Universidade de Campinas

Instituto de computação - IC

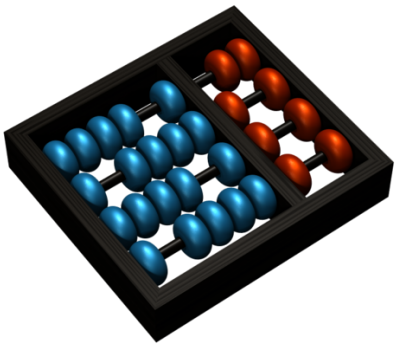
Introdução a programação paralela - MO644

Estabilização de Vídeo

Autor:
John Leandro Silva
RA: 191082

Professor:
Dr. Guido Araujo

June 29, 2017



1 Introdução

Na era da informação boa parte dos habitantes do mundo possuem algum dispositivo capaz de gravar vídeos, essa facilidade permite que milhões de vídeos sejam gravados diariamente, sendo assim, é natural que as vezes algum movimento brusco, da câmera ou do objeto em questão afete a qualidade final da gravação.

Para amenizar o impacto causado por movimentos indesejados na hora da captura de um vídeo, foram criadas técnicas de estabilização, cuja a finalidade é suavizar a trajetória da câmera ou dos objetos da cena.

Buscando resolver o problema descrito de forma paralela, este trabalho fez uso das bibliotecas OpenMP, Pthread e OpenCV.

2 Descrição geral do problema

O problema consiste de um ou mais vídeos, que podem ou não conter trechos de movimentação brusca de translação e rotação, assume-se também, que os vídeos de entrada tem transições suaves entre os quadros de forma que haja pontos de interesse entre dois quadros consecutivos.

A técnica utilizada por este trabalho encontra uma transformação do quadro anterior para o atual usando o fluxo óptico para todos os quadros. A transformação apenas consiste em três parâmetros: dx, dy, da (ângulo), é basicamente uma transformada euclidiana rígida sem escala.

O método acumula as transformações para obter a trajetória para x, y e ângulo em cada quadro. Posteriormente, é utilizado uma janela média deslizante para suavizar a trajetória. Por fim, é criada uma nova transformação descrita pela equação 1

$$T_f = T + (t_s - t) \tag{1}$$

em que T_f é transformação final, T é a transformação entre dois quadros consecutivos , t_s é a trajetória suavizada e t é a trajetória gerada para x, y e ângulo, então, a nova transformação T_f é aplicada a todos os frames do vídeo, gerando um novo vídeo estabilizado.

3 Resultados

Esta seção apresenta os principais artefatos gerados por esse projeto: conjunto de dados, perfilamento do código serial e gráficos de speedup e eficiência.

3.1 Programa Sequencial

A região sequencial escolhida para ser paralelizada foi um procedimento chamado **transformationFrame**. Ele recebe com entrada um conjunto de frames, itera sob os mesmos, a cada iteração são lidos 2 frames, estes são convertidos para escala cinza.

Na sequência, são selecionados os cantos mais significativos em uma imagem através do procedimento **goodFeaturesToTrack**, em seguida, é calculado o fluxo óptico para os 2 frames em questão, utilizando o procedimento **calcOpticalFlowPyrLK**.

Por fim, é calculada uma transformação afim ótima entre os dois conjuntos de pontos 2D gerado pelo passo anterior, utilizando o procedimento **estimateRigidTransform**. Essa transformação é mais conhecida como homografia, representada como uma matriz 3x3, ela é decomposta e aramzenada em dx, dy e da(ângulo).

3.2 Profiling

Para realizar o profiling foi utilizada a ferramenta Vtune, como pode ser visto na Figura 1, o procedimento **transformationFrame** consome mais de 80% do tempo de utilização da CPU no programa sequencial.

O procedimento **transformationFrame** foi escolhido para ser paralelizado não só porque é o mais caro de todos, mas porque dada a sequêcia de transformações de um vídeo $T_1(f_1, f_2), T_2(f_2, f_3), \cdots, T_{n-1}(f_{n-1}, f_n)$, em que T_i é o conjunto de todas as operações utilizadas para calcular uma transformação rígida entre dois frames e f_i um frame, os pares de frames são processados de forma independente, de modo que a transformação T_i não depende do resultado das transformações anteriores a ela.

