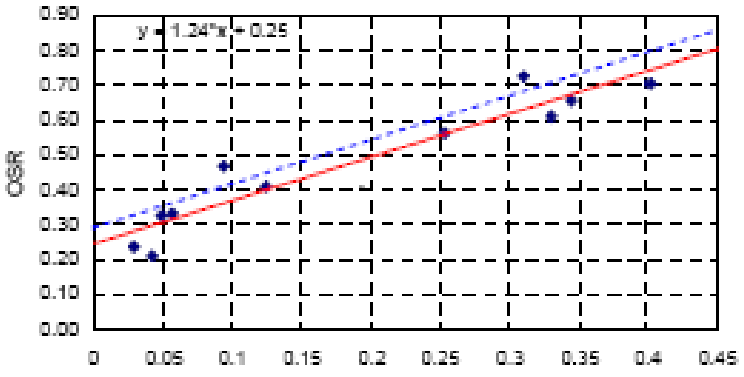

Figura 45- Um modelo linear de ESR e OSR

Fonte: Ref [31]

PUC-Rio - Certificação Digital Nº 0421021/CA

Finalmente, formula-se um modelo linear de ESR e OSR como segue:

$$OSR = 1{,}24 \times ESR \pm 0{,}3 \qquad (5.3)$$

Como o objetivo aqui é permitir uma pequena degradação na qualidade de vídeo às custas de um menor tempo de processamento, uma primeira modificação no método apresentado em [31], realizada neste trabalho, foi a seguinte:

Variou-se empiricamente, mas de forma que a reta se encontre na região entre os pontos, o coeficiente angular e linear da equação 5.3 nos intervalos de [1,1; 1,26] e [0,12 ; 0,28], respectivamente, com variação de 0,1 para três seqüência (akiyo.qcif, foreman.qcif e container.cif). A cada par de coeficientes, o tempo de processamento era medido e a equação que forneceu, na média, o menor tempo de estimação de movimento e, conseqüentemente, menor tempo total de processamento, associado à pequena redução do desempenho em termos de taxa-distorção foi a seguinte:

$$OSR = 1{,}123 \times ESR \pm 0{,}2689 \qquad (5.4)$$

5.4.3
Ajuste dinâmico do limite de SAD e a segunda modificação do algoritmo proposto

Aqui descreve-se um procedimento de programação para o ajuste dinâmico do limite de SAD. Este procedimento foi inserido no código fonte do método P021 disponível em http://ftp3.itu.int/av-arch/jvt-site/, mais especificamente nos módulos global.h, image.c e mv-search.c. Por sua vez P021, como descrito anteriormente, é uma adaptação do software de referência JM. Há que se destacar o grande tempo gasto durante a pesquisa para o entendimento do código fonte do software de referência JM e do método P021, necessários nesta fase. As variáveis usadas na implementação e apresentadas no pseudo-código a seguir são as seguintes:

Block: variável que determina o número de blocos no quadro;

Block_Search: variável que determina o número de blocos que sofrem pesquisa com precisão de subpixel ;

PUC-Rio - Certificação Digital Nº 0421021/CA

Block_Useful: variável que determina o número de blocos que apresentam uma pesquisa com precisão de sub-pixel efetiva;

SAD_Threshold: variável que determina o valor do limite de SAD;

SAD0: constante que apresenta o valor de SAD inicial (tipicamente igual a 600);

MinCost: variável que determina o valor de SAD após a pesquisa com precisão de pixel inteira.

As alterações no módulo mv_search foram referentes à:

- inicialização das variáveis *Block, Block_Search, Block_Useful, SAD_Threshold, e SAD0;*
- verificação de valores das variáveis Min Cost e SAD_Threshold ao final de cada macrobloco
- verificação de valores das variáveis ASR, ESR e OSR ao final de cada quadro.

PUC-Rio - Certificação Digital Nº 0421021/CA

O pseudo-código está apresentado a seguir:

Passo 1: Ajustamos:

Block = 0;

Block_Search = 0;

Block_Useful = 0;

SAD_Threshold = *SAD0.*

Passo 2: Para cada macrobloco no quadro

Block++;

Realiza a pesquisa de pixel inteira, obtendo *MinCost;*

Se *MinCost* > *SAD_Threshold* então

Realiza a pesquisa com precisão de sub-pixel, *Block_Search*++;

Se a pesquisa com precisão de sub-pixel é efetiva, *Block_Useful*++.

