HƯỚNG DẪN THỰC HÀNH XỬ LÝ ẢNH

Nguyễn Hải Triều

Khoa CNTT-Trường ĐH Nha Trang

trieunh@ntu.edu.vn

Ngày 25 tháng 11 năm 2023

Mục lục

1 LOADING, DISPLAYING, AND SAVING

Ví dụ đọc ảnh “”*beach.png*” từ thư mục **images**. Script này phù hợp với Jupyter Notebook.

```
#import các thư viện cần thiết
import matplotlib.pyplot as plt
%matplotlib inline
import cv2

img=cv2.imread('./images/beach.png') # đọc ảnh bằng OpenCV
# in ra thông số của ảnh
print(f"width of image: {img.shape[1]} pixels")
print(f"Height of image: {img.shape[0]} pixels")
print(f"Channels of image: {img.shape[2]}")
# vẽ ảnh bằng matplotlib.
fig=plt.figure()
plt.imshow(img) #B=0, G=1, R=2
# plt.imshow(cv2.cvtColor(img,cv2.COLOR_BGR2RGB)) #B=0, G=1, R=2
plt.show()
```

```
# lưu lại ảnh
cv2.imwrite("trieu_newfigure.png",img)
```

Examining the output of the script, you should also see some basic information on our image. You'll note that the image has a width of 350 pixels, a height of 233 pixels, and 3 channels (the RGB components of the image). **Represented as a NumPy array**, our image **has a shape of (223,350,3)**.

The **NumPy shape may seem reversed** to you (specifying the height before the width), but in terms of a **matrix definition, it actually makes sense**. When we define matrices, it is common to write them in the form (*# of rows* × *# of columns*). Here, **our image has a height of 223 pixels (the number of rows) and a width of 350 pixels (the number of columns)**, thus, the NumPy shape makes sense (although it may seen a bit confusing at first) [3].

Thực thi Script bằng terminal (viết code chuyên nghiệp hơn)

```
# USAGE
# python load_display_save.py --image ../images/beach.png

# Import the necessary packages
from __future__ import print_function
import argparse
import cv2

# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required = True,
  help = "Path to the image")
args = vars(ap.parse_args())

# Load the image and show some basic information on it
image = cv2.imread(args["image"])
print("width: {} pixels".format(image.shape[1]))
print("height: {} pixels".format(image.shape[0]))
print("channels: {}".format(image.shape[2]))

# Show the image and wait for a keypress
cv2.imshow("Image", image)
cv2.waitKey(0)

# Save the image -- OpenCV handles converting filetypes
# automatically
cv2.imwrite("newimage.jpg", image)
```

2 IMAGE BASICS

2.1 Truy xuất và xử lý một điểm ảnh.

In order to access a pixel value, we **just need to supply the x and y coordinates of the pixel** we are interested in. It's important to note that **OpenCV stores RGB channels in reverse order**. While we normally think in terms of Red, Green, and Blue, **OpenCV**

actually stores them in the order of Blue, Green, and Red.

On Line 6, we grab the **pixel located at (0, 0): the top-left corner of the image**. This pixel is represented as a tuple. Then, Line 9 prints out the values of each channel to our console.

```
1  import matplotlib.pyplot as plt
2  %matplotlib inline
3  import cv2
4  img=cv2.imread('./images/trex.png')
5  # lấy pixel góc trên cùng bên trái theo thứ tự Blue, Green và Red
6  (b_tl,g_tl,r_tl)=img[0,0,:]
7  # lấy pixel góc dưới cùng bên phải theo thứ tự Blue, Green và Red
8  (b_br,g_br,r_br)=img[-1,-1,:]
9  print(f"Pixel at (0, 0) - Red: {r_tl}, Green: {g_tl}, Blue: {b_tl}")
10 print(f"Pixel at ({img.shape[0]-1}, {img.shape[1]-1}) - Red: {r_br}, Green:
       {g_br}, Blue: {b_br}")
11 plt.imshow(cv2.cvtColor(img,cv2.COLOR_BGR2RGB))
12 plt.show()
```

Just as *NumPy makes it easy to access pixel values*, it also makes it **easy to manipulate pixel values**.

2.2 Truy xuất và xử lý một vùng điểm ảnh.

Ví dụ chương trình lấy 100 × 100 pixels điểm ảnh của ảnh "*trex.png*" ở góc trên cùng bên trái tại dòng số 5.

```
1  import matplotlib.pyplot as plt
2  %matplotlib inline
3  import cv2
4  img=cv2.imread('./images/trex.png')
5  img2=img[0:100,0:100,:]
6  plt.imshow(cv2.cvtColor(img2,cv2.COLOR_BGR2RGB)) #B=0, G=1, R=2
7  cv2.imwrite("crop0100.png",img2)
8  plt.show()
```