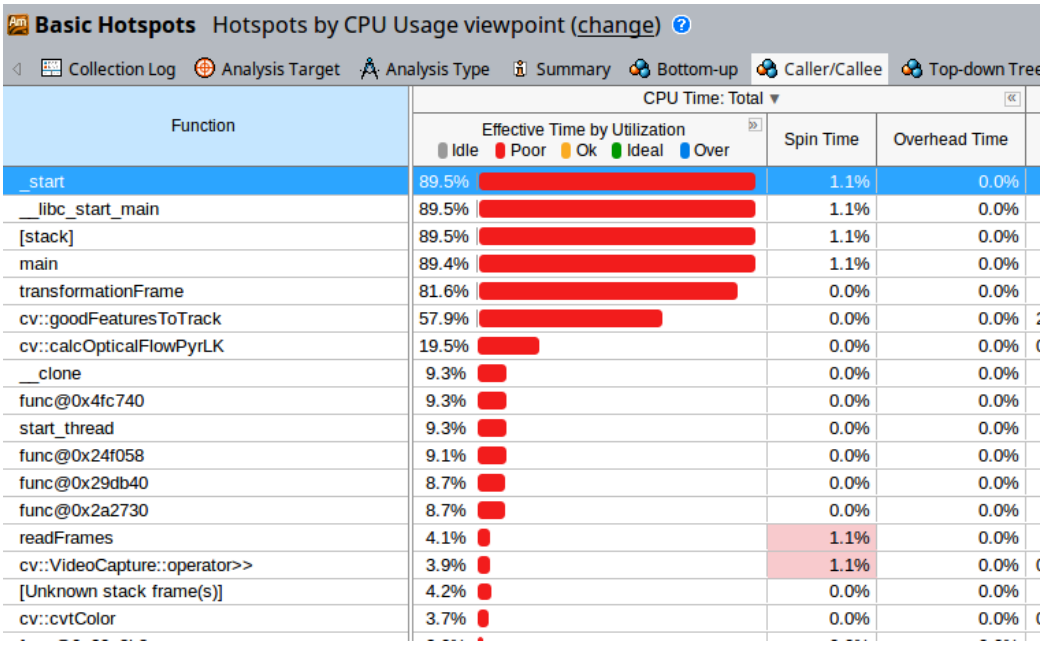


Figure 1: Tempo de utilização da CPU do programa sequencial.

3.3 Paralelização

A paralelização do procedimento **transformationFrame** utilizando as bibliotecas **Pthread** e **OpenMP** implementa a arquitetura ilustrada na Figura 2. Nessa arquitetura, um vídeo composto por um conjunto de frames é subdividido em subconjuntos, de modo que cada subconjunto é atribuído a uma determinada thread.

A paralelização do procedimento transformationFrame utilizando Pthread não faz uso de quaisquer artifícios como semáforos, mutex e etc. A versão OpenMP de transformationFrame é similar a versão Pthread, exceto que são usados alguns pragmas que evitam a recriação de threads, são usadas também algumas barreiras para sincronizar o trabalho das threads.

3.4 Testes

Antes que os programas paralelizados fossem testados, foi criado um conjunto de 4 vídeos, cada um composto por três resoluções: HQ (480p), HD (1080) e 4k (3840p), formando assim um total de 12 vídeos. Os vídeos estão nos formatos .mp4 e .webm, que são compactos e ajudam poupar memória. Todos

