Galaxy Random Search: Algoritmo e Arquitetura para Estimação de Movimento em Vídeos de Alta Definição

Resumo. *Atualmente os vídeos em alta definição vêm sendo utilizados em larga escala. Estes vídeos apresentam uma alta incidência de mínimos locais, o que tem influência direta nos resultados da Estimação de Movimento (ME - Motion Estimation). Esta característica prejudica os resultados de qualidade dos algoritmos rápidos iterativos para a ME, que são mais facilmente atraídos a mínimos locais, degradando a qualidade final do processo. O processo de estimação de movimento deve ter um compromisso entre o custo computacional, a taxa de compressão e a qualidade final obtida. Para atender a esta expectativa, a eficiência do algoritmo de busca usado na ME é muito importante. Neste artigo, é proposto um novo algoritmo de busca para a ME em vídeos de alta definição, chamado Galaxy Random Search (GRS). Este algoritmo utiliza a aleatoriedade como forma de evitar a escolha de mínimos locais, obtendo, assim, melhores resultados de qualidade. Este artigo também apresenta o projeto arquitetural para o algoritmo GRS. Esta arquitetura foi projetada para ser implementada em FPGA e processar vídeos HD 1080p em tempo real (30 quadros por segundo).*

1. Introdução

Atualmente, vídeos em alta definição vêm sendo utilizados em larga escala e o número de dispositivos que os utilizam tem crescido no mercado mundial. Essa popularização ocorre, principalmente, devido à qualidade visual obtida com o aumento de resolução, aumentando a satisfação dos usuários. Entretanto, apesar da grande quantidade de dispositivos que suportam aplicações neste tipo de resolução de vídeo, existe uma grande variedade em suas características. Em geral, estes dispositivos têm a demanda por soluções que visam pequena área física, baixo consumo de potência e alta qualidade. Neste contexto, tem aumentado o esforço de pesquisa na academia e na indústria, buscando soluções eficientes que satisfaçam estas necessidades.

A quantidade de bits necessários para representar toda informação contida em um vídeo digital de alta definição, sem compressão, é extremamente elevada. Isso torna inviável a transmissão, o processamento e o armazenamento desses vídeos. Então é necessário que esses vídeos sejam codificados, onde suas redundâncias de informação são exploradas e é possível reduzir drasticamente o volume de dados usados na representação do vídeo. Os codificadores de vídeo exercem um papel fundamental para o avanço na utilização de vídeos com maiores resoluções, visto que atingem taxas de compressões elevadas e, consequentemente, permitem o uso de vídeos de maior resolução.

A Estimação de Movimento (ME - *Motion Estimation*) é a etapa mais importante dos codificadores de vídeo atuais. Esta etapa representa 80% do esforço computacional [Puri 2004] dos codificadores. Contudo, é na ME onde se concentram os maiores ganhos de compressão. A ME visa reduzir a redundância temporal entre imagens vizinhas de um vídeo. O desempenho da estimação de movimento está diretamente relacionado com o algoritmo de busca utilizado. Os algoritmos de ME se dividem em duas classes: ótimo e sub-ótimos (rápidos). O algoritmo *Full Search* (FS) [Puri 2004] é o único que apresenta o resultado de qualidade ótimo, visto que ele avalia todos os blocos candidatos dentro de uma área de pesquisa. Entretanto, seu custo computacional é extremamente elevado. Os algoritmos sub-ótimos utilizam heurísticas para acelerar a convergência da ME. O custo computacional é reduzido de forma expressiva em relação ao algoritmo FS, no entanto, a aceleração os torna suscetíveis a escolha de mínimos locais. Dessa forma, quando o mínimo global não é encontrado, a qualidade da estimação de movimento sofre perdas em relação ao resultado ótimo. No entanto, algoritmos de busca atuais visam aliar baixo custo em complexidade com alto desempenho em termos de qualidade.

A codificação de vídeos digitais em alta definição é um processo complexo, que demanda altas taxas de processamento, principalmente para aplicações em tempo real (30 quadros por segundo). Um computador pessoal até pode codificar e decodificar vídeos Full HD 1080p em software e em tempo real. Entretanto, um sistema embarcado como um terminal de acesso de TV digital, por exemplo, não possui um processador de última geração e, assim, esse tipo de dispositivo não consegue atingir o desempenho mínimo para a codificação. Para dispositivos com restrições que não permitem a codificação em software, é necessária a utilização de arquiteturas dedicadas que codificam os vídeos em hardware. Neste caso, o projeto em hardware deve levar em conta as restrições de hardware impostas por cada dispositivo.

Neste trabalho, será apresentado um novo algoritmo rápido para a estimação de movimento em vídeos de alta definição, chamado de *Galaxy Random Search* (GRS). Este algoritmo utiliza a escolha aleatória de blocos candidatos dentro da área de busca como estratégia para evitar a escolha de mínimos locais. O GRS também explora características importantes da estimação de movimento, como a questão da localidade (explorando a região central da área de pesquisa) e também o refinamento iterativo (proporcionado a convergência para regiões de maior similaridade após a etapa aleatória). Este trabalho também apresenta uma proposta arquitetural para o algoritmo GRS, visando o processamento de vídeos *full* HD (1920x1080 pixels) em tempo real (30 quadros por segundo).

Este artigo está organizado da seguinte forma: Na seção 2 são apresentados conceitos sobre codificação de vídeo. Essa seção também mostrará a etapa de estimação de movimento assim como alguns algoritmos que a compõem, além das diferenças desses algoritmos na compressão de vídeos de alta definição. Na seção 3 é apresentado o algoritmo GRS. Nesta seção, também são avaliados os resultados de qualidade e custo computacional do algoritmo proposto, bem como as comparações com outros algoritmos da literatura. Em seguida é apresentado, na seção 4, o projeto da arquitetura e as comparações com outros trabalhos. Na seção 5 são apresentadas as conclusões deste artigo.

2. Codificação de vídeo

Um vídeo digital é composto por uma sequência de imagens estáticas, denominadas de quadros. Para gerar sensação de movimento ao olho humano, são necessários que aproximadamente 30 quadros sejam apresentados a cada segundo. Cada quadro é divido em blocos, que são compostos por um conjunto de pontos (pixels). Para vídeos coloridos, os pixels são representados em um espaço de cores, como RGB ou YCbCr. Neste caso, cada pixel é formado por três amostras distintas, que indicam a cor daquele ponto da imagem. As amostras representam a menor unidade de informação em um vídeo.

Imagens vizinhas normalmente são muito similares, devido à taxa de amostragem de aproximadamente 30 quadros por segundo. Esta similaridade entre os quadros vizinhos é denominada redundância temporal [Ghanbari 2003]. O aumento na capacidade de captação das câmeras filmadoras foi observado durante a realização da copa do mundo da África, onde "super câmeras" captavam milhares de quadros por segundo, exibindo detalhadamente cada movimento de bola e de jogadores. Essa tecnologia também é muito utilizada em programas de televisão, para filmagem de detalhes que, até então, eram difíceis de serem observados, como por exemplo, o impacto de uma bala em uma superfície ou o estouro de uma bolha de sabão. Esta tecnologia também será utilizada durante a realização da copa do mundo no Brasil. O aumento na quantidade de quadros captados por segundo também acarreta em um aumento na redundância de informações, pois a similaridade entre os quadros vizinhos aumenta significativamente. Sendo assim, são necessários codificadores de vídeo eficientes para reduzir essa grande quantidade de dados, mantendo a qualidade visual dos vídeos codificados.