Passo 3: Fazemos

ESR = *Block_Useful* / *Block*;

ASR = *Block_Search* / *Block;*

Calculamos *OSR usando a* Eq. (5.4)

Passo 4: Ajustamos *SAD_Threshold* para o próximo quadro usando a equação (5.5) que será apresentada a seguir.

Pode-se ajustar o SAD_Threshold no passo 4 de duas formas. Uma forma consiste em ajustar o limite com um tamanho de passo fixo. Contudo, o resultado é muito sensível ao tamanho de passo selecionado. Se o tamanho de passo é pequeno, o número de pesquisas com precisão de subpixel será alto e a velocidade de conversão será muito baixa. De outra forma, se o tamanho de passo é grande, podem ocorrer oscilações. Outra forma, que foi utilizada em [31], deve adaptativamente mudar o tamanho de passo em função da diferença entre ASR e OSR. Em [31], a equação de ajuste do limite de SAD é apresentada a seguir. Ainda, segundo [31], o esforço computacional para o cálculo do limite de SAD é desprezível.

$$SAD\_Threshold_{new} = SAD\_Threshold_{i-1}\left(1+\frac{ASR-OSR}{2xOSR}\right) \qquad (5.5)$$

Uma segunda modificação no algoritmo proposto em [31] foi a seguinte. Durante o processo de codificação, observou-se que, ao final de cada quadro, a taxa de pesquisa efetiva (ESR) assumia valores bastante pequenos e a taxa de pesquisa atual (ASR) valores bastante elevados, conforme apresentado na Tabela 5. Esta tabela ilustra um exemplo destes valores para a seqüência akiyo.qcif (disponível em http://www.video-processing.pe.kr/, com resolução de 176x144, freqüência de quadros de 10 Hz, quadros IPPP (baseline profile), RDOptimization = 1 e 298 quadros sendo 100 quadros a serem codificados). A definição destes parâmetros será apresentada no item 6.1. Por simplicidade, foi mostrado apenas os valores para os 30 primeiros quadros:

Quadro	ESR	ASR	OSR
1	0	0.003203	0.2689
2	0	0.008623	0.2689
3	0.000246	0.016753	0.2691
4	0.021434	0.069722	0.2929
5	0.056418	0.1507	0.3322
6	0.05912	0.1736	0.3353

PUC-Rio - Certificação Digital Nº 0421021/CA

7	0.044592	0.149791	0.3189
8	0.025622	0.1818	0.2976
9	0.028	0.2384	0.30044
10	0.045331	0.2197	0.3198
11	0.039911	0.2370	0.3137
12	0.014	0.2668	0.2852
13	0.03153	0.2606	0.3043
14	0.03941	0.2357	0.3131
15	0.0559	0.3146	0.3317
16	0.00024	0.208	0.2691
18	0.0280	0.256	0.3004
19	0.0266	0.2702	0.2987
20	0.0298	0.3284	0.3023
21	0.01186	0.33752	0.2813
22	0.008623	0.2823	0.2785
23	0.0300	0.3318	0.3026
24	0.0266	0.3530	0.2987
25	0.0174	0.2658	0.2885
26	0.012	0.3197	0.2824
27	0.0157	0.329	0.2866
28	0.0182	0.2599	0.2893
29	0.0640	0.3217	0.3408
30	0.0640	0.3217	0.3408

Tabela 5 – Valores de ESR, ASR e OSR para os 30 primeiros quadros codificados da seqüência akiyo.qcif

Para reduzir o valor de ASR deve-se aumentar o valor do limite de SAD, conforme apresentado em 5.4.3. Pela análise da equação 5.5 e dos dados da Tabela 5, isto só seria possível se for alterado o fator 2 de multiplicação de OSR . Variou-se empiricamente este valor, no intervalo de [1,5;4,0] com variação de 0,1 para três seqüência (akiyo.qcif, foreman.qcif e container.cif), com 100 quadros codificados e freqüência de 10 Hz. As características destas seqüências serão apresentadas no item 6.2. Para cada uma das seqüências usou-se o parâmetro QP

PUC-Rio - Certificação Digital Nº 0421021/CA

igual a 8, 18, 28 e 38 e as configurações com RDO=0 e IPPP, RDO=0 e IBBP, RDO=1 e IPPP, RDO=2 e IPPP. Os parâmetros QP, RDO, IPPP e IBBP serão definidos no item 6.1. Os tempos totais de processamento foram medidos, e as médias destes tempos para cada fator multiplicador está apresentado na Figura 46. A partir desta Figura, observou-se que usando o fator 3,0, este apresentava a menor média dos tempos totais de processamento, associado a desempenho em termos de taxa-distorção similar ao dos outros fatores multiplicadores.

