

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ



BÁO CÁO
BÀI TẬP LỚN XỬ LÝ ẢNH
Chủ đề: Spot the Difference

Lớp: INT3404E 21

Giảng viên: Nguyễn Thị Ngọc Diệp

Họ và tên: Mai Ngọc Duy

Mã sinh viên: 21020512

Mục lục

I. Giới thiệu	3
II. Nội dung:	3
1. Tiền xử lý (Pre – processing):	3
2. Xử lý hình ảnh (Processing/ Difference Generator):	6
2.1. Level 1: Sinh hình tại vị trí ngẫu nhiên	6
2.2. Level 2: Chèn hình tự chọn tại vị trí ngẫu nhiên	8
2.3. Level 3: Xoay một phần của ảnh	9
2.4. Level 4: Đổi màu một chi tiết của ảnh	13
3. Xử lý sự khác biệt (Difference Detection):	16
III. Đánh giá:	20
IV. Tổng kết:	21
V. Tham khảo:	21

Source code:

https://drive.google.com/drive/folders/1bfA-54G71o7eCQPuZzNeYYXYcyXZBCGf?usp=share_link



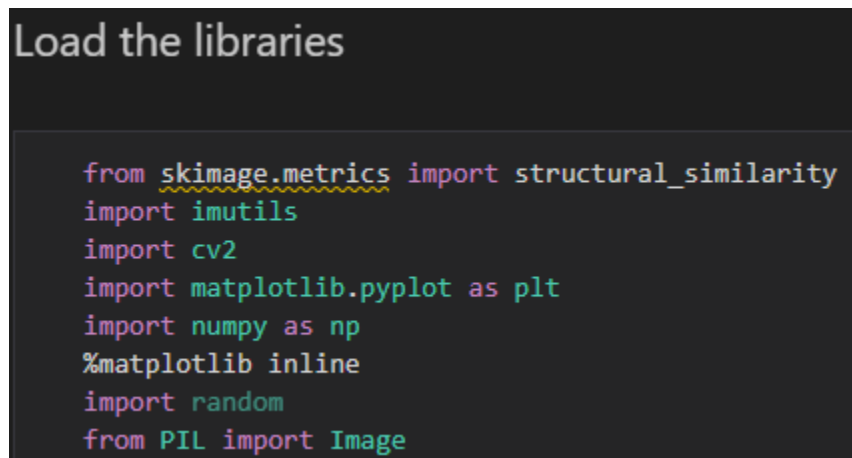
I. Giới thiệu

Spot the Difference (*tạm dịch: Tìm sự khác biệt*) là một loại trò chơi giai đồ mà người chơi phải tìm sự khác biệt giữa hai bức ảnh khá tương đồng với nhau. Spot the Difference có thể được thấy ở rất nhiều phương tiện truyền thông, chủ yếu dưới ở trong các sách cho trẻ em, bài báo hoặc trò chơi điện tử. Đây là một trò chơi giúp rèn luyện cho trẻ tính tập trung, kiên nhẫn cũng như phát triển thị giác và tư duy.

Trong bài tập lớn môn Xử lý ảnh lần này, ứng dụng các kiến thức đã học, ta sẽ tìm hiểu về cơ chế và một số cách thiết lập để có thể tạo dữ liệu ảnh đầu ra đơn giản từ ảnh đầu vào cho trước, cũng như sử dụng chúng để tìm sự khác biệt. Dưới đây là phần tìm hiểu của em về bài tập này.

II. Nội dung:

1. Tiền xử lý (Pre – processing):



```
Load the libraries

from skimage.metrics import structural_similarity
import imutils
import cv2
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
import random
from PIL import Image
```

Hình 1.1 Các thư viện sử dụng

Một số thư viện sử dụng cho việc thiết lập và xử lý hình ảnh:

- *Numpy*: Cung cấp các hàm toán học cấp cao để quản lý các mảng đa chiều, ma trận
- *Matplotlib*: Giúp tạo ra các đồ thị, thao tác và biểu diễn hình ảnh
- *OpenCV*: Cho phép đọc, ghi, thay đổi dữ liệu ảnh
- *Scikit-image (Skimage)*: Cung cấp các hàm, thuật toán xử lý ảnh. Ở đây ta sử dụng phương pháp tính Độ tương đồng về cấu trúc (*Structural_Similarity - SSIM*) phục vụ cho bài toán tìm sự khác biệt giữa hai bức ảnh
- *Pillow (PIL)*: Chứa các hàm xử lý ảnh. Ở đây ta sử dụng PIL để lấy thông số RGB của một pixel nào đó
- *Imutils*: Sử dụng để xử lý ảnh cơ bản

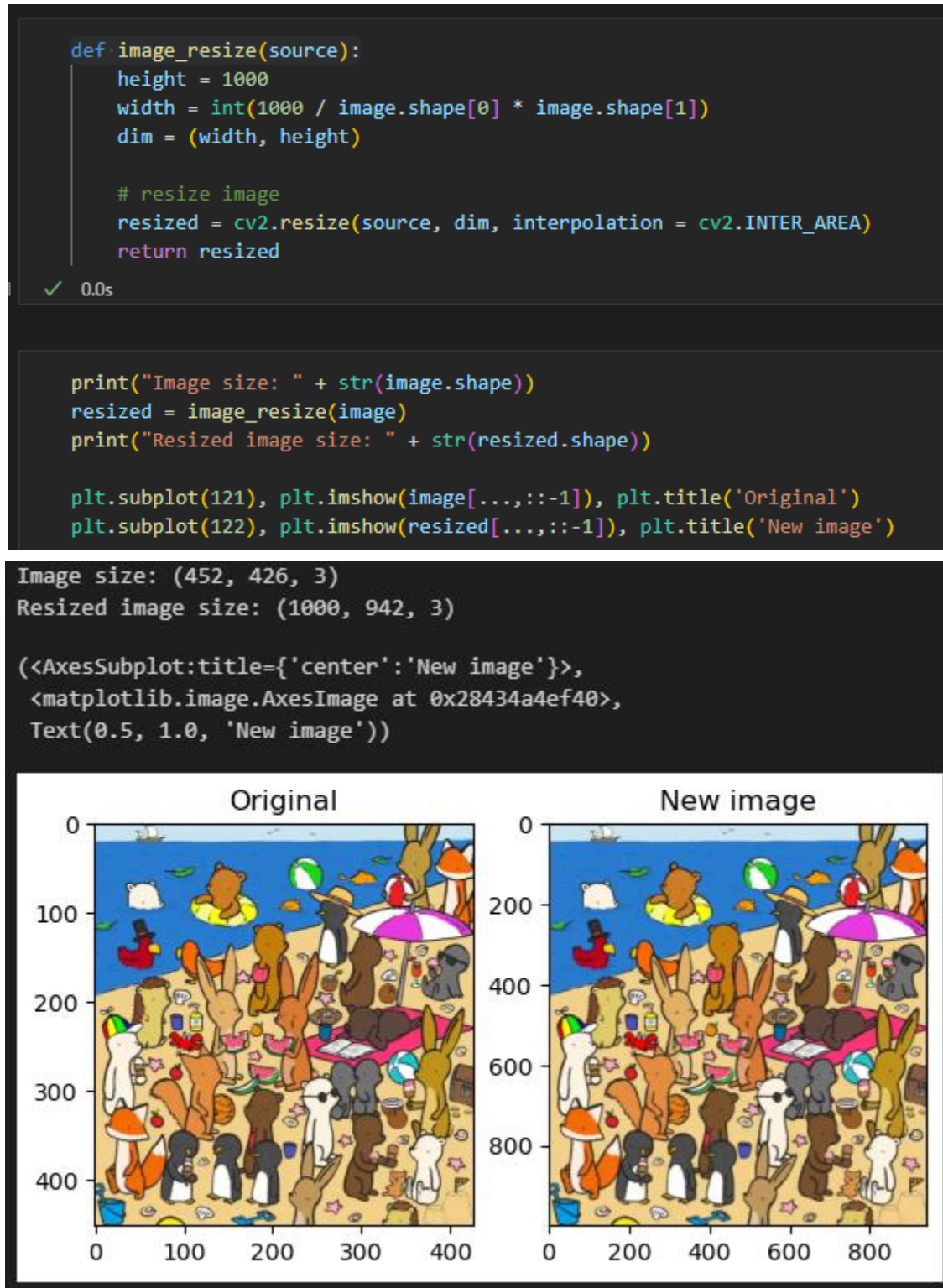
- Random: Phục vụ sinh số ngẫu nhiên

Tiếp theo ta tiến hành load hình ảnh:



Hình 1.2 Load hình ảnh sẽ sử dụng

Để đảm bảo tính thống nhất và ổn định cho việc tính toán lúc sau, ta resize lại hình ảnh đầu vào với tỉ lệ giữ nguyên, chiều cao 1000px



Hình 1.3 Điều chỉnh kích thước hình ảnh sẽ dùng

2. Xử lý hình ảnh (Processing/ Difference Generator):

2.1. Level 1: Sinh hình tại vị trí ngẫu nhiên

Một trong những cách đơn giản nhất để tạo sự khác biệt giữa hai hình ảnh là chèn thêm vào một hình ảnh khác. Ở đây, để đơn giản hoá bài toán, ta chèn thêm một hình vuông hoặc hình tròn tại một vị trí bất kỳ trong ảnh (*ta hoàn toàn có thể sinh các hình đơn giản khác*)

```
def getDifferentPicture1(source, maxNumOfDif, shape):
    source = image_resize(source)
    img_res = source.copy()

    height = img_res.shape[0]
    width = img_res.shape[1]

    square_width = square_height = 100
    circle_radius = 100

    for i in range(0, maxNumOfDif):
        h = np.random.randint(0, height)
        w = np.random.randint(0, width)
        if (shape == 'square'):
            cv2.rectangle(img_res, (w, h), (w + square_width, h + square_height), (0, 255, 0), -1)
        if (shape == 'circle'):
            cv2.circle(img_res, (w, h), circle_radius, (255, 0, 0), -1)

    return img_res
```

Hình 2.1.1 Hàm tạo ảnh Khác biệt 1 – Sinh hình tại vị trí ngẫu nhiên

Các bước được thực hiện như sau:

- Resize lại ảnh gốc, tạo ảnh img_res từ ảnh gốc để thao tác trên đó
- Khởi tạo các kích thước cần có (độ dài cạnh nếu là hình vuông, bán kính nếu là hình tròn)
- Random vị trí điểm sẽ đặt
- Thực hiện vẽ hình tròn hoặc hình vuông theo yêu cầu

Hình ảnh đầu ra được thêm các hình vuông hoặc hình tròn theo yêu cầu dựa trên số lượng maxNumOfDif (*các hình có thể bị mất một phần do vượt quá border ảnh, có thể hạn chế bằng cách giống như hàm Khác biệt 2 ở dưới*)



Hình 2.1.2 Đầu ra Level 1 với lựa chọn hình vuông



Hình 2.1.3 Đầu ra Level 1 với lựa chọn hình tròn

2.2. Level 2: Chèn hình tự chọn tại vị trí ngẫu nhiên

Một cách khác để chèn đó là ta chèn một hình tự chọn vào ảnh. Trước đó, để đảm bảo ảnh không quá to hoặc quá nhỏ so với ảnh ta đang sử dụng, ta cần resize lại ảnh sẽ chèn vào (giống hàm *resize* trên nhưng không theo tỉ lệ, mặc định kích thước 200x200)

```
def resizeInsertImage(source):  
    height = 200  
    width = 200  
    dim = (width, height)  
  
    # resize image  
    resized = cv2.resize(source, dim, interpolation = cv2.INTER_AREA)  
    return resized
```

