

Laboratório Semanal 03

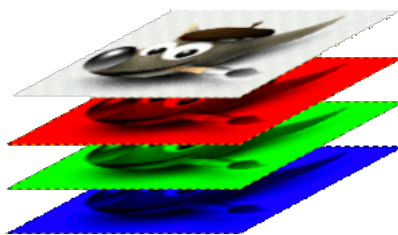
Prof. Arnaldo Moura e Prof. Lehlton Pedrosa

Prazo para entrega: 01/05/2016 às 21:59:59

1 PROCESSAMENTO DE IMAGENS

Como um dedicado aluno de computação, você gostaria de treinar os novos conceitos aprendidos na aula de matrizes e vetores em C. Desta forma, você aplicará estes conceitos na área de processamento de imagens em 3 passos: Descomprimir uma imagem colorida; Transformá-la em uma imagem em escala de cinza; Aplicar filtros sobre a imagem.

As imagens digitais são compostas por uma malha de *pixels*. A cor de cada *pixel* é atribuída de acordo com o sistema RGB que, através da composição aditiva das cores primárias Vermelho (R-*Red*), Verde (G-*Green*) e Azul (B-*Blue*), geram as cores derivadas. Na imagem digital, a composição de cores de cada *pixel* é formada por 3 valores de intensidade de cor variando de [0-255]. Desta forma, uma imagem é resultado da sobreposição de 3 malhas (matrizes) de intensidade de cores, como na figura abaixo.



Para reduzir o espaço de armazenamento de uma imagem, ela pode ser comprimida. Um dos algoritmos mais simples de compressão é o RLE (*Run Length Encoder*). Ele funciona reduzindo o tamanho físico de uma sequência de repetição de caracteres. Na imagem, para cada matriz (ou malha de intensidade de cores), uma sequência repetida da mesma intensidade é substituída pelo par (numero de vezes que se repete, intensidade). O exemplo abaixo ilustra esta codificação para uma matriz X . A codificação Y indica que em X existe 1 valor 15, seguido de 2 valores 12, e assim por diante.

$$\begin{array}{cccccc}
 & 15 & 12 & 12 & 10 & 12 & 12 & 15 \\
 X = & 12 & 12 & 10 & 10 & 10 & 12 & 12 \\
 & 12 & 10 & 10 & 10 & 10 & 10 & 12 \\
 & 10 & 10 & 10 & 10 & 10 & 10 & 10
 \end{array}
 \quad
 \begin{array}{cccccc}
 & 01 & 15 & 02 & 12 & 01 & 10 & 02 & 12 & 01 & 15 \\
 Y = & 02 & 12 & 03 & 10 & 03 & 12 & & & & \\
 & 05 & 10 & 01 & 12 & & & & & & \\
 & 07 & 10 & & & & & & & &
 \end{array}$$

Desta forma, como primeiro passo da sua atividade, você deverá descomprimir as 3 matrizes de intensidade de cor de uma imagem, habilitando o seu uso para processamentos futuros. Isto é, dado Y , você deverá gerar X .

O segundo passo da sua atividade é transformar esta imagem colorida em escala de cinza. Neste processamento, as três matrizes devem ser combinadas para gerar uma única matriz de intensidade, que representa os tons cinzentos entre 0 (preto absoluto) e 255 (branco absoluto). A figura abaixo ilustra a intensidade dos *pixels* na pequena região em destaque.



Para obter os valores em escala de cinza, você deverá calcular a média da intensidade das componentes RGB de cada *pixel*. Para este processo, considere apenas o valor inteiro da média, desconsidere suas casas decimais. As matrizes abaixo ilustra o processo, onde:

$$Cinza_{x,y} = \frac{R_{x,y} + G_{x,y} + B_{x,y}}{3}$$

$$\begin{array}{ccc}
 R = & \begin{array}{ccc} 250 & 34 & 234 \\ 250 & 0 & 10 \\ 34 & 0 & 30 \end{array} & \begin{array}{ccc} G = & \begin{array}{ccc} 100 & 50 & 95 \\ 100 & 100 & 100 \\ 90 & 50 & 0 \end{array} & \begin{array}{ccc} B = & \begin{array}{ccc} 0 & 90 & 120 \\ 250 & 134 & 44 \\ 240 & 100 & 110 \end{array}
 \end{array}
 \end{array}$$

$$\begin{array}{ccc}
 & 116 & 58 & 149 \\
 Cinza = & 200 & 78 & 51 \\
 & 121 & 50 & 46
 \end{array}$$

