UFV - Universidade Federal de Viçosa
DPI - Departamento de Informática
Prof. André Gustavo dos Santos

INF 110 - Programação I
Trabalho 3
Prazo: segunda, 11/jun/2018

Tratamento e manipulação de imagens

Este trabalho consiste em criar um programa para ler, alterar, e gravar imagens digitais. As imagens usadas serão codificadas no formato PNM, que é um formato bitmap, píxel a píxel, sem nenhum tipo de compressão. Imagens desse tipo podem ser lidas e gravadas por editores de imagens, como o GIMP[1]. Detalhes do formato PNM são dados na seção seguinte.

Este documento detalha e exemplifica o que deve ser desenvolvido no trabalho, e define o que e como o trabalho deve ser entregue. A última página contém exemplos de resultados esperados.

O trabalho pode ser feito individualmente ou em dupla. Os autores podem ser convidados a apresentar o trabalho ao professor para explicar detalhes da implementação e dos resultados.

1. Formato PNM

Imagens gravadas no formato PNM contêm os valores de cada píxel da imagem, linha por linha. Existem dois tipos de formato PNM, o formato ASC (ou ASCII) e o formato RAW (ou cru). Nesse trabalho usaremos apenas o formato ASC.

No arquivo de uma imagem no formato ASC, a primeira linha contém o tipo da imagem, sendo `P2` para uma imagem em tons de cinza e `P3` para uma imagem colorida. A segunda linha contém algum comentário gravado no momento de criação da imagem. Na verdade, pode haver mais de uma linha de comentário, e todas são iniciadas com '`#`'. Neste trabalho, elas podem ser descartadas. A terceira linha contém três valores, que indicam a dimensão da imagem, ou seja, a largura L e a altura A em píxels, e o valor máximo M de um píxel (assumiremos $M = 255$). Por fim, virão vários números inteiros, de valores no intervalo $[0, M]$, que indicam a cor de cada píxel. A quantidade de números depende da dimensão da imagem e dela ser colorida ou em tons de cinza.

Imagens em tons de cinza

Uma imagem em tons de cinza tem $L \times A$ píxels, cada um deles podendo variar entre 0 (preto) e 255 (branco). Veja o exemplo na Figura 1. A primeira linha indica que o arquivo armazena os dados de uma figura em tons de cinza, já que o tipo da imagem é `P2`. Os valores 5 e 2 na terceira linha definem a largura e altura da imagem (i.e., tamanho 5×2). O valor 0 indica píxel preto; 255 píxel branco; 100 cinza escuro; e 180 cinza claro. O exemplo contém 5 valores por linha para facilitar o entendimento do formato PNM, mas na prática, os valores podem ser escritos todos em uma mesma linha, apenas um valor por linha, ou de alguma outra maneira, mas sempre estarão na ordem dos píxels da imagem (da esquerda pra direita e de cima pra baixo).

```
P2
# CREATOR: GIMP PNM Filter Version 1.1
5 2 255
0 100 100 255 0
0 180 255 255 0
```

Figura 1: Exemplo de imagem em tons de cinza: arquivo PNM e a imagem representada (ampliada muitas vezes, para facilitar o entendimento; cada quadrado representa um píxel)

[1] O GIMP é livre, gratuito e multiplataforma. Pode ser baixado em `http://www.gimp.org`

Imagens coloridas

A Figura 2 mostra uma imagem colorida (tipo `P3`) de dimensões 4×2. Note que, embora a imagem tenha $4 \times 2 = 8$ píxels, o arquivo PNM tem $4 \times 2 \times 3 = 24$ números. Isso acontece pois, em imagens coloridas, cada píxel é definido por três valores entre 0 e 255. Para um determinado píxel, o primeiro dos três valores representa a intensidade de vermelho, o segundo a de verde, e o terceiro a de azul - esse padrão é conhecido como padrão RGB (*red*, *green*, *blue*).

Conforme pode ser visto no exemplo, os valores `255 0 0` indicam $R = 255$, $G = 0$, $B = 0$, ou seja, vermelho puro; os valores `255 255 255` indicam branco (intensidade total das três cores); os valores `0 0 0` indicam preto (nada das três cores); os valores `255 255 0` indicam amarelo (combinação de vermelho e verde); e assim por diante. São 256^3 combinações possíveis, mais de 16 milhões de cores.

```
P3
# CREATOR: GIMP PNM Filter Version 1.1
4 2 255
255 0 0 255 255 255 255 255 0 0 0 255
0 0 0 255 153 0 0 255 0 100 100 100
```

Figura 2: Exemplo de imagem colorida: arquivo PNM e a imagem representada

2. Estrutura de dados

Para armazenar uma imagem deve-se utilizar uma matriz com capacidade de armazenar as informações de seus píxels. Assim, para imagens em tons de cinza, deve-se ter uma matriz de tamanho pelo menos $L \times A$. Para imagens coloridas, pode-se utilizar uma matriz de três dimensões, $L \times A \times 3$, ou três matrizes de tamanho $L \times A$, uma para cada banda de cor.

