THỰC HÀNH 1: ẢNH KỸ THUẬT SỐ & MÀU

Sau bài thực hành này, sinh viên có thể

- Sử dụng thư viện PIL để nạp hình với nhiều định dạng JPG, GIF, Bitmaps, PNG
- Viết được chương trình nạp hình có độ phân giải intensity và spatio
- Viết được chương trình xử lý màu của ảnh
- Viết được chương trình lọc ảnh

1. CÀI ĐẶT THƯ VIỆN

```
pip install imageio
```

```
pip install scipy
```

```
pip install scikit-image
```

2. VIẾT CHƯƠNG TRÌNH XỬ LÝ ẢNH CƠ BẢN

2.1. Nạp ảnh sử dụng thư viện Python Image Library (PIL)

PIL là thư viện mã nguồn mở kèm theo ngôn ngữ lập trình Python. Thư viện này đực dùng để mở, xử lý và lưu ảnh với nhiều định dạng khác nhau.

```
from PIL import Image
import numpy as np
img = Image.open('bird.png')
img.show()
```

2.2. Nạp ảnh sử dụng lệnh thư viện imageio

Imageio là thư viện dễ sử dụng dùng để đọc và ghi dữ liệu ảnh, ảnh động

```
import numpy as np
import imageio.v2 as iio
import matplotlib.pylab as plt
data = iio.imread('bird.png')
plt.imshow(data)
plt.show()
```

2.3. Độ phân giải ảnh

Intensity resolution biểu diễn giá trị cho mỗi pixel.

Spatial resolution biểu diễn tổng số pixel cho kích thước ảnh.

- Viết chương trình nạp ảnh màu và chuyển sang ảnh grayscale

```
import numpy as np
import imageio.v2 as iio
import matplotlib.pylab as plt
data = iio.imread('bird.png', as_gray=True)
plt.imshow(data)
plt.show()
```

- Viết chương trình nạp ảnh grayscale và giảm 4 bit thấp, giữ 4 bit cao. Lưu ảnh với tên là birdF0.png

```
import numpy as np
import imageio.v2 as iio
import matplotlib.pylab as plt
data = iio.imread('bird.png', as_gray=True).astype(np.uint8)
cl = data & 0xF0
iio.imsave('birdf0.png',cl)
tmp = iio.imread('birdf0.png')
plt.imshow(tmp)
plt.show()
```

Bài tập: sinh viên viết chương trình tạo ảnh mới birdc0.png và bird80.png với mức giảm intensity resolution là 0xc0 và 0x80. Xem kết quả

2.4. Màu sắc với hệ RGB

Để biểu diễn màu, mỗi pixel trong một ảnh sử dụng 3 giá trị ứng với 3 kênh (red, green, blue).

Viết chương trình hiển thị ảnh với kênh màu blue and green

```
import numpy as np
import imageio.v2 as iio
import matplotlib.pylab as plt
data = iio.imread('bird.png')
bdata = (data[:,:,1] + data[:,:,2])
plt.imshow(bdata)
plt.show()
```

2.5. Màu sắc với hệ HSV

Hệ HSV biểu diễn dữ liệu trong ba kênh là Hue, Saturation, và Value. Kênh Value biểu diễn thông tin intensity. Kênh Hue biểu diễn tỉ lệ giữa hai số lớn nhất từ 3 giá trị RGB cho mỗi pixel. Kênh Saturation xác định độ đậm đặc màu qua tỉ lệ khác nhau giữa màu thấp nhất với màu lớn nhất trong giá trị RGB.

Bài tập: Sinh viên nhập đoạn mã chuyển đổi giữa 2 hệ màu RGB→HSV và HSV→RGB và xem kết quả

```
import numpy as np
import imageio.v2 as iio
import matplotlib.pylab as plt
import colorsys

a = colorsys.rgb_to_hsv(255, 0, 0)
print(a)
b = colorsys.rgb_to_hsv(1, 0, 0)
print(b)
c = colorsys.rgb_to_hsv(0, 255, 0)
print(c)
d = colorsys.hsv_to_rgb(1, 1, 255)
print(d)
```

2.6. Ứng dụng chuyển đổi hệ màu

Bài tập: Viết chương trình thay thế giá trị kênh Hue của các pixel trong một ảnh bằng bằng phương giá trị ban đầu của Hue.

