ICC2 – Trabalho 6

Entrega: 30/09/2013 as 23h59
Sistema de Submissão de Programas: https://ssp.icmc.usp.br
Data máxima para encaminhamento de dúvidas sobre o trabalho: 25/09/2013

Faça um programa para ler um arquivo de imagem no formato PGM. Esse formato é composto por um cabeçalho que tem duas linhas iniciais (que você pode desprezar) e, em seguida, há uma linha de texto com o número de pixels de largura e de altura da imagem. A linha seguinte contém o valor máximo de escala de cinza para a imagem. Posteriormente são listados os bytes que compõem a imagem. Cada byte representa um pixel.

Exemplo:

```
P2
# CREATOR: GIMP PNM Filter Version 1.1
800 600
255
<valor do pixel em um unsigned char>...<valor do pixel em um unsigned char>
```

A primeira linha tem o texto P2, o qual deve ser descartado. Em seguida, descarte a segunda linha, a qual apresenta detalhes do software que gerou a imagem. A terceira linha contém o número de pixels que define a largura da imagem (neste caso 800 pixels de largura) e, em seguida, a altura da imagem (neste caso 600 pixels). Na linha seguinte há o valor máximo de escala de cinza para a imagem, o qual, neste caso, é 255.

Após essa parte o arquivo ***deve ser lido byte a byte (pode ler todos os bytes de uma vez também), no formato binário***. Isso porque o arquivo tem o cabeçalho ASCII e o conteúdo binário.

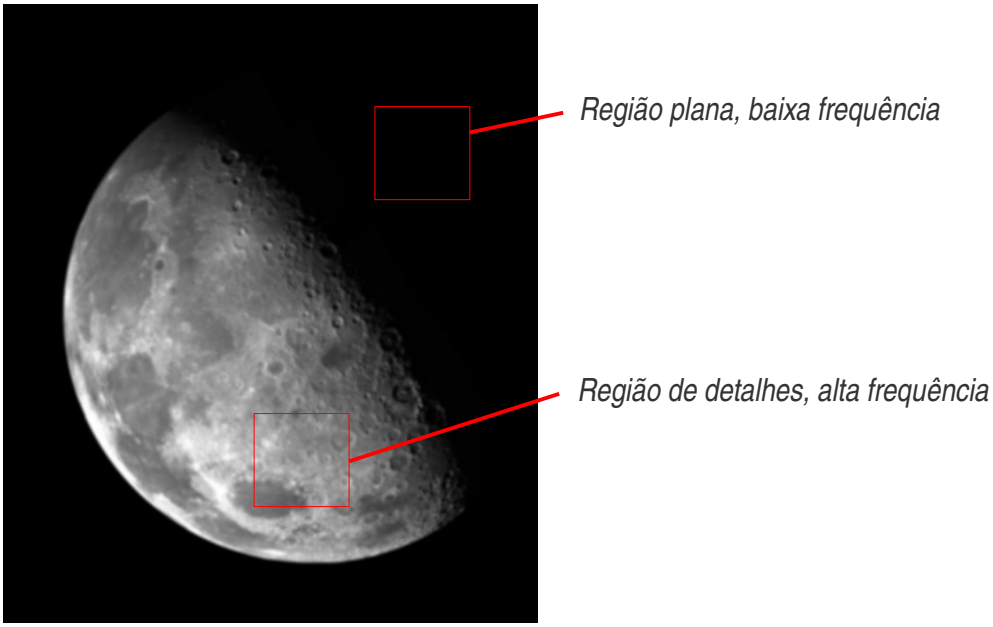
O valor máximo 255 indica que a imagem contém pixels com valor 0 até 255. Em que o 0 indica a cor preta e 255 representa a cor branca. Demais valores intermediários representam diferentes níveis de cinza.

Você deve ler o cabeçalho desse arquivo PGM e os bytes que compõem a imagem. Em seguida deve atender às tarefas abaixo listadas.

