Tratamento de Imagens

Trabalho Prático 3 — Programação I

Departamento de Informática
Centro de Ciências Exatas
Universidade Federal de Viçosa

Entrega dia 19/06

Introdução

Este trabalho consiste em criar um programa para ler, alterar, e gravar imagens digitais. As imagens usadas serão codificadas no formato PNM (descrito em abaixo), que é um formato bitmap, pixel a pixel, sem nenhum tipo de compressão. Imagens desse tipo podem ser lidas e gravadas por editores de imagens como o GIMP.[1]

Esse trabalho será desenvolvido em dupla. Cada dupla irá entregar uma cópia do código desenvolvido com instruções de como executa-lo e um relatório descrevendo os resultados obtidos. **O relatório e código deverão ser entregues até o dia 19/06**.

Formato PNM

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

A primeira linha contém o tipo da imagem, onde $P2$ indica uma imagem em tons de cinza e $P3$ indica uma imagem colorida. A segunda linha do arquivo, iniciada com o caracter '#', contém algum comentário gravado no momento de criação da imagem. Nesse trabalho não utilizaremos comentários. A terceira linha contém três valores, que indicam a dimensão da imagem, ou seja, a largura L e a altura A em pixels, e o valor máximo do pixel M (iremos assumir $M = 255$). Em seguida virão vários números, todos de 0 a M, que indicam a cor de cada pixel. A quantidade depende do tamanho da imagem, e dela ser colorida ou em tons de cinza. Se a imagem for em tons de cinza, haverá $L \times A$ pixels, cada um deles representado por um valor entre 0 (preto) e 255 (branco).

Imagens em Tons de Cinza

Veja o exemplo de arquivo PNM a seguir.

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

[1] Você pode baixar o GIMP em https://www.gimp.org

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., a imagem possui tamanho 5×2). O valor 0 indica pixel preto; 255 branco; 100 cinza escuro; e 180 cinza claro. Esse exemplo contém apenas 5 valores por linha para facilitar o entendimento da formatação PNM, pois 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 pixels da imagem (da esquerda para a direita e de cima para baixo). A Figura 2 mostra a imagem do arquivo PNM mostrado acima.

Figura 1: Figura em tons de cinza com dimensões 5×2.

Imagens Coloridas

O exemplo abaixo mostra uma imagem colorida no formato PNM (tipo $P3$) de dimensões 4×2.

```
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
```

Repare que embora a imagem tenha $4 \times 2 = 8$ pixels, o arquivo contém $8 \times 3 = 24$ números. Isso acontece pois, em imagens coloridas, cada pixel é definido por três valores entre 0 e 255. Para um determinado pixel, 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 o padrão *red, green, blue* (RGB).

Por exemplo, no arquivo acima, os valores 255 0 0 indicam $R = 255$, $G = 0$, $B = 0$, ou seja, vermelho puro; os valores 0 0 0 indicam preto (nada das três cores); os valores 255 255 255 indicam branco (intensidade total das três cores) e os valores 255 255 0 indicam amarelo (combinação de vermelho e verde). A Figura 2 mostra a imagem resultante.

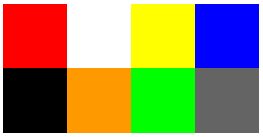

Figura 2: Figura colorida com dimensões 4×2.

Estruturas de Dados

Para armazenar as informações dos pixels de uma imagem deve-se utilizar uma matriz com capacidade de armazenar as informações da imagem. Isto é, para imagens em tons de cinza, a matriz deve ter um tamanho de pelo menos $L \times A$. Para imagens coloridas, pode-se utilizar uma matriz de três dimensões, $L \times A \times 3$, ou então três matrizes de tamanho $L \times A$, uma para cada banda de cor.

Recomenda-se utilizar o tipo *unsigned char* da linguagem C++, 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 $255 + 10 = 9$. 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.

Operações com Imagens

Esta seção contém a descrição de algumas das operações que serão implementadas nesse 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 sob todos os valores de pixel e subtrair o valor de 50. Lembre-se que o valor zero deve ser atribuído a pixels que atingirem valores negativos.

No caso de imagens coloridas o valor de k deve ser subtraído de todas três bandas de cores.

Clarear

A operação de clarear é análoga à operação de escurecer, com a diferença de que o valor de k deve ser somado aos valores de cada pixel da imagem.

Negativo

Na operação de negativo os pixels 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).

Em imagens coloridas deve-se fazer esse mesmo processo com cada banda de cor.

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.

Copiar

Para implementar os filtros explicados abaixo necessita-se de uma cópia da matriz original da imagem. Isso acontece pois ao modificar um pixel da imagem, precisam-se dos valores originais da matriz, e não dos valores já modificados pelo filtro. Portanto, uma operação importante é a de cópia da matriz de pixels de uma imagem.

Filtros

Esta seção descreve como aplicar filtros em imagens. Para um exemplo do resultado da aplicação de um filtro—no exemplo o filtro de Sobel (explicado abaixo)—veja as imagens mostradas nas Figuras 3e e 3f.

A matrix F de tamanho 3×3 abaixo, onde os valores f_{ij} são valores inteiros, define um filtro.

