HARDWARE DESIGN DE UM NOVO ALGORITMO PARA ESTIMAÇÃO DE MOVIMENTO EM VÍDEOS DE ALTA DEFINIÇÃO

DALL'OGLIO, Pargles[1]; CRISTANI, Cássio[1]; PORTO, Marcelo[2]; AGOSTINI, Luciano[2]

[1]UFPel – Curso de Ciência da Computação. Email: {pwdalloglio, crcristani}@inf.ufpel.edu.br
[2]UFPel – Centro de Desenvolvimento Tecnológico. Email: {porto, agostini}@inf.ufpel.edu.br

1 INTRODUÇÃO

O número de aparelhos eletrônicos capazes de reproduzir vídeos de alta definição (*High Definition* - HD) vem crescendo intensamente, como por exemplo, aparelhos celulares, computadores, tocadores de blu-ray e televisores. A maioria destes dispositivos demanda por soluções que visam hardware de baixo custo, de baixo consumo de energia e que reproduzam vídeos de alta definição em tempo real. No entanto, a quantidade de dados para representar um vídeo de alta definição sem compressão é proibitiva, superior à capacidade de processamento, transmissão e armazenamento destes dispositivos. Para transmitir em tempo real ou armazenar estes vídeos de forma a manter uma boa qualidade e um baixo custo, torna-se necessário o estudo e o desenvolvimento de novos e eficientes algoritmos para a compressão de vídeos, assim como o desenvolvimento de hardware dedicado.

A estimação de movimento (EM) é a etapa que representa mais de 80% da complexidade computacional de um codificador de vídeo atual (PURI, 2004), no entanto, é a principal ferramenta responsável pela redução do tamanho de um vídeo.

Um vídeo digital é composto por uma sequência de imagens, denominadas quadros. Para gerar sensação de movimento, é necessário que aproximadamente 30 quadros sejam apresentados a cada segundo. Cada quadro é divido em blocos e estes são compostos por um conjunto de pixels. Em um vídeo *Full HD*, existem 1920x1080 pixels em cada quadro do vídeo.

Quadros vizinhos normalmente são muito similares e esta similaridade é denominada redundância temporal. A EM é a etapa responsável por identificar e reduzir a redundância temporal entre os quadros vizinhos. Para isso, a EM utiliza uma área de pesquisa em um quadro de referência (que foi anteriormente processado), aonde cada bloco do quadro atual é comparado com os blocos da área de pesquisa. Ao final, a EM indica o bloco com maior similaridade (melhor casamento) ao bloco atual, através de um vetor de movimento.

Um algoritmo de busca determina com que ordem os blocos serão comparados dentro da área de pesquisa. O algoritmo de busca completa (*Full Search* - FS) (RICHARDSON, 2003) é o algoritmo que apresenta a melhor qualidade, mas também possui o maior custo computacional, pois compara todos os blocos da área de pesquisa. Entre os algoritmos rápidos mais conhecidos na literatura, cita-se a busca em diamante (*Diamond Search* – DS) (ZHU, 2000) e a busca em quatro passos (*Four Step Search* - FSS) (TASDIZEN, 2009), cita-se também um algoritmo rápido voltado para vídeos de alta definição que utiliza uma busca com múltiplos pontos (*Multi Point Diamond Search* - MPDS) (SANCHEZ, 2011). Os algoritmos rápidos são baseados em heurísticas que visam convergir a busca mais rapidamente, assim, é possível atingir uma grande redução no custo computacional, porém, com perdas de qualidade.

A qualidade objetiva dos resultados da EM é avaliada através do PSNR (*Peak Signal-to-Noise Ratio*) (RICHARDSON, 2003) enquanto que a quantidade de blocos candidatos calculados (BCC) é utilizada como métrica de desempenho. Para algoritmos sequencias, quanto menor for a quantidade de BCCs calculados, menor será seu tempo de processamento, enquanto que algoritmos paralelizáveis (sem dependência de dados) podem ser mais rápidos que algoritmos sequencias mesmo quando possuem um número maior de BCCs.

Este trabalho apresenta um novo algoritmo rápido para estimação de movimento em vídeos de alta definição, chamado *Iterative Random Search* (IRS) e parte de seu projeto de hardware. Este algoritmo usa a aleatoriedade como estratégia para ampliar a qualidade da EM, pois permite que blocos mais distantes na área de pesquisa sejam avaliados. Além disso, o uso da estratégia aleatória torna esse algoritmo facilmente paralelizável, pois não existe dependência de dados entre os pontos aleatórios.