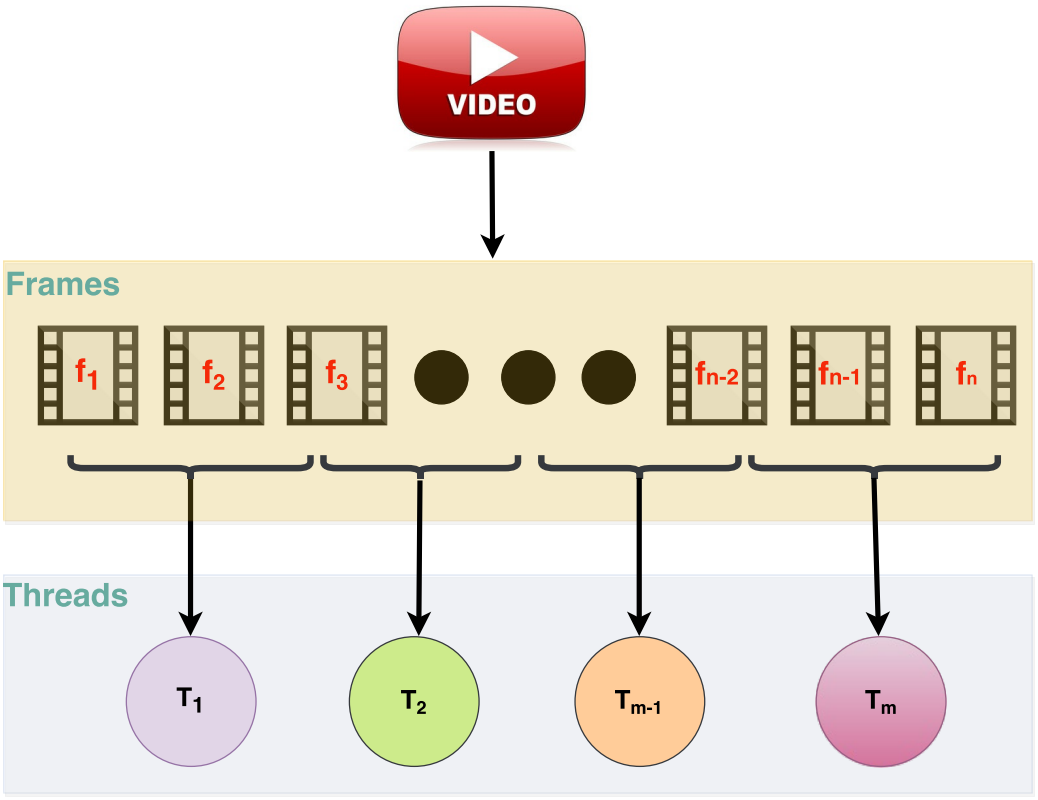
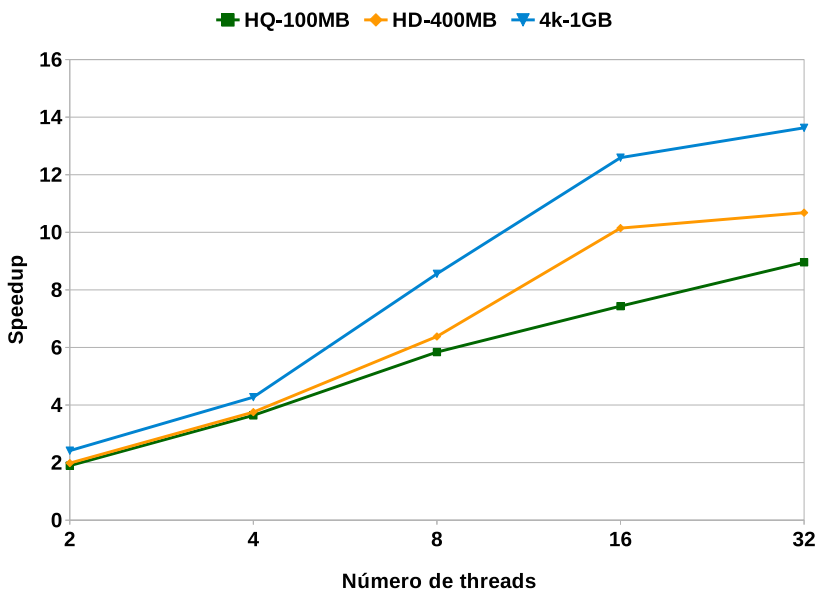


Figure 2: Arquitetura paralela do problema de estabilização de vídeo.

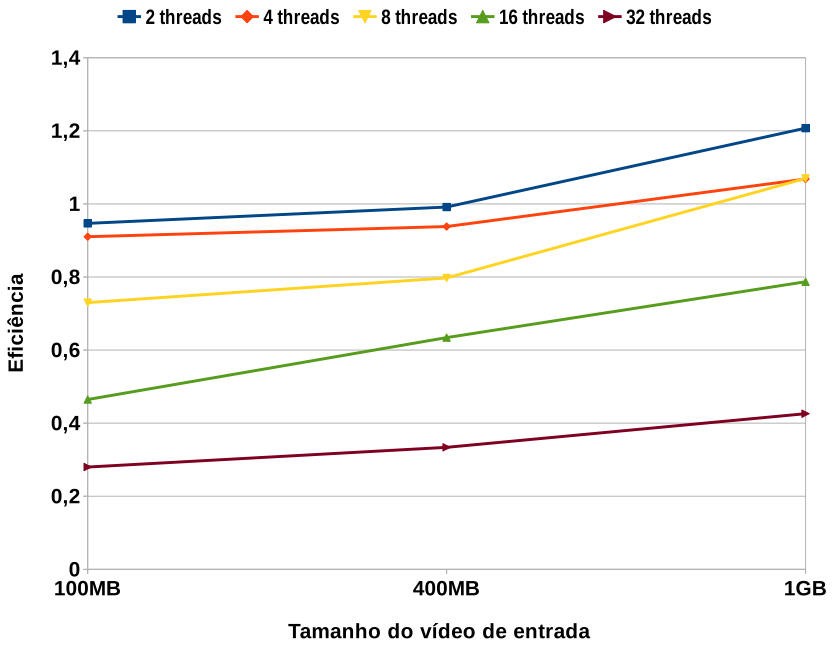
os vídeos são transmitidos a uma taxa de 30 FPS.

