Projeto Prático 04 *

Processamento de imagens: conceitos básicos

Uma imagem digital é tipicamente definida como uma função bidimensional, $f(x, y)$, em que x e y são coordenadas espaciais (plano), e a amplitude de f no par de coordenadas (x, y) é denominado nível de intensidade ou nível de cinza da imagem naquele ponto. O número de elementos que representam uma imagem, denominados pixels (*picture elements*), é determinado pela taxa de amostragem sendo considerada. Esta é responsável pela resolução da imagem, realizada a uma melhor ou pior visualização dos detalhes de cada objeto.

O valor que cada pixel pode assumir é determinado pela quantização considerada para codificar os níveis de intensidade. Imagens em níveis de cinza usualmente consideram uma codificação em oito bits, em que a intensidade de um pixel varia de zero (preto) a 255 (branco). A Figura 1 ilustra um exemplo.

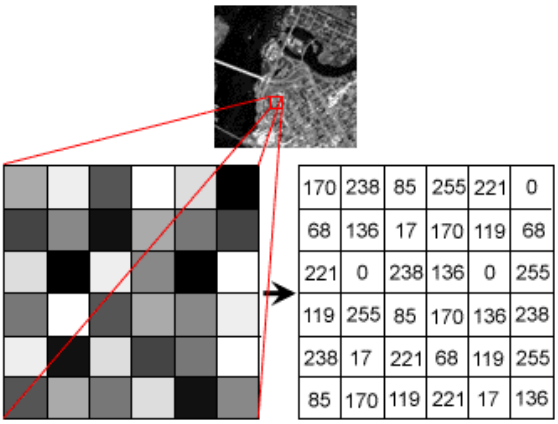


Figura 1: Imagem digital.

As imagens binárias, por sua vez, consideram apenas dois níveis de intensidade: 0 (preto) e 1 (branco). Assim como é o caso para matrizes, em imagens o canto superior esquerdo tem coordenadas $(0, 0)$.

Outro conceito relevante é o de vizinhança. A vizinhança-8 de um pixel p nas coordenadas (x, y) considera seus vizinhos horizontais, verticais e diagonais, cujas coordenadas são dadas por

$$(x+1, y) \quad (x-1, y) \quad (x, y+1) \quad (x, y-1)(x+1, y+1) \quad (x+1, y-1) \quad (x-1, y+1) \quad (x-1, y-1)$$

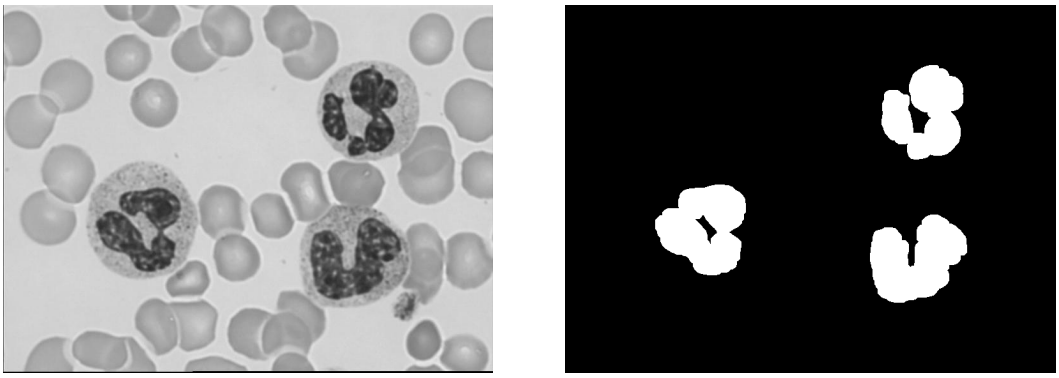
Outra vizinhança frequentemente utilizada é a que considera apenas os vizinhos horizontais e verticais (vizinhança-4). **IMPORTANTE:** Note que a vizinhança não está definida para os pares de índices muito próximos à "borda" da matriz.

Na sequência, serão brevemente descritas duas operações muito comuns: limiarização e filtragem.

*Atenção: você deve usar funções para o desenvolvimento das questões - organize o código de forma organizada e coerente. Fique atento aos requisitos de codificação (comentários, nomes significativos, etc).

(A) Limiarização

Basicamente, a limiarização é um método que converte uma imagem em níveis de cinza em uma imagem binária, a qual idealmente deve separar os objetos de interesse do fundo. A Figura 4 ilustra um exemplo (no caso, o objetivo é identificar as regiões da imagem correspondentes ao núcleo das células).



(a) (b)

Figura 2: Exemplo de limiarização: (a) Imagem original e (b) Imagem limiarizada.

Para tal é preciso especificar um valor, denominado limiar, que define se um determinado pixel pertence ao objeto ou ao fundo. Em suma, qualquer pixel cujo valor de intensidade é maior que o limiar é definido como pertencente ao objeto. Caso contrário, o ponto é classificado como pertencente ao fundo. Em outras palavras, a imagem limiarizada $g(x, y)$ é definida como:

$$g(x,y) = \begin{cases} 255, & \text{se} \quad f(x,y) \geq T \\ 0 & \text{se} \quad f(x,y) < T \end{cases} \tag{1}$$

em que T denota o limiar e os nível de intensidade 255 e 0 representam, respectivamente, objeto (branco) e fundo (preto).