2 METODOLOGIA (MATERIAL E MÉTODOS)

O algoritmo *Iterative Random Search* (IRS), proposto neste trabalho, foi implementado na linguagem de programação C e é composto por quatro etapas principais. A primeira etapa divide a área de pesquisa em quatro regiões iguais e sorteia *N* blocos candidatos, igualmente divididos em cada região. A segunda etapa é responsável pela execução de um processo iterativo em cada um dos quatro melhores pontos de cada quadrante, obtidos na etapa anterior. Na terceira etapa, um processo iterativo é executado na posição central da área de pesquisa, garantindo bons resultados para vídeos de baixa movimentação. Os cinco processos iterativos (quatro da etapa aleatória e um da avaliação central) podem ser executados em paralelo, pois não há dependências de dados entre eles. Ao final desse processo, na quarta etapa, os resultados obtidos pelos cinco processos iterativos são comparados e então é gerado um vetor de movimento.

O número de blocos candidatos sorteados (*N*) e a área de pesquisa influenciam diretamente na qualidade e na complexidade do vídeo codificado. Áreas de busca maiores podem conduzir a maiores valores de PSNR, contudo, exigem um maior valor de *N* de modo a explorar esta ampla área. O aumento no *N* produz resultados com melhor qualidade, porém, exigem maior número de comparações. Para determinar o valor ideal do *N* para cada tamanho da área de pesquisa, diversos testes foram realizados analisando a variação destes parâmetros. Através destes testes foi possível encontrar o melhor valor de *N* como 16 e de tamanho da área de pesquisa como 96x96 pixels.

A arquitetura proposta neste trabalho está focada apenas na etapa de busca aleatória do algoritmo IRS e utiliza blocos de 16x16 pixels com uma taxa de sub-amostragem de pixel de 4:1. A Fig.1 ilustra o projeto de hardware da arquitetura, assim como os componentes necessários para seu desenvolvimento. A utilização de sub-amostragem de pixels reduz o tamanho da Memória de Referência para 32x32 bytes. O Gerador Aleatório é implementado em hardware utilizando um *Linear Feedback Shift Register* (LFSR) (BEKER, 1983) composto por 32 bits onde utiliza-se apenas os primeiros 10 bits desse registrador, sendo 5 bits para X e 5 bits para Y. Essas coordenadas informam o endereço de memória para cada bloco sorteado.

Na inicialização, a Memória de Referência está vazia, sendo necessário preenchê-la a partir de uma memória externa (onde o quadro de referencia está

armazenado). Em paralelo com esta operação, a memória de bloco atual é preenchida com os dados do quadro original (sem compressão). Nesta etapa também são geradas todas as posições aleatórias. Após preenchida a primeira linha da Memória de Referência, ela começa a ser lida linha a linha. Se existem blocos sorteados na linha corrente, eles começam a ser armazenados nas Memórias Locais (ML), que são pequenas memórias de 8x8 bytes que armazenam a informação de cada um dos *N* blocos sorteados.

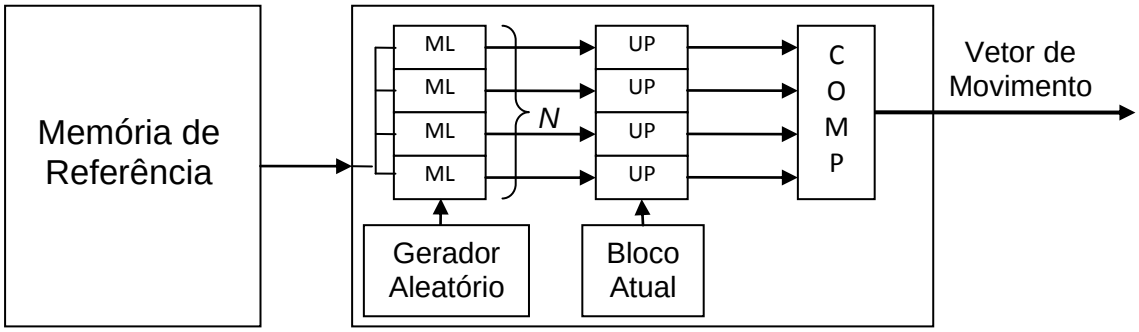

Figura 1 – Projeto de Hardware do Algoritmo IRS

O próximo passo é ler cada ML e processar seus conteúdos nas Unidades de Processamento (UP), que são responsáveis por calcular o nível de similaridade entre o Bloco Atual e os blocos analisados. Ao final desse processo, os níveis de similaridade são comparados (COMP) gerando o vetor de movimento. O processo é então repetido para os demais blocos do quadro.