A Figura 1 apresenta a divisão de um vídeo digital em uma sequência de quadros e a divisão de cada quadro em blocos. Os blocos são conjuntos de pixels, normalmente de tamanhos 4x4, 8x8 ou 16x16 pixels. Um quadro pode ser visto como uma matriz, e quanto maior o tamanho dessa matriz, maior será a resolução da imagem e, consequentemente, maior será sua qualidade.

Uma sequencia de 10 minutos de um vídeo *full* HD apresentado a 30 quadros por segundo gera uma quantidade de 112GB de dados. Esse elevado volume de dados a ser processado, armazenado ou transmitido requer altas taxas de compressão. Na compressão com perdas, utilizado em compressores de vídeos atuais, as informações de baixa relevância visual são eliminadas [Richardson 2003], alcançando elevadas taxas de compressão com uma pequena redução na qualidade visual, que muitas vezes é imperceptível.

2.1. Estimação de Movimento (ME)

Para identificar e reduzir (ou até mesmo eliminar) a redundância temporal entre os quadros vizinhos de um vídeo, a ME utiliza uma área de pesquisa ou área de busca, de forma a comparar os blocos do quadro atual em relação a um quadro de referência (anteriormente processado). A área de pesquisa é formada por um conjunto de blocos (blocos candidatos) e um limite (*range)* de pixels indica a fronteira da área.

A menor diferença, ou a maior similaridade (melhor casamento), entre o bloco atual e os blocos candidatos de uma área de busca é obtida com a aplicação de um critério de similaridade. O critério de similaridade utilizado neste trabalho é a Soma das Diferenças Absolutas (*Sum of Absolute Differences* - SAD) [Richardson 2003]. Quando o bloco com menor SAD é encontrado, um vetor de movimento é gerado para indicar a posição desse bloco na área de busca.

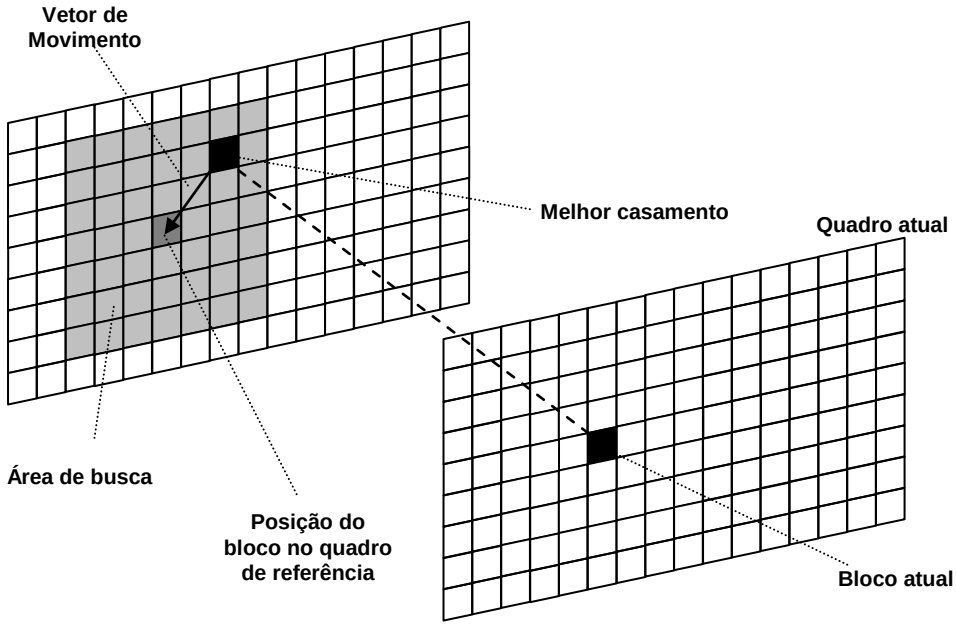

Figura 1. Elementos que compõe a estimação de movimento

O algoritmo de busca determina a forma em que a análise será realizada dentro da área de busca. Existem diversos algoritmos de busca publicados na literatura, cada um deles com características distintas. A escolha do algoritmo influencia diretamente a qualidade dos vetores de movimento gerados. Os algoritmos utilizados pela ME devem ter um compromisso entre qualidade e custo computacional, de modo a obter uma boa compressão do vídeo, permitindo que seja possível a transmissão ou reprodução de vídeos em alta definição em tempo real. Por este motivo, os esforços no desenvolvimento de novos algoritmos de ME vem sendo um tema de grande relevância na comunidade científica atualmente.

A etapa de estimação de movimento apresenta a maior complexidade computacional dos codificadores de vídeo atuais, no entanto, é a etapa que gera os maiores ganhos em compressão. Após a geração dos vetores de movimento pela ME, esses vetores são utilizados pela etapa de compensação de movimento (*Motion Compensation* - MC), etapa responsável por remontar os quadros a partir dos vetores de movimento gerados pela ME.

2.1.1. Estimação de movimento em vídeo de alta definição

Vídeos de alta definição possuem uma quantidade muito maior de pixels para representar a mesma informação do que vídeos de baixa definição. Por esse motivo, muitos algoritmos rápidos não conseguem manter bons resultados para vídeos de alta

definição. Para ilustrar essa diferença, a Figura 2 apresenta um gráfico com as curvas de qualidade média em termos de PSNR para o algoritmo ótimo FS e para o algoritmo rápido *Diamond Seach* (DS) [Zhu 2000] para 10 sequências de teste *full* HD: blue_sky, man_in_car, pedestrian_area, rush_hour, station2, sun_flower, riverbed, rolling_tomatoes, traffic e tractor [Xiph.org 2011]. O algoritmo DS foi escolhido por ser um dos algoritmos rápidos mais eficientes e utilizados na literatura.

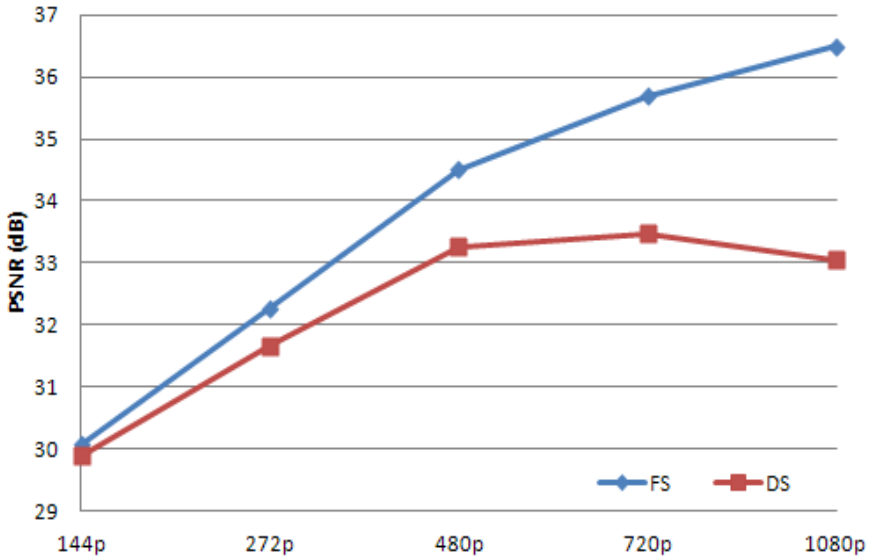

Figura 2. Diferença de qualidade para os algoritmos FS e DS em diferentes resoluções

A Figura 2 ilustra as curvas avaliando as 10 sequencias de teste em cinco diferentes resoluções: 144p (256x144 pixels), 272p (480x272 pixels), 480p (864x480), 720p (1280x720 pixels) e 1080p (1920x1080 pixels), utilizando diferentes áreas de busca (*ranges*) para cada resolução: [-16,+16], [-24, + 24], [-32, +32], [-64, +64] e [-128, + 128] respectivamente.

