# Exploração da Multiprogramação Leve no Processo de Estimação de Movimento utilizando OpenMP

**João Alberto Vortmann, Rafael Petry,**
**Luciano Volcan Agostini, Gerson Geraldo H. Cavalheiro**

Universidade Federal de Pelotas
Departamento de Informática
Bacharelado em Ciência da Computação

{jvortmann_ifm, rpetry_ifm, agostini, gerson.cavalheiro}@ufpel.edu.br

***Resumo.*** *A ampla utilização de vídeo digital pelas mais diferentes mídias, torna o processo de codificação fundamental. Neste processo, a etapa de estimação de movimento é a mais complexa, demandando elevado tempo de processamento. Este trabalho apresenta o uso de técnicas e ferramentas de multiprogramação leve em arquiteturas multi-core para a realização da estimação de movimento. Para tanto, a técnica de Diamond Search e a ferramenta OpenMP foram utilizados. Os resultados de duas implementações, uma seqüencial e outra utilizando OpenMP para paralelismo de quadro, são apresentados e discutidos. Como esperado, a versão concorrente obteve um tempo de processamento, em média, 34,2% menor que a versão seqüencial.*

## 1. Introdução

O grande interesse da indústria na codificação de vídeo digital é reflexo da popularização das diversas aplicações de vídeo em diferentes mídias, como telefones celulares, DVD players, câmeras e mesmo no Sistema Brasileiro de Televisão Digital (SBTVD). Em conseqüência, diversos padrões de codificações surgiram, entre os quais o H.264/AVC [JVT 2003] é o que atinge a maior taxa de compressão, até duas vezes maior em relação a padrões anteriores.

A estimação de movimento é a etapa mais complexa do processo de compressão de vídeo. Para obter, em software, uma implementação que ofereça tempo real (24 a 30 quadros por segundo), esta etapa deve explorar técnicas e recursos de programação eficientes. Este trabalho propõe o uso de arquiteturas multi-core e programação em OpenMP [OpenMP 2005] para implementar a técnica de *Diamond Search* [Kuhn 1999] de estimação de movimento.

Este trabalho está organizado como segue. A Seção 2 introduz o processo de estimação de movimento e a Seção 3 a técnica *Diamond Search*. A Seção 4 apresenta uma avaliação dos resultados obtidos e a seção seguinte conclui o trabalho.

## 2. Estimação de Movimento

Um vídeo digital é formado por uma seqüência de quadros estáticos, que devem ser apresentados a uma taxa de 24 a 30 quadros por segundo. É natural que entre quadros vizinhos existam informações semelhantes. A estimação de movimento (EM) é a operação de um codificador que busca minimizar a redundância temporal entre quadros vizinhos. Para tanto, a EM utiliza um quadro previamente codificado como quadro de

referência, e divide o quadro atual (alvo da codificação) em blocos. Para cada bloco, o algoritmo irá procurar o bloco mais semelhante no quadro de referência utilizando uma função de similaridade. Ao final da busca, é gerado um vetor de movimento que indica o bloco mais semelhante no quadro de referência. Essa informação é utilizada pelo decodificador para remontar a imagem a partir do quadro de referência. Com isso não é necessário enviar todo o novo quadro, sendo reaproveitados blocos já recebidos.

Para diminuir o tempo de processamento requerido na etapa de EM, a busca pelo bloco mais semelhante é geralmente feita em uma região limitada denominada área de pesquisa, como mostrado na Figura 1. Também podem ser empregados algoritmos que implementam heurísticas para diminuir o número de operações necessárias para se determinar o vetor de movimento.

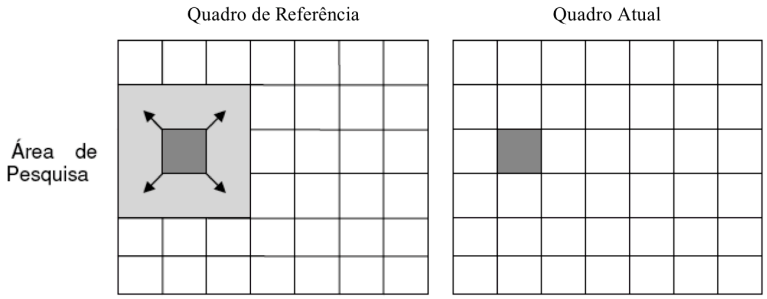


**Figura 1. Determinação do vetor de movimento para um bloco.**

## 3. Algoritmo *Diamond Search*

Uma análise de diversas técnicas de EM foi apresentada em [Rosa 2007]. Este trabalho indica que o algoritmo *Full Search* apresenta a melhor qualidade de compressão. Neste mesmo estudo, o *Diamond Search* permitiu atingir um resultado próximo ao *Full Search* requerendo menos operações, diminuindo a quantidade de recursos e de tempo para a etapa de EM, sem perda significativa de qualidade.

O algoritmo *Diamond Search* utiliza dois padrões (*patterns*) de busca em forma de diamante e é realizado em duas fases. O *Large Diamond Search Pattern* (LDSP) é utilizado nas iterações da etapa inicial e o *Small Diamond Search Pattern* (SDSP) na etapa final de refinamento. Esses padrões são ilustrados na Figura 2.

