Tutorial: Image Processing with Python(I)

- Course: Digital Image Processing(ELEC4245)
- Tutor: Dong Jinping (RSC G96, jpdong@eee.hku.hk)
- Venue: CB103
- Date: 14 Mar, 2017

Outline

- Brief Introduction of Python
  - Why Python
  - Vector Programming, Modular Programming
- Digital Image Processing (DIP)
  - Windowed operation in
    - *Spatial domain*
    - *Spectral domain(FFT)*
  - Advanced Image Analysis (Histogram)
  - Color Space Conversion
- Summary & Resources for further study

Why Python

- Why not Photoshop/Image J
  - Powerful tools to deal with images
  - **Only application of existing DIP techniques**
- Python
  - Programming
    - General-purpose language, open-source, object-oriented, etc.
    - Useful skill, help to find a job
  - For image processing
    - **Help to understand DIP principles**
    - **Further development of DIP techniques**
    - Batch processing, extensible, tunable, repeatable
    - Steep learning curve, frustration with bugs
  - Python(x,y)[1], Spyder (Scientific Python Development EnviRonment)
    - *Numpy* (linear algebra), *Scipy* (signal and image processing)
    - *Matplotlib* (interactive 2D/3D plotting), *OpenCV-Python*
    - *Skimage* (scikit-image, collection of image processing algorithms)

1. https://python-xy.github.io/

Vector Programming

- C-style code vs Matlab-style code

```
from scipy import misc

faceA = misc.imread('letterA.png')
faceB = misc.imread('letterB.png')

faceAB = np.zeros((500,500))
for i in range(500):
    for j in range(500):
        faceAB[i,j] = 0.5*faceA[i,j] + 0.5*faceB[i,j]
```

VS

```
from scipy import misc

faceA = misc.imread('letterA.png')
faceB = misc.imread('letterB.png')

faceAB = 0.5*faceA + 0.5*faceB
```

(A+B)/2= 

Images from: http://graphemica.com/unicode/characters

Modular Programming

```
from scipy import misc

faceA = misc.imread('letterA.png')
faceB = misc.imread('letterB.png')

faceAB = 0.5*faceA + 0.5*faceB
```

VS

1. Import Libraries and APIs (unlike Matlab)

```
from scipy import misc
```

2. Define Functions

```
def average(a,b, weight=0.5):
    '''Perform elemenrwise average of two images'''
    return weight*a + (1-weight)*b
```

3. Main procedure

```
# Main procedure
faceA = misc.imread('letterA.png')
faceB = misc.imread('letterB.png')
faceAB = average(faceA, faceB, 0.5)
```

Tune parameters here

(A+B)/2= 

Images from: http://graphemica.com/unicode/characters

Practice I

- 5 mins
- Start Python(x,y), enter Spyder
- Run the two scripts for some basic ideas
  - *A+B_c-style.py*
  - *A+B_matlab-style.py*
- Write the ‘modular programming’ code
  - Take average of two images
  - Tunable weight
  - Try to tune the weight parameter
- Tips
  - *%matplotlib qt*
  - *dir()*

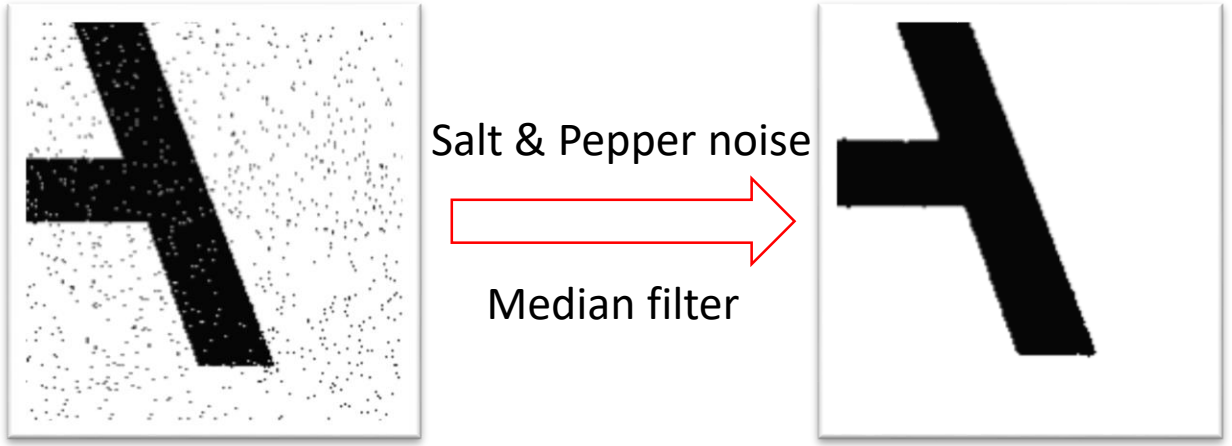
Syntax

```
def functionname( parameters ):
   "function_docstring"
   function_suite
   return [expression]
```

```
from scipy import misc
import numpy as np
import matplotlib.pyplot as plt

def [redacted]
    '''Perform elemenrwise average of two images'''
    return [redacted]

# Main procedure
faceA = misc.imread('letterA.png')
faceB = misc.imread('letterB.png')
faceAB = [redacted]

# Display images
faceAB=np.uint8(faceAB)
plt.figure('modular programming example')
plt.imshow(faceAB,cmap='gray')
misc.imsave('letterA+B.png',faceAB)
```