Os testes dos programas paralelos Pthread e OpenMP foram executados na infraestrutura da Azure em uma máquina intel Xeon(R) CPU E5-2698B v3 @ 2.00GHz, 32 cores e 441 GB de memória RAM.

Foram executadas sessões de testes com 2, 4, 8, 16 e 32 threads, para todos os 12 vídeos, os resultados de speedup e eficiência apresentados nesse trabalho foram extraídos do teste que produziu o maior speedup para um determinado vídeo. Os resultados dos testes de speedup e eficiência estão ilustrados nas Figuras 3 e 4, por último, a Figura 5 ilustra a comparação entre os speedups de Pthread e OpenMP.

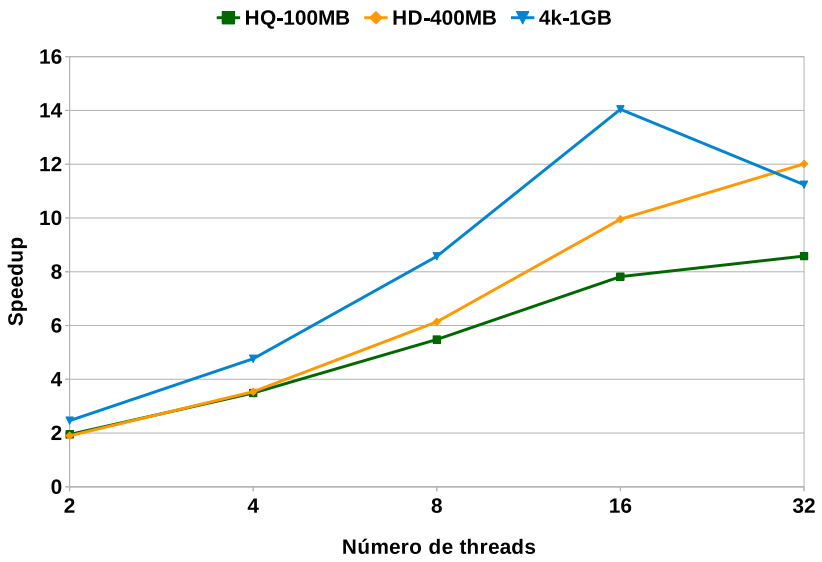


(a) Speedup

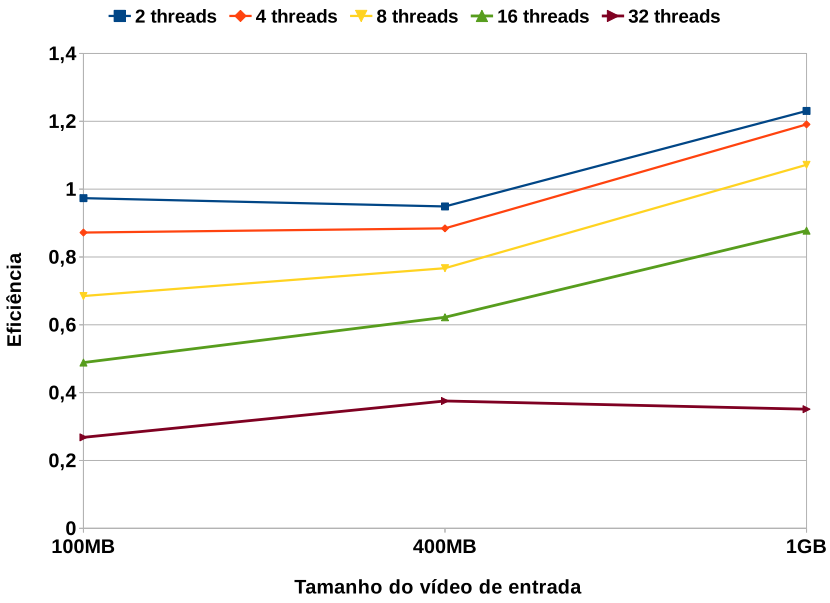


(b) Eficiência

Figure 3: Speedup e Eficiência para o programa paralelo em Pthread.