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

Seja M uma matriz com os valores dos pixels de uma determinada imagem. Ao aplicarmos um filtro F em M, devemos substituir cada entrada $m_{i,j}$ de M pelo seguinte valor.

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

Por exemplo, considere o seguinte trecho de matriz definindo parte de uma imagem em tons de cinza.

$$M = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 10 & 15 & 80 & 85 & 95 & \cdots \\ \cdots & 15 & \boxed{40} & 120 & 130 & 131 & \cdots \\ \cdots & 16 & 41 & \boxed{100} & 120 & 120 & \cdots \\ \cdots & 17 & 43 & 80 & 110 & 130 & \cdots \\ \cdots & 19 & 50 & 70 & 42 & 20 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

Considere também o seguinte filtro.

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

Ao aplicar o filtro definido por G_y à imagem M, o valor de 40 destacado em M será substituído por,

$$10\cdot 1+15\cdot 2+80\cdot 1+(-1)\cdot 16+(-2)\cdot 41+(-1\cdot 100) = -78$$

No entanto, como $-78 < 0$ e a matriz que define a imagem em PNM aceita apenas valores não-negativos, substituímos o valor de 40 por 0 ao aplicar G_y em M.

O valor de 100 (destacado em M) será substituído pelo seguinte valor após a aplicação de G_y em M.

$$40\cdot 1+120\cdot 2+130\cdot 1+(-1)\cdot 43+(-2)\cdot 80+(-1)\cdot 110 = 129$$

Filtro de Sobel

Considere uma imagem definida por M. O filtro de Sobel consiste em aplicar o filtro G_y, como explicado na seção anterior, gerando uma imagem modificada M'. Em seguida, aplica-se a M o filtro definido pela matriz G_x mostrada abaixo, gerando M''.

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

Deve-se então criar uma terceira matriz M_s (imagem resultante da aplicação do filtro de Sobel) que leva em consideração M' e M''. M_s pode ser gerada através da média (geométrica ou aritmética) dos valores de M' e M'', ou através da seleção do maior valor de cada posição de M' e M'', dentre outras possibilidades—experimente formas diferentes de criar M_s e verifique os resultados.

Outros Filtros

Essa seção descreve outros filtros que podem ser aplicados seguindo o processo descrito anteriormente. Repare que uma vez que o processo de aplicação de filtro é implementado, precisa-se apenas definir uma nova matriz para implementar um novo filtro. Essa seção limita-se a descrever filtros definidos por matrizes 3×3, mas outros filtros podem requerer matrizes de dimensões maiores.

O filtro abaixo é utilizado para realçar uma imagem.

$$G_r = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}.$$

O seguinte filtro desfoca a imagem (desfocagem gaussiana).

$$G_{dg} = \frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

O próximo filtro, conhecido com o filtro de Laplace, detecta as bordas de uma image.

$$G_L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}.$$

Operações e Filtros a Implementar

O programa a ser implementado nesse trabalho deverá incluir obrigatoriamente as seguintes funcionalidades.

- **Leitura de Imagem**: o programa deverá perguntar o nome do arquivo PNM e carregar as informações dos pixels na memória do computador.
- **Clarear/Escurecer Imagem**: o programa receberá como entrada um fator k de modificação e aplicar esse fator na imagem.
- **Espelhar a Imagem**: a imagem carregada na memória deverá ser espelhada em seu eixo vertical.
- **Negativo**: será criado o negativo da imagem carregada na memória.
- **Tons de Cinza**: transformar a imagem colorida carregada na memória em uma imagem em tons de cinza.
- **Filtros**: implementar a aplicação de filtros definidos por matrizes 3×3, como descrito acima. O programa deverá implementar pelo menos dois filtros.
- **Gravar**: o programa deverá perguntar o nome do arquivo e gravar a imagem no formato PNM.
- **Outra Operação**: o programa deverá efetuar alguma outra operação não descrita nesse documento (as melhores ideias serão recompensadas com um ponto extra).

O que Entregar

Deverá ser entregue um arquivo com instruções de utilização do programa informando como executar as funcionalidades implementadas. O relatório deve conter também exemplos de utilização de cada uma das funcionalidades implementadas utilizando fotos diferentes das disponibilizadas com esse documento.

Como Entregar

O trabalho deve ser feito em dupla e entregue por e-mail. Favor enviar um único email para levilelis@gmail.com com o assunto "INF110 - T3 xxxxx yyyyy" (onde xxxxx e yyyyy deverão ser substituídos pelo número de matrícula dos autores do trabalho) contendo o código em C++, devidamente identado e comentado, e o relatório. **Não envie o programa executável.** Data limite para entrega: 19 junho. Trabalhos entregues atrasados podem sofrer desconto na nota, a critério do professor.

Resultados Esperados

A Figura 3 mostra alguns resultados esperados para as operações de espelhar, negativo e Sobel.

(a) Figura original em tons de cinza.

(b) Versão espelhada da figura original.

(c) Figura original colorida.

(d) Negativo da figura original colorida.

(e) Figura original colorida.

(f) Figura original colorida com efeito de Sobel.

Figura 3: Resultados esperados para as operações de espelhar, negativo e Sobel.