Universidade Estadual de Campinas - UNICAMP
EA876 - Compiladores e Sistemas operacionais
Turma A - 1s/2018
Mariane Tiemi Iguti (RA 147279)
Gabriela Akemi Shima (RA 135819)

EA876A - Relatório 2

Introdução

Este trabalho consiste de três programas que aplicam um filtro de área blur, sendo o primeiro deles implementado em uma solução single-thread, o segundo em multithread e o terceiro em multiprocessos. O objetivo será medir o *tempo de execução* de cada um dos programas para a discussão e comparação entre as três propostas de tratamento da aplicação.

Método

Cada imagem é constituída de 3 matrizes R[ ]. G[ ] e B[ ], sendo estes na verdade vetores que serão tratados como matrizes por meio de cálculos com os seus índices (**Figura 1**). Em cada uma das células existe um valor que vai de 0 a 255 que determina a cor daquele pixel para construir a cor final RGB da imagem.

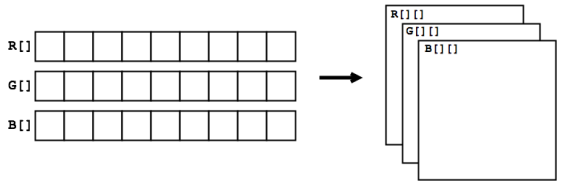

Figura 1: Diagrama de representação matricial e vetorial da imagem.

O efeito Blur que foi implementado, consiste em dado um N, que é aplicado em uma área ao redor do pixel (i,j) de (i-N),(j-N) a (i+N),(j+N), substituindo seu valor pela média de todas as células inclusas nesta área, como é mostrado na **Figura 2**. Caso o pixel em questão esteja na borda ou em uma posição que não possua esta área completa, o cálculo é feito com os pixels que compõem a área disponível.

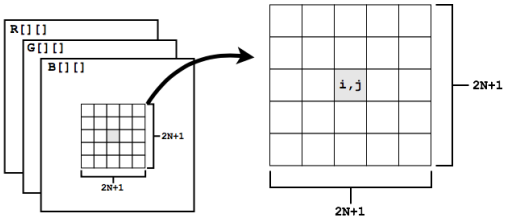

Figura 2: Diagrama de como o filtro blur atua com uma entrada de área N.

A primeira implementação consiste em um processo único e simples que executa o algoritmo de modo descrito anteriormente. As duas seguintes seguem uma solução de programação paralela para fins de comparação entre as três implementações, uma delas em multiprocesso, outra em multithread.

O algoritmo consiste em deixar um fluxo, no caso thread ou processo, para cada vetor: R[ ], G[ ], B[ ]. Cada vetor é dividido ao meio, sendo a primeira metade o cálculo da célula será responsabilidade de um novo fluxo criado, e a metade restante da própria thread/processo.

Deste modo os cálculos foram paralelizados em cada matriz, e também dentro de cada fluxo, com a possibilidade de ser implementados tanto em um cenário de multithread quanto de multiprocesso. Além de nesta solução não haver o problema de concorrência pois cada fluxo escreve em um espaço diferente da memória.

.

Resultados

O trabalho consiste de três programas que funcionam separadamente. Os tempos obtidos na tabela 1: *single_t, multi_t* e *multi_p* são correspondentes ao tempo de execução dos programas que processam a imagem em solução *singlethread* (fluxo único), *multithread* (threads múltiplas) e *multiprocessos* (múltiplos processos), respectivamente.

Cada entrada (imgXXX.jpg) foi executada por cada um dos programas 5 vezes, os valores obtido da tabela 1 são as médias de tempo de execução para as 21 entradas distintas.

arquivo	dimensão (px)	single_t (ms)	multi_t (ms)	multi_p (ms)
img001.jpg	1920 x 1280	6209.954200	5496.454600	3.222400
img002.jpg	1024 x 683	1684.345600	1477.493200	1.129400
img003.jpg	1920 x 1280	5395.209800	5014.412800	1.191200
img004.jpg	1920 x 1280	5563.635800	4991.559400	1.232200
img005.jpg	1920 x 1280	5320.261200	4960.587600	1.441400
img006.jpg	1024 x 683	1678.241200	1470.660200	1.086800
img007.jpg	620 x 400	478.455400	451.414200	788200
img008.jpg	1024 x 683	1673.979800	1478.889600	1.003000
img009.jpg	1920 x 1280	5288.795800	4891.995600	1.391400
img010.jpg	1920 x 1280	7123.360600	5866.085600	1.287600
img011.jpg	1920 x 1280	5983.588800	6215.371200	1.240200
img012.jpg	1920 x 1280	5336.079600	5319.726800	1.147400
img013.jpg	840 x 560	966.609800	882.532000	915600
img014.jpg	1920 x 1280	5327.953000	5146.039000	1.268000
img015.jpg	1920 x 1279	7091.738200	6323.422800	1.165200
img016.jpg	840 x 560	1488.038600	1720.652000	1.223600
img017.jpg	1920 x 1280	7454.245000	7883.777800	1.672600
img018.jpg	833 x 625	1331.628000	1415.703000	1.451800
img019.jpg	2048 x 1462	8201.661400	8856.250400	1.955800
img020.jpg	1920x 1280	5806.157200	5804.887200	1.387200

Tabela 1: Tabela de resultados de execução.

O que se pode observar é que não houve uma queda tão elevada da solução single thread para multithread, mas houve uma melhora significativa em tempo de execução para o multiprocessos.