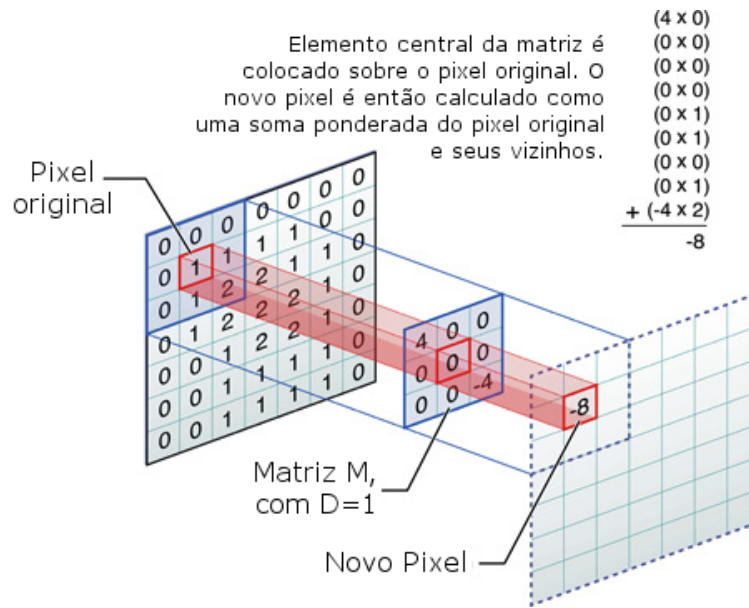
O terceiro passo desta atividade é a aplicação de filtros sobre a imagem/matriz em escala de cinza. A filtragem, aplicada a uma imagem digital, é uma operação local que modifica os valores de cada *pixel* da imagem considerando o contexto atual do *pixel*, isto é sua vizinhança. Existem diversos filtros para manipular imagens, tais como *Blur*, *Sharpen*, *Edge detection*, *Emboss*, etc. Muitos desses efeitos são realizados através de somente um algoritmo, a **convolução de matrizes**.

Uma operação de convolução consiste em, para cada *pixel* $p(x,y)$ de uma imagem, aplicar uma função matemática sobre a vizinhança de $p(x,y)$ e guardar o resultado em uma nova imagem p' . O efeito resultante dependerá dessa forma do tipo de função aplicada.

Dada uma matriz $M = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ e um divisor D , a função de convolução é aplicada para cada *pixel* $p'_{x,y}$, onde x e y são valores inteiros de um plano cartesiano, como segue:

$$p'_{x,y} = \frac{a \cdot p_{x-1,y-1} + b \cdot p_{x,y-1} + c \cdot p_{x+1,y-1} + d \cdot p_{x-1,y} + e \cdot p_{x,y} + f \cdot p_{x+1,y} + g \cdot p_{x-1,y+1} + h \cdot p_{x,y+1} + i \cdot p_{x+1,y+1}}{D}$$

A imagem abaixo ilustra como a operação é realizada para o *pixel* $p_{1,1}$ usando a matriz M com $a = 4$ e $i = -4$ (0 para os demais valores) e um divisor $D = 1$. Nesta imagem, a parte sombreada indica os vizinhos de $p_{1,1}$.



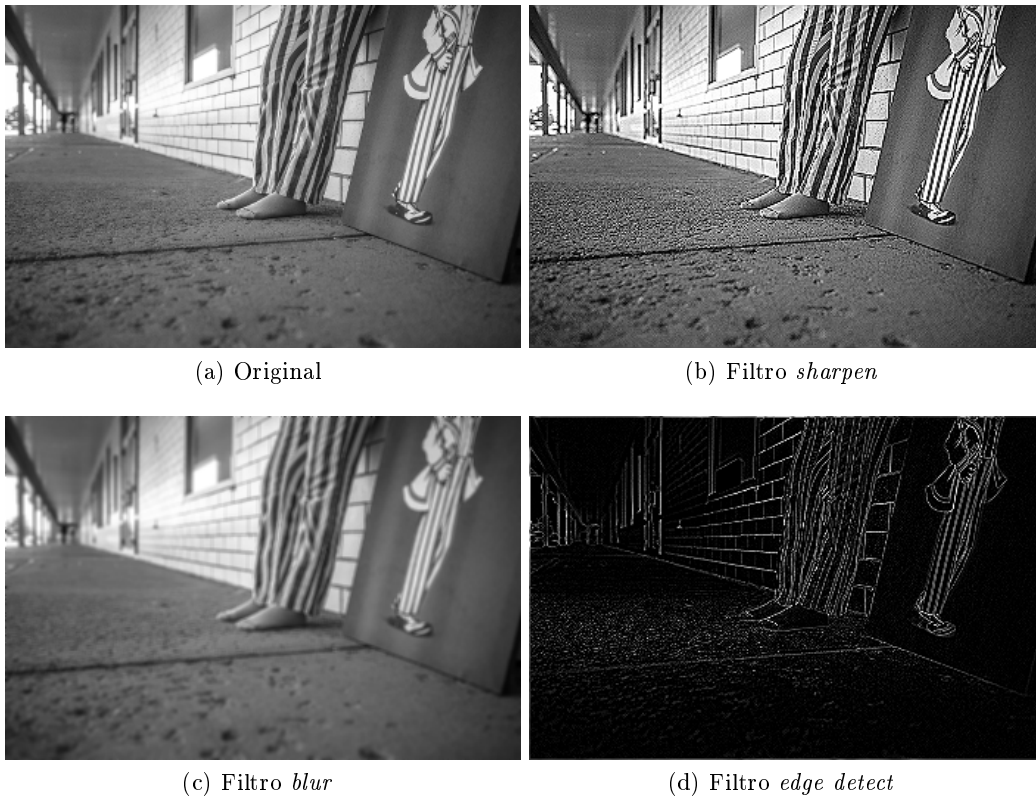
A matriz M é sempre quadrada e sua ordem tem valor ímpar (3×3 , 5×5 , $7 \times 7 \dots$). Neste caso, para aplicar a função de convolução, a vizinhança do *pixel* central deve ser expandida para que tenha a mesma ordem da matriz M . Para os *pixels* da borda (cuja vizinhança é menor que o tamanho da matriz de convolução M), os valores devem ser copiados para a matriz resultante sem qualquer operação. Note que na imagem acima o valor do *pixel* resultante é -8 , mas a imagem só armazena valores de $[0-255]$. Portanto, os valores menores que 0 e maiores que 255 devem ser truncados para 0 e 255, respectivamente. Use sempre valores inteiros pois, como no segundo passo da atividade, a divisão é truncada, desprezando as casas decimais.