Windowed Operation in Spatial Domain

- Gaussian filter, Median filter
  - Gaussian noise vs Salt & Pepper noise



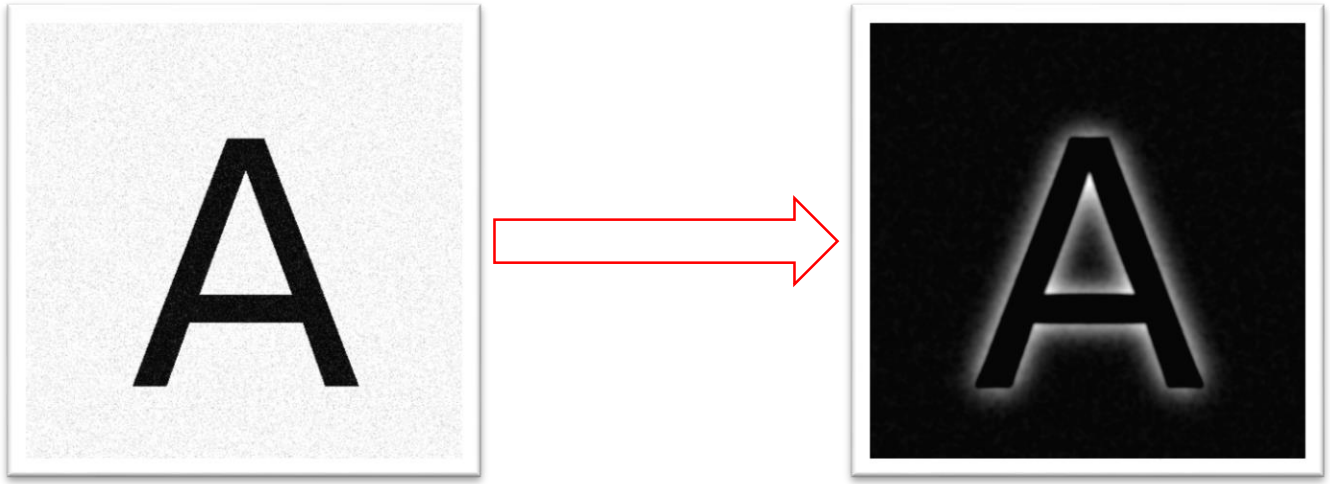
Practice II

- 15 mins
- Add noise
  - Add to ‘letterA.png’
    - Gaussian noise, Salt & Pepper noise
    - *skimage.util.random_noise*
  - Save image
    - *skimage.misc.imsave*
- Denoise
  - Complete *denoise_g_m.py*
  - Choose different noise for filtering
  - What have you observed?
- Tips
  - *help(skimage.filters)/ skimage.filters?*

```
import matplotlib.pyplot as plt
from scipy import misc
from skimage.filters import [redacted]
from skimage.morphology import rectangle

# Main procedure
#A_noise=misc.imread('img_A_gaussian_noise.png')   # choose noise
A_noise=misc.imread('img_A_saltpepper_noise.png')

A_noise_filtered1=[redacted](A_noise,[redacted])
A_noise_filtered2=[redacted](A_noise,rectangle(3,3))

# Display images
plt.figure('noisy image')
plt.imshow(A_noise,cmap='gray')
plt.figure('gaussian filter')
plt.imshow(A_noise_filtered1,cmap='gray')
plt.figure('median filter')
plt.imshow(A_noise_filtered2,cmap='gray')

misc.imsave('img_A_filtered.png',A_noise_filtered2)
```