Hình 2.2.1 Hàm *resize* ảnh chèn

Tiếp đến ta tiến hành thêm ảnh chèn vào:

```
def getDifferentPicture2(source, maxNumOfDif):  
    source = image_resize(source)  
    img_res = source.copy()  
  
    insert_image = cv2.imread("smiley_face.png", cv2.IMREAD_UNCHANGED)  
    insert_image = resizeInsertImage(insert_image)  
  
    height = img_res.shape[0]  
    width = img_res.shape[1]  
  
    iheight = insert_image.shape[0]  
    iwidth = insert_image.shape[1]  
  
    for i in range(0, maxNumOfDif):  
        y = np.random.randint(0, height - iheight)  
        x = np.random.randint(0, width - iwidth) #Inserted picture fully inside the image  
  
        alpha_channel = insert_image[:, :, 3] / 255 # convert from 0-255 to 0.0-1.0  
        overlay_colors = insert_image[:, :, :3]  
  
        alpha_mask = np.dstack((alpha_channel, alpha_channel, alpha_channel))  
        background_subsection = img_res[y: y+iheight, x: x+iwidth]  
  
        composite = background_subsection * (1 - alpha_mask) + overlay_colors * alpha_mask  
  
        img_res[y: y+iheight, x: x+iwidth] = composite  
  
    return img_res
```

Hình 2.2.2 Hàm tạo ảnh Level 2 – Chèn hình tự chọn tại vị trí ngẫu nhiên

Các bước được thực hiện như sau:

- Resize lại ảnh gốc, tạo ảnh `img_res` từ ảnh gốc để thao tác trên đó
- Khởi tạo hình ảnh sẽ chèn
- Khởi tạo các kích thước cần có
- Random vị trí điểm sẽ đặt (*giới hạn lại vị trí random theo kích thước ảnh insert để tránh mất hình khi insert*)
- Thực hiện chèn hình sau khi đã lọc ra background (*thông qua `alpha_mask`*)

Hình ảnh sau xử lý cho ra hình ảnh insert tại các vị trí ngẫu nhiên phù hợp với số lượng `maxNumOfDif` yêu cầu



Hình 2.2.3 Đầu ra với Level 2

2.3. Level 3: Xoay một phần của ảnh

Thay vì thêm vào ảnh, ta sẽ thao tác trực tiếp vào ảnh. Một trong những cách thay đổi là xoay 1 phần ảnh theo 1 góc nào đó.

Có nhiều cách để thực hiện bài toán. Cách đơn giản nhất, tương tự như trên, là chọn một điểm bất kỳ, tạo một hình để xoay từ điểm đó và xoay hình theo mong muốn.

```
def getDifferentPicture3(source, maxNumOfDif):
    source = image_resize(source)
    img_res = source.copy()

    height = img_res.shape[0]
    width = img_res.shape[1]

    for i in range(maxNumOfDif):
        w = 100
        h = 100
        y = np.random.randint(0, height - h)
        x = np.random.randint(0, width - w)
        crop_img = img_res[y:y+h, x:x+w]
        rotated_crop_img = rotateImage(crop_img, 180) #Angle of choice
        img_res[y:y+h, x:x+w] = rotated_crop_img

    return img_res
```

Hình 2.3.1. Hàm tạo ảnh Level 3 – Xoay một phần của ảnh (Cách 1)

Cách làm trên hoàn toàn hợp lý, tuy nhiên việc chọn random điểm đối với 1 ảnh có xác suất rất nhỏ có thể khiến ảnh xoay được cắt ra xoay không thay đổi gì với ảnh trước (*giả sử pick random phải một vùng chỉ có duy nhất 1 màu, khi xoay ảnh, ảnh đó không hề thay đổi gì*). Mặt khác do chọn ngẫu nhiên một phần ảnh, hình sau khi xoay và thêm vào gần như rất dễ phát hiện bởi người chơi.

Một cách làm “tự nhiên” hơn chút, là ta xác định các contour trong hình, xây dựng bounding box quanh chúng và lựa chọn những bounding box phù hợp để xoay. Cách làm này tuy vẫn có sự ngẫu nhiên trong đó, hình ảnh sau khi xoay thực tế cũng khá dễ phát hiện, nhưng đảm bảo hơn sẽ có sự thay đổi trong ảnh và tăng nhỏ xác suất rằng ảnh được xoay, vì đã được chọn theo contour, sau khi xoay sẽ có thể “hòa” vào xung quanh nó, tăng độ khó trò chơi! 😊 (*chẳng hạn như bên trong bounding box đối xứng, hoặc texture của ảnh là những đường cong vẽ ngẫu nhiên,...*)

