Paralelização do algoritmo Full-Search para estimação de movimentos em vídeos com OpenMP

Mathaus C. Huber

[1]Universidade Federal de Pelotas (UFPEL) – Discentes do Curso Superior de Ciência da Computação
R. Gomes Carneiro, 1 - Centro – 96075-630 – Pelotas – RS – Brazil

mchuber@inf.ufpel.edu.br, trporto@inf.ufpel.edu.br, wcsilveira@inf.ufpel.edu.br

Abstract. *This article describes the practical work of the discipline Introduction to Parallel and Distributed Processing given to the seventh semester of the Computer Science course at the Federal University of Pelotas, which refers to the compression of a video using the FullSearch exhaustive search algorithm to estimate motion in videos, using a parallelization with the OpenMP tool in order to gain performance.*

Resumo. *Este artigo descreve o trabalho prático da disciplina de Introdução ao Processamento Paralelo e Distribuído conferida ao sétimo semestre do curso de Ciência da Computação da Universidade Federal de Pelotas, que se refere a compressão de um vídeo utilizando o algoritmo de busca exaustiva FullSearch para estimar movimento em vídeos, utilizando uma paralelização com a ferramenta OpenMP a fim de ganhar desempenho.*

1. Informação Geral

A estimação de movimento em vídeos (EM), também conhecido como previsão entre frames, pode nos fornecer ganhos de codificação muito importantes ao reduzir a redundância temporal entre os frames, justamente os que tendem a ser muito semelhantes devido à alta taxa de quadros usada para dar ao observador a sensação de movimento. Esta técnica emprega algoritmos de busca para encontrar em quadros de referência (os quadros reconstruídos anteriormente codificados) as regiões mais semelhantes ao quadro atual. Com o intuito de maximizar desempenho, visamos trazer uma abordagem paralela de um algoritmo exaustivo de busca, conhecido como Full Search, utilizando o OpenMP.

Em geral, o OpenMP foi projetado para a programação de computadores paralelos com memória compartilhada, sendo uma interface de programação de aplicativo para a programação multi-threading de memória compartilhada em múltiplas plataformas. Permitindo acrescentar simultaneidade aos programas escritos em C, C++ e Fortran sobre a base do modelo de execução fork-join.

2. Algoritmo

O processo de EM é realizado bloco a bloco, para cada bloco de um frame de vídeo, usando um algoritmo de casamento de blocos para encontrar a melhor correspondência em um frame de referência. Isso é definido através um critério de similaridade, neste trabalho utilizamos a Soma de Diferenças Absolutas (SAD).

O algoritmo Full Search visa encontrar a melhor correspondência entre o bloco do frame atual e todas as posições possíveis no frame de referência, com isso, temos algo relacionado a uma força bruta, tornando o algoritmo computacionalmente mais caro, em termos de tempo, comparado a outros algoritmos existentes na literatura. Apesar do baixo desempenho, este algoritmo é considerado ótimo porque é capaz de gerar vetores de movimento melhorados, resultando na melhor qualidade de vídeo e melhor eficiência de codificação.

3. Implementação

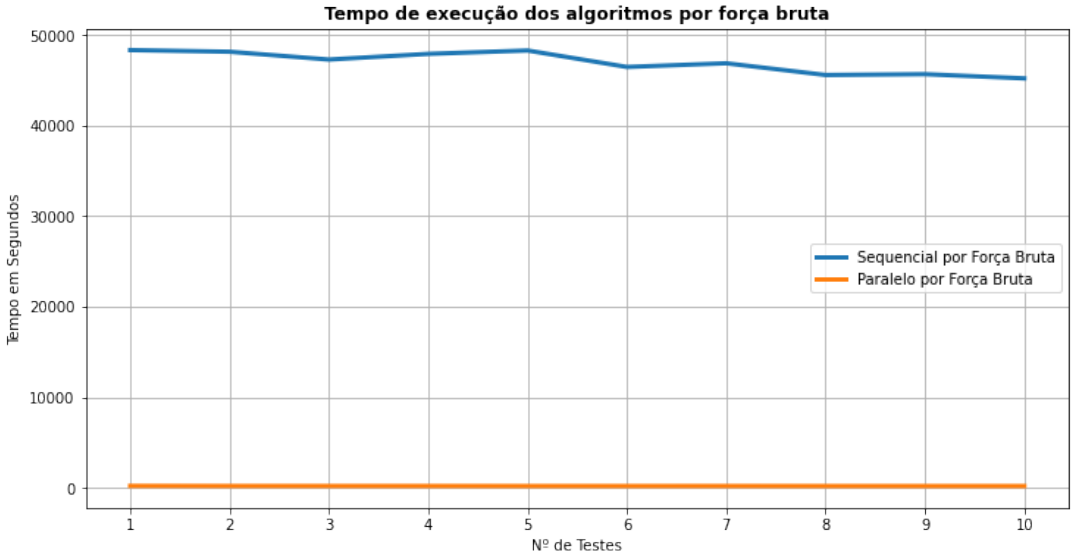
Neste trabalho, foi usado um vídeo em formato .yuv. Este formato é uma representação dos três componentes do tipo de sinal para componentes de vídeo, um para luminosidade e outros dois para informação de cor. No que se remete a estimação de movimento, focamos no canal de luminosidade, onde sua incidência é maior.