Windowed Operation in Spectral Domain(FFT)

- Edge enhancing filter
  - Bandpass filter
  - Filtering
    - $i(x,y) = g(x,y) * h(x,y) \Leftrightarrow I(f_x, f_y) = G(f_x, f_y)H(f_x, f_y)$
    - Fast Fourier transform (FFT)



Practice III

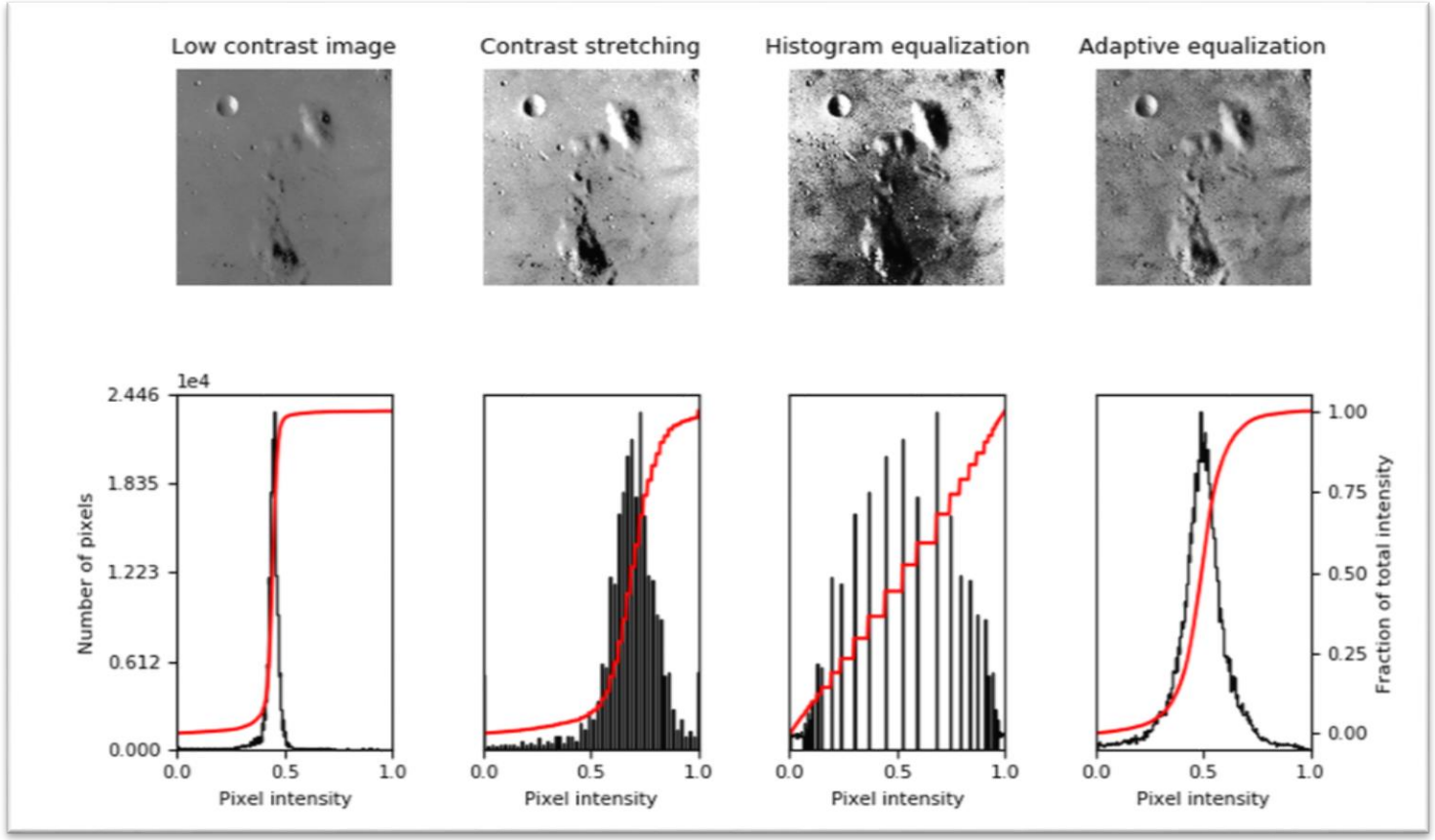
- 15 mins
- Bandpass filter & Convolution
  - Complete *edge_enhance.py*
  - Bandpass filter
    - From Gaussian kernel
    - *bp_filter.py*
- Tips:
  - *plt.subplot()*
  - *plt.axis('off')*

```python
from scipy import misc,fftpack
import matplotlib.pyplot as plt
import numpy as np

def gaussian_kernel(N, sigma):
    '''Construct a blur kernel'''
    coord = np.arange(-N/2, N/2)
    xx,yy = np.meshgrid(coord,coord)
    rr = -0.5/[redacted]**2 * ([redacted] + [redacted])
    return np.exp(rr)

def fft_convolve(img, H):
    '''Apply filter in spectral domain'''
    if img.shape != H.shape:
        return -1
    IMG = fftpack.fft2(img)
    power_spectrum = fftpack.fftshift(np.log(np.abs(IMG)))
    IMG *= fftpack.ifftshift(H)
    new_img = [redacted](IMG)
    return new_img.real, power_spectrum
```

```python
# Load image
faceA = misc.imread('img_A_gaussian_noise.png',flatten=True)
faceA /= np.[redacted](faceA)
N = faceA.shape[0]

# Bandpass filter
#H = 1.
H = gaussian_kernel(N, sigma=40)
H -= gaussian_kernel(N, sigma=5)
H /= np.linalg.norm(H,1)

# Apply filter
face_edge,FACE = fft_convolve(faceA, H)
face_edge[face_edge<0] = 0.
```