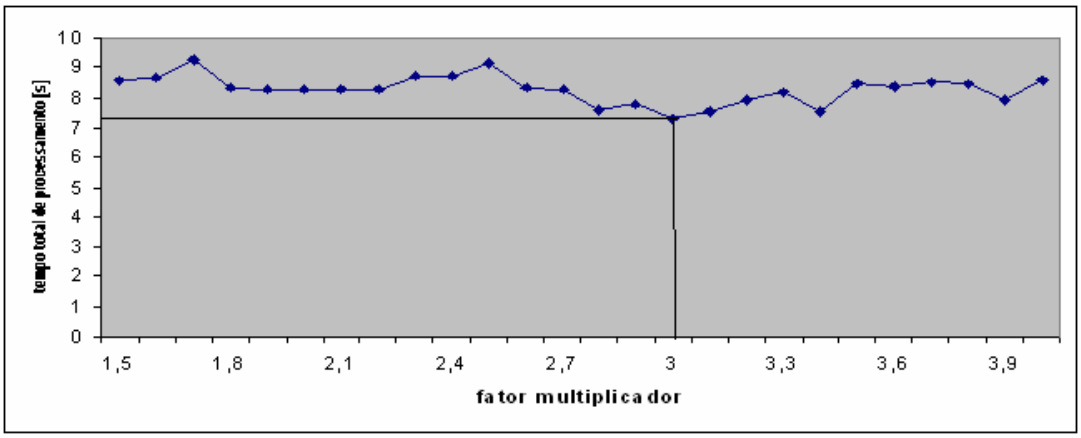

Figura 46 – Média dos tempos totais de processamento para cada fator multiplicador da Eq. 5.5.

Assim, a equação 5.5 é modificada e será usada neste método proposto como segue:

$$SAD\_Threshold_{new} = SAD\_Threshold_{i-1}\left(1+\frac{ASR-OSR}{3xOSR}\right) \qquad (5.6)$$

5.5
Resumo e conclusão do capítulo

Neste capítulo foram apresentados e discutidos os seguintes métodos rápidos utilizados para estimação de movimento: Full Search, JM98, P021 e o método proposto.

A estimação de movimento para CODEC H.264/AVC é computacionalmente custosa se o método Full Search é usado. Este método

PUC-Rio - Certificação Digital Nº 0421021/CA

consiste em verificar em todos os pontos da janela de pesquisa, qual apresenta o menor SAD, ou seja, o menor resultado da equação 2.3, conforme apresentado na Figura 35. A fim de reduzir o tempo de codificação, o atual software de referência JM adota um método rápido de estimação de movimento para pixel inteiro chamado UMHexagonS e um método para pixel fracionário chamado CBFPS, que foram descritos neste capítulo. Um método proposto por Xiaoquan Yi, Jun Zhang, Nam Ling e Weijia Shang [2] aqui chamado, por simplicidade, de P021, apresenta, em comparação com o software de referência JM, uma forma melhor e simplificada de estimação de movimento para aumentar a velocidade do processo de codificação e manter o desempenho em termos da taxa-distorção. A partir do método P021, foi proposto um método de estimação de movimento que não realize o processo de estimação de movimento de meio-pixel para certos blocos baseado num modelo de seleção linear. Assim, obtém-se uma forma de estimação de movimento com velocidade de codificação superior e desempenho em termos da taxa-distorção semelhante, se comparado aos três métodos acima (Full Search, JM98 e P021). Este método sofreu duas modificações em relação à sua forma originalmente apresentada em [31] com o objetivo de diminuir o custo computacional, preservando o desempenho em termos de taxa-distorção. As modificações foram introduzidas nas equações (5.3) e (5.5).

PUC-Rio - Certificação Digital Nº 0421021/CA