Introduction to Big Data and Machine Learning
Image Processing in Python

Dr. Mihail

October 24, 2019

Image processing

- Images are 2D numerical arrays
- Medical imagery such as CTs or MRIs are 3D tensors, or even 3D+time such as fMRI
- In Python, while imagery can be stored with almost any container data type, we will use numpy.array
- We will use numpy for basic array manipulation
- We will use skimage.io for loading/saving images
- We will use scipy.ndimage for image processing routines

```
from scipy import ndimage
```

Image processing

Tasks

- Input/Output, displaying images
- Basic manipulations: cropping, flipping, rotating
- Image filtering: denoising, sharpening
- Image segmentation: labeling pixels corresponding to different objects
- Classification
- Feature extraction
- Registration
- etc.

Opening an image

Code

```
from skimage.io import imread
import numpy as np
import matplotlib.pyplot as plt

im = imread('https://mypages.valdosta.edu/rpmihail/teaching/F19/
    CS4731/futuristic.jpg')


print('Data type: ', im.dtype)
print('Shape: ', im.shape)
```

Displaying an image

Red, Green and Blue Channels

```
fig, ax = plt.subplots(1, 4, figsize=(20, 20))
ax[0].imshow(im), ax[0].axis('off'), ax[0].set_title('Original')
ax[1].imshow(im[:, :, 0], cmap=plt.cm.gray), ax[1].axis('off'),
    ax[1].set_title('Red')
ax[2].imshow(im[:, :, 1], cmap=plt.cm.gray), ax[2].axis('off'),
    ax[2].set_title('Green')
ax[3].imshow(im[:, :, 2], cmap=plt.cm.gray), ax[3].axis('off'),
    ax[3].set_title('Blue')
```

Basic manipulations

Masking

```
from scipy import misc
im = misc.face(gray=True)

lx, ly = face.shape
X, Y = np.ogrid[0:lx, 0:ly]
mask = np.sqrt((X - lx / 2) ** 2 + (Y - ly / 2) ** 2) > 300

im[mask] = 0

plt.imshow(im)
```

Statistical information

Stats

```
im = misc.face(gray=True)
print(im.mean())
print(im.max(), im.min())
```

Basic manipulations

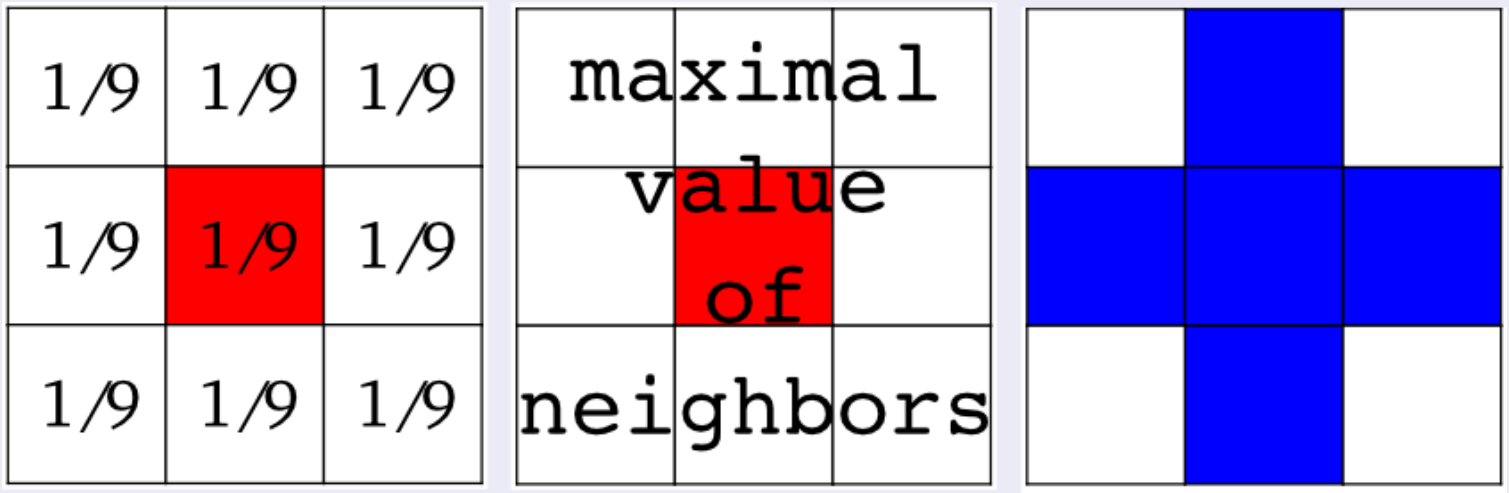
Transformations

```
from scipy import ndimage
im = misc.face(gray=True)
lx, ly = im.shape
# Cropping
crop_im = im[lx // 4: - lx // 4, ly // 4: - ly // 4]
# up <-> down flip
flip_ud_im = np.flipud(im)
# rotation
rotate_im = ndimage.rotate(im, 45)
rotate_im_noreshape = ndimage.rotate(im, 45, reshape=False)

plt.figure()
plt.imshow(rotate_im)

plt.figure()
plt.imshow(rotate_im_noreshape)
```

