Instituto de Computação da UNICAMP

Disciplina MC102: Algoritmos e Programação de Computadores - Turmas EF

Primeiro Semestre de 2015

Laboratório Nº 17. Peso 2.

Prazo de entrega: **19/06/2015 às 23:59:59**

PROFESSOR: Alexandre Xavier Falcão
MONITORES: João do Monte Gomes Duarte
MONITORES: Jadisha Yarif Ramírez Cornejo
MONITORES: Takeo Akabane
MONITORES: Eduardo Spagnol Rossi
MONITORES: Guilherme Augusto Sakai Yoshike

Convolução de Imagens

Esta tarefa tem por objetivo exercitar o processamento de arquivos.

Em programas de edição e processamento de imagens como Gimp e Adobe Photoshop, existem diversos filtros utilizados para se manipular imagens, tais como Blur, Sharpen, Edge detection, Emboss, etc. Muitos desses efeitos são realizados através de somente um algorítmo, a **convolução** de matrizes.

Ao ver uma imagem em níveis de cinza (tons que variam do preto ao branco) de perto, notamos que a mesma é formada por uma coleção de pontos. Cada um desses pontos é denominado de pixel (*picture element*). Além disso, esses pixels estão arranjados sobre uma malha retangular, ou matriz. Nessa matriz, cada pixel ocupa uma posição, (x, y), e tem um certo valor. Esse valor descreve a intensidade do ponto, que varia do preto ao branco e, neste trabalho, supomos que essas intensidades estão na faixa [0, 255]. As posições do pontos, ou coordenadas, seguem a ordem *raster* (de cima para baixo, da esquerda para direita). A Figura 1 apresenta um exemplo que caracteriza essa forma de representação de imagens. Um pixel na linha x e coluna y é representado por $p(x, y)$.

(a) Imagem, em níveis de cinza, com uma pequena região destacada.

87	65	86	121	174	202	204
90	65	50	72	163	204	204
112	77	56	66	157	204	204
108	74	53	87	177	207	204
72	57	66	126	197	209	206
55	65	113	173	207	211	210
77	107	160	198	208	206	208

(b) Zoom na região em destaque. Cada valor representa a intensidade de um pixel.

Figura 1: Visualização de uma pequena região da imagem de Lena.

Suponha que a sua imagem de entrada esteja armazenada em uma matriz $n \times m$, tal que um pixel na posição (x, y) tem valor $p(x, y)$. Uma operação de convolução consiste em para cada pixel $p(x, y)$ de uma imagem, aplicar uma função matemática Λ 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.

Para especificar a função a ser aplicada sobre cada pixel, é definida uma matriz "núcleo" M e um divisor inteiro D. Dessa forma, dado a matriz $M = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$ e o divisor D, a função a ser aplicada para realizar uma convolução $\Lambda(p, x, y, M, D)$ terá como resultado

$$p'(x, y) = \Lambda(p, x, y, M, D) = \frac{a \cdot p(x-1, y-1) + b \cdot p(x, y-1) + c \cdot p(x+1, y-1) + d \cdot p(x-1, y) + e \cdot p(x, y) + f \cdot p(x+1, y) + g \cdot p(}{D}$$

O método de convolução consiste em aplicar essa função descrita acima para cada pixel da imagem original sucessivamente, formando uma nova imagem. Para simplificar a operação, **não** são considerados os pixels de borda (pixels com $x = 0$, $x = n - 1$, $y = 0$ ou $y = m - 1$), que deverão ser simplesmente copiados para a nova imagem. Como exemplo, dado uma matriz núcleo M, onde $a = 1$ e demais valores são 0, realizamos a operação para a imagem 5×5 abaixo representada como uma matriz.

$$\text{Convolução}\left(p = \begin{bmatrix} 87 & 65 & 86 & 121 & 174 \\ 90 & 65 & 50 & 72 & 163 \\ 112 & 77 & 56 & 66 & 157 \\ 108 & 74 & 53 & 87 & 177 \\ 72 & 57 & 66 & 126 & 197 \end{bmatrix}, M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, D = 1 \right) = \begin{bmatrix} 87 & 65 & 86 & 121 & 174 \\ 90 & 87 & 65 & 86 & 163 \\ 112 & 90 & 65 & 50 & 157 \\ 108 & 112 & 77 & 56 & 177 \\ 72 & 57 & 66 & 126 & 197 \end{bmatrix}.$$

Ainda, abaixo é dado um outro exemplo ilustrando como a operação é realizada, para um M com $a = 4$ e $i = -4$ (sendo o resto zerado) e com divisor $D = 1$.