Recomenda-se utilizar o tipo `unsigned char`, pois ele armazena valores de 0 a 255, gastando, portanto, apenas 1 byte de memória. Se fosse utilizado o tipo `int`, também seria possível guardar os valores de 0 a 255, porém gastando 2 ou 4 bytes de memória (dependendo do computador) para cada entrada.

Observe que, ao efetuar contas com os valores armazenados em variáveis do tipo `unsigned char`, qualquer valor acima de 255 ou abaixo de 0 será convertido em algo dentro do intervalo $[0, 255]$. Por exemplo, $255 + 1 = 256$, será armazenado como 0. Da mesma forma, $0 - 1 = 255$ e $255 + 10 = 9$. Por isso, ao efetuar uma operação envolvendo variáveis do tipo `unsigned char`, utilize uma variável do tipo `int`. Ao terminar a operação, antes de converter de volta para `unsigned char`, considere qualquer valor acima de 255 como 255, e qualquer valor abaixo de 0 como 0.

3. Operações simples com imagens

A seguir são descritas algumas operações que deverão ser implementadas neste trabalho.

Escurecer

Na operação de escurecer uma imagem, deve-se diminuir os valores de cada pixel da imagem por um fator k. Por exemplo, para $k = 50$, deve-se iterar por todos os valores de píxel e subtrair o valor de 50. Lembre-se que o valor zero deve ser atribuído a píxels que atingirem valores negativos. No caso de imagens coloridas, o fator k deve ser subtraído de todas três bandas de cores. A Figura 3 mostra um exemplo desta operação.

Clarear

A operação de clarear é análoga à operação de escurecer, com a diferença de que o fator k deve ser somado aos valores de cada píxel da imagem. A Figura 4 mostra um exemplo desta operação.

```
P2
# CREATOR: GIMP PNM Filter Version 1.1
5 2 255
0 100 100 255 0
0 180 255 255 0
```

```
P2
# CREATOR: GIMP PNM Filter Version 1.1
5 2 255
0 50 50 205 0
0 130 205 205 0
```

Figura 3: Exemplo de escurecimento de imagem por fator $k = 50$

```
P3
# CREATOR: GIMP PNM Filter Version 1.1
4 2 255
255 0 0 255 255 255 255 255 0 0 0 255
0 0 0 255 153 0 0 255 0 100 100 100
```

```
P3
# CREATOR: GIMP PNM Filter Version 1.1
4 2 255
255 50 50 255 255 255 255 255 50 50 50 255
50 50 50 255 203 50 50 255 50 150 150 150
```

Figura 4: Exemplo de clareamento de imagem com fator $k = 50$

Negativo

Na operação de negativo, os píxels claros ficam escuros e os escuros ficam claros. Isto é, o valor de preto (0) vira branco (255) e o branco (255) vire preto (0). Os quase pretos (1) viram quase brancos (254), e os quase brancos (254) viram quase pretos (1), e assim por diante. Em imagens coloridas esta operação gera imagens como nos filmes fotográficos (exemplo na Figura 5).

```
P3
# CREATOR: GIMP PNM Filter Version 1.1
4 2 255
255 0 0 255 255 255 255 255 0 0 0 255
0 0 0 255 153 0 0 255 0 100 100 100
```

```
P3
# CREATOR: GIMP PNM Filter Version 1.1
4 2 255
0 255 255 0 0 0 0 0 255 255 255 0
255 255 255 0 102 255 255 0 255 155 155 155
```

Figura 5: Exemplo de negativo de imagem

Espelhar

Para espelhar uma imagem deve-se trocar a primeira coluna da imagem com a última, a segunda com a penúltima, e assim por diante. Em imagens coloridas deve-se fazer esse mesmo processo com cada banda de cor.

4. Filtros

A aplicação de diversos filtros em imagens pode ser feita por operação de convolução de uma matriz que define o filtro sobre a matriz que define a imagem. Por exemplo, efeitos de realce e desfocagem de imagens, detecção de contornos, entre outros, podem ser implementados da mesma forma, por convolução de matrizes, modificando-se apenas a matriz que define o filtro. Este método é explicado abaixo, e em seguida são dados detalhes do filtro de Sobel, que é parte deste trabalho.