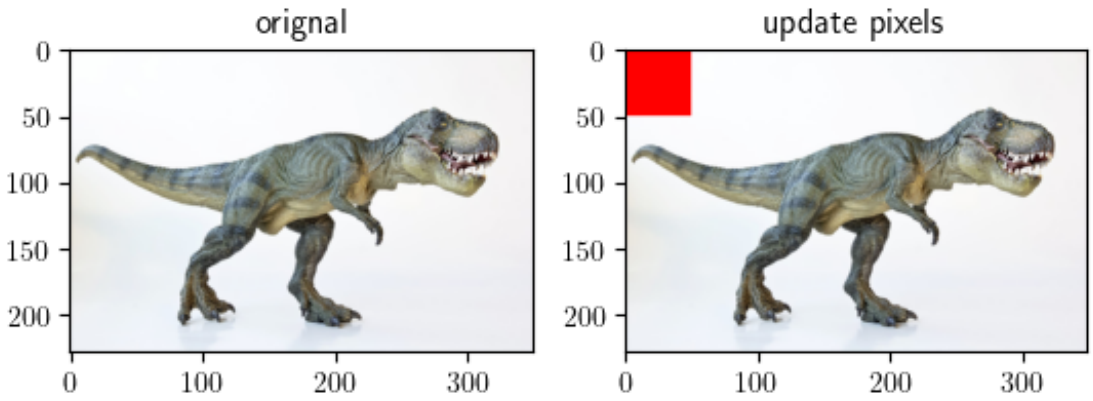
2.3 Thay đổi giá trị của các điểm ảnh.

Chúng ta có thể thay đổi trực tiếp các giá trị của pixel thành các giá trị mới. Ví dụ ta thay các pixels trong vùng 0-50 của góc trên trái bằng các pixels màu đỏ có giá trị trong hệ màu BGR là (0,0,255) (xem dòng 10 trong code).

```
1  import matplotlib.pyplot as plt
2  %matplotlib inline
3  import cv2
4  plt.rcParams.update({'text.usetex':True})
5  img=cv2.imread('./images/trex.png')
6  fig=plt.figure()
7  ax1,ax2=fig.subplots(1,2)
8  ax1.imshow(cv2.cvtColor(img,cv2.COLOR_BGR2RGB))
9  ax1.set_title("orignal")
10 img[0:50,0:50]=(0,0,255)
```

Hình 1: Kết quả crop ảnh trong vùng có toạ độ 0-100. Ta thu được ảnh mới có kích thước 100×100.

```
11  ax2.imshow(cv2.cvtColor(img,cv2.COLOR_BGR2RGB))
12  ax2.set_title("update pixels")
13  plt.show()
```



Hình 2: Update các pixels màu đỏ.

3 DRAWING

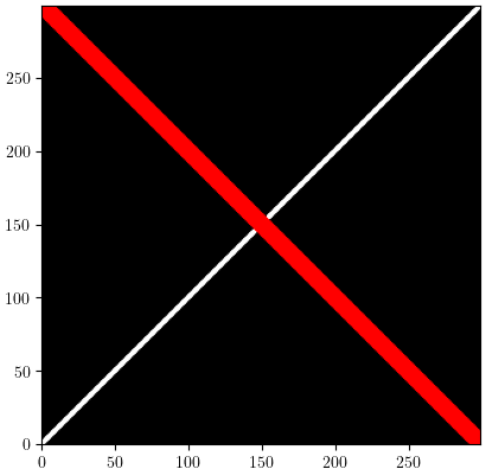
OpenCV provides convenient, easy-to-use methods to draw shapes on an image. We'll review the three most basic methods to draw shapes: **cv2.line, cv2.rectangle, and cv2.circle**.

3.1 Lines and rectangles

Đầu tiên, chúng ta tự tạo ra một ảnh màu có nền đen bằng **numpy** có kích thước 300 × 300 pixels (xem dòng 6) với kiểu dữ liệu là số nguyên không dấu 8 bit. Sử dụng phương thức **cv2.line** với cú pháp như dòng 9, 10 để *vẽ một đường thẳng vào canvas với đỉnh bắt đầu, kết thúc, màu của đường và độ đậm của nét vẽ*. Xem kết quả ở Hình 3.

```
1  import matplotlib.pyplot as plt
```

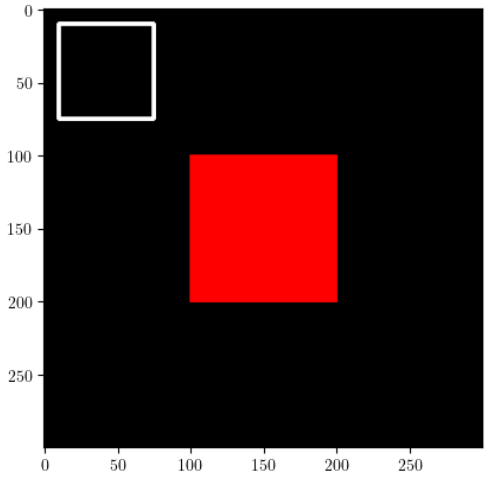
```
%matplotlib inline
import cv2
import numpy as np
# Tự tạo ra một hình nền đen bằng numpy
canvas=np.zeros((300,300,3),dtype='uint8')
white=(255,255,255)
red=(255,0,0)
cv2.line(canvas, (0,0), (canvas.shape[0],canvas.shape[1]), white)
cv2.line(canvas, (canvas.shape[0],0), (0,canvas.shape[1]), red)
plt.imshow(canvas)
plt.xlim([0,300-1])
plt.ylim([0,300-1])
plt.show()
```



Hình 3: Vẽ các đường với điểm bắt đầu và kết thúc trong hình canvas.