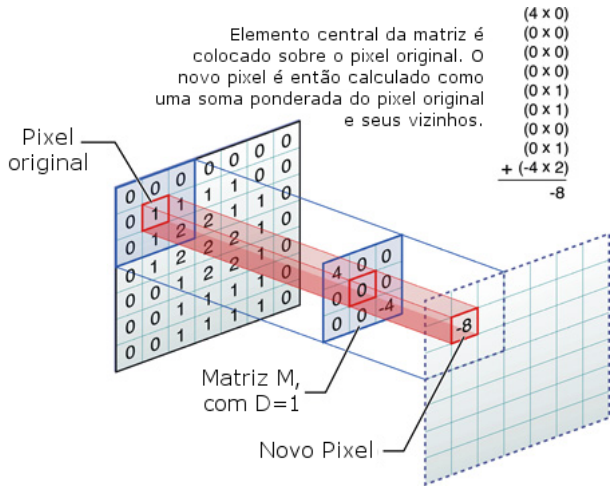

Figura 2: Imagem ilustrando o conceito de convolução, adaptado de [ios].

Para mais detalhes sobre convolução, podem ser acessados estes links: [gimp], [wikipedia], [ios], [demo].

Para este laboratório, o tamanho da matriz M será sempre 3×3. Para realizar a operação, considere a divisão por D como uma **divisão inteira**. Como o resultado da convolução a ser executada deve ser guardado como uma imagem, caso o resultado da operação em um pixel seja menor que 0 ou maior que 255, esses valores devem ser truncados para 0 ou 255, respectivamente, uma vez que o pixel da imagem resultante não pode ter valores fora do intervalo $[0, 255]$.

Como exemplos de como essa operação pode resultar em imagens diversas, veja os exemplos abaixo:

(a) Imagem original.

(b) Imagem com filtro "sharpen":

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

(a) Imagem com filtro "blur":

$$M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, D = 9$$

(b) Imagem com filtro "edge detect":

$$M = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}, D = 1$$

Figura 3: Visualização de diversas convoluções possíveis para uma imagem.

Nesse laboratório seu programa deve receber como parâmetro do programa o nome de um arquivo texto contendo uma imagem no formato PGM (descrito abaixo) e como segundo parâmetro o nome de um arquivo texto contendo a matriz M e divisor D. Seu programa deverá imprimir na saída padrão o resultado da convolução da imagem dada, utilizando-se M e D.

Os arquivos de entrada serão dados como parâmetros para o programa. Dessa forma, você deve incluir os argumentos (`int argc, char *argv[]`) na chamada da sua função main. O primeiro arquivo estará em `argv[1]` e o segundo em `argv[2]`, e ambos devem ser lidos em modo texto.

Entrada

Você receberá como parâmetro do programa através do vetor de strings `argv` dois nomes de arquivos. Em `argv[1]` estará o nome do arquivo contendo a imagem original, e em `argv[2]` o nome do arquivo texto com o divisor D e a matriz de convolução M.

Para a imagem, as primeiras três linhas representam o cabeçalho de uma imagem em formato PGM. A primeira linha da entrada contém apenas a string `P2` que indica o formato do arquivo e deve ser desconsiderada. A segunda linha contém dois números inteiros indicando o número de colunas e linhas da imagem. A terceira linha indica o valor máximo de cinza que a imagem poderia conter (no caso da entrada sempre será 255). Os números nas linhas seguintes representam os valores de pixel da imagem em escala de cinzas, na ordem raster.

Para o arquivo em `argv[2]`, são dados 10 números inteiros em sequência. O primeiro valor será o divisor D, e os próximos 9 valores correspondem a matriz M, seguindo a ordem $a\ b\ \ldots\ i$.

Exemplo do arquivo em `argv[1]`:

```
P2
9 10
255
    0    0    0    0    8   49   73   98  125
    0    0    0    0   16   66   90  126  132
    0    0    0    0   33   82  106  123  140
    0    0    0    0   56   90  107  132  142
    0    0    0    8   66   99  123  140  157
    0    0    0   24   74  107  126  142  164
    0    0    0   56   90  109  132  156  174
    0    0    8   74   99  123  142  165  181
    0    0   56   90  107  140  147  106   57
    0   33   73   89   88   47   15    0    0
```

Exemplo do arquivo em `argv[2]`:

```
1
0 -1 0
-1 4 -1
0 -1 0
```

Saída

A saída será impressa na saída padrão e terá o formato de um arquivo PGM contendo a imagem resultante da convolução. As primeiras três linhas devem conter os dados de cabeçalho (iguais a imagem original). A primeira linha da saída deverá conter apenas a string `P2` que indica o formato do arquivo. A segunda linha conterá dois números inteiros indicando o número de colunas e linhas da imagem. A terceira linha indicará o valor máximo que um pixel da imagem conterá (que sempre será 255). As linhas seguintes deverão conter os valores da imagem resultante da convolução, separados por exatamente um caractere em branco entre cada valor. Cada linha deste arquivo deverá terminar com uma quebra de linha.