A matriz F de tamanho 3×3 abaixo, onde f_{ij} são valores inteiros, define um filtro genérico. Tais valores representam o peso dos píxels vizinhos na aplicação do filtro. Seja M a matriz com os valores dos píxels da imagem. A aplicação do filtro F à imagem M consiste em substituir cada valor m_{ij} da matriz M pelo valor da expressão a seguir:

$$F = \begin{bmatrix} f_{1,1} & f_{1,2} & f_{1,3} \\ f_{2,1} & f_{2,2} & f_{2,3} \\ f_{3,1} & f_{3,2} & f_{3,3} \end{bmatrix}$$

$$\begin{aligned} m_{i,j} = {} & f_{1,1} \cdot m_{i-1,j-1} + f_{1,2} \cdot m_{i-1,j} + f_{1,3} \cdot m_{i-1,j+1} \\ & + f_{2,1} \cdot m_{i,j-1} + f_{2,2} \cdot m_{i,j} + f_{2,3} \cdot m_{i,j+1} \\ & + f_{3,1} \cdot m_{i+1,j-1} + f_{3,2} \cdot m_{i+1,j} + f_{3,3} \cdot m_{i+1,j+1} \end{aligned}$$

Como exemplo, considere o seguinte trecho da matriz de píxels de uma imagem em tons de cinza e o filtro de realce (*sharpening*) F_r definido pela matriz de filtro abaixo.

$$M = \begin{bmatrix} \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & 10 & 15 & 80 & 85 & 95 & \dots \\ \dots & 15 & \boxed{30} & \boxed{110} & 130 & 131 & \dots \\ \dots & 16 & 41 & 100 & 120 & 120 & \dots \\ \dots & 17 & 43 & 80 & 110 & 130 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \qquad F_r = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Ao se aplicar o filtro F_r à imagem M, o valor 110 destacado em M será substituído por:
$(-1) \cdot 80 + (-1) \cdot 30 + 5 \cdot 110 + (-1) \cdot 130 + (-1) \cdot 100 = 210$.

Já o outro valor destacado, 30, seria substituído por:
$(-1) \cdot 15 + (-1) \cdot 15 + 5 \cdot 30 + (-1) \cdot 110 + (-1) \cdot 41 = -31$. No entanto, como $-31 < 0$, e imagens PNM não aceitam valores negativos, o valor 30 deverá ser substituído por 0 ao se aplicar F_r em M.

Note que a aplicação do filtro em um píxel é independente de sua aplicação nos demais píxels. Os píxels destacados, por exemplo, são vizinhos e um influencia no resultado do outro. No entanto, na aplicação do filtro no píxel 30, o cálculo usa o valor 110 do píxel vizinho e, na aplicação do filtro no píxel 110, o cálculo usa o valor 30. Assim, independentemente da ordem em que eles são processados pelo filtro, o cálculo usa os valores originais da matriz de píxels, e não os valores 0 e 210 resultantes de sua aplicação. Então, nesta operação (e possivelmente em outras), será necessário fazer uma cópia da matriz para uma matriz auxiliar de mesmo tamanho, antes de sua aplicação.

Outros exemplos de filtros de convolução são os de desfocagem gaussiana e o de Laplace para detecção de bordas, definidos respectivamente pelas seguintes matrizes. Não são necessários neste trabalho, mas você pode experimentá-los por curiosidade

$$F_{dg} = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad F_L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Filtro de Sobel

O filtro Sobel pode ser usado para detecção de contornos (bordas) em imagens digitais, e consiste na aplicação de duas matrizes, que detectam os contornos na vertical e na horizontal:

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \text{ e } G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}.$$

Aplicando-se o filtro definido pela matriz G_y a todos os píxels de uma matriz M, a imagem resultante M_y dará ênfase aos contornos dos objetos da imagem (onde há mudança brusca de cor). Da mesma forma, a aplicação da matriz G_x à matriz M produzirá uma imagem M_x com detecção de contornos, porém em outra direção. Deve-se então criar uma terceira matriz M_s (imagem resultante da aplicação do filtro de Sobel) que leva em consideração M_y e M_x. M_s pode ser gerada por média (geométrica ou aritmética) dos valores de M_y e M_x, pela seleção do maior valor de cada píxel de M_y e M_x, dentre outras possibilidades. Experimente formas diferentes de criar M_s e verifique os resultados. Veja exemplos na Figura 8 na última página.

5. Tarefas do trabalho

Desenvolver um programa que realiza todas as seguintes funcionalidades:

- **Ler**: perguntar o nome do arquivo PNM e carregar as informações dos píxels da imagem;
- **Clarear/Escurecer**: perguntar o valor de um fator k de modificação e aplicar esse fator na imagem carregada;
- **Negativo**: gerar o negativo da imagem carregada;
- **Espelhar**: espelhar a imagem carregada em seu eixo vertical;
- **Tons de cinza**: transformar uma imagem colorida em uma imagem em tons de cinza;
- **Filtro de Sobel**: aplicar o filtro de Sobel à imagem carregada;
- **Outra opção de filtro**: aplicar algum dos outros filtros explicados acima (ou outro similar);
- **Gravar**: perguntar o nome do arquivo e gravar a imagem modificada no formato PNM;
- **Outra operação**: efetuar alguma outra operação não descrita nesse enunciado (as melhores serão recompensadas com ponto extra).