Gợi ý:

1. Chuyển ảnh từ hệ màu RGB → HSV

2. Tính bình phương giá trị kênh Hue

3. Tạo ảnh mới sử dụng giá trị H mới và giá trị Saturation và Values của ảnh ban đầu

4. Chuyển ảnh từ hệ màu HSV → RGB

Hàm rgb_to_hsv chỉ làm trên 1 pixel. Dùng hàm np.vectorize để tạo 1 hàm có thể áp dụng cho một ảnh.

```
import numpy as np
import imageio.v2 as iio
import matplotlib.pylab as plt
import colorsys

rgb = iio.imread('bird.png')
rgb2hsv = np.vectorize(colorsys.rgb_to_hsv)
h, s, v = rgb2hsv(rgb[:,:,0], rgb[:,:,1], rgb[:,:,2])
h *= h
hsv2rgb = np.vectorize(colorsys.hsv_to_rgb)
rgb2 = hsv2rgb(h, s, v)
rgb2 = np.array(rgb2).transpose((1, 2, 0))
plt.imshow(rgb2)
plt.show()
```

2.7. Lọc ảnh

Lọc ảnh được dùng để khử nhiễu (noise) trong ảnh. Có 2 loại khử nhiễu

- Linear filter: mean, Laplacian và Laplacian của Gaussian
- Non-linear filter: median, maximum, minimum, Sobel, Prewitt và Canny

Bộ lọc thường là một cửa sổ 2 chiều di chuyển trên toàn bộ ảnh. Mỗi số trên bộ lọc gọi là hệ số. Hệ số xác định ảnh hưởng của bộ lọc và kết quả đầu ra của ảnh.

F_1	F_2	F_3
F_4	F_5	F_6
F_7	F_8	F_9

Với bộ lọc 3 x 3 trên, nếu (i, j) là pixel trong ảnh thì các pixel xung quanh (i, j) có cùng chiều với bộ lọc được xem xét thực hiện lọc. Giá trị pixel được lọc được cập nhật tùy theo loại lọc sử dụng. Ví dụ, với phương pháp mean filter, giá trị pixel là $F_i = \frac{1}{N}$, với N là số phần tử trong bộ lọc.

Lọc ảnh với mean filter

```
import numpy as np
import imageio.v2 as iio
import scipy.ndimage as sn
import matplotlib.pylab as plt
import colorsys

# opening the image and converting it to grayscale
a = iio.imread('bird.png', as_gray=True)

# initializing the filter of size 5 by 5
# the filter is divided by 25 for normalization
k = np.ones((5,5))/25

# performing convolution
b = sn.convolve(a, k).astype(np.uint8)
iio.imsave('bird_mean_filter.png', b)
# b is converted from an ndarray to an image
print(b)
plt.imshow(b)
plt.show()
```

Lọc ảnh với median filter

Tính phần tử trung vị của các số trong cửa sổ lọc và gán giá trị trung vị cho pixel (i, j).

```
import numpy as np
import imageio.v2 as iio
import scipy.ndimage as sn
import matplotlib.pylab as plt
import colorsys

# opening the image and converting it to grayscale
a = iio.imread('bird.png', as_gray=True).astype(np.uint8)

# performing median filter
## size = 5: convolution
## footprint: a boolean array of the same dimension of image.
##The pixels in the input image corresponding to the points to the
##footprint with true values are considered for filtering
## mode: padding (constant, reflect, nearest)
b = sn.median_filter(a, size=5, footprint=None, output=None,
                     mode='reflect', cval=0.0, origin=0)
iio.imsave('bird_median_filter.png', b)
# b is converted from an ndarray to an image
print(b)
plt.imshow(b)
plt.show()
```

Lọc ảnh với Max filter

Kỹ thuật này làm tăng độ sáng cho ảnh. Giá trị lớn nhất trong ảnh phụ sẽ thay thế cho giá trị tại pixel (i, j)

```
import numpy as np
import imageio.v2 as iio
import scipy.ndimage as sn
import matplotlib.pylab as plt
import colorsys

# opening the image and converting it to grayscale
a = iio.imread('bird.png', as_gray=True).astype(np.uint8)

# performing maximum filter
## size = 5: convolution
## footprint: a boolean array of the same dimension of image.
##The pixels in the input image corresponding to the points to the
##footprint with true values are considered for filtering
## mode: padding (constant, reflect, nearest)
b = sn.maximum_filter(a, size=5, footprint=None, output=None,
                      mode='reflect', cval=0.0, origin=0)
iio.imsave('bird_max_filter.png', b)
# b is converted from an ndarray to an image
print(b)
plt.imshow(b)
plt.show()
```