Tương tự như vẽ line, chúng ta có thể gọi phương thức **cv2.rectangle** để vẽ một hình chữ nhật. Lưu ý, với tham số độ đậm của nét ở dòng 10 bằng -1 để tô đầy hình chữ nhật (xem Hình 4).

```
1  import matplotlib.pyplot as plt
2  %matplotlib inline
3  import cv2
4  import numpy as np
5  plt.rcParams.update({'text.usetex':True})
6  canvas=np.zeros((300,300,3),dtype='uint8')
7  white=(255,255,255)
8  red=(255,0,0)
9  cv2.rectangle(canvas, (10,10), (canvas.shape[0]//4,canvas.shape[1]//4), white,
       2)
10 cv2.rectangle(canvas, (100,100), (200,200), red, -1)
11 plt.imshow(canvas)
12 plt.show()
```

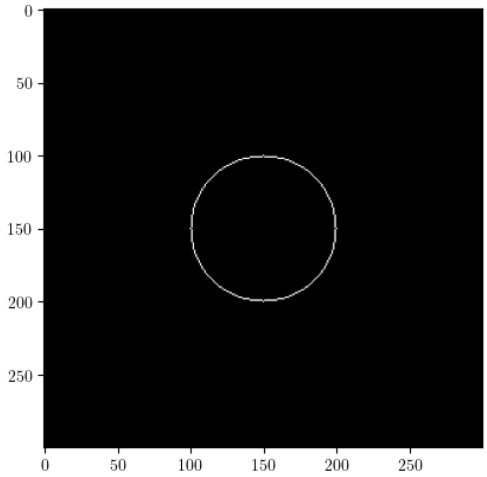


Hình 4: Sử dụng phương thức **cv2.rectangle** để vẽ các hình chữ nhật

3.2 Circles

Để vẽ được hình tròn trong OpenCV, chúng ta sử dụng phương thức **cv2.circle** với các tham số như dòng 8: *canvas, tâm hình tròn, bán kính, màu vẽ, độ đậm của nét* (xem Hình 5).

```
1  import matplotlib.pyplot as plt
2  %matplotlib inline
3  import cv2
4  import numpy as np
5  canvas=np.zeros((300,300,3),dtype='uint8')
6  white=(255,255,255)
7  red=(255,0,0)
8  cv2.circle(canvas, (canvas.shape[0]//2,canvas.shape[1]//2), 50 , white, 5)
9  plt.imshow(canvas)
10 plt.show()
```

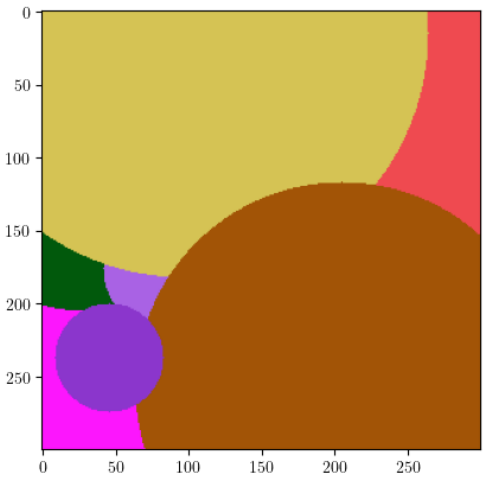


Hình 5: Sử dụng phương thức **cv2.circle** để vẽ các hình tròn

Bài tập: Sinh viên hãy vẽ các đường tròn đồng tâm có bán kính là các số nguyên tố trong khoảng từ [0,175] pixel tại tâm của canvas.

Ví dụ vẽ các đường tròn với tâm, bán kính, màu ngẫu nhiên như Hình 6

```
import matplotlib.pyplot as plt
# %matplotlib inline
import cv2
import numpy as np
plt.rcParams.update({'text.usetex':True})
canvas=np.zeros((300,300,3),dtype='uint8')
for i in range(0, 25):
    radius = np.random.randint(5, high = 200)
    color = np.random.randint(0, high = 256, size = (3,)).tolist()
    pt = np.random.randint(0, high = 300, size = (2,))
    cv2.circle(canvas, tuple(pt), radius, color, -1)
plt.imshow(canvas)
plt.plot()
```



Hình 6: Vẽ các hình tròn ngẫu nhiên

Tài liệu

[1] Ravishankar Chityala, Sridevi Pudipeddi. Image Processing and Acquisition using Python (2020), Second Edition, Chapman & Hall/CRC.

[2] Rafael C. Gonzalez, Richard E. Woods. Digital Image Processing (2018), Fourth Edition, Global Edition, Pearson.

[3] Adrian Rosebrock. Practical Python and OpenCV: An Introductory, Example Driven Guide to Image Processing and Computer Vision, Third Edition.