EA876 – Introdução Software de Sistema

Paralelização na aplicação de filtro em imagens

Rodrigo Caus (186807) e Victor Ferrão Santolim (187888).

Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação, Campinas-SP.

Introdução e Objetivo

Uma imagem digital pode ser compreendida por um conjunto de três matrizes, em que cada célula destas indica a quantidade de vermelho, verde e azul presente em cada pixel da imagem, em um valor de 8 bits[1]. Esse sistema de cor é chamado de RGB (do inglês *Red, Green* e *Blue*). A aplicação de filtros em imagens consiste no tratamento dessas matrizes com uma outra matriz chamada de núcleo (*kernel*) em um procedimento denominado de **convolução**[2].

O objetivo deste trabalho é implementar um programa que permita identificar se a aplicação de um determinado filtro em uma imagem pelo processo de convolução é mais rápida se ocorrer em uma linha única de execução, em múltiplas *threads* ou em múltiplos processos.

Metodologia

Utilizamos funções de uma biblioteca externa de manipulação de imagens para realizar as leituras, efetuar a descompressão das imagens JPEG e gerar as matrizes RGB correspondentes, assim como obter as suas dimensões. O uso de uma biblioteca externa para a manipulação dos arquivos de imagem é decorrente da não trivialidade em se obter as matrizes RGB que desejamos, pois essas informações encontram-se comprimidas no formato JPEG e o método de compressão/descompressão pode variar entre o conjunto de imagens teste selecionadas.[3]

A etapa seguinte foi definir como seriam tratados os casos de borda, ou seja, aqueles onde a vizinhança do pixel sendo calculado no momento transpassa os limites dimensionais da matriz original. Decidimos que nesses casos os pixels de borda seriam expandidos aritmeticamente e portanto haveria um "contorno" de espessura igual à parte inteira da divisão da ordem da matriz de filtro por dois. Desse modo, durante a convolução, se houvesse a tentativa de acessar um pixel fora dos limites das matrizes RGB, seria realizado alternativamente o acesso ao pixel mais próximo localizado dentro do limite.

Implementamos uma rotina que realizava de forma sequencial a convolução de um kernel (ou máscara) com as matrizes RGB de uma imagem, de modo a gerar novas matrizes de dimensões iguais às originais e que correspondem ao resultado da filtragem em cada canal de cor. A convolução foi implementada com quatro loops aninhados que tratam simultaneamente as operações nas três matrizes originais (RGB). Se o valor de algum canal do pixel obtido pela convolução for maior que 255 ele é salvo como 255, se for negativo é salvo como zero. A operação de truncamento impede overflow e mantém o padrão de cor RGB 8 bits.

Para paralelizar a operação de filtragem usando threads ou processos, decidimos que haveria uma pré divisão em áreas de trabalho para cada trabalhador (thread ou processo que realiza os cálculos). Como sabemos a altura H da imagem antes de filtrá-la, aquela pode ser subdividida verticalmente em N regiões de tamanho aproximadamente igual, onde N é o número de trabalhadores que serão usados. A figura 1 ilustra essa divisão.

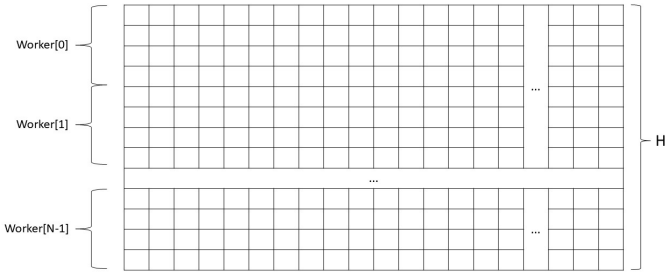

Figura 1 - Ilustração da divisão dos workers por linhas da imagem.

O modo como foi escolhida a divisão de trabalho impede a condição de corrida pois cada trabalhador escreve em regiões diferentes da matriz de saída, e na matriz de entrada há apenas leitura. Por fim, as matrizes de saída são convertidas em JPEG por funções da biblioteca externa.

Resultado

Utilizando um conjunto de 9 imagens JPEG com dimensões entre 256x256 e 8192x5461 pixels, aplicamos um filtro *emboss*[2] modificado de ordem cinco. O número de trabalhadores escolhidos foi 8, tanto na implementação com multithreads quanto na com multiprocessos. O computador utilizado para a executar o programa possui 4 núcleos físicos e 8 threads lógicas, operando em uma frequência base de 2,6 GHz e dispondo de 16 GB de memória RAM.

Figura 2 - Aplicação do filtro *emboss* por linha única de execução.

A imagem com dimensões 256x256 pixels demorou 9ms para ser filtrada em multithread, 28ms em linha única de execução e 32ms em multiprocesso. É observado aqui o impacto constante do *overhead* de criação de processos, gerenciada pelo sistema operacional, no tempo da filtragem em imagens pequenas, superior ao da disparada de uma *thread*. Essa teoria é suportada ao verificar que o tempo gasto pelo filtro multiprocessado para realizar a filtragem em uma imagem 1023x409 é de 53ms, ou seja, menos que o dobro de tempo para operar sobre 6,4 vezes mais pixels.

No caso de uma imagem de tamanho intermediário (1024x768 pixels), esse impacto é melhor diluído no tempo total de processamento, mais significante. Utilizando threads, essa filtragem foi realizada em 61ms, multiprocessos estão similares, com tempo de 78ms, o método sequencial levou 260ms, mais que o triplo que as versões paralelizadas. O padrão observado até então se acentua no teste da maior imagem. Multithread levou 3,35s para aplicar o filtro, seguido de multiprocesso com 3,83s. A linha de execução sequencial levou 14,27s para concluir a operação, 4 vezes pior que threads e processos. Em todos as vezes que os conjuntos de testes foram feitos, os resultados se mantiveram similares, com desvios menores que 5% de tempo em relação ao primeiro resultado obtido.

Foi possível concluir dessa forma que a aplicação de filtros em imagens é uma operação que pode ser beneficiada com paralelismo em mais de 4 vezes em comparação com linha única de execução, considerando a configuração, hardware e software descritos. Da forma como implementamos a divisão de tarefas, multithreading é mais rápido em qualquer tamanho de imagem testada. A diferença do tempo gasto por multiprocesso e multithreading diminui (em proporção ao tempo total) com o aumento do tamanho da imagem, devido principalmente à menor influência do tempo de overhead de criação de processos no valor total gasto na operação.

Referências

[1] Pesco, Dirce U. Bortolossi, Humberto J. **Matrizes e Imagens Coloridas**. Disponível em <http://www.uff.br/cdme/matrix/matrix-html/matrix_color/matrix_color_br.html>.

[2] Gimp. **Filtros Genéricos:** Matriz de convolução. Disponível em <https://docs.gimp.org/2.8/pt_BR/plug-in-convmatrix.html>.

[3] Tom Lane. **JPEG image compression FAQ**. Disponível em <http://www.faqs.org/faqs/jpeg-faq/part1/>.