Processamento de Imagens

Configuração do ambiente Windows

1 - Instale o pacote WinPython 3.10. Você pode instalar a versão para máquinas de 32 bits em máquinas de 32 e 64 bits. O WinPython inclui todos os pacotes necessários para as atividades: matplotlib, numpy e scipy.

```
https://sourceforge.net/projects/winpython/files/WinPython_3.10/
```

Configuração do ambiente Linux - Ubuntu

1 - Rode

```
sudo apt-get install python3 idle python3-numpy python3-scipy
python3-matplotlib python3-dateutil python3-pyparsing
```

Configuração do ambiente Linux - Archlinux

1 - Rode

```
sudo pacman -S python python-numpy python-scipy python-matplotlib
python-dateutil python-pyparsing cblas gcc
```

Dicas sobre Python

O Python é uma linguagem interpretada muito flexível. Seu ambiente de programação, o **idle**, permite que o programador rode instruções de maneira interativa, o que facilita os testes do código sendo desenvolvido. Existem centenas de bibliotecas disponiveis que podem ser usadas para aumentar as funcionalidades da linguagem. A indentação é obrigatória e é usada para delimitar blocos, como as chaves na linguagem C.

Para usar funções de uma bibllioteca, usar o comando import. Um exemplo de biblioteca bastante útil é a biblioteca **os**. Para importá-la, rodar:

```
import os
```

Algumas funções úteis da biblioteca **os**:

`os.listdir("."")` lista os arquivos do diretório “.”, ou seja, do diretório atual.
`os.chdir("..")` acessa o diretório “..”, ou seja, o pai do diretório atual.
`os.getcwd()` mostra o nome do diretório atual.

Uma vez que sua biblioteca foi carregada, ela fica na memória, e modificações no arquivo fonte são ignoradas até que a sua biblioteca seja recarregada com a função `reload` da biblioteca **importlib**.

Preparando o ambiente

Abra um prompt de comandos chamando:
`cmd` se você estiver no Windows
`xterm` ou `terminal` se você estiver no Linux

A partir do prompt de comandos, crie um diretório para este projeto, onde serão armazenadas as imagens e o arquivo da biblioteca usando:
`md` se você estiver no Windows
`mkdir` se você estiver no Linux

Acesse o diretório usando `cd`. Execute o comando `idle` para abrir o Python Shell. Verifique se a versão é a 3. Observe que o Python Shell possui um prompt como este `>>>`

De dentro do Python Shell, execute os comandos abaixo para testar se o matplotlib e o numpy estão corretamente instalados. Se não retornarem mensagens de erro, significa que a instalação está correta.

```
>>> import matplotlib.pyplot as mp
>>> import numpy as np
```

Salve um conjunto de imagens no diretório do projeto.

Todas as funções abaixo devem funcionar tanto em imagens RGB quanto em escala de cinza, exceto a questão 14, que precisa funcionar apenas em escala de cinza.

A grande maioria recebe imagens em formato `numpy.ndarray`. Apenas duas recebem o nome do arquivo contendo a imagem, nas questões 2 e 6.

Exercícios

0.1 - Crie uma bibliotca em Python para armazenar suas funções.

0.2 - Crie uma função chamada `imread` que recebe um nome de arquivo e retorna a imagem lida. O tipo da imagem retornada deve ser `numpy.ndarray` e o de seus pixels, `uint8`.
a) Abra uma imagem colorida e a exiba usando o Python Shell.
b) Abra uma imagem em escala de cinza e a exiba usando o Python Shell.
c) Abra uma imagem pequena, com até 50 pixels de lado, e a exiba usando o Python Shell.

```
def imread(filename):
    im = mp.imread(filename)
    if (im.dtype == "float32"):
        im = np.uint8(255*im)
    if (len(im.shape) >= 3 and im.shape[2] > 3):
        im = im[:, :, 0:3]
    return im
```

0.3 - Crie uma função chamada `imshow` que recebe uma imagem como parâmetro e a exibe. Se a imagem for em escala de cinza, exiba com **colormap gray**. Sempre usar interpolação **nearest** para que os pixels apareçam como quadrados ao dar zoom ou exibir imagens pequenas.

```
def imshow(im):
    plot = mp.imshow(im, cmap=mp.gray(), origin="upper")
    plot.set_interpolation('nearest')
    mp.show()
```

1 - Crie uma função chamada `nchannels` que retorna o número de canais da imagem de entrada.