Lọc ảnh với Min filter

Kỹ thuật này làm tăng độ tối cho ảnh. Giá trị nhỏ nhất trong ảnh phụ sẽ thay thế cho giá trị tại pixel (i, j)

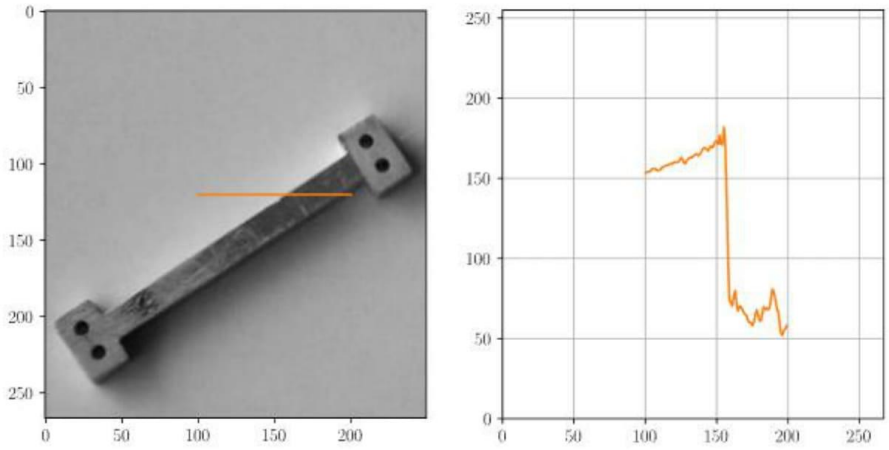
```
import numpy as np
import imageio.v2 as iio
import scipy.ndimage as sn
import matplotlib.pylab as plt
import colorsys

# opening the image and converting it to grayscale
a = iio.imread('bird.png', as_gray=True).astype(np.uint8)

# performing minimum filter
## size = 5: convolution
## footprint: a boolean array of the same dimension of image.
##The pixels in the input image corresponding to the points to the
##footprint with true values are considered for filtering
## mode: padding (constant, reflect, nearest)
b = sn.minimum_filter(a, size=5, footprint=None, output=None,
                      mode='reflect', cval=0.0, origin=0)
iio.imsave('bird_min_filter.png', b)
# b is converted from an ndarray to an image
print(b)
plt.imshow(b)
plt.show()
```

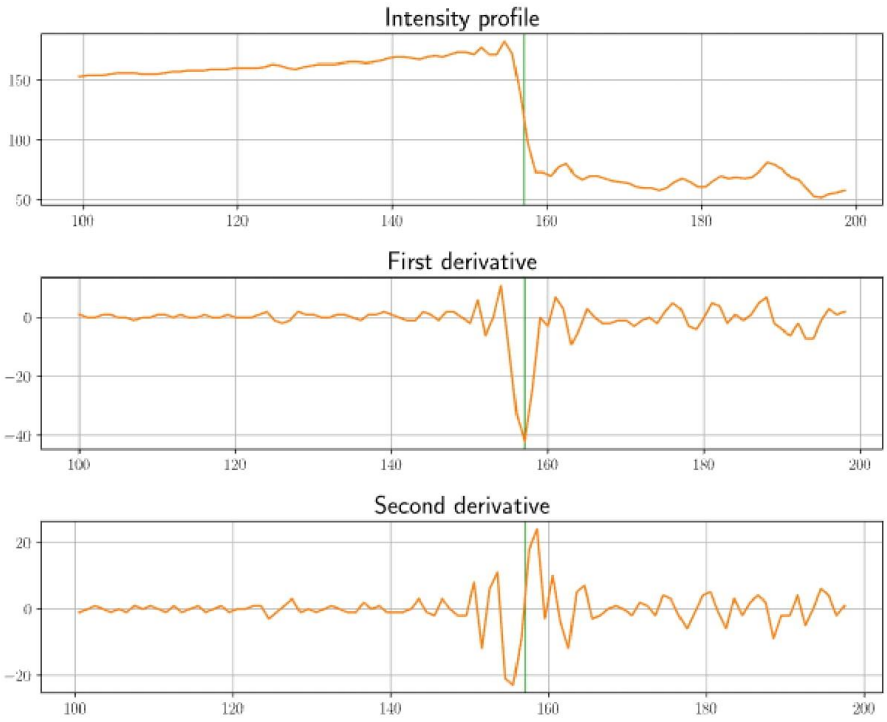
2.8. Dò cạnh biên của ảnh

Là kỹ thuật nhận dạng hoặc đo lường các đối tượng trong ảnh qua đường biên giữa các đối tượng.