Abaixo veja alguns resultados da aplicação de filtros.

Filtro *sharpen*: $M = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}, D = 1$

Filtro *blur*: $M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, D = 9$

Filtro *edge detect*: $M = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}, D = 1$



Entrada:

- Uma linha contendo um inteiro D , representando o divisor da terceira fase (filtragem).
- Uma linha contendo um inteiro X representando a ordem da matriz quadrada M , usada na terceira fase da atividade (filtragem). X é um número ímpar menor que 10.
- A matriz quadrada M com X^2 números inteiros, dispostas em X linhas e X colunas.
- Uma linha com dois inteiros positivos, menores ou igual a 600, representando a resolução da imagem em *pixels* (Largura e Altura).
- Uma sequência de pares de números inteiros (frequência, intensidade) representando uma imagem colorida comprimida com o algoritmo RLE. Esta entrada é composta apenas pelos números separados por espaço, sem vírgula ou parênteses. Esta sequência apresenta a compressão da matriz R , seguida da G e seguida da B . Note que, você sabe quantos *pixels* existem em cada matriz ($Largura \times Altura$), desta forma é possível descobrir/calcular onde a sequência da entrada encerra a codificação da matriz R e inicia a G , bem como encerra a G e inicia a B .

Saída:

A saída deve ser uma imagem em escala de cinza no formato PGM (*Portable Gray Map*). O formato permite que a saída, em forma de texto, da matriz de intensidade de cinza seja visualizada como imagem. Para isto você deverá imprimir:

- Uma linha com os caracteres **P2**.
- Uma linha com dois inteiros separados por espaço, informando o tamanho da imagem em *pixels*, na ordem largura e depois altura.
- Uma linha contendo o número **255** que representa o valor máximo de intensidade de um *pixel*.

- A matriz de intensidade com A linhas e L colunas, onde A é altura e L é a largura da imagem em *pixels*.

Exemplo:

Entrada:

```
1
3
0 1 0
1 -4 1
0 1 0
12 7
13 0 4 255 2 0 4 100 2 0 1 255 5 0 1 100 5 0 3 255 3 0 3 100 3 0 1 255 5 0 1 100 5 0 1 255 5 0 4 100 13
0 19 0 4 170 8 0 1 170 11 0 3 170 9 0 1 170 11 0 4 170 13 0 13 0 4 255 2 0 4 100 2 0 1 255 5 0 1 100 5
0 3 255 3 0 3 100 3 0 1 255 5 0 1 100 5 0 1 255 5 0 4 100 13 0
```

Saída:

P2

12 7

255

```
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 170 123 0 0 0 0 0
0 0 255 255 170 0 123 0 255 246 123 0
0 0 0 0 170 0 123 0 0 0 123 0
0 0 255 170 0 0 123 0 255 246 123 0
0 0 170 0 0 0 123 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
```

Resultados Parciais do Exemplo:

Etapa 1: Descompactar imagem colorida

R=

0	0	0	0	0	0	0	0	0	0	0	0	0
0	255	255	255	255	0	0	100	100	100	100	0	0
0	255	0	0	0	0	0	100	0	0	0	0	0
0	255	255	255	0	0	0	100	100	100	0	0	0
0	255	0	0	0	0	0	100	0	0	0	0	0
0	255	0	0	0	0	0	100	100	100	100	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