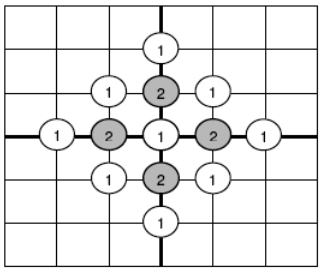


**Figura 2. Padrões LDSP (1) e SDSP (2).**

O algoritmo inicia aplicando o LDSP, blocos identificados por (1), no centro da área de pesquisa. Caso o bloco de menor erro (maior grau de similaridade) esteja localizado em uma posição que não seja a central deste diamante, esse bloco é definido

como o novo centro, e o algoritmo repete a fase LDSP. Existem duas possibilidades de iteração nessa etapa: por aresta e por vértice. Elas são ilustradas na Figura 3.

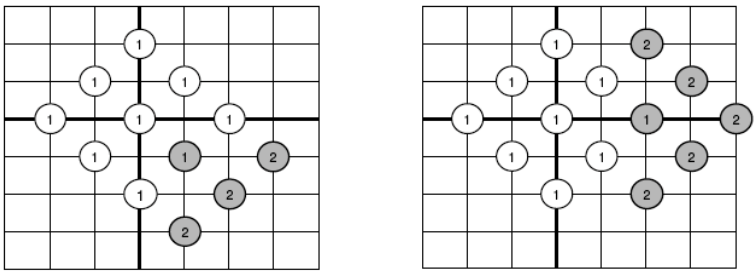

**Figura 3. Busca por uma aresta (lado esquerdo) e busca por um vértice (lado direito).**

Quando a posição central for a que possuir o menor erro, é aplicado o SDSP para refinar o resultado. Nessa etapa são verificados quatro novos blocos que são comparados com o centro, e o melhor resultado é escolhido.

## 4. Estudo de caso

A técnica *Diamond Search* foi implementada na linguagem C em duas versões, seqüencial e concorrente com OpenMP. A versão OpenMP utiliza paralelismo por quadro, ou seja, cada *thread* é responsável pelo processamento de um quadro. *O* desempenho destas implementações foi medido utilizando como entrada os primeiros cem quadros de dez vídeos digitais (Figura 4) não comprimidos em resolução SDTV (720x480 pixels). O SAD – soma das diferenças absolutas – foi definido como critério de similaridade e a área de busca máxima foi de 208*x*208 pixels. Os vídeos são variados, apresentando diferentes graus de movimento. Os experimentos foram realizados em um computador com processador Intel Core 2 Duo (2.0 GHz) com 4 MB de memória cache compartilhada e 2 GB de RAM sob Mac OS X.

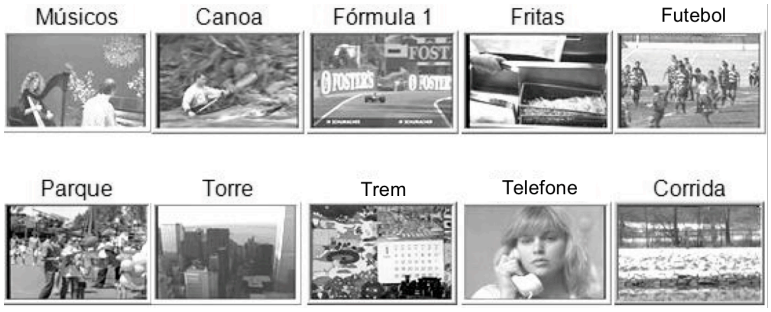


**Figura 4. Primeiro quadro dos vídeos utilizados nas simulações.**

A Tabela 1 mostra os resultados de tempo de processamento e taxa de quadros por segundo obtidas. O número de threads informados corresponde ao número de threads de serviço de OpenMP.

**Tabela 1. Tempos e taxas de processamento das implementações**

| | Seqüencial | 2 Threads | 4 Threads |
|---|---|---|---|
| Tempo de processamento (s) | 8,36 | 5,51 | 5,45 |
| Quadros por segundo (FPS) | 11,96 | 18,16 | 18,36 |

Conforme a Tabela 1, ocorre diminuição no tempo de processamento com utilização de paralelismo por quadro. Na execução com suporte de duas *threads*, a redução no tempo chegou a 34,2%. Execuções com maior número de *threads* de execução não apresentaram ganho significativo em relação à execução com duas. Além do fato de o processador ter apenas dois cores, isso também pode ser explicado pela grande quantidade de dados acessados na memória, que faz com que as *regiões paralelas* disputem o acesso aos dados.

## 5. Conclusões

A técnica *Diamond Search*, uma heurística para diminuição da quantidade de operações na estimação de movimento, apresenta, em sua versão seqüencial, ganho em tempo de processamento sem apresentar significativa perda de qualidade em relação à técnica *Full Search*. O experimento de implementação concorrente da técnica *Diamond Search* relatado neste trabalho, utilizando OpenMP e executando em arquitetura dual-core, é capaz de atingir um *throughput* 51,8% maior que a sua versão seqüencial.

Os resultados indicam que o uso de técnicas e ferramentas de programação concorrente, como paralelismo de dados usando OpenMP, e de arquiteturas de processadores multi-core podem representar um importante papel em aplicações de vídeo e, em particular, no contexto da TV Digital.

## Referências

JVT - Joint Video Team of ITU-T and ISO/IEC JTC 1 (2003). "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 or ISO/IEC 14496-10 AVC)".

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

Rosa, Leandro Zanetti Paiva da (2007). "Investigação sobre Algoritmos para a Estimação de Movimento na Compressão de Vídeos Digitais de Alta Definição: Uma Análise Quantitativa".

OpenMP Architecture Review Board (2005). “OpenMP Application Program Interface”. http://www.openmp.org/mp-documents/spec25.pdf