Dưới mô tả cách làm trên:

```

def getDifferentPicture3(source, maxNumOfDif):
    source = image_resize(source)
    img_res = source.copy()

    bounding_box_contour = getBoundingBoxContour(img_res, maxNumOfDif)

    for i in range(len(bounding_box_contour)):
        x = bounding_box_contour[i][0]
        y = bounding_box_contour[i][1]
        w = bounding_box_contour[i][2]
        h = bounding_box_contour[i][3]
        crop_img = img_res[y:y+h, x:x+w]
        rotated_crop_img = rotateImage(crop_img, 180) #Angle of choice
        img_res[y:y+h, x:x+w] = rotated_crop_img

    return img_res

```

Hình 2.3.2. Hàm tạo ảnh Khác biệt 3 – Xoay một phần của ảnh (Cách 2)

Các bước được thực hiện như sau:

- Resize lại ảnh gốc, tạo ảnh img_res từ ảnh gốc để thao tác trên đó
- Tạo một list bounding box từ contour của hình
 - + Vì một hình có rất nhiều contour nên chỉ lấy ngẫu nhiên 1 lượng nhỏ theo maxNumOfDif
 - + Tiêu chí chọn bounding box ta xét theo diện tích của nó (không quá to và quá nhỏ)

```

def getContours(source):
    img = source.copy()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    imgCanny = cv2.Canny(gray, 150, 220)
    contours, hierarchy = cv2.findContours(imgCanny, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    return contours

```

Hình 2.3.3. Hàm tạo Contour của ảnh

```

def getBoundingBoxContour(source, maxNumOfDif):
    contours = getContours(source)

    bounding_box_contour = []

    count = 0

    while(True):
        if (count == maxNumOfDif):
            break
        randomNumber = np.random.randint(0, len(contours))
        contourTest = contours[randomNumber]
        (x, y, w, h) = cv2.boundingRect(contourTest)
        areaBoundingBox = w*h
        if (areaBoundingBox >= 1000 and areaBoundingBox <= 100000):
            bounding_box_contour.append(cv2.boundingRect(contourTest))
            count += 1

    return bounding_box_contour

```

Hình 2.3.4. Hàm tạo Bounding box của một lượng Contour nhất định của ảnh

- Với mỗi bounding box trong list đã chọn, lấy các thông số, dựa trên thông số đó lấy phần ảnh trùng với vị trí bounding box và xoay nó (góc có thể tùy chọn, ở đây để 180 độ)
- Đè lên hình đang xét ảnh vừa xoay tại chính vị trí lúc trước, tương tự đến khi đủ maxNumOfDif, ta sẽ được ảnh mới

```

pic3 = getDifferentPicture3(image, 2)

plt.figure(figsize = (12, 3))
plt.subplot(121),plt.imshow(image[...,:-1]),plt.title('Original')
plt.subplot(122),plt.imshow(pic3[...,:-1]),plt.title('New image')

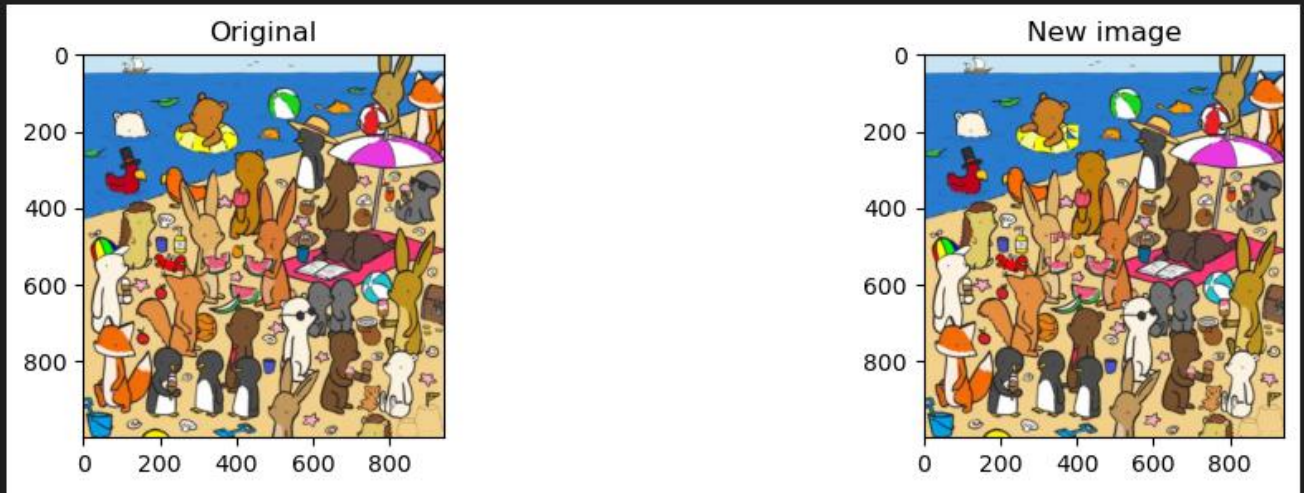
```

