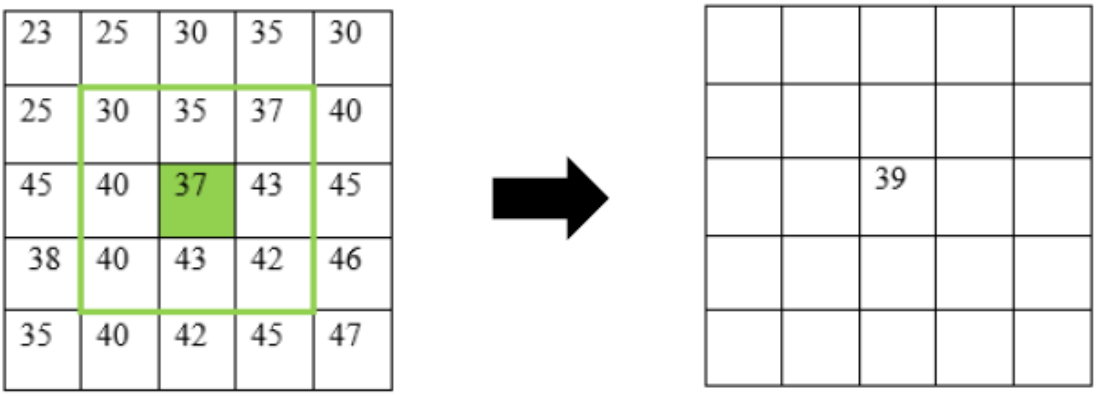
Laboratório de Programação Concorrente

Lab2 - Image Noise - 24.2

Objetivo

Nesse laboratório, vocês irão trabalhar com o melhoramento de desempenho de um programa que reduz o ruído em imagens. O ruído em imagens digitais consiste em variações indesejadas no brilho ou nas cores, que ocorrem de forma aleatória e comprometem a qualidade visual. Essas variações podem surgir por diferentes motivos, como limitações no sensor da câmera, condições inadequadas de iluminação, ou até mesmo problemas durante a transmissão e compressão de dados [1]. Abaixo, apresentamos um exemplo de imagem digital que possui ruído, e a mesma imagem após o processamento de remoção de ruído utilizando um kernel 3x3.

Hoje, vocês irão trabalhar com o método de filtro de média [2]. O processo por trás dele é como o de uma janela deslizante que passa por partes da imagem e troca cada entrada pela média da própria entrada com os seus vizinhos. A figura abaixo exemplifica esse processo. Observe que a entrada atual é 37 e temos um filtro de dimensão 3x3, também chamado de kernel 3x3. Nesse caso, todos os vizinhos (30, 35, 37, 40, 43, 40, 43 e 42) e a entrada (37) são somados, e o resultado da média entre esses números irá substituir a entrada na imagem com ruído reduzido.



No código base vocês encontrarão uma implementação serial em Java que recebe como argumento o “path” da imagem que desejamos remover o ruído. A sua implementação deve adicionar concorrência a esse processo para o melhorar o desempenho do processamento e deve receber como argumento tanto o “path” da imagem quanto a quantidade de threads que seu código irá utilizar (considere a utilização de no mínimo 2 threads). Vocês devem experimentar quantidades diferentes de threads para encontrar um bom valor utilizando como base as imagens disponíveis no diretório /data do código deste laboratório.

A entrega, detalhada nas seções seguintes, envolverá o código fonte. Iremos avaliar tanto as possibilidades de plágio entre os alunos quanto a geração automática de código.

