Avaliação da Performance de Filtros em Vídeo Usando Processamento Paralelo com Threads

1. Introduction

Applying filters to videos is an area with increasing demand for power computational. Real-time pixel manipulation, especially in vídeo high resolution, requires significant processing resources. The use of processing parallel with threads emerges as a promising strategy to optimize the performance and achieve greater efficiency in the application of these filters.

This article evaluates the impact of using threads when applying filters to videos, with focus on the black and white filter as a case study. The objective is to identify the gain of performance (speedup) relative to sequential execution and determine the configuration optimal number of threads to maximize performance. To do this, we use an environment controlled with a Ryzen 7 5700G processor, 16GB of RAM (dual Channel) and a SSD, seeking to achieve a speedup of 2.0.

1. Introdução

A aplicação de filtros em vídeos é uma área com crescente demanda por poder computacional. A manipulação de pixels em tempo real, especialmente em vídeos de alta resolução, exige recursos significativos de processamento. O uso de processamento paralelo com threads surge como uma estratégia promissora para otimizar o desempenho e alcançar maior eficiência na aplicação desses filtros.

Este artigo avalia o impacto do uso de threads na aplicação de filtros em vídeos, com foco no filtro preto e branco como estudo de caso. O objetivo é identificar o ganho de desempenho (speedup) em relação à execução sequencial e determinar a configuração ideal de threads para maximizar a performance. Para isso, utilizamos um ambiente controlado com um processador Ryzen 7 5700G, 16GB de RAM (dual Channel) e um SSD, buscando alcançar um speedup de 2.0.

2. Implementação

A implementação da aplicação de filtros em vídeo foi realizada em Python, utilizando a biblioteca OpenCV para manipulação de imagens e vídeos. O processamento paralelo foi implementado com o uso de threads, dividindo a carga de trabalho entre múltiplas threads para acelerar a aplicação dos filtros.

O script implementado utiliza filas para gerenciar o fluxo de frames entre as threads. Uma thread é responsável por ler os frames do vídeo de entrada e colocá-los na fila de entrada. As demais threads consomem os frames da fila de entrada, aplicam o filtro escolhido e colocam os frames processados na fila de saída. Por fim, uma thread de escrita consome os frames da fila de saída e os escreve no vídeo de saída.

3. Resultados

A avaliação experimental foi conduzida utilizando um vídeo de 1 hora e 30 minutos a 720p ("Resident Evil 2") como entrada. O tempo de execução para a aplicação do filtro preto e branco foi medido em diferentes configurações de threads, e a média de cinco execuções foi calculada para cada configuração.

Durante os testes, algumas configurações não se mostraram eficazes:

- **Carregamento completo do vídeo:** A tentativa de carregar o vídeo inteiro na memória resultou em estouro da memória de 16GB, impossibilitando a análise.
- **Processamento de 100 frames por thread:** A abordagem de cada thread processar 100 frames de uma vez não apresentou os resultados esperados, consumindo mais tempo.
  (batch_size sendo o tanto de frame a consumir de uma vez da fila)

```
• for _ in range(num_threads):
•         threading.Thread(target=process_frames,
  args=(batch_size,)).start()
```

Após essas tentativas, a configuração com uma fila de tamanho 10 para o gerenciamento dos frames e a threads pegando apenas um frame por vez mostrou-se a mais eficiente.

```
# Fila de entrada e saída
input_queue = queue.Queue(maxsize=10)
output_queue = queue.Queue(maxsize=10)
```

Os resultados obtidos com essa configuração são apresentados a seguir:

Configuração	Tempo Médio (min:seg)	Speedup
Sem Threads	10:10	1.0
2 Threads	6:42	1.53
4 Threads	6:39	1.55
6 Threads	6:31	1.59
8 Threads	6:40	1.54
10 Threads	6:41	1.54

Os resultados demonstram que o uso de threads proporcionou um speedup significativo em relação à execução sequencial. A configuração com 6 threads atingiu o melhor resultado, com um speedup de 1.59, aproximando-se da meta de 2.0. No entanto, o acréscimo de mais threads não resultou em melhorias adicionais de desempenho, indicando que o sistema atingiu um ponto de saturação, possivelmente devido à sobrecarga de gerenciamento das threads e à contenção de recursos.

4. Conclusão

Este estudo avaliou o impacto do uso de threads na aplicação de filtros em vídeos, demonstrando um ganho de desempenho considerável em relação à execução sequencial. A configuração com 6 threads mostrou-se a mais eficiente para o ambiente de teste utilizado, atingindo um speedup próximo da meta de 2.0.

No entanto, é importante ressaltar que o desempenho pode variar dependendo do hardware utilizado e das características do vídeo de entrada. Em trabalhos futuros, pretende-se explorar outras configurações de threads e avaliar o impacto em vídeos de diferentes resoluções e codecs, além de propor uma matriz onde o usuário pode colocar o próprio filtro e a otimização do código para alcançar um speedup ainda maior, e trabalha com processamento ao vivo de filtro aplicados em tempo real.