3 RESULTADOS E DISCUSSÃO

Nas diversas avaliações realizadas com o algoritmo IRS, foram utilizadas dez sequências de teste *Full HD* (XIPH.ORG, 2011). Utilizou-se a quantidade de blocos candidatos calculados (BCC) como métrica de complexidade e o PSNR (dB) para avaliar a qualidade dos vídeos.

Através da Tab. 1 é possível observar os resultados médios de qualidade e complexidade do algoritmo ótimo FS e dos algoritmos rápidos: MPDS, IRS, DS e FSS. O algoritmo IRS obteve uma diferença de apenas 1,44 dB e 0,30 dB em relação aos algoritmos FS e MPDS, porém com uma redução de complexidade de aproximadamente 169 vezes e 3 vezes, respectivamente. Se comparado aos algoritmos DS e FSS o algoritmo proposto obtém ganhos de 1,43 dB e 2,06 dB respectivamente, com um aumento de aproximadamente 2 vezes na complexidade.

Tabela 1 – Comparação de qualidade e custo do algoritmo desenvolvido

Algoritmo	PSNR (dB)	BCC's x 10^6
FS	35,89	14.662,59
MPDS	34,75	307,51
IRS	**34,45**	**91,82**
DS	33,02	48,06
FSS	32,39	58,03

A Tab. 2 contém os dados das arquiteturas dos algoritmos MPDS e DS relacionados com a arquitetura proposta. Através dela é possível observar que a arquitetura proposta requer doze vezes menos hardware em relação à arquitetura MPDS, pois a arquitetura relacionada é voltada pra vídeos de alta definição e utiliza cinco instâncias do algoritmo DS para obter uma melhor qualidade.

Tabela 2 – Comparação entre arquiteturas

Algoritmo	Quantidade de UP's	Tamanho da UP	Tamanho da Memória	Ciclos por Bloco
Este Trabalho	**16**	**8**	**1.024**	**120**
MPDS	45	16	12.000	560
DS	9	16	2.040	184

Este trabalho não possui resultados de frequência, pois essa arquitetura não foi implementada em FPGA ou *Standard Cells*. Contudo, como a arquitetura DS necessita de 184 ciclos para processar um bloco em vídeos *Full HD* em tempo real, a arquitetura proposta também deve ser capaz de processar os mesmos vídeos.

4 CONCLUSÃO

Este trabalho apresentou um novo algoritmo para estimação de movimento em vídeos de alta definição, chamado *Iterative Random Search* (IRS) e também propôs uma arquitetura em hardware para implementar a etapa aleatória desse algoritmo. O algoritmo proposto introduz buscas aleatórias e refinamentos iterativos para a estimação de movimento. Através dessas estratégias o IRS é três vezes mais rápido em software e doze vezes menos custoso em hardware quando comparado ao algoritmo MPDS, com diferença de qualidade de apenas 0,30 dB.

Analisando a arquitetura proposta, é possível concluir que ela será capaz de processar vídeos *Full HD* (1080p) em tempo real (a 30 quadros por segundo) utilizando poucos recursos de hardware e deve atingir um baixo consumo de energia, pois necessita de poucos ciclos para gerar um vetor de movimento.

Estes resultados são especialmente relevantes no cenário atual da área de vídeo digital, já que vídeos digitais de elevada resolução, como *Full* HD, estão cada vez mais presentes nas aplicações atuais.

5 REFERÊNCIAS

BEKER, H. **Cipher Systems: The Protection of Communications**. John Wiley & Sons Inc, 1983.

PURI, A. et al. Video Coding Using the H.264/MPEG-4 AVC Compression Standard. **Elsevier Signal Processing: Image Communication**, 2004.

RICHARDSON, I. **H.264/AVC and MPEG-4 Video Compression – Video Coding for Next-Generation Multimedia**. Chichester: John Wiley and Sons, 2003.

SANCHEZ, G. et al. A Real Time HDTV Motion Estimation Architecture for the New MPDS Algorithm. **IEEE EUROCON**, 2011.

TASDIZEN, O. et al. Dynamically variable motion estimation algorithm and dynamically reconfigurable hardware for its implementation. **IEEE Transactions on Consumer Electronics**, 2009

XIPH.ORG. **Test Media.** Disponível em: <http://media.xiph.org/video/derf/>, Acesso em: 12 agosto de 2011.

ZHU, S. et al. A new Diamond Search Algorithm for Fast Block-Matching Motion Estimation. **IEEE Transactions on Image Processing**, 2000.