Através da Figura 2 é possível notar um aumento contínuo na qualidade para o algoritmo FS com o aumento da resolução do vídeo, pois com o aumento da área de busca, o algoritmo FS pode explorar um número maior de blocos candidatos, aumentando as chances de encontrar um bloco de menor SAD. No entanto, para algoritmos rápidos como o DS, esse aumento da área de pesquisa não gera os mesmos benefícios. Observando o gráfico da Figura 2 pode-se perceber que, para a menor resolução, o algoritmo DS consegue obter bons resultados de qualidade, apenas 0,2 dB inferior ao obtido pelo algoritmo FS. No entanto, essa diferença em qualidade aumenta de acordo com o aumento da resolução do vídeo, chegando a uma diferença de 3,45 dB na resolução mais elevada (*full* HD). Esses dados demonstram que a eficiência do algoritmo DS diminui de acordo com o aumento da resolução. Isso ocorre, pois a heurística utilizada durante a busca satisfaz a sua condição de parada muito cedo, fazendo como que o algoritmo fique preso a mínimos locais próximos ao centro da área de busca e impedindo a obtenção do melhor casamento dos blocos (mínimo global).

2.1.2. Mínimos locais

A perda de eficiência dos algoritmos rápidos em relação ao algoritmo FS com o aumento da resolução está diretamente ligada ao aumento das ocorrências de escolha de mínimos locais. O aumento da área de pesquisa permite que o algoritmo possa obter blocos com menor SAD mais distantes do centro da área de pesquisa. A Figura 3 ilustra um mapa de SAD, composto por um gráfico em três dimensões (3D) com os valores de SAD para a sequência de teste sun_flowers (*full* HD) para uma área de pesquisa de 128x128 pixels e utilizando blocos de 16x16 pixels. Valores altos de SAD são representados por picos, enquanto que valores baixos de SAD são representados por vales.

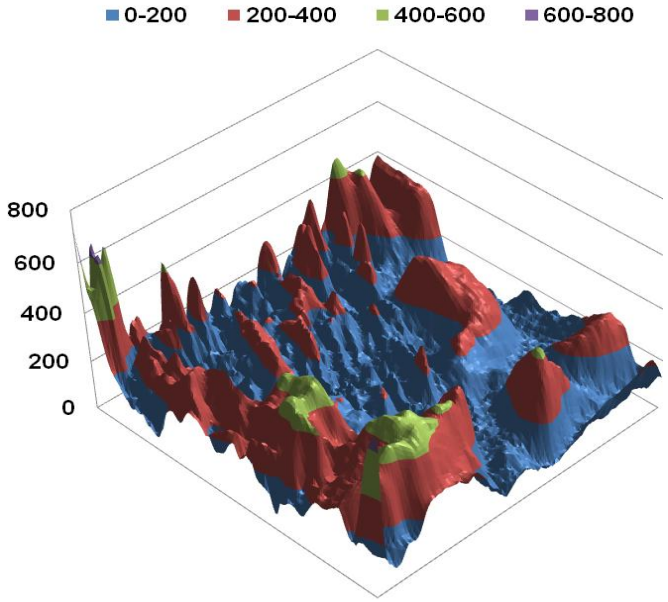

Figura 3. Mapa de SAD para uma área de busca em *full* HD

Analisando a Figura 3 é possível observar a grande ocorrência de picos e vales. A maioria dos algoritmos rápidos inicia sua heurística de busca no centro da área de pesquisa e itera algum padrão de busca enquanto menores resultados de SAD forem sendo encontrados. Esta característica acaba restringindo os algoritmos rápidos a escolha de mínimos locais próximos a região central da área de busca, visto que sua heurística de busca impossibilita a transposição de um pico de SAD. Isto torna impossível a escolha de um mínimo global que esteja localizado após um pico de SAD em volta da região central da área de busca.

A Figura 4 é utilizada para melhor demonstrar a homogeneidade da área de pesquisa em diferentes resoluções e para demonstrar o efeito que essa característica acarreta em algoritmos rápidos. As Figuras 4(a), (b), (c), (d) e (e) apresentam os mapas de SAD (em forma de mapas de calor) para o vídeo sun_flower nas resoluções 144p, 272p, 480p, 720p e 1080p, respectivamente. O tamanho de bloco utilizado foi 16x16 pixels e o tamanho da área de busca para cada resolução foi o mesmo utilizado para o gráfico da Figura 2.Tons escuros representam valores baixos de SAD, enquanto que valores altos de SAD são representados por tons na cor vermelha.

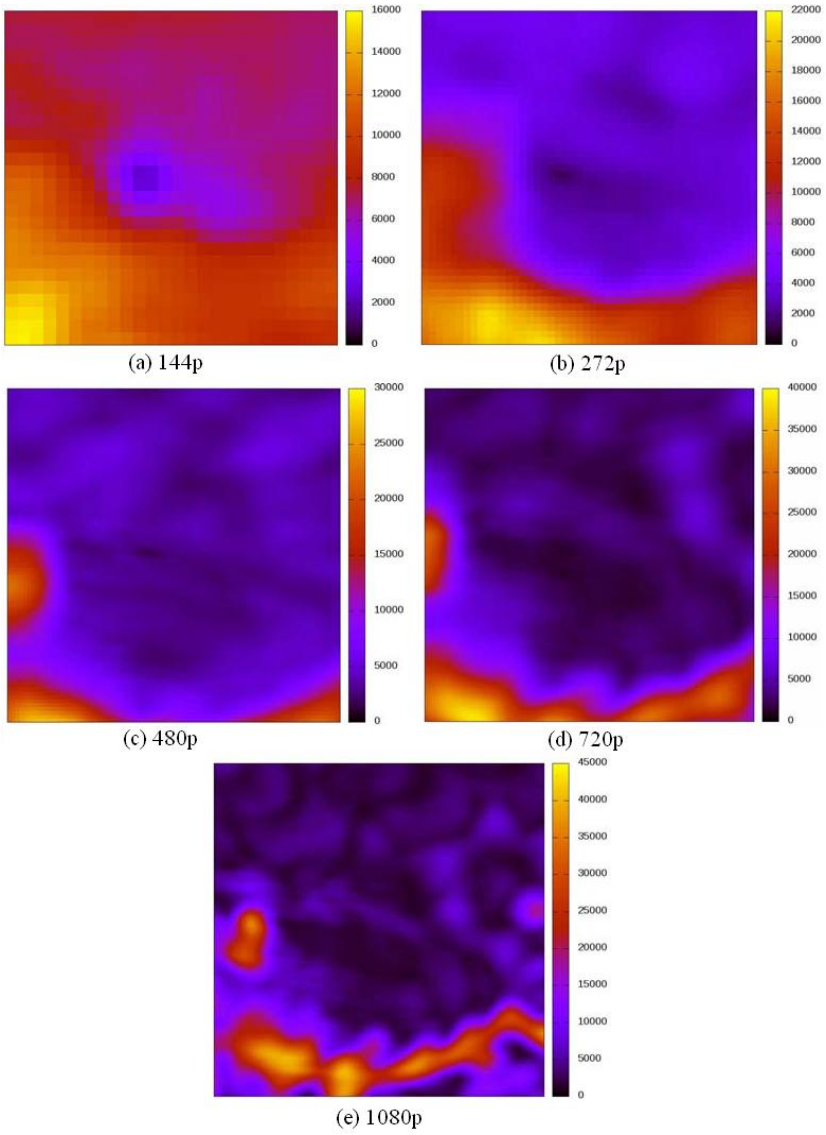

Figura 4. Mapas de SAD para diferentes resoluções de vídeo

A partir da observação da Figura 4(a) é possível notar que os blocos com valores baixos de SAD concentram-se na região central da área de busca. Isto permite que o algoritmo DS consiga avaliar blocos muito próximos (ou iguais) aos blocos avaliados pelo algoritmo ótimo FS. Porém, conforme ilustrado nas Figuras 4(b), (c) e (d), o aumento da resolução resulta em uma maior homogeneidade da área de pesquisa, aumentando assim o número de mínimos locais e tornando mais difícil a busca do bloco com menor valor de SAD (mínimo global) por algoritmos rápidos. Por outro lado, o algoritmo FS sempre encontrará o bloco ótimo, pois explora todos os blocos candidatos da área de pesquisa.