Descrição:

Uma das ferramentas mais úteis na análise e processamento de imagens é a Transformada Discreta de Fourier 2D (DFT-2D). Essa transformada permite obter um coeficiente (que é um par de valores) para cada observação (pixel) da imagem original.

De forma grosseira, esses coeficientes representam o *tamanho de padrões que se repetem na imagem*, desde padrões de menor frequência (regiões planas da imagem – com baixa variação de cor), até padrões de maior frequência (regiões com texturas e detalhes – com alta variação de cor). Veja, por exemplo a imagem a seguir:



Para estudar a repetição de padrões de menor ou maior frequência em imagens e sinais, a técnica da Transformada de Fourier utiliza descritores matemáticos de periodicidade (ou seja que permitam representar repetição de padrões). Há diversas funções matemáticas periódicas, sendo que as funções **seno** e **cosseno** são especialmente interessantes pois estão associadas com o objeto mais simples que se repete regularmente: o círculo.

Assim, a Transformada de Fourier de uma imagem realiza a multiplicação de uma exponencial complexa (que representa ao mesmo tempo seno e cosseno) e faz a soma dessa multiplicação considerando toda a imagem (ou seja, todos os pixels). Com isso, a imagem é transformada do domínio espacial, para o domínio da frequência, permitindo processar separadamente coeficientes de frequência de diferentes magnitudes (mais altas ou mais baixas).

A equação que representa a Transformada de Fourier 2D de uma imagem $f(x,y)$ de tamanho $M \times N$ para o ponto em frequência (u,v) é

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M+vy/N)},$$

Repare que essa equação só computa o coeficiente para uma determinada frequência (u,v). O resultado da transformada seria uma função $F(u,v)$ que representa as frequências u = 0 até M-1 e v = 0 até N-1.

Por exemplo, o cálculo de F(0,0) é dado por:

$$F(0,0) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)$$

Isso indica que o coeficiente (0,0) é simplesmente a soma de todos os pixels da imagem. Exceto pelo coeficiente (0,0) que é um número real, todos os outros serão coeficientes complexos, pois é possível decompor esses coeficientes em dois valores, sua parte real e sua parte imaginária. Isso é necessário para que a DFT-2D capture corretamente as diferentes frequências presentes na imagem.

Após computar todos os coeficientes, podemos obter a magnitude:

$$|F(u,v)| = \left[R^2(u,v) + I^2(u,v)\right]^{1/2}$$

Essa magnitude indica se uma determinada frequência no ponto (u,v) possui maior ou menor magnitude na imagem. Com isso podemos por exemplo processar apenas frequências referentes a regiões planas da imagem, ou apenas frequências referentes a regiões de detalhe na imagem.

Após processar as frequências, é possível retornar para o domínio do espaço, reconstruindo a imagem já com seus coeficientes alterados. Essa operação é chamada de Transformada Inversa de Fourier 2D, definida pela equação:

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi(ux/M + vy/N)}$$

Repare que há grande semelhança com a equação da DFT-2D, exceto pela troca de sinal da exponencial, e pela divisão pelo número de pixels realizada. Abaixo, uma função em linguagem C que realiza a Transformada Inversa de Fourier 2D (IFT-2D):

```
unsigned char *ift2d(double complex *coefficients, int width, int height)
{
       int x, y, u, v;
       unsigned char *newimg = (unsigned char *) malloc(sizeof(unsigned char)*width*height);

       for (x = 0; x < height; x++)
       {
          for (y = 0; y < width; y++)
          {
             double complex newPixel = 0;
             for (u = 0; u < width; u++)
             {
                for (v = 0; v < height; v++)
                {
                    double complex expon = 2.0 * M_PI * (
                                           (x*u*1.0)/(1.0*height) +
                                           (y*v*1.0)/(1.0*width) ) * I;

                    newPixel += coefficients[(v*width)+u] * cexp(expon);
                }
             }
             newimg[(x*width)+y] = (unsigned char) ( (int) creal(
                                    (1.0/sqrt(height* width*1.0))*newPixel) );
          }
       }
       return newimg;
}
```

