Processamento de imagens de grandes dimensões com MPI

D

Em processamento de sinal uma das operações mais importantes é a remoção de ruído que, estando sobreposto, degrada a qualidade da informação de qualquer sinal. Em imagens é comum existir ruído de alta frequência que se manifesta de forma estática ao longo da imagem. A solução pode ser obtida usando uma técnica básica de remoção do ruído de alta frequência das componentes da imagem passando-a por um filtro "passa-baixo".

Existem basicamente dois tipos de técnicas para remoção do ruído, as que são aplicadas no dominio temporal ou espacial e as que são aplicadas no domínio das frequências. Para usar o segundo tipo em imagens podemos recorrer à Transformada Discreta de Fourier (DFT) bidimensional.

D.1 O Cálculo da DFT

Considerando uma sequência de números complexos $x_0, x_1, ..., x_{n-1}$ os valores correspondentes do vector da transformada são dados por

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi \frac{kn}{N}},$$

onde, segundo a fórmula de Euler,

$$e^{i\theta} = \cos(\theta) + i\sin(\theta).$$

A transformada inversa pode ser aplicada a uma sequência de números complexos (do domínio das frequências) como

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{i2\pi \frac{kn}{N}}.$$

No caso bidimensional os valores da transformada, também ela bidimensional, são dados por

$$F(k,l) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-i2\pi\left(\frac{km}{M} + \frac{ln}{N}\right)}.$$

Com uma manipulação simples temos

$$F(k,l) = \frac{1}{MN} \sum_{m=0}^{M-1} \left[\sum_{n=0}^{N-1} f(m,n) e^{i2\pi\left(\frac{ln}{N}\right)} \right] e^{i2\pi\left(\frac{km}{M}\right)}$$

ou seja temos que a transformada bidimensional pode ser obtida calculando primeiro a transformada unidimensional a cada uma das linhas e de seguida aplicar a mesma transformada a cada uma das colunas que resultaram do passo anterior.

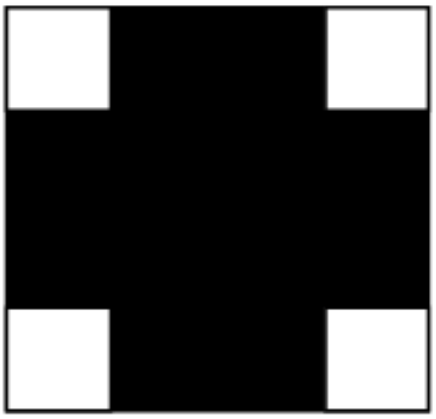

Figura D.1: Máscara de filtragem passa baixo em que os píxeis pretos correspondem aos valores zero e os brancos aos valores um. Os quadrados brancos têm lado igual a 1/4 do lado da imagem. *Note-se que a linha exterior a preto apenas representa os limites da máscara (nesta figura) e não faz parte da mesma.*

A aplicação de um filtro passa-baixo a uma imagem passa a ser muito simples através da Transformada de Fourier, através dos seguintes passos:

1. A partir da imagem a filtrar crie uma matriz com os valores de luminância (níveis de cinzento) e uma segunda matriz com a parte imaginária correspondente que será zero para todos os elementos.

2. Calcule a transformada de Fourier discreta que irá resultar também em duas matrizes, uma para as componentes reais e outra para as imaginárias.

3. Usando uma matriz de iguais dimensões cujos elementos serão zeros nas zonas correspondentes às frequências que queremos eliminar e uns nas que queremos "deixar passar"(ver figura H.1), vamos multiplicar elemento a elemento as duas matrizes obtidas no passo anterior.

4. Calcule a transformada de Fourier inversa da imagem filtrada

D.2 Estruturas de dados

Foi disponibilizado o código necessário para ler e escrever ficheiros contendo imagens no formato PGM. Para simplificar a passagem de imagens entre funções foram definidos os seguintes tipos:

```
struct Matrix{
  int rows;
  int cols;
  unsigned char * data;
  int widthStep;
};
typedef struct Matrix Image;

struct MatrixF{
  int rows;
  int cols;
  double * data;
  int widthStep;
};
typedef struct MatrixF ImageF;
```

Note-se que as matrizes (imagens) foram pensadas para conter apenas um canal de cor, por isso suportam apenas imagens com níveis de cinzento. Foram definidos 2 tipos de imagens:

- **Image** que armazena dados do tipo unsigned char, ou seja valores inteiros sem sinal entre 0 e 255 (1 byte).
- **ImageF** que armazena dados do tipo double, ou seja virgula flutuante de dupla precisão.

O tipo ImageF é o necessário para as operações de filtragem por envolverem a DFT, mas terão de ser convertidos de e para Image na leitura e na escrita de ficheiros PGM.