Desenvolvemos duas estratégias para testar o ganho de desempenho na paralelização, em um primeiro momento utilizamos o algoritmo Full Search percorrendo bloco a bloco tentando aproveitar o significativo potencial de paralelismo inerente ao algoritmo Full Search sem nenhum uso de heurísticas. Em um segundo momento utilizamos uma heurística de percorrer os frames através de uma área de busca, primeiro calculando os valores do SAD para todos os blocos candidatos dentro da área de busca e posteriormente comparando os valores de SAD de todos os blocos candidatos com o intuito de encontrar as melhores correspondências.

4. Testes e Conclusão

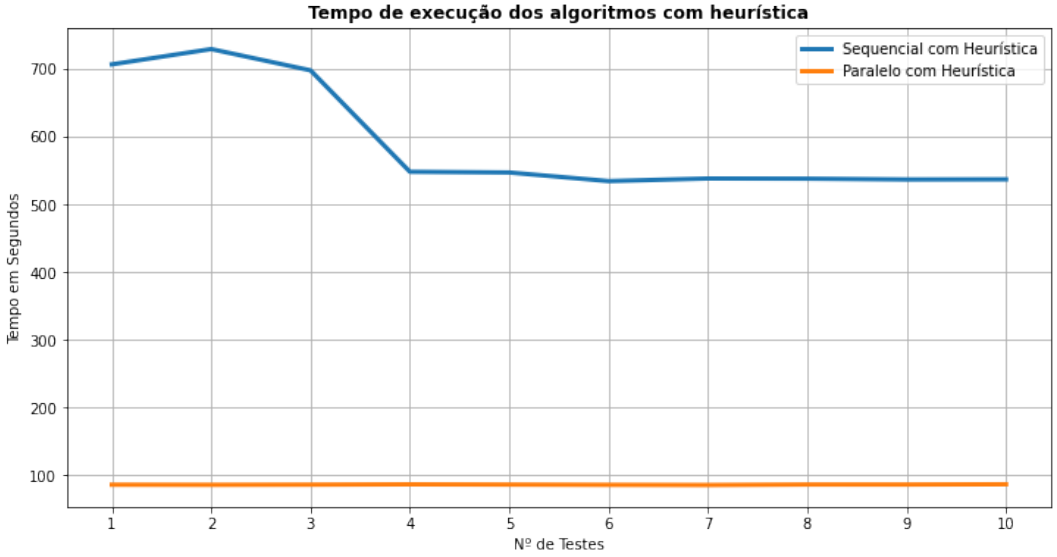
No primeiro algoritmo, temos uma boa diminuição do tempo de execução do algoritmo Full Search por força bruta quando usamos o paralelismo, tentamos utilizar o método de redução para os valores de SAD, porém não se mostrou vantajoso. Como era esperado, pelo alto custo computacional, este foi o algoritmo que demandou mais tempo.