2 - Crie uma função chamada `size` que retorna um vetor onde a primeira posição é a largura e a segunda é a altura em pixels da imagem de entrada.

3 - Crie uma função chamada `rgb2gray`, que recebe uma imagem RGB e retorna outra imagem igual à imagem de entrada convertida para escala de cinza. Para converter um pixel de RGB para escala de cinza, faça a média ponderada dos valores (R, G, B) com os pesos (0.299, 0.587, 0.114) respectivamente.
ATENÇÃO: verifique se a imagem de entrada permanece inalterada após o término da execução.

4 - Crie uma função chamada `imreadgray`, que recebe um nome de arquivo e retorna a imagem lida em escala de cinza. Deve funcionar com imagens de entrada RGB e escala de cinza.

5 - Crie uma função chamada `thresh`, que recebe uma imagem e um valor de limiar. Retorna uma imagem onde cada pixel tem intensidade máxima onde o pixel correspondemte da imagem de entrada tiver intensidade maior ou igual ao limiar, e intensidade mínima caso contrário.

6 - Crie uma função chamada `negative`, que recebe uma imagem e retorna sua negativa.

7 -Crie uma função chamada `contrast`, que recebe uma imagem `f`, real `r` e um real `m`. Retorna uma imagem `g = r(f - m) + m`.
a) Modifique a função `imshow` para que exiba a imagem sem modificar as escalas de cinza.

8 - Crie uma função chamada `hist`, que retorna uma matriz coluna onde cada posição contém o número de pixels com cada intensidade de cinza. Caso a imagem seja RGB, retorne uma matriz com 3 colunas.

9 - Crie uma função chamada `showhist`, que recebe a saída da função anterior e mostra um gráfico de barras. Caso a matriz recebida tenha três colunas, ou seja, se referente a uma imagem RGB, desenhe para cada intensidade uma barra com cada uma das três cores.

10 - Altere a função anterior, adicionando um segundo parâmetro opcional chamado `binsize`. Seu valor padrão deve ser 1, o tipo é inteiro e serve para agrupar os itens do vetor recebido no primeiro parâmetro. Ou seja, se `binsize == 5`, cada barra corresponderá a um grupo de 5 intensidades consecutivas.

11 - Crie uma função chamada `histeq`, que calcula a equalização do histograma da imagem de entrada e retorna a imagem resultante. Deve funcionar para imagens em escala de cinza.

12 - Crie uma função chamada `convolve`, que recebe uma imagem de entrada e uma máscara com valores reais. Retorna a convolução da imagem de entrada pela máscara. Nesta e nas próxomas questões, quando necessário extrapolar, use o valor do pixel mais próximo pertencente à borda.

13 - Crie uma função chamada `maskBlur`, que retorna a máscara 1/16 * [[1, 2, 1], [2, 4, 2], [1, 2, 1]].

14 - Crie uma função chamada `blur`, que convolve a imagem de entrada pela máscara retornada pela função maskBlur.

15 - Crie uma função chamada `seSquare3`, que retorna o elemento estruturante binário [[1, 1, 1], [1, 1, 1], [1, 1, 1]].

16 - Crie uma função chamada `seCross3`, que retorna o elemento estruturante binário [[0, 1, 0], [1, 1, 1], [0, 1, 0]].

17 - Crie uma função chamada `erode`, que recebe uma imagem e um elemento estruturante binário. Retorna uma imagem onde cada pixel `(i, j)` da saída é igual ao menor valor presente no conjunto de pixels definido pelo elemento estruturante centrado no pixel `(i, j)` da entrada. São considerados apenas os pixels diferentes de zero no elemento estruturante.

18 - Crie uma função chamada `dilate`, semelhande `erode` da questão anterior, retornando porém o maior valor no lugar do menor. Na dilatação, o elemento estruturante deve ser espelhado em todas as dimensões.