G=

0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	170	170	170	170	0	0
0	0	0	0	0	0	0	170	0	0	0	0	0
0	0	0	0	0	0	0	170	170	170	0	0	0
0	0	0	0	0	0	0	170	0	0	0	0	0
0	0	0	0	0	0	0	170	170	170	170	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 255 & 255 & 255 & 255 & 0 & 0 & 100 & 100 & 100 & 100 & 0 \\ 0 & 255 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 255 & 255 & 255 & 0 & 0 & 0 & 100 & 100 & 100 & 0 & 0 \\ 0 & 255 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 255 & 0 & 0 & 0 & 0 & 0 & 100 & 100 & 100 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Etapa 2: Montar imagem em escala de cinza

$$Cinza = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 170 & 170 & 170 & 170 & 0 & 0 & 123 & 123 & 123 & 123 & 0 \\ 0 & 170 & 0 & 0 & 0 & 0 & 0 & 123 & 0 & 0 & 0 & 0 \\ 0 & 170 & 170 & 170 & 0 & 0 & 0 & 123 & 123 & 123 & 0 & 0 \\ 0 & 170 & 0 & 0 & 0 & 0 & 0 & 123 & 0 & 0 & 0 & 0 \\ 0 & 170 & 0 & 0 & 0 & 0 & 0 & 123 & 123 & 123 & 123 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Etapa 3: Aplicação de filtro

$$Cinza' = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -340 & -340 & -340 & -510 & 170 & 123 & -246 & -246 & -246 & -369 & 0 \\ 0 & -340 & 510 & 340 & 170 & 0 & 123 & -246 & 369 & 246 & 123 & 0 \\ 0 & -170 & -340 & -510 & 170 & 0 & 123 & -123 & -246 & -369 & 123 & 0 \\ 0 & -340 & 340 & 170 & 0 & 0 & 123 & -246 & 369 & 246 & 123 & 0 \\ 0 & -510 & 170 & 0 & 0 & 0 & 123 & -246 & -246 & -246 & -369 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Após este passo, basta truncar os valores maiores que 255 e menores que 0.

Dicas e lembretes sobre o problema:

- Existem 6 casos de teste no Susy utilizando uma matriz de convolução com ordem 3.
- Todos os valores manipulados nas matrizes são inteiros, então use variáveis inteiras para não haver problema com precisão e/ou arredondamento.
- Perceba que os valores das matrizes variam sempre de [0-255], portanto você pode economizar espaço utilizando variáveis de tamanho menor que as variáveis do tipo *int*. No entanto, nem todas as variáveis devem ser limitadas a 255, o resultado da convolução pode gerar um valor maior que 255 ou menor que 0.
- Lembre que os valores resultantes da convolução de matrizes deve ser atribuído a uma nova matriz. Desta forma, os novos valores são apenas influenciados pela vizinhança da matriz original.
- Caso você queira visualizar a imagem resultante (no formato PGM), você deverá salvar a saída do programa em um arquivo com extensão .pgm e utilizar o programa IrfanView no Windows. Instale o programa e seus *plugins*. No Linux, os visualizadores de imagem tradicionais já oferece esta visão. Você poderá visualizar também a imagem em escala de cinza original, desde que atribua os valores da matriz original a um arquivo no formato PGM.
- Extra: Para manipular e visualizar imagens coloridas, os dados devem estar no arquivo de formato PPM. Este arquivo contém uma única matriz com 3 valores para cada *pixel*, correspondentes ao sistema RGB. Filtros também podem ser aplicados sobre a imagem colorida. No entanto, ele

deve ser aplicado em cada componente/malha separadamente. Mantendo o formato de arquivo PPM, você poderá visualizar os resultados da sua imagem após a aplicação de algum filtro.

- **Observações**

- *O programa deve ser submetido em C (labSemanal03.c)*
- *Faça comentários e indentação do seu código*
- *O aluno pode assumir que todas as linhas da entrada terminam com o fim-de-linha*
- *Todas as linhas da saída devem terminar com o fim-de-linha*
- *É obrigatório o uso de matrizes para manipular os dados*
- *É proibido o uso de manipulação de arquivo para ler as entradas. Isto quer dizer que você não pode usar uma variável do tipo FILE, ou funções como: fseek, fopen, fscanf, etc.*
- *Apenas por curiosidade, as imagens utilizadas na entrada estão no susy.*
- *O número máximo de submissões é 15*
- *O comando de compilação será:*
gcc -std=c99 -pedantic -Wall -lm labSemanal03.c -o labSemanal03
- *O comando de execução será:*
./labSemanal03 ou ./labSemanal03 <arq00.in >saida.pgm
- *Você poderá comparar os arquivos de saída com o comando diff*