Exemplo de saída:

```
P2
9 10
255
   0   0   0   0   8  49  73  98 125
   0   0   0  16  43   0  11   0 132
   0   0   0  33  22   0   0  12 140
   0   0   0  64   0   0  23   0 142
   0   0   8  58   0   0   0   0 157
   0   0  24  42   0   0   0  18 164
   0   0  64   0   0  16   5   0 174
   0   8  98   0   0   0   0   0 181
   0  89   0   0   0   0   0   0  57
   0  33  73  89  88  47  15   0   0
```

Observações e Dicas:

- A imagem de entrada conterá no máximo 250000 pixels, sendo que tanto o número de linhas quanto o de colunas nunca será maior que 600.
- Não se esqueça de utilizar argc e argv para a sua função main, no seguinte formato: `int main(int argc, char *argv[])`.
- Não é permitido modificar os arquivos a serem lidos na entrada. Dessa forma, você deve abrir os arquivos somente para leitura.
- Nos arquivos auxiliares podem ser encontrados as versões em formato PNG das entradas e saídas dos testes abertos. Por exemplo, para a entrada 10 (denominada arq10), o link aux/arq10.png tem a imagem original convertida para PNG e o link aux/arq10.res.png contém o resultado

esperado do seu programa, também convertido para PNG.
- É possível converter um arquivo PGM para um outro formato de imagem utilizando por exemplo um conversor online como esse, para melhor verificar seus resultados.
- Para executar o programa dado uma imagem `imagem.pgm` e um arquivo de filtro `matriz.txt` e guardar o resultado em um arquivo `resultado.pgm`, utilize em um terminal a linha de comando:
  - Linux e Mac: `./convolucao imagem.pgm matriz.txt > resultado.pgm`
  - Windows: `convolucao imagem.pgm matriz.txt > resultado.pgm`

Exemplos de execução:

Exemplo 1:

argv[1]:

```
P2
5 5
255
  10 20 30 40 50
  50 40 30 20 10
  10 20 30 40 50
  50 40 30 20 10
  10 20 30 40 50
```

argv[2]:

```
2
0 2 0
2 9 2
0 2 0
```

Saída:

```
P2
5 5
255
  10  20  30  40  50
  50 255 255 210  10
  10 210 255 255  50
  50 255 255 210  10
  10  20  30  40  50
```

Exemplo 2:

argv[1]:

```
P2
9 10
255
    0    0    0    0    8   49   73   98  125
    0    0    0    0   16   66   90  126  132
    0    0    0    0   33   82  106  123  140
    0    0    0    0   56   90  107  132  142
    0    0    0    8   66   99  123  140  157
    0    0    0   24   74  107  126  142  164
    0    0    0   56   90  109  132  156  174
    0    0    8   74   99  123  142  165  181
    0    0   56   90  107  140  147  106   57
    0   33   73   89   88   47   15    0    0
```

argv[2]:

```
1
0 -1 0
-1 4 -1
0 -1 0
```

Saída:

```
P2
9 10
255
   0   0   0   0   8  49  73  98 125
   0   0   0   0   0  27   0  61 132
   0   0   0   0   0  33  22   0 140
   0   0   0   0  35  16   0  16 142
   0   0   0   0  27  10  20   6 157
   0   0   0   0   9  20   0   0 164
   0   0   0  36  22   0   0  11 174
   0   0   0  43   2   2   1  75 181
   0   0  53  34  11 136 185  55  57
   0  33  73  89  88  47  15   0   0
```

Nota: Textos em azul denotam dados de entrada do programa. Textos em vermelho denotam dados de saída do programa.

Observações Gerais:

- O número máximo de submissões é 15;
- O seu programa deve estar completamente contido em um único arquivo denominado `convolucao.c`;
- Para a realização dos testes automáticos, a compilação se dará da seguinte forma:
  `gcc convolucao.c -o convolucao -Wall -std=c99 -pedantic`;
- Não se esqueça de incluir no início do programa uma breve descrição dos objetivos, da entrada, da saída, seu nome e RA;
- Para comparar seu arquivo de saída com um arquivo de saída do Susy pode usar o programa **kdiff3 (**http://kdiff3.sourceforge.net). Disponivel para Linux, Mac Os X e Windows;

---

6. Critérios Importantes

O não cumprimento dos critérios abaixo acarretará nota zero na atividade, independentemente dos resultados dos testes do SuSy.

- os únicos headers aceitos para inclusão são `stdio.h` e `stdlib.h`.