A Figura 4(e) ilustra a área de pesquisa de um vídeo *full* HD (1080p), foco deste trabalho. É possível observar a grande quantidade de mínimos locais, e a necessidade da avaliação e desenvolvimento de novas heurísticas algorítmicas, de modo a transpor picos de SAD com o objetivo de encontrar o mínimo global, mantendo os resultados de qualidade próximos aos obtidos pelo algoritmo ótimo FS.

3. O algoritmo GRS

A estimação de movimento visa encontrar as redundâncias temporais dos vídeos digitais, reduzindo o volume de informação com pequenas, ou até imperceptíveis, perdas na qualidade visual. No entanto, a ME é a etapa mais complexa dos codificadores de vídeo atuais. Os algoritmos de ME devem ser rápidos, apresentando baixa complexidade computacional, e devem gerar distorções mínimas ou nulas na qualidade dos vídeos. O algoritmo *Galaxy Random Search* (GRS), utiliza a aleatoriedade como forma de reduzir a escolha dos mínimos locais. A escolha aleatória de blocos candidatos dentro de uma área de busca possibilita ao algoritmo vencer eventuais picos de SAD, que seriam barreiras instransponíveis apenas com refinamento iterativo. Contudo, a busca aleatória aplicada de forma isolada não considera diversas características importantes da estimação de movimento. Além da busca aleatória, o algoritmo GRS realiza uma busca iterativa na região central da área de busca, que garante bons resultados para vídeos que apresentam baixa movimentação, onde bons vetores se encontram próximo ao centro. Estas duas etapas do algoritmo serão descritas detalhadamente nos próximos parágrafos.

No primeiro passo, o algoritmo explora duas características importantes na estimação de movimento. A primeira é que blocos candidatos que estão ao centro da área de busca tendem a gerar bons resultados de resíduo. Isso ocorre, pois considerando o bloco atual, o bloco candidato do quadro de referência situado na mesma posição espacial, tende a representar a mesma região no quadro. Em vista desta característica, o GRS explora o bloco central, bem como seus quatro vizinhos. É realizado o cálculo do SAD para estes blocos. A segunda característica é o refinamento iterativo, que deve ser repetido até que dada condição de parada seja satisfeita. Isso faz com que os algoritmos possam convergir para regiões que apresentam menores valores de SAD dentro da área de busca, através de n iterações. Porém, esta abordagem, se utilizada de forma isolada, apresenta grande suscetibilidade à escolha de mínimos locais. Após a avaliação estática no centro, é iniciado o processo iterativo. Se algum vizinho apresentar resultado melhor que o bloco do centro, este passa a ser o bloco central da próxima iteração. Da mesma forma que no passo anterior, seus quatro vizinhos são analisados. As iterações têm fim quando o bloco central apresenta melhor resultado que seus vizinhos. Esse processo iterativo possibilita ao algoritmo a convergência para uma região de menor SAD.

No segundo passo, o algoritmo GRS sorteia N blocos candidatos aleatoriamente dentro da área de busca. Essa etapa utiliza uma função randômica para realizar o sorteio. Esta estratégia visa aumentar as chances do algoritmo iniciar sua busca em uma região além dos picos de SAD, que seriam intransponíveis com algoritmos iterativos. Após o sorteio, é escolhido o bloco com melhor SAD, dentre os N blocos candidatos sorteados na área de busca. Esta região é explorada por um refinamento final iterativo que busca identificar blocos com menor resíduo. Este refinamento é idêntico ao aplicado ao centro da área de busca, no primeiro passo. Para isso, o mesmo padrão é adotado, avaliando o bloco de menor SAD como sendo o centro do padrão e seus quatro vizinhos. Ao final do refinamento, o melhor bloco candidato do segundo passo é encontrado. Por fim, é realizada uma comparação entre o melhor bloco do primeiro e do segundo passo, e o bloco de menor SAD é escolhido.

A Figura 5 apresenta o fluxograma do algoritmo GRS. Nela, os dois passos do algoritmo são realizados sequencialmente. Entretanto, estas duas etapas podem ser

executadas em paralelo, já que não existem dependências de dados entre elas. Assim, aplicações paralelas em software (usando multi-cores ou GPU) ou arquiteturas dedicadas em hardware podem explorar o paralelismo para aumentar a taxa de processamento. Quando os melhores blocos de ambos os passos estão definidos, uma comparação final identifica o melhor resultado da área de pesquisa processada.

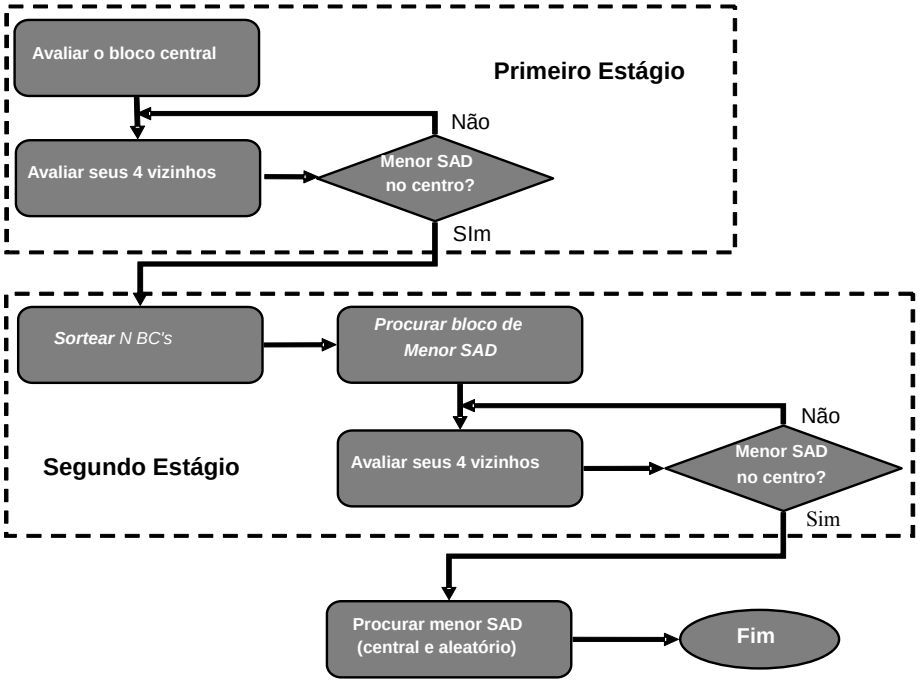

Figura 5. Fluxograma do algoritmo GRS

A Figura 6 apresenta um exemplo de busca do algoritmo GRS para $N = 6$. Nesta figura podem ser observadas as duas etapas presentes no algoritmo. Na primeira etapa o GRS executa o refinamento no centro da área de busca, indicado pelos blocos "c" na Figura 6, destacados em laranja. O bloco central e seus vizinhos (blocos "c_0" na Figura 6) são comparados. A iteração tem inicio porque o vizinho direito apresenta um SAD menor. Contudo, já na primeira iteração (blocos "c_1" na Figura 6) o processo se encerra devido ao centro apresentar o melhor resultado. Assim, o bloco identificado com "C" é o melhor bloco obtido no primeiro passo.