(B) Filtragem

Diversas aplicações em processamento de imagens envolvem um processo denominado filtragem, que consiste em uma operação que modifica o valor de um pixel com base nos níveis de cinza dos pixels vizinhos. Para implementar este processo utiliza-se uma operação denominada convolução, que calcula o novo valor de um pixel com base no somatório dos níveis de cinza dos pixels vizinhos ponderados pelos respectivos pesos de uma máscara. Dependendo dos valores presentes na máscara, pode-se filtrar ruídos ou destacar contornos, por exemplo. A Figura 3 ilustra o processo para a operação de convolução.

Alguns exemplos de máscaras são ilustrados abaixo (neste trabalho, você vai utilizar apenas a primeira).

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} /9 \qquad \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \qquad \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

(a) Filtro da média 3×3 (b) Sobel (mudanças verticais) (c) Sobel (mudanças horizontais)

O filtro da média é uma transformação frequentemente usada para suavizar ruídos em sinais e imagens digitais. Dada uma imagem f, representada por uma matriz de inteiros com m linhas e n colunas, o filtro da média produz uma matriz transformada g, com as mesmas dimensões que f, definida da seguinte maneira: para cada par de índices (i, j) o elemento $g(i, j)$ da matriz transformada corresponde à média dos elementos da vizinhança-8 de (i, j) na imagem de entrada f.

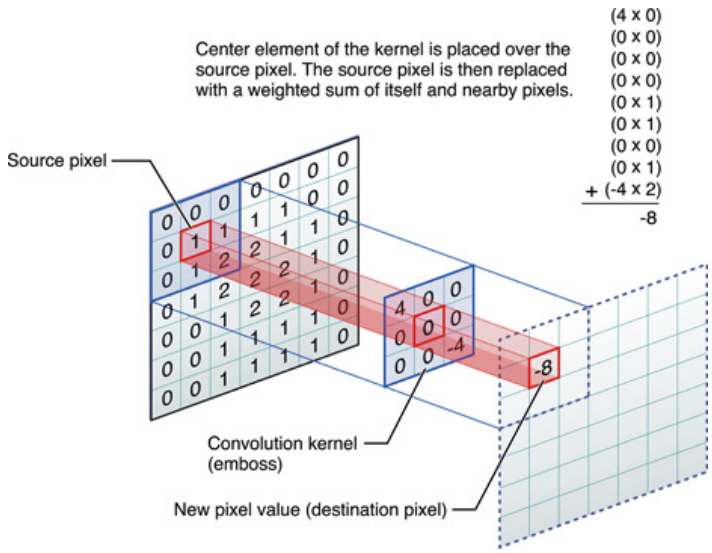


Figura 3: Imagem digital.

(C) Rotulação de componentes conexos

A rotulação (ou extração) de componentes conexos é de extrema importância em diversas aplicações de visão computacional e reconhecimento de padrões, tais como reconhecimento de texto em documentos e classificação de imagens. Basicamente, consiste no procedimento de atribuição de um único rótulo a cada objeto (ou componente conexo) da imagem. A partir dessa divisão, é possível extrair diversas informações de interesse, tais como estrutura espacial, tamanho médio ou tamanho máximo do componente conexo.

Dois pixels p e q estão conectados de 4 se obedecem ao critério de similaridade e se são vizinhos horizontais ou verticais. Da mesma forma, são ditos conectados de 8 se obedecem ao critério de similaridade e são vizinhos horizontais, verticais ou diagonais. Um componente conexo de uma imagem é dado por um conjunto de pixels que estão conectados entre si, ou seja, dado um pixel p do componente conexo, existe pelo menos um caminho de p a todos os outros pixels deste componente.

São diversos os algoritmos para executar esta tarefa, sendo classificados em *multi-pass*, *two-pass* e *one-pass*. De forma geral, tais algoritmos atribuem rótulos provisórios para cada pixel. No Anexo A, é descrita a abordagem mais simples (mas não a mais eficiente) para rotulação de componentes conexos.

1 Tarefas

Você vai utilizar o pacote de arquivos `pp04_PDI.zip`. **Após analisar cada função do código-fonte**:

1. Implemente a função `void binariza(Imagem* img, Imagem* out, int limiar)`. Depois, faça os seguintes testes:
   (a) Utilize a função desenvolvida para tentar segmentar o núcleo presente na imagem `celula01.bmp` considerando diferentes limiares. Qual o valor utilizado para obter o melhor resultado?
   (b) A imagem `celula02.bmp` pode ser segmentada utilizando o mesmo limiar da questão acima? Caso não, qual valor você considera o melhor?
   (c) Tente segmentar as demais imagens presentes na pasta `img2`. Quais os limiares utilizados e os problemas encontrados?
2. Implemente a função `void filtroMedia(Imagem* img, Imagem* out, int winSize)`, que realiza a filtragem de ruído utilizando o filtro da média considerando uma janela de dimensão `winSize`×`winSize`. Depois:
   (a) filtre a imagem `lenna.bpm` utilizando uma máscara 3×3. Qual a sua conclusão?
   (b) filtre a imagem `borboleta.bpm` utilizando máscaras 3×3 e 11×11. Qual a sua conclusão?
3. Implemente uma função que faça a rotulação de componentes conexos (defina um protótipo adequado). Depois, processe uma das imagens geradas pelo código do projeto final. Com base na rotulação realizada, crie **outra função** para contar quantos objetos existem em cada imagem.