Basic manipulations

Image filtering

- Local filters: replace the value of pixels by a function of the values of neighboring pixels.
- Neighbourhood: square (choose size), disk, or more complicated structuring element.



Kernels

Basic manipulations

Filters

```
from scipy import misc
im = misc.face(gray=True)
blurred = ndimage.gaussian_filter(im, sigma=3)
very_blurred = ndimage.gaussian_filter(im, sigma=5)
local_mean = ndimage.uniform_filter(im, size=11)



fig, ax = plt.subplots(1, 3, figsize=(20, 20))
ax[0].imshow(blurred), ax[0].axis('off'), ax[0].set_title('
    Blurred')
ax[1].imshow(very_blurred), ax[1].axis('off'), ax[1].set_title('
    Very blurred')
ax[2].imshow(local_mean, cmap=plt.cm.gray), ax[2].axis('off'), ax
    [2].set_title('Uniform')
```

Basic manipulations

Sharpening

```
from scipy import misc
im = misc.face(gray=True).astype(float)
blurred_f = ndimage.gaussian_filter(im, 3)
filter_blurred_im = ndimage.gaussian_filter(blurred_f, 1)
alpha = 30

#increase the weight of edges by adding an approximation of the
    Laplacian:
sharpened = blurred_f + alpha * (blurred_f - filter_blurred_im)

fig, ax = plt.subplots(1, 2, figsize=(20, 20))
ax[0].imshow(blurred_f), ax[0].axis('off'), ax[0].set_title('
    Blurred')
ax[1].imshow(sharpened), ax[1].axis('off'), ax[1].set_title('
    Sharpened')
```

Basic manipulations

Edge detection

```
im = np.zeros((256, 256))
im[64:-64, 64:-64] = 1
im = ndimage.rotate(im, 15, mode='constant')
im = ndimage.gaussian_filter(im, 8)
sx = ndimage.sobel(im, axis=0, mode='constant')
sy = ndimage.sobel(im, axis=1, mode='constant')
sob = np.hypot(sx, sy)
plt.figure(figsize=(16, 5)), plt.subplot(141)
plt.imshow(im, cmap=plt.cm.gray), plt.axis('off')
plt.title('square', fontsize=20), plt.subplot(142)
plt.imshow(sx), plt.axis('off')
plt.title('Sobel (x direction)', fontsize=20)
plt.subplot(143), plt.imshow(sob), plt.axis('off')
plt.subplots_adjust(wspace=0.02, hspace=0.02, top=1, bottom=0,
    left=0, right=0.9)
plt.show()
```