Có 2 dạng làm nổi bật đường biên đối tượng:

- First derivative: xác định các giá trị có sẵn của ảnh tới cạnh biên
- Second derivative: xác định các điểm gia nhau bằng 0





Thực hành: xác định biên của đối tượng trên first derivative sử dụng Sobel filter

Sobel filter sử dụng 2 kênh tìm cạnh ngang và dọc của biên ảnh

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

```
import numpy as np
import imageio.v2 as iio
import scipy.ndimage as sn
from skimage import filters
import matplotlib.pylab as plt
import colorsys

# opening the image and converting it to grayscale
a = iio.imread('bird.png', as_gray=True)
b = filters.sobel(a).astype(np.uint8)

iio.imsave('bird_sobel_filter_edge_detection.png', b)
plt.imshow(b)
plt.show()
```

Thực hành: xác định biên của đối tượng trên first derivative sử dụng Prewitt filter

```
import numpy as np
import imageio.v2 as iio
import scipy.ndimage as sn
from skimage import filters
import matplotlib.pylab as plt
import colorsys

# opening the image and converting it to grayscale
a = iio.imread('bird.png', as_gray=True)
b = filters.prewitt(a).astype(np.uint8)

iio.imsave('bird_prewitt_filter_edge_detection.png', b)
plt.imshow(b)
plt.show()
```

Thực hành: xác định biên của đối tượng trên first derivative sử dụng Canny filter

```
import numpy as np
import imageio.v2 as iio
import scipy.ndimage as sn
from skimage import feature
import matplotlib.pylab as plt
import colorsys

# opening the image and converting it to grayscale
a = iio.imread('bird.png', as_gray=True)
b = feature.canny(a, sigma=3).astype(np.uint8)

iio.imsave('bird_canny_filter_edge_detection.png', b)
plt.imshow(b)
plt.show()
```

Xác định biên đối tượng sử dụng Second derivative

Laplacian detection chỉ sử dụng 1 kênh để xác định biên của đối tượng

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

```
import numpy as np
import imageio.v2 as iio
import scipy.ndimage as sn
from skimage import filters
import matplotlib.pylab as plt
import colorsys

# opening the image and converting it to grayscale
a = iio.imread('bird.png', as_gray=True)
b = sn.laplace(a, mode='reflect').astype(np.uint8)

iio.imsave('bird_laplace_filter_edge_detection.png', b)
plt.imshow(b)
plt.show()
```

3. BÀI TẬP

1. Viết chương trình nạp một ảnh và lưu thành 3 ảnh với 3 màu khác nhau

2. Viết chương trình nạp một ảnh và hoán đổi giá trị các màu. Lưu các ảnh vào máy.

3. Viết chương trình nạp một ảnh, chuyển thành hệ màu HSV và lưu 3 ảnh với 3 màu khác nhau.

4. Viết chương trình nạp một ảnh, chuyển sang hệ màu HSV. Lưu ảnh mới với kênh H_{new}= 1/3 H_{old}, V_{new} = ¾ V_{old}.

5. Viết chương trình sử dụng mean filter cho các hình trong thư mục Exercise

6. Viết chương trình sử dụng các filter khử nhiễu đã thực hành cho các hình trong thư mục Exercise. Cho biết filter nào khử nhiễu tốt nhất?

7. Viết chương trình sử dụng các filter xác định biên của các hình trong thư mục Exercise. Lưu các hình vào máy. (Khử nhiễu trước khi xác định biên)

8. Viết chương trình đổi màu RGB ngẫu nhiên của các hình trong thư mục Exercise. Lưu hình mới vào máy. (Khử nhiễu trước khi đổi màu)

9. Viết chương trình đổi màu HSV ngẫu nhiên nhưng không trùng của các hình trong thư mục Exercise. Lưu hình mới vào máy. (Khử nhiễu trước khi đổi màu)