Na segunda etapa, o algoritmo sorteia N blocos candidatos aleatoriamente, marcados com a letra "a" na Figura 6, e destacados em verde. Neste caso, serão seis blocos sorteados. O bloco identificado com "A" na Figura 6 é o bloco que apresentou o menor SAD dentre os seis blocos candidatos sorteados. Sobre este bloco é realizado o refinamento final, destacado em azul na Figura 6. Na primeira etapa do refinamento, os seus quatro blocos vizinhos ao bloco "A" são avaliados (blocos "r0" na Figura 6). O processo iterativo tem inicio pois o vizinho direito (bloco "R0" na Figura 6) obteve melhor resultado que o bloco "A" (centro do refinamento). Com isso, o bloco "R0" passa a ser o novo centro e o algoritmo itera com a avaliação de seus três vizinhos (blocos "r1" na Figura 6). Novamente, o melhor resultado desta iteração também foi encontrado no bloco à direita. Na segunda iteração, após a comparação do centro com os vizinhos (blocos "r2" na Figura 6), é encontrado o melhor resultado no centro (bloco

“R” na Figura 6), e a busca acaba. Este bloco central é o melhor bloco do segundo passo.

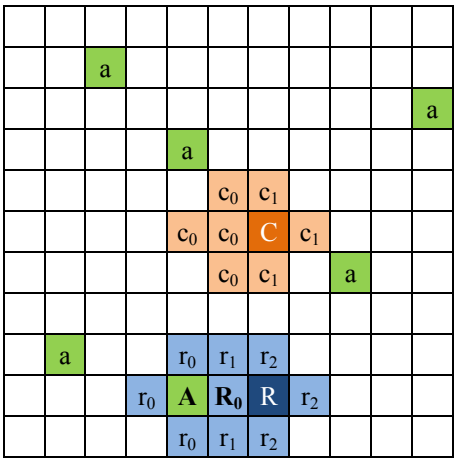

Figura 6. Exemplo para o algoritmo GRS com N=6 e área hipotética

A última etapa do GRS é a identificação do melhor bloco entre os resultados da avaliação central (bloco C) e do refinamento final (bloco R). O bloco com menor SAD será escolhido e para ele será gerado o vetor de movimento.

Uma característica do algoritmo GRS é a geração de vetores diferentes caso a estimação seja aplicada mais de uma vez para a mesma área de pesquisa. Isto ocorre devido à etapa aleatória, que pode escolher blocos candidatos diferentes para uma nova execução sobre a mesma área de pesquisa. No entanto, uma avaliação do desvio médio demonstrou que esta variação é muito pequena, a ponto de ser desprezada.

3.1. Avaliação de Qualidade

O algoritmo proposto foi implementado na linguagem C e avaliado em um *framework* contendo a ME e também a MC, assim os quadros remontados podem ser comparados com os originais e os resultados de qualidade da ME podem ser avaliados de modo isolado, não considerando as demais etapas de um codificador de vídeo. As avaliações foram realizadas para os duzentos frames iniciais de dez sequências de teste *full* HD [Xiph.org 2011]. Os videos escolhidos foram: (a) blue_sky, (b) man_in_car, (c) pedestrian_area, (d) rush_hour, (e) station2, (f) sun_flower, (g) riverbed (h) rolling_tomatoes, (i) traffic e (j) tractor.

Alguns fatores podem influenciar diretamente no resultado obtido pela ME. Um fator determinante na qualidade do algoritmo GRS é o número de blocos candidatos sorteado. Valores maiores de *N* tendem a gerar melhores resultados de qualidade, visto que um maior número de blocos candidatos será avaliado. Outro fator importante é o tamanho da área de busca utilizada. Áreas maiores oferecem um número maior de blocos candidatos, possibilitando a escolha de blocos com maior qualidade. Entretanto, o aumento no tamanho da área pode ocasionar o aumento do custo computacional, além do tamanho da memória física para armazená-la.

A Figura 7 apresenta um gráfico com os resultados de qualidade do algoritmo GRS, considerando a variação do tamanho da área de busca e também o valor de *N*. O tamanho de bloco utilizado foi 16x16 pixels e cada uma das curvas representa o

resultado obtido para um dado valor de número de blocos sorteados (*N*). Ao analisar o gráfico, é possível perceber que conforme aumenta o número de blocos candidatos sorteados a qualidade também é incrementada. O crescimento de qualidade observado entre uma curva e outra é praticamente proporcional. Assim, se for levado em conta apenas a qualidade, o melhor resultado, para uma mesma área de pesquisa, será do maior *N*.

O range define a extensão da área de busca, em número de pixels, a partir do bloco atual centralizado. No gráfico da Figura 7 pode ser observado que para um *N* fixo, 32 por exemplo, a qualidade aumenta conforme é aumentado o range da área de busca. Isso ocorre, pois, na medida em que a área de pesquisa é aumentada, o algoritmo tem novas regiões no quadro para fazer a busca por blocos candidatos melhores. Entretanto, a qualidade tende a uma estabilização. A partir do range 48 [-48, +48], o algoritmo deixa de aproveitar as regiões mais afastadas do centro, geralmente porque os processos iterativos satisfazerem os critérios de parada antes mesmo de chegarem à nova área, aproveitando pouco os novos blocos disponíveis.

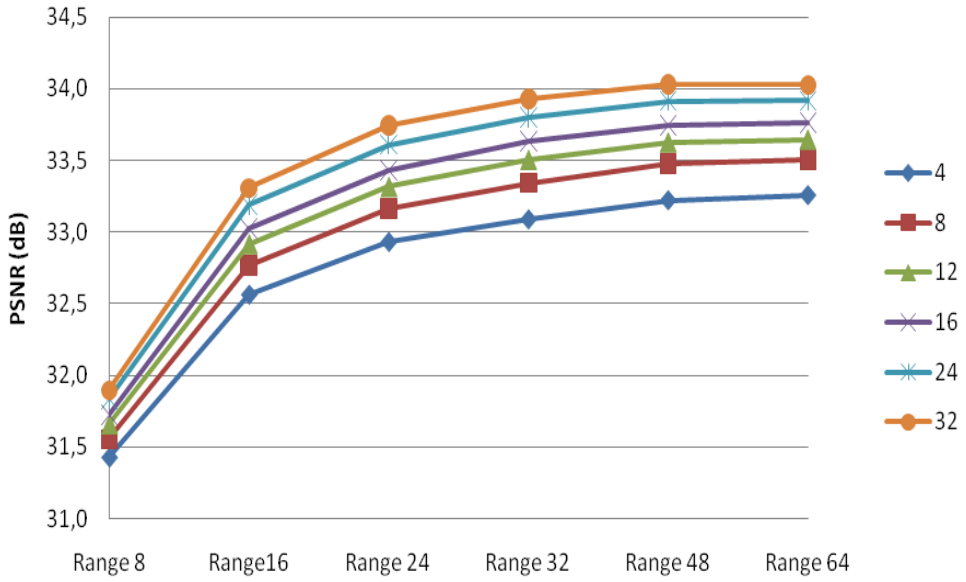

Figura 7. Resultados médios de PSNR para o algoritmo GRS com a variação do tamanho da área de busca e do valor de N

3.2. Avaliação de Custo Computacional

Os impactos, causados pela variação do número de blocos sorteados e do tamanho da área, também são perceptíveis no custo computacional. Estes impactos estão ilustrados na Figura 8, que apresenta uma avaliação do custo computacional do algoritmo GRS, medida em número de blocos candidatos calculados (BCC), na escala dos milhões ($x10^6$). Na Figura 8 é possível observar que conforme mais blocos candidatos são sorteados, o número de BCCs também cresce. Analisando o aumento da área de busca pode ser observado que ocorre uma estabilização, da mesma forma que acontece na Figura 7, a partir do range 48.