(a) Speedup



(b) Eficiência

Figure 4: Speedup e Eficiência para o programa paralelo em OpenMP.

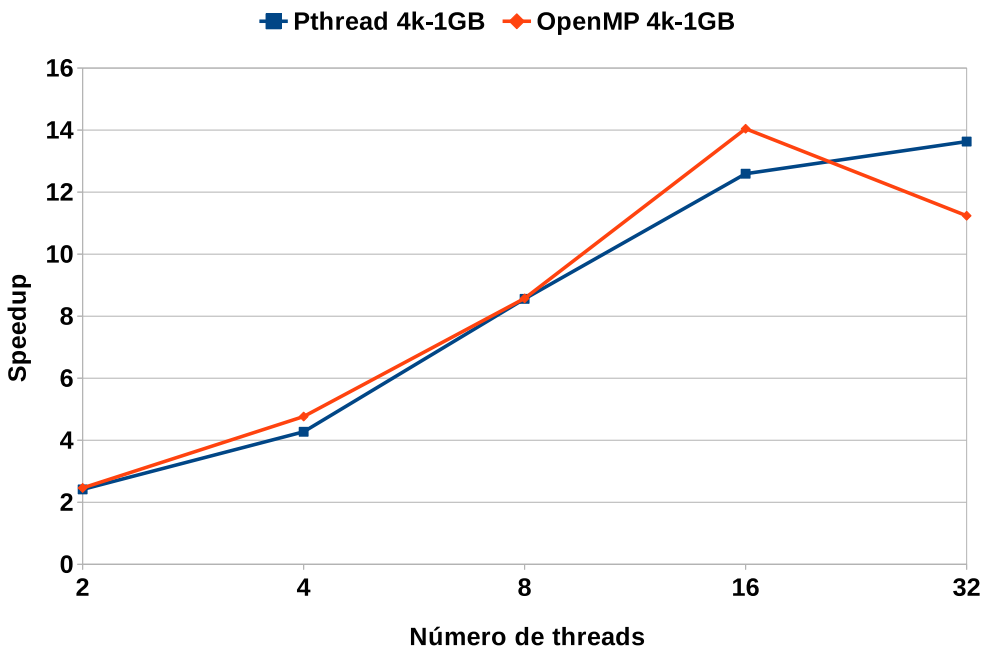


Figure 5: Speedup para um vídeo 4k - 1GB, Pthread vs OpenMP.

4 Análise dos Resultados

Os gráficos das Figuras 3a e 4a apresentam uma leve diferença na curvas dos vídeos 4k-1G, em que a versão do programa paralelo em OpenMP atinge seu ápice com um speedup de 14, porém, ao utilizar 32 threads o resultado diminui. Já a versão Pthread, apresenta um crescimento, de modo que com 32 threads atinge seu ápice com um speedup de 13.62.

Para os gráficos das Figuras 3b e 4b pode-se notar que para 2, 4 e 8 threads, a eficiência ultrapassa o valor de 1, caracterizando-se como super linear. Para os demais valores a eficiência tende a crescer.

Isso ocorre, segundo Gustafson (1988) porque o speedup não está limitado a parte serial do código, na realidade o paralelismo aumenta de acordo com o tamanho do problema, sendo assim, ao aumentar o tamanho do problema, o percentual da porção serial do programa irá diminuir, resultando numa maior eficiência. Portanto, os programas paralelos Pthread e OpenMP são escaláveis quando o tamanho do problema aumenta.

A Figura 5 mostra uma comparação entre os speedups das versões parale-

las Pthread e OpenMP, pode-se notar que a variação é pequena, isso implica que as curvas de eficiência entre eles também irá variar pouco.

5 Conclusão

Embora esse trabalho tenha alcançado um ganho razoável de speedup utilizando as bibliotecas Pthread e OpenMP, os resultados teriam sido mais expressivos se o programa paralelo fosse implementado na platafroma de computação cuda, isso reduziria drasticamente o tempo de execução do programa, aumentando assim o speedup, portanto, a paralelização utilizando CPUs é interessante apenas para vídeos pequenos.

6 Dificuldades

As principais dificuldades encontradas para realizar esse trabalho foram:

- Construção de um conjunto de dados adequado.
- Gasto de tempo elevado para realizar os testes.
- Diferenças entre as aquiteturas cuda e OpenCV.

7 Referências

http://nghiaho.com/?p=2093
http://opencv.org/
https://github.com/lecasax/video-stabilization
https://software.intel.com/en-us/intel-vtune-amplifier-xe