- Média de tempo dos testes do Algoritmo Sequencial por Força Bruta 46957.85s
- Média de tempo dos testes do Algoritmo Paralelo por Força Bruta 197.94s
- Desvio Padrão dos testes do Algoritmo Sequencial por Força Bruta 1137.31s
- Desvio Padrão dos testes do Algoritmo Paralelo por Força Bruta 4.85s
- Variância dos testes do Algoritmo Sequencial por Força Bruta 1293477.58s
- Variância dos testes do Algoritmo Paralelo por Força Bruta 23.54s

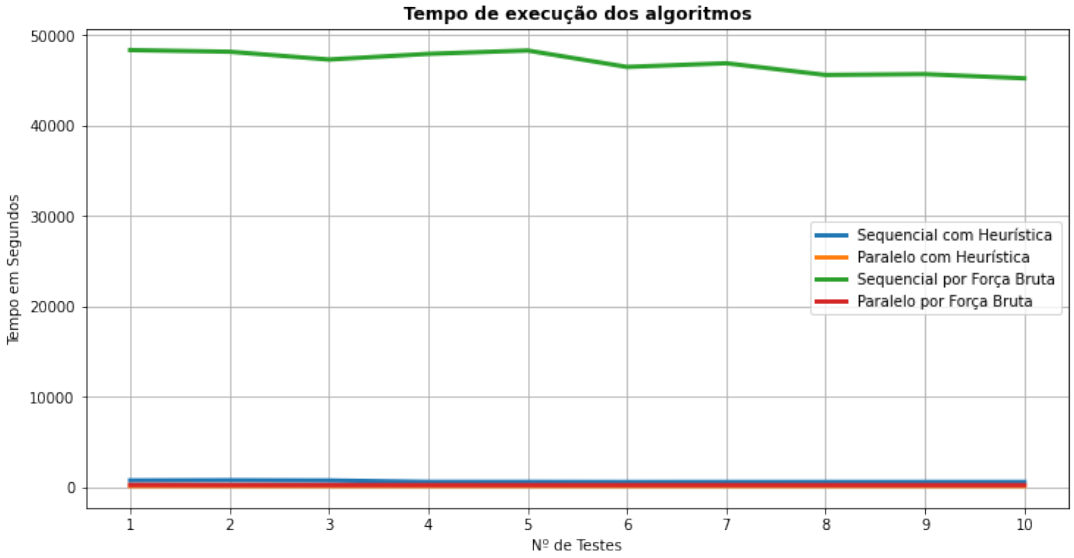


No segundo algoritmo, utilizando uma heurística, o tempo de execução já na sua implementação sequencial é muito diminuído. Tendo um ganho de tempo na paralelização quase 10x melhor.

- Média de tempo dos testes do Algoritmo Sequencial com Heurística 591.14s
- Média de tempo dos testes do Algoritmo Paralelo com Heurística 86.30s
- Desvio Padrão dos testes do Algoritmo Sequencial com Heurística 78.94s
- Desvio Padrão dos testes do Algoritmo Paralelo com Heurística 0.32s
- Variância dos testes do Algoritmo Sequencial com Heurística 6230.96s
- Variância dos testes do Algoritmo Paralelo com Heurística 0.10s



Abaixo podemos notar no gráfico os seus respectivos ganhos de desempenho através do uso de técnicas de paralelização.



Este trabalho almejou paralelizar ao máximo os dois algoritmos utilizando o OpenMP, com o intuito de ganhar desempenho. Foi nítido a diferença, em ganho de tempo, entre os algoritmos sequenciais e paralelos, tanto o que faz uso de heurística quanto o algoritmo exaustivo Full-Search, se mostram muito mais rápidos quando paralelizados.

5. Links

Código do Algoritmo Sequencial Exaustivo: `https://github.com/HuberM1998/Parallel-Fullsearch/blob/main/src/fs_heuristica.c`
Código do Algoritmo Paralelo Exaustivo: `https://github.com/HuberM1998/Parallel-Fullsearch/blob/main/src/fullsearch.c`
Código do Algoritmo Sequencial com Heurística: `https://github.com/HuberM1998/Parallel-Fullsearch/blob/main/src/fs_seq_heuristica.c`
Código do Algoritmo Paralelo com Heurística: `https://github.com/HuberM1998/Parallel-Fullsearch/blob/main/src/fs_heuristica.c`
Todo o código e relatórios estão disponíveis no repositório do GitHub, em: `https://github.com/HuberM1998/Parallel-Fullsearch`