O código envolve cálculo com números imaginários. Assim, mostramos abaixo um exemplo mais simples para definir um número complexo usando linguagem C:

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <complex.h>

int main(int argc, char *argv[]) {
	double complex numero_complexo = 3.0 + 4.0 * _Complex_I;

	double parte_real = __real__ numero_complexo;
	double parte_imaginaria = __imag__ numero_complexo;

	printf("Parte real: %lf\n", parte_real);
	printf("Parte imaginaria: %lf\n", parte_imaginaria);

	return 0;
}
```

Para compilar esses códigos use a opção -lm na compilação. Ela é necessária para termos acesso à libmath, essa biblioteca contém a função cexp() dentre outras.

Tarefas:

1) Leia do teclado o nome da imagem a ser aberta e realize a leitura da imagem

2) Leia do teclado o número de coeficientes que devem ser utilizados. Esse número inteiro K será indicado pelo usuário.

3) Aplique a Transformada de Fourier sobre a imagem

Dessa maneira, serão produzidos os seguintes coeficientes após aplicar a transformada: C_1, C_2, …, C_{MN}, ou seja, o índice dos coeficientes varia de 1 a MN, sendo MN o número total de pixels na imagem.

4) Calcule a magnitude para esses coeficientes. Em seguida ordene o vetor de coeficientes de maneira **decrescente** segundo suas magnitudes, ou seja, o coeficiente de maior magnitude será o primeiro do vetor, em seguida o segundo de maior magnitude e assim sucessivamente. Será preciso, no entanto, guardar qual é a posição original de cada coeficiente.

5) Atribua zero aos coeficientes das posições maiores do que K. Devem ser mantidos no vetor de coeficientes somente os valores dos K primeiros coeficientes.

6) Em seguida, volte os coeficientes para suas posições originais, ou seja, para as posições que eles tinham antes da ordenação.

7) Utilizando como entrada os coeficientes processados (obtidos no passo 6), aplique a Transformada Inversa de Fourier. Note que a Inversa poderá gerar valores de ponto flutuante, que deverão, na hora da escrita, ser convertidos para inteiros.

8) O programa deverá mostrar como saída (na tela do computador) a sequência ordenada (de forma decrescente) das magnitudes dos coeficientes das posições de 1 a K, convertidas para um inteiro (*casting* para *int*).

9) Produza um arquivo de saída que contenha a nova imagem, após aplicação da Transformada Inversa de Fourier. Esse arquivo deve ter como cabeçalho:

```
P5
# Created by Trabalho 05
<largura da imagem> <altura da imagem>
<valor máximo da escala de cinza>
<bytes que correspondem aos pixels da imagem>
```

Observações:

Em alguns tipos de imagens essa operação permite, por exemplo, representar uma imagem por meio de K coeficientes de Fourier, obtendo um tipo de compressão de imagem para armazenamento e transmissão. Teste seu programa com imagens utilizando diferentes valores de K para verificar a eficácia do método.

Se tiver curiosidade teste com diferentes tipos de imagens repare que esse método funciona melhor para imagens com maior quantidade de detalhes (por exemplo texturas) e portanto com valores mais distribuídos no domínio da frequência. Já em imagens mais simples, com predominância de baixas frequências, fica mais evidente o aparecimento de artefatos (ruído) decorrente da eliminação de coeficientes.

Formato de Entrada e Saída:

Entrada:

```
<nome da imagem de entrada>
<K>
<nome da imagem de saida>
```

Saída:

```
<numero total de pixels na imagem de entrada>\n
<magnitude dos coeficientes 1 a K ordenados de forma decrescente>\n
```

Exemplo (conforme imagem fornecida juntamente com o trabalho):

Entrada:

```
imgA.pgm
6
newimgA.pgm
```

Saída:

```
64
1291 222 222 118 118 88
```