D.3 Trabalho a entregar

Este trabalho consiste em desenvolver uma solução em MPI para filtrar imagens de muito alta resolução tipicamente adquiridas via satélite e que deve ser testado usando o cluster disponibilizado.

Para simplificar a implementação, sugere-se que a máquina "root"faça a distribuição do trabalho pelas restantes. Essa distribuição deverá ser em termos de um subconjunto das linhas para estas calcularem a DFT das mesmas. De seguida em vez de se fazer a DFT por colunas, pode-se transpor as matrizes e repetir a DFT por linhas.

A filtragem deve ser feita na máquina "root"e repete-se o procedimento anterior para a transformada inversa.

Para desenvolver de forma coerente este trabalho deve seguir as seguintes etapas, criando as funções pedidas e depois otimizá-las para MPI registando o ganho conseguido.

Exercício D.1

5%

Uma função que devolve a máscara de filtragem, cujo protótipo é o seguinte:

```
ImageF *  genlpfmask(int rows, int cols);
```

- **rows**,**cols** - número de linhas e colunas da matriz (máscara) de saída.
- (retorna) - ponteiro para matriz de "doubles"com zeros e uns de acordo com a figura H.1.

Exercício D.2

15%

Uma função que calcula a DFT ou a DFT inversa de uma imagem, cujo protótipo é o seguinte:

```
void fti(ImageF * in_re, ImageF *in_img,
                ImageF* out_re, ImageF * out_img, int inverse);
```

- **in_re** - ponteiro para a matriz com as partes reais de entrada armazenada em memória linha a linha.
- **in_img** - ponteiro para a matriz com as partes imaginárias de entrada armazenada em memória linha a linha.
- **out_re** - ponteiro para a matriz com as partes reais de saída armazenada em memória linha a linha.
- **out_img** - ponteiro para a matriz com as partes imaginárias de saída armazenada em memória linha a linha.
- **inverse** - valor 0 ou 1 dependendo de se pretender a transformada de Fourier direta ou inversa
- (retorna) - Nada.

Exercício D.3

Uma função que multiplique elemento a elemento os valores de cada uma das matrizes "in_re"e "in_im"pelos da matriz "mask"e retorne os resultados respetivos em "out_re"e "out_im", de acordo com o seguinte protótipo: 10%

```
void dofilt(ImageF * in_re, ImageF * in_im, ImageF * mask,
                    ImageF * out_re, ImageF * out_im);
```

- **in_re** - ponteiro para a matriz com as partes reais de entrada armazenada em memória linha a linha.
- **in_img** - ponteiro para a matriz com as partes imaginárias de entrada armazenada em memória linha a linha.
- **mask** - ponteiro para a matriz com os valores da máscara de filtragem.
- **out_re** - ponteiro para a matriz com as partes reais de saída armazenada em memória linha a linha.
- **out_img** - ponteiro para a matriz com as partes imaginárias de saída armazenada em memória linha a linha.
- (retorna) - Nada.

Exercício D.4

Integre estas funções com o código fornecido e teste com as imagens fornecidas ou outras no formato PGM. 10%

Nota: Pode instalar o pacote ImageMagick para converter qualquer imagem para o formato PGM suportado usando o comando

```
$ convert -colorspace Gray imagem.jpg imagem.pgm
```

As imagens geradas podem ser visualizadas com o comando display do mesmo pacote, fazendo

```
$ display imagem.pgm
```

em que o $ representa o "prompt"da linha de comandos.

Exercício D.5

Faça as adaptações necessárias para tirar partido do MPI procurando melhorar o desempenho das funções:

- genlp(...) 10%
- fti(...) 30%

- dofilt(...) 20%

Para isso:

1. Teste com as imagens contidas na pasta /home/ACDATA/Imagens/ da máquina mpi-01.deec.lan e para cada uma compare os tempos de execução com e sem MPI. Note que este arquivo contém imagens com e sem ruído em subpastas separadas.

2. Crie uma pasta com todos os ficheiros necessários para compilar o código (tal como aquela que descarregou do Inforestudante) e um relatório no formato PDF onde explica a implementação e apresenta e discute os resultados obtidos. O relatório poderá ter até 3 páginas.

3. Entregue um arquivo ZIP com os ficheiros necessários à compilação incluindo o ficheiro Makefile e um relatório que deve explicar os pormenores de implementação, o modo de testar e os resultados obtidos.

   Note que deve ser possível testar descompactando apenas o ficheiro ZIP e correndo o comando make.

D.4 Bónus

Esta parte apenas será considerada se tiver completado o trabalho anteriormente pedido. Se estiver correto poderá compensar 10% da classificação não conseguida de um outro trabalho.

Exercício D.6
Repita a filtragem distribuída desta vez usando uma máscara de convolução Gaussiana 11×11. Não esquecer de analisar os ganhos e tomar em atenção as situações de fronteira.