Advanced Image Analysis

- Histogram analysis
- Contrast stretching
- Histogram equalization
- Adaptive equalization



Images and histograms[1]

1. http://scikit-image.org/docs/dev/auto_examples/color_exposure/plot_equalize.html#sphx-glr-auto-examples-color-exposure-plot-equalize-py

Practice IV

- 15 mins
- Complete main procedure of *hist.py*
  - *data.moon()*
- Plot image and its histogram of
  - Original image
  - Contrast stretching
  - Histogram equalization
  - Adaptive equalization
- Which is the best? Metric?
- Tips:
  - *np.percentile(), exposure.rescale_intensity()*
  - *exposure.equalize_hist()*
  - *exposure.equalize_adapthist()*

```python
import matplotlib.pyplot as plt
import numpy as np
from skimage import exposure,data

def plot_img_hist(img,img_name='',bins=256):
    '''plot image and its histogram'''
    plt.figure(img_name)
    plt.subplot(1,2,1)
    plt.imshow(img,cmap='gray')
    plt.subplot(1,2,2)
    plt.hist(img.ravel(),bins)
    plt.show

# Main procedure
img_moon = data.moon()  # Load data

# Low contrast image
```

Color Space Conversion

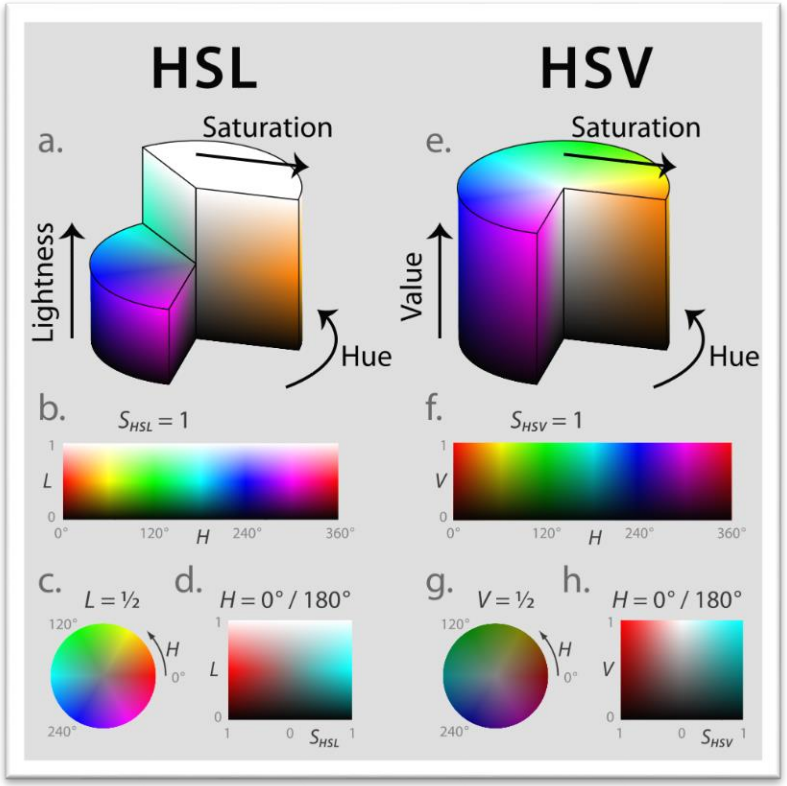
Example： Chroma Key[1]