ATENÇÃO: As imagens geradas, bem como a análise solicitada em cada item, devem ser apresentadas no formato de um relatório, o qual deve estar no modelo fornecido pela Sociedade Brasileira de Computação.

Anexo A

São diversos os algoritmos para executar esta tarefa, sendo classificados em *multi-pass*, *two-pass* e *one-pass*. De forma geral, tais algoritmos atribuem rótulos provisórios para cada pixel. Se a varredura for feita da esquerda para a direita e de cima para baixo, ela é denominada *forward scan*. Se for feita da direita para a esquerda e de baixo para cima, é denominada *backward scan*. **Atenção:** observe que a imagem de entrada deve ser binária, contendo apenas os valores 0 (para fundo) e 1 (para objeto(s)).

Algoritmo convencional - *multi-pass*

A matriz que representa a imagem de entrada será representada por I. Como neste trabalho a rotulação será feita em imagens binárias, assuma que o pixel na coordenada (i, j) pertence ao fundo (*background*) se $I(i, j) = 0$ e ao objeto (*foreground*) se $I(i, j) = 1$. A imagem rotulada será armazenada na matriz L (que possui a mesma dimensão que I).

Seja r um valor inteiro inicializado em 1. A primeira etapa deste algoritmo consiste na atribuição dos rótulos provisórios da seguinte forma:

$$L(i,j) = \begin{cases} 0, & \text{se } I(i,j) = 0; \\ r, (r \leftarrow r+1), & I(k,l) = 0, \text{ para } (k,l) \in V; \\ \min(L(k,l)), \text{ para } (k,l) \in V, & \text{caso contrário} \end{cases} \tag{2}$$

Segundo esta regra, um pixel pertencente ao fundo não tem seu valor alterado (primeira consideração). Caso todos os vizinhos de uma posição (i, j) pertençam ao fundo, é atribuído o rótulo r a esta posição na imagem resultado L e o valor de r é incrementado (segunda consideração). Caso um ou mais vizinhos de $I(i, j)$ já tenham um rótulo atribuído, o menor destes rótulos é atribuído à $L(i, j)$ (terceira consideração).

A vizinhança depende do tipo de varredura sendo realizada. As Tabelas 1 e 2 ilustram a vizinhança considerada para *forward scan* e *backward scan*, respectivamente. Os vizinhos são denotados pelas letras a, b, c e d.

a	b	c
d	(i,j)	

Tabela 1: Máscara *forward scan*

	(i,j)	d
c	b	a

Tabela 2: Máscara *backward scan*

Após essa primeira etapa, todos os pixels pertencentes a um objeto terão um rótulo temporário atribuído. O problema é que pixels pertencentes a um mesmo objeto conexo poderão ter rótulos diferentes. Suponha que a imagem de entrada, I, é representada pela matriz da Tabela 3. Após uma aplicação do procedimento acima considerando a varredura *forward scan*, a rotulação resultante seria aquela ilustrada na Tabela 4, a qual resultou na atribuição dos rótulos 2 e 3 para um mesmo objeto conexo.

Para resolver este problema, o algoritmo *multi-pass* mais simples irá aplicar repetidamente as varreduras *forward* e *backward scan* segundo o procedimento acima até que nenhuma alteração seja feita na imagem resultante L (observe que, após todos os pixels pertencentes a um objeto receberem um rótulo inicial, apenas a terceira regra será de fato utilizada nas demais varreduras).

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	**1**	**1**	0	0	0	0	0	0
0	0	**1**	**1**	0	0	0	0	0	0
0	0	0	0	0	0	0	**1**	0	0
0	0	0	0	**1**	**1**	**1**	**1**	0	0
0	0	0	0	0	0	**1**	**1**	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

(a)

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	**1**	**1**	0	0	0	0	0	0
0	0	**1**	**1**	0	0	0	0	0	0
0	0	0	0	0	0	0	**2**	0	0
0	0	0	0	**3**	**3**	**2**	**2**	0	0
0	0	0	0	0	0	**2**	**2**	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

(b)

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	**1**	**1**	0	0	0	0	0	0
0	0	**1**	**1**	0	0	0	0	0	0
0	0	0	0	0	0	0	**2**	0	0
0	0	0	0	**2**	**2**	**2**	**2**	0	0
0	0	0	0	0	0	**2**	**2**	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

(c)

Figura 4: Exemplo de execução do algoritmo *multi-pass*: (a) *forward scan*, (b) *backward scan* e (c) *forward scan*.