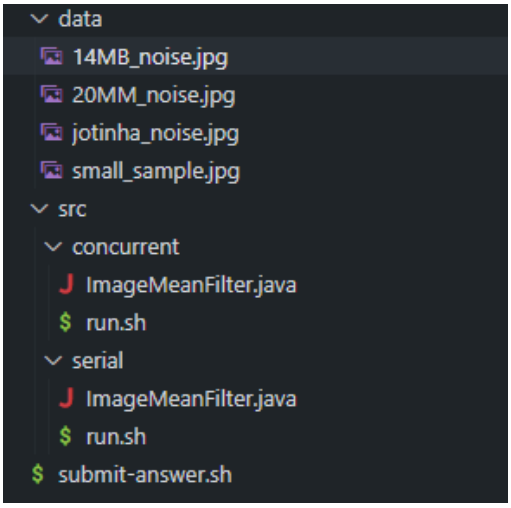
Prazo

05/12/2024 às 18h00

Visão Geral do Código Base

https://github.com/thiagomanel/fpc/tree/master/2024.2/Lab2

O código está organizado na seguinte hierarquia:



- data/:
  Contém as imagens de entrada para o processamento. As imagens disponibilizadas são:
  - small_sample.jpg: Uma imagem pequena para testes rápidos.
  - 14MB_noise.jpg, 20MM_noise.jpg e jotinha_noise.jpg: Imagens maiores para avaliar o desempenho.

- src/serial/:
  Implementação serial que será o ponto de partida para entender o funcionamento do filtro de média.

- src/concurrent/:
  Diretório onde você implementará a versão concorrente. Essa versão deve receber, além do caminho da imagem, a quantidade de threads a ser utilizada. **Note que, por padrão, esse diretório contém a mesma implementação que a serial, então você deve alterá-la para implementar a concorrência!**

- run.sh:
  Scripts para compilar e executar as implementações serial e concorrente, em seus respectivos diretórios.

- submit-answer.sh:
  Script que será utilizado para a submissão do laboratório.

Preparação

1. Clone o repositório do código base

   git clone [link do repositório]

Faça uma comparação de desempenho entre as versões serial e concorrente (detalhes abaixo):

Versão Serial

1. Navegue até o diretório da implementação serial:
   cd fpc/2024.2/lab2/src/serial
2. Execute a versão serial especificando a imagem:
   ./run.sh ../../data/small_sample.jpg

Versão Concorrente

1. Navegue até o diretório da implementação concorrente:
   cd fpc/2024.2/lab2/src/concurrent
2. Execute a versão concorrente, especificando a imagem e o número de threads:
   run.sh ../../data/small_sample.jpg 4

Comparação de Desempenho

Entendendo o output do script run.sh:

- real: o tempo total decorrido
- user: o tempo total que o processo gastou utilizando a CPU em modo usuário
- sys: o tempo total que o processo gastou utilizando recursos do kernel

Interpretação

- real: é o tempo que você veria em um cronômetro
- user + sys: representa o tempo efetivamente gasto pela CPU no processamento

Se o programa usar múltiplas threads em um sistema com vários núcleos, o valor de user pode ser maior que real, já que múltiplas threads podem trabalhar simultaneamente.

Entrega

Você deve criar e manter um repositório privado no GitHub com a sua solução. No entanto, a entrega do laboratório deverá ser realizada por meio de submissão online utilizando o script submit-answer.sh, disponibilizado na estrutura de arquivos do próprio laboratório. Uma vez que você tenha concluído sua resposta, seguem as instruções:

1) Crie um arquivo lab2_matr1_matr2.tar.gz somente com o "src" do repositório que vocês trabalhou. Para isso, supondo que o diretório raiz de seu repositório privado chama-se lab2_pc, você deve executar:
   tar -cvzf lab2_matr1_matr2.tar.gz lab2_pc/src

2) Submeta o arquivo lab2_matr1_matr2.tar.gz usando o script submit-answer.sh, disponibilizado no mesmo repositório do laboratório:
   bash submit-answer.sh lab2 path/to/lab2_matr1_matr2.tar.gz

Lembre-se que você deve manter o seu repositório privado no GitHub para fins de comprovação em caso de problema no empacotamento ou transmissão online. Alterações no código realizadas após o prazo de entrega não serão analisadas.