- Color space conversion
  - RGB to HSL/HSV[2]
    - Cartesian → Cylindrical
    - More intuitive for human
    - Easy to capture color using H
- OpenCV-Python
  - *cv2.cvtColor(input_image, flag)*
  - HSV range[3]
    - Hue [0,179]
    - Saturation [0,255]
    - Value [0,255]
  - Image, Video

1. https://en.wikipedia.org/wiki/Chroma_key
2. https://en.wikipedia.org/wiki/HSL_and_HSV
3. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html#converting-colorspaces

Color Space Conversion

- Color space conversion
  - RGB to HSL/HSV[1]
    - Cartesian → Cylindrical
    - More intuitive for human
    - Easy to capture color using H
- OpenCV-Python
  - *cv2.cvtColor(input_image, flag)*
  - HSV range[2]
    - Hue [0,179]
    - Saturation [0,255]
    - Value [0,255]
  - Image, Video

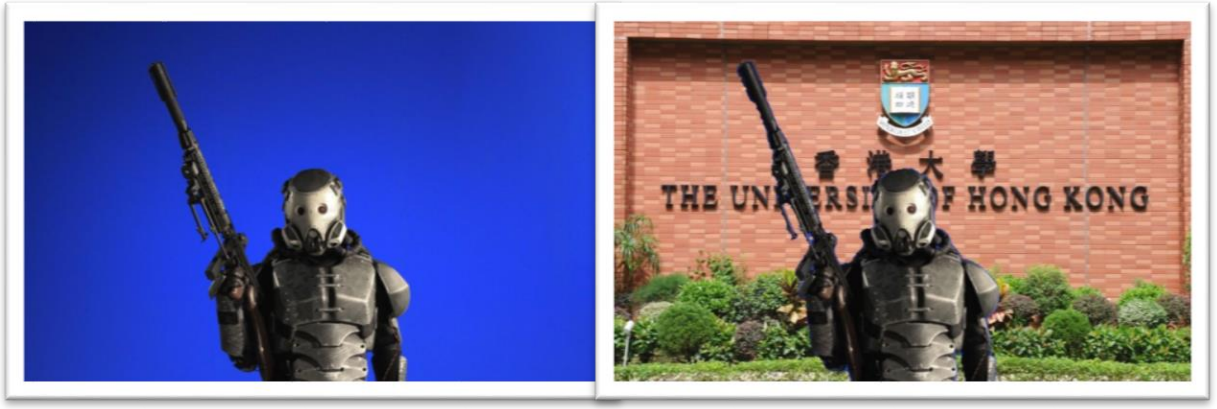


HSL & HSV[1]

1. https://en.wikipedia.org/wiki/HSL_and_HSV
2. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html#converting-colorspaces

Practice V

- 20 mins
- Fuse two images
  - Run *opencv_img.py*
  - What if the background is green?
    - HSV value of green?
    - How to change the background to green?
      - *blue2other.py*
    - Try to use the green one to fuse
- Blue object tracking in real-time video[1]
  - Turn on your camera
  - Run *opencv_vid.py* by double-click
  - Show a blue object to the camera
  - Track other color?
- Tips
  - *cv2.inRange(), cv2.bitwise_and()*
  - *cv2.cvtColor(input_image, cv2.COLOR_BGR2HSV)*
  - Coordinates start from 0 in *Numpy*!




```
>>> green = np.uint8([[[0,255,0 ]]])
>>> hsv_green = cv2.cvtColor(green,cv2.COLOR_BGR2HSV
>>> print hsv_green
[[[ 60 255 255]]]
```

1. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_colorspaces/py_colorspaces.html#converting-colorspaces

Summary & Resources for further study

- Summary
  - *Numpy, Scipy, Matplotlib, Skimage, OpenCV-Python*
- Python programming
  - Help system, Python(x,y) documents
  - https://scipy.org/index.html
  - NumPy for Matlab users
    - https://docs.scipy.org/doc/numpy-dev/user/numpy-for-matlab-users.html#numpy-for-matlab-users
- DIP
  - http://scikit-image.org/
  - https://opencv-python-tutroals.readthedocs.io/en/latest/

Acknowledgement

- Parts of the materials were developed by Antony Chan in ELEC4245 tutorial of last year

- Hope this tutorial is helpful, thank you!