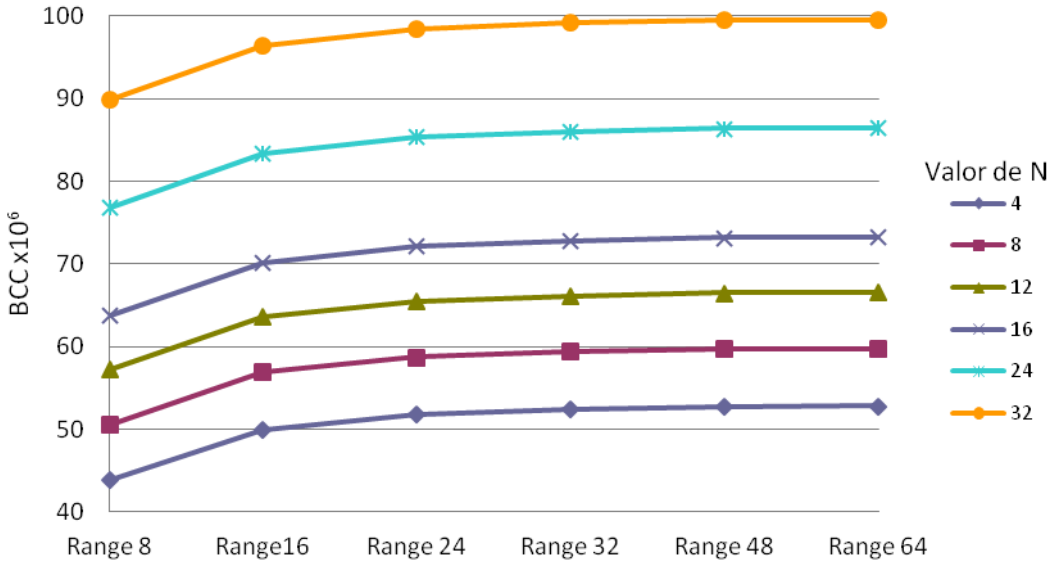

Figura 8. Resultados de número de BCCs para o algoritmo GRS com a variação do tamanho da área de busca e do valor de N

A Tabela 1 apresenta os resultados de incremento de PSNR e número de BCCs para vários valores de *N*, tomando como base os resultados obtidos para $N = 4$. A diferença em qualidade, entre o maior e o menor número de blocos sorteados, chega a 0,77 dB. Entretanto, para chegar à qualidade máxima, o custo é elevado. A taxa de crescimento no custo, até $N = 16$, é praticamente proporcional. Contudo, após esse ponto, a taxa tem uma variação maior e os ganhos em PSNR não justificam o aumento no número de cálculos. Com isso, foi definido que o valor de $N = 16$ apresentou o melhor custo-benefício para uso no GRS. Neste ponto, o ganho em qualidade foi de 0,5 dB, com um custo 38,6% superior ao $N = 4$.

Tabela 2. Variação de N

Sorteios (*N*)	PSNR (dB)	BCC(x10^6)
4	33,26	52,80
8	+0,25	+6,97
12	+0,39	+13,73
16	+0,50	+20,40
24	+0,66	+33,63
32	+0,77	+46,78

A variação do tamanho da área de busca também interfere no custo e na qualidade obtidos pelo algoritmo GRS. A Tabela 2 apresenta o comportamento do custo computacional e da qualidade conforme o *range* é variado, considerando $N = 16$. O *range* adotado como base neste caso é de oito pixels [-8, +8]. Com base na Tabela 2, é possível observar que o custo cresce a uma taxa mais lenta do que observado com a variação do valor de *N*. Porém, áreas de busca maiores fazem com que mais dados sejam carregados para a memória. Isso deve ser evitado quando os ganhos em qualidade são pequenos. O resultado médio de PSNR para o range de 48 apresenta um ganho de qualidade de apenas 0,01dB inferior ao obtido para o range 64, e o número de BCC é apenas 0,05x10^6 inferior. Neste caso, apesar do range de 64 pixels não apresentar um

aumento significativo no número de BCCs, também não resultou melhorias significativas em qualidade. Assim, o range de 48 pixels obteve o melhor custo-benefício para o algoritmo GRS com $N = 16$. Sua diferença de qualidade, em relação ao range de oito, foi de 2,07dB. O custo computacional representou um aumento de apenas 12,4%.

Tabela 2. Variação do Range

Range	PSNR (dB)	BCC(x10^6)
8	31,84	76,87
16	+1,35	+6,51
24	+1,77	+8,45
32	+1,96	+9,15
48	+2,07	+9,51
64	+2,08	+9,56

3.3. Resultados Comparativos

Os resultados obtidos para o algoritmo GRS foram comparados aos algoritmos FS, DS, HEX [Zhu 2002], TSS [Jing 2004] e FSS [Tasdizen 2009]. Na Tabela 3 são apresentados os resultados de qualidade, considerando o PSNR, e custo. O custo computacional é medido em número de blocos candidatos comparados (BCCs). Esta medida é a soma de todos bloco candidatos analisado por dado algoritmo.

Tabela 3. Resultados comparativos dos algoritmos GRS, FS, DS, HEX, FSS e TSS

Algoritmo	PSNR (dB)	BCC (x10^6)
FS	36,16	25.626,46
GRS	**33,74**	**73,20**
DS	33,06	36,58
HEX	32,79	32,52
FSS	32,40	58,03
TSS	30,94	43,51

O algoritmo proposto conseguiu melhorar o resultado de qualidade em relação aos demais algoritmos rápidos analisados. Com isso, a diferença entre o resultado ótimo e o melhor algoritmo sub-ótimo, foi reduzida de 3,1 dB (sem incluir o GRS) para 2,4 dB (incluindo o GRS). O ganho em qualidade do algoritmo GRS em relação ao DS foi de 0,7 dB, melhor resultado dentre os demais algoritmos rápidos comparados. O aumento na qualidade é função da exploração de regiões mais afastadas do centro da área de pesquisa propiciada pela busca aleatória e pelo uso de dois refinamentos iterativos.

Analisando o custo computacional, o GRS calculou cerca de 350 vezes menos blocos candidatos que o algoritmo FS. Porém, é valido observar que em relação aos algoritmos rápidos, houve um acréscimo no custo. O DS apresenta um custo computacional quase 50% inferior ao apresentado pelo algoritmo GRS. Apesar do algoritmo GRS apresentar um custo mais elevado, as duas etapas do algoritmo podem ser realizadas em paralelo. Com isso, mesmo com um número maior de comparações, a taxa de processamento em quadros por segundo pode ser equivalente, ou até mesmo superior, ao obtido por muitos algoritmos rápidos.

4. Projeto arquitetural

A Figura 9 apresenta o diagrama de blocos da arquitetura de hardware proposta para o algoritmo GRS. Esta arquitetura usa blocos de tamanho 16x16 pixels e uma sub-amostragem em nível de pixel de 4:1. Esta sub-amostragem faz com que apenas 25% dos pixels de cada bloco sejam avaliados no cálculo de SAD. Esta estratégia reduz a quantidade de memória interna necessária e também acelera o processo de cálculo de SAD.