Encontra-se disponível no PVANet um código que realiza leitura, aplicação de uma das operações acima e gravação de uma imagem PNM no formato tons de cinza. Ele pode ser usado como base para o desenvolvimento do trabalho.

O que entregar

Deverá ser entregue um documento contendo instruções de utilização do programa, informações sobre as operações implementadas e exemplos de utilização de cada uma dessas funcionalidades, utilizando imagens diferentes das disponibilizadas com esse documento.

Como entregar

O trabalho pode ser feito individualmente ou em dupla e deve ser entregue por e-mail, para `andreufv@gmail.com` com o assunto "`INF110 - TP3 xxxxx yyyyy`", sendo `xxxxx` e `yyyyy` substituídos pelo número de matrícula dos autores do trabalho. Enviar tudo (código em C++, devidamente indentado e comentado, o relatório, e imagens utilizadas (compactadas!)) em um mesmo e-mail. **Não envie o programa executável!**.

6. Exemplos de resultados esperados

Figura 6: Imagens originais, colorida e em tons de cinza

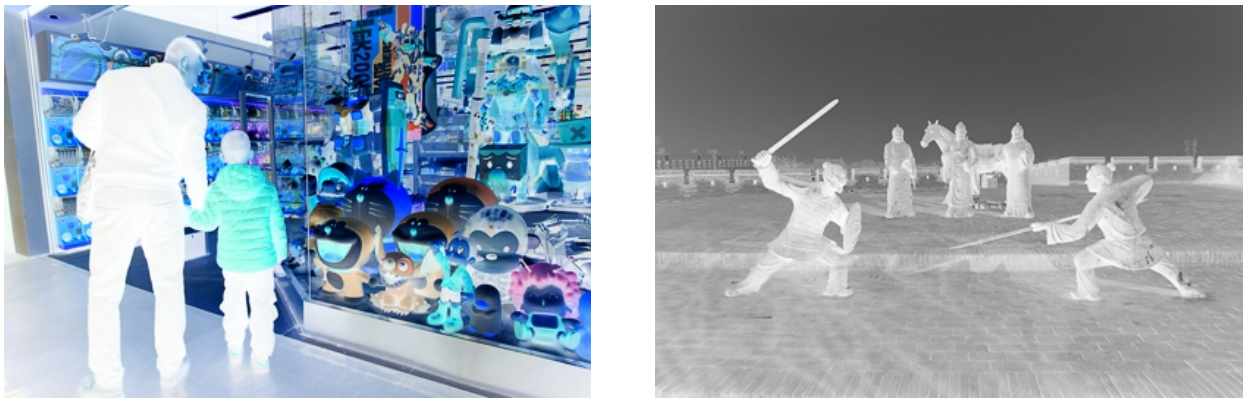

Figura 7: Negativos das imagens

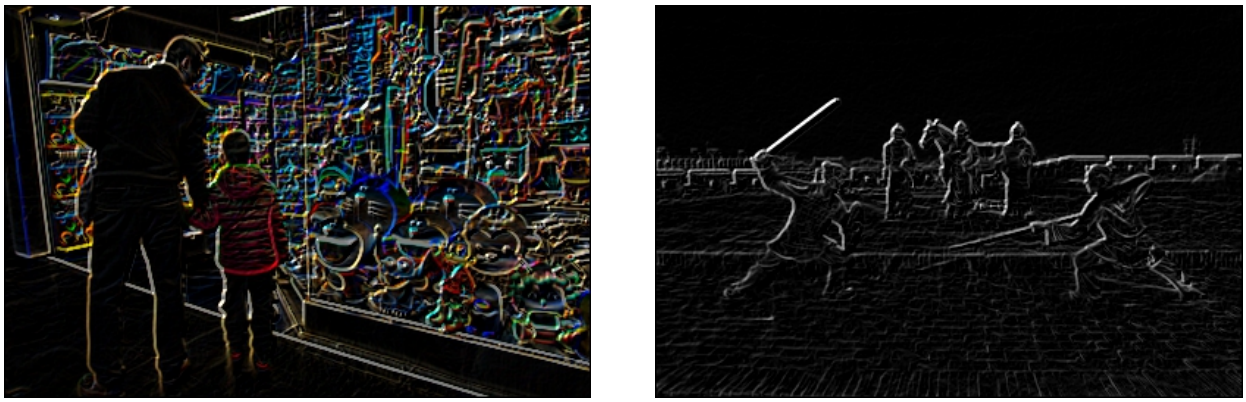

Figura 8: Filtro de Sobel com média aritmética