✓ 0.5s

```

(<AxesSubplot:title={'center':'New image'}>,
 <matplotlib.image.AxesImage at 0x2844dbf7490>,
 Text(0.5, 1.0, 'New image'))

```



Hình 2.3.5 Đầu ra với Level 3

2.4. Level 4: Đổi màu một chi tiết của ảnh

Để tăng độ khó cho trò chơi, một trong những cách làm là ta đổi màu một chi tiết nhỏ nào đó của bức ảnh đi thành một màu gần giống với nó.

Có nhiều cách để chọn màu sắc trong trường hợp này để giống với cái mình đang xét nhất, một trong những cách là ta tìm tâm của contour đang xét, lấy giá trị màu RGB của nó và thay đổi 3 hệ số màu 1 lượng nhất định, sau đó sử dụng drawContours() với chức năng cv2.FILLED để phủ màu đó lên contour đang xét.

```

def getDifferentPicture4(source, maxNumOfDif):
    source = image_resize(source)
    source_pil = Image.fromarray(source)
    pix = source_pil.load()
    img_res = source.copy()

    contours = getContours(source)
    count = 0

    while(True):
        if (count == maxNumOfDif):
            break
        randNumber = np.random.randint(0, len(contours))
        contourTest = contours[randNumber]

        (x, y, w, h) = cv2.boundingRect(contourTest)
        areaBoundingBox = w*h

        if (areaBoundingBox >= 1000 and areaBoundingBox <= 100000):
            center = findCentroidOfContour(contourTest)
            cx = center[0]
            cy = center[1]

            diff_rgb = 100 #tuy chọn
            new_tuple = (diff_rgb, diff_rgb, diff_rgb)
            new_color = tuple(abs(np.subtract(pix[cx, cy], new_tuple))) #Tru 2 contours

            a = int(new_color[0])
            b = int(new_color[1])
            c = int(new_color[2])

            cv2.drawContours(img_res, contours, randNumber, (c, b, a), cv2.FILLED)
            count += 1

    return img_res

```

Hình 2.4.1. Hàm tạo ảnh Level 4 - Đổi màu một chi tiết của ảnh

Các bước được thực hiện như sau:

- Resize lại ảnh gốc, tạo ảnh img_res từ ảnh gốc để thao tác trên đó
- Tạo thêm một ảnh source_pil để có thể lấy giá pixel cần tính lúc sau
- Tìm contour của ảnh (như Hình 2.3.3)
- Chọn ngẫu nhiên contour với bounding box kích thước phù hợp
- Tìm tâm (centroid) contour đó theo công thức

$$\text{Centroid: } \{\bar{x}, \bar{y}\} = \left\{ \frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right\}$$

(tham khảo thêm tại: https://en.wikipedia.org/wiki/Image_moment)


```
def findCentroidOfContour(cnt):
    M = cv2.moments(cnt)

    if (M['m00'] != 0):
        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])

    center = [cx, cy]
    return center
```

Hình 2.4.2. Hàm tìm tâm (Centroid) của Contour

- Xác định giá trị RGB của pixel tại tâm, sau đó trừ đi một khoảng diff_rgb tùy chọn
- Phủ kín bên trong contour đang xét bằng màu mới