A arquitetura GRS contém uma memória de referência, que armazena a área de busca de 48x48 amostras (*range* 48). Entretanto, como é utilizada a técnica de sub-amostragem a área é reduzida para 24x24 bytes de memória, com um byte por amostra. A arquitetura também contém uma Unidade Randômica (UR), que é responsável por gerar e armazenar todos os blocos sorteados e encontrar o bloco com maior similaridade com o bloco em processamento. A arquitetura também tem dois iteradores (IT), cada qual representa um estágio do algoritmo. Ambos realizam tarefas similares que podem ser realizadas em paralelo. O primeiro é responsável pela etapa de refinamento iterativo no centro da área de busca. Já o segundo é responsável pela etapa de refinamento iterativo final no melhor bloco dentre os que foram sorteados na etapa aleatória. Um comparador (COMP na Figura 9) é o responsável por encontrar o menor SAD entre o resultado gerado por cada processo iterativo.

Em uma visão mais detalhada, pode ser percebido que a UR é composta por: (1) um banco de memórias locais (ML), que contêm uma memória de 8x8 bytes para cada bloco candidato sorteado; (2) um grupo de Unidades de Processamento (UP) (contendo *N* UPs, uma para cada bloco candidato sorteado); (3) um comparador para definir o bloco candidato de menor SAD; e (4) um gerador de números aleatórios.

A UP é responsável por realizar o cálculo de SAD entre o bloco candidato e o bloco em processamento. As UPs usadas neste trabalho recebem como entrada uma linha de cada bloco por ciclo de relógio, dessa forma, cada bloco necessita de oito ciclos para ser processado (por conta da subamostragem). O comparador de saída da UR seleciona o bloco candidato com o menor SAD dentre os *N* blocos sorteados. Neste bloco selecionado o IT realiza o refinamento iterativo, avaliando o SAD dos vizinhos e iterando até o melhor SAD ser encontrado no centro.

O Gerador Aleatório (GA) é implementado em hardware usando o *Linear Feedback Shift Register* (LFSR) [Liang 2010]. O LSFR é implementado para 32 bits, e os bits menos significativos são utilizados para definir os blocos candidatos sorteados. O GA define o endereço de memória para cada bloco candidato sorteado.

Os dois módulos iteradores, IT na Figura 9, realizam o refinamento iterativo. Cada IT possui quatro UPs, quatro memórias locais, um comparador e uma unidade para identificar a posição da iteração atual e para armazenar o resultado final do refinamento.

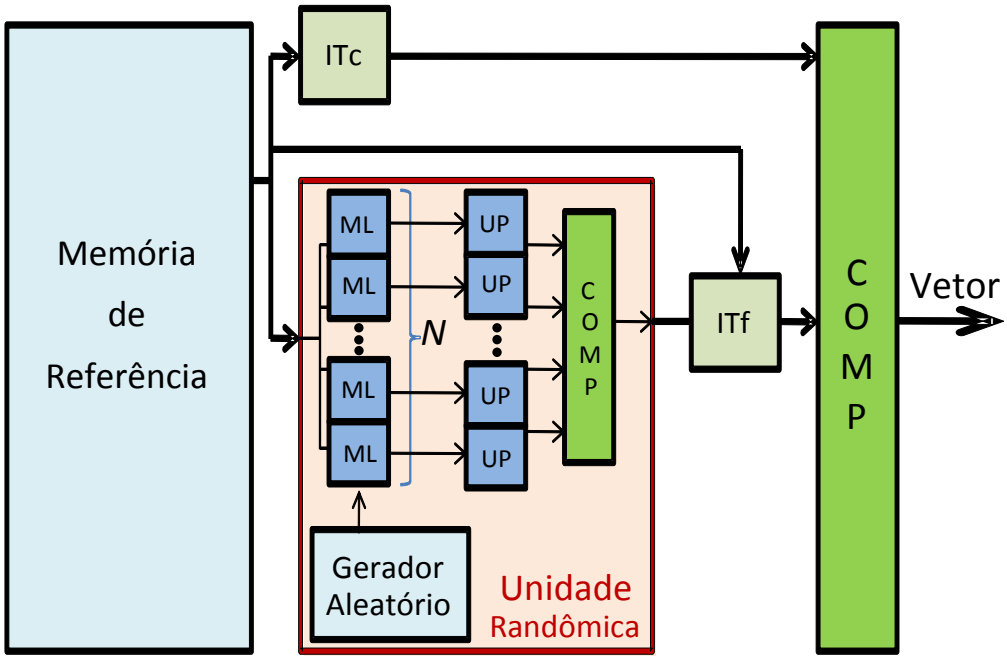

Figura 9. Arquitetura para o algoritmo GRS

Inicialmente, a memórias de referência e do bloco atual são preenchidas com os dados lidos da memória externa, que contêm o vídeo sem compressão. Em paralelo, a UR gera os endereços dos blocos candidatos sorteados e, em seguida, acessa a memória de referência e armazena nas memórias locais (LMs) todos os blocos candidatos sorteados.

Assim que a UR preencher as suas memórias LMs, o ITc, que atua no centro da área de pesquisa, inicia a etapa de avaliação da região central da área de busca. A UR e o ITc podem trabalhar em paralelo, pois não existe dependência de dados entre eles e apenas o ITc fará acessos a memória de referência.

A UR calcula o SAD dos *N* blocos candidatos sorteados e o comparador define o bloco de menor SAD. Este bloco é passado para o ITf, para o cálculo de refinamento final iterativo. Na última etapa, um comparador final identifica o bloco candidato de menor SAD entre os resultados obtidos nos módulos ITc e ITf.

4.1. Trabalhos Relacionados

A Tabela 4 apresenta uma comparação da arquitetura proposta com outros trabalhos desenvolvidos no grupo de pesquisa e publicados na literatura. O trabalho [Referência 1] propõe uma arquitetura para o algoritmo DS que utiliza bloco de tamanho 8x8. Em [Referência 1] são geradas amostras para a estimação de movimento com pixel fracionário. Esta arquitetura também possui um compensador de movimento integrado. A arquitetura apresentada em [Referência 2] propõe uma arquitetura para o algoritmo MPDS (*Multi-Point Diamond Search*). Este algoritmo também tem foco na ME em vídeos de alta definição. Em [Referência 2] o tamanho de bloco utilizado é 8x8, sem subamostragem. A arquitetura do algoritmo GRS utilizou 16 blocos candidatos sorteados ($N = 16$) e bloco de 16x16 pixels com sub-amostragem 4:1, fazendo com que o tamanho físico de bloco também seja de 8x8 amostras.

Tabela 4. Comparativo entre as arquiteturas dos algoritmos GRS, DS e MPDS

	GRS (N=16)	Referência 1	Referência 2
Número de PU's	**24**	**9**	**45**
Tamanho da PU	**8**	**16**	**16**
Tamanho de Memória (Kbits)	**17 408**	**10 368**	**51 840**
Ciclos por Bloco	**180**	**184**	**560**
PSNR (dB)	**32,24**	**31,45**	**33,14**
Frequência mínima para *full* HD à 30 qps (MHz)	**43,74**	**178,85**	**544,32**

Comparando o custo de hardware com o trabalho [Referência 1], desconsiderando os registradores usados para compensação de movimento, a arquitetura para o GRS com $N = 16$ utiliza um número muito maior de UPs, no entanto, o tamanho das UPs na arquitetura do algoritmo GRS é a metade do utilizado por [Referência 1], que calcula duas linhas por ciclo de relógio. A UR tem seu custo definido basicamente pela soma do hardware das UPs utilizadas, levando ambas as arquiteturas a ter o custo em hardware muito similar. O tamanho da memória usado em [Referência 1] é 32% menor em relação à arquitetura do GRS. Apesar disso, a banda de memória necessária é menor já que a hierarquia da memória da arquitetura do GRS é mais bem organizada. Neste trabalho a memória de referência é preenchida apenas uma vez, enquanto que no trabalho [Referência 1] o preenchimento ocorre a cada iteração do algoritmo DS. O algoritmo GRS também alcança uma qualidade de vídeo superior ao apresentado em [Referência 1]. Quando comparado com [Referência 2], a arquitetura proposta para o algoritmo GRS utiliza um número muito menor de UPs. Além disso, as UPs da arquitetura apresentada em [Referência 2] também possuem o dobro do tamanho das UPs utilizadas na arquitetura do algoritmo GRS. O algoritmo MPDS utiliza cinco instancias do algoritmo DS, logo, seu custo em hardware é em torno de cinco vezes maior do que a arquitetura apresentada em [Referência 1]. Comparada com a arquitetura proposta para o algoritmo GRS, o custo em hardware é aproximadamente quatro vezes maior, e a quantidade de memória interna é três vezes maior. Contudo, em [Referência 2] a qualidade obtida foi 0,9dB superior. Em ambos os trabalhos comparados é considerado apenas o tamanho da memória de referência. Contudo, as comparações apresentadas na Tabela 4 levam em consideração o custo total em memória requerido por cada arquitetura. Neste custo está incluído a memória para os blocos candidatos e bloco atual.

Neste trabalho não é possível apresentar os resultados de frequência de operação, pois esta arquitetura ainda não foi prototipada para um FPGA. Entretanto, avaliando o número de ciclos necessários por bloco, a quantidade de blocos por quadro e o número de quadros; foi calculada a frequência mínima necessária para o processamento em tempo real, a 30 quadros por segundo, para vídeos *full* HD. A frequência mínima para a arquitetura do GRS é de 44MHz. Esta frequência de operação é quatro vezes menor do que a frequência mínima exigida pela arquitetura apresentada em [Referência 1]. Comparando com a arquitetura apresentada em [Referência 2], a frequência mínima exigida pelo GRS é 12 vezes menor. A arquitetura do GRS necessita de um número

menor de ciclos de relógio se comparada a [Referência 1] e a [Referência 2], pois a etapa aleatória, executada pela UR, e a etapa iterativa, executada pelas ITs, podem ser paralelizadas.

6. Conclusões

Este trabalho apresentou um novo algoritmo para a estimação de movimento em vídeos de alta resolução, chamado *Galaxy Randon Search* (GRS) Também foi apresentada uma proposta arquitetural para a codificação de vídeos *full* HD em tempo real. Este algoritmo utiliza a busca aleatória como forma reduzir a incidência da escolha de mínimos locais. O GRS também explora duas características importantes no processo de estimação de movimento. Uma delas é a exploração da região central da área de busca, pois em vídeos com pouca movimentação os melhores blocos estão próximo ao centro. A outra é a utilização de um refinamento final iterativo, visando a convergência para regiões de maior similaridade.

O número de blocos sorteados e o tamanho da área de busca exercem influência sobre os resultados de qualidade e custo computacional. Neste trabalho, foi analisado o desempenho do algoritmo GRS para a variação destes parâmetros. O melhor custo-benefício foi obtido pela área com range de 48 amostras e o número de blocos sorteados (*N*) igual a 16. Nesta configuração, o algoritmo GRS apresentou um PSNR médio de 33,74 dB, condirando a média obtida para 10 sequencias *full* HD. O número médio de blocos candidatos avaliados foi de 73,20 $x10^6$.

O algoritmo GRS apresentou o melhor resultado em termos de qualidade, dentre todos os algoritmos rápidos comparados, obtendo um ganho de 0,7dB em relação ao algoritmo DS, por exemplo. No entanto, o custo computacional, medido em número de blocos candidatos comparados, foi maior. Entretanto, a taxa de processamento do algoritmo pode ser equivalente ou até superior aos demais algoritmos rápidos, devido a possibilidade de paralelização de suas etapas, uma vez que não existe dependência de dados entre elas. Os outros algoritmos avaliados possuem estas dependências, impossibilitando uma exploração maior do paralelismo. Assim, uma implementação em hardware ou em software com dois processos paralelos pode reduzir pela metade o tempo de execução do algoritmo GRS.

Se comparado ao Full Search, o custo computacional do algoritmo GRS foi 350 vezes menor, com uma perda de qualidade de apenas 2,42dB. Essa diferença em relação ao FS foi a menor entre os algoritmos rápidos desenvolvidos.

A arquitetura de hardware projetada para o algoritmo GRS apresenta baixo consumo de recursos de hardware e desempenho suficiente para processar vídeos *full* HD em tempo real (30 quadros por segundo) a uma frequência de 44Mhz.

Referências

Bhaskaran, V.; Konstantinides, K. 1999. Image and Video Compression Standards: Algorithms and Architectures. 2nd ed. Boston: Kluwer Academic Publishers, 1999.

Ghanbari, M. 2003, Standard Codecs: Image Compression to Advanced Video Coding. United Kingdom: The Institution of Electrical Engineers, 2003.

Jing, X, Chau, L, 2004, An efficient three-step search algorithm for Block motion estimation. IEEE Transactions on Multimedia, [S.l.], Vol. 6, No. 3, 2004, pp. 435-438.

Kuo, C. et al. 2009, A Novel Prediction-Based Directional Asymmetric Search Algorithm for Fast Block-Matching Motion Estimation. IEEE Transactions on Circuits and Systems for Video Technology, [S.l.], v. 19, n. 6, p. 893-899, Jun. 2009.

Liang, W. et al 2010, A cryptographic algorithm based on Linear Feedback Shift Register. IEEE ICCASM, 2010.

Ndili, O.; Ogunfunmi, T. 2010, Hardware-Oriented Modified Diamond for Motion Estimation in H.246/AVC. In: ICIP 2010 – IEEE International Conference on Image Processing. Proceedings Hong Kong: IEEE, 2010, p. 749-752.

Puri, A. et al. 2004, Video Coding Using the H.264/MPEG-4 AVC Compression Standard. Elsevier Signal Processing: Image Communication, [S.l.], n. 19, p.793–849, 2004.

Richardson, I. 2003, H.264 and MPEG-4 Video Compression : Video Coding for Next-Generation Multimedia. Chichester: John Wiley and Sons, 2003.

Tasdizen, O., et al. 2009, Dynamically Variable Step Search Motion Estimation Algorithm and a Dynamically Reconfigurable Hardware for Its Implementation. IEEE Transactions on Consumer Electronics, Vol. 55, No. 3, 2009, pp. 1645-1653.

Xiph.org 2011: Test media, available at <http://media.xiph.org/video/derf/>, May, 2011.

Zhu, C.; LIN, X. Chau, L. 2002, Hexagon-based Search pattern for fast Block motion estimation. IEEE Transactions on Circuits and Systems for Video Technology, Nova York, v. 12, n. 5, p. 349 – 355, Maio 2002

Zhu, S., Ma, K. 2000, A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation. In: IEEE Transactions on Image Processing, Vol. 9, No. 2, 2000, pp. 287-290.

“Referência 1, omitida para a revisão cega”

“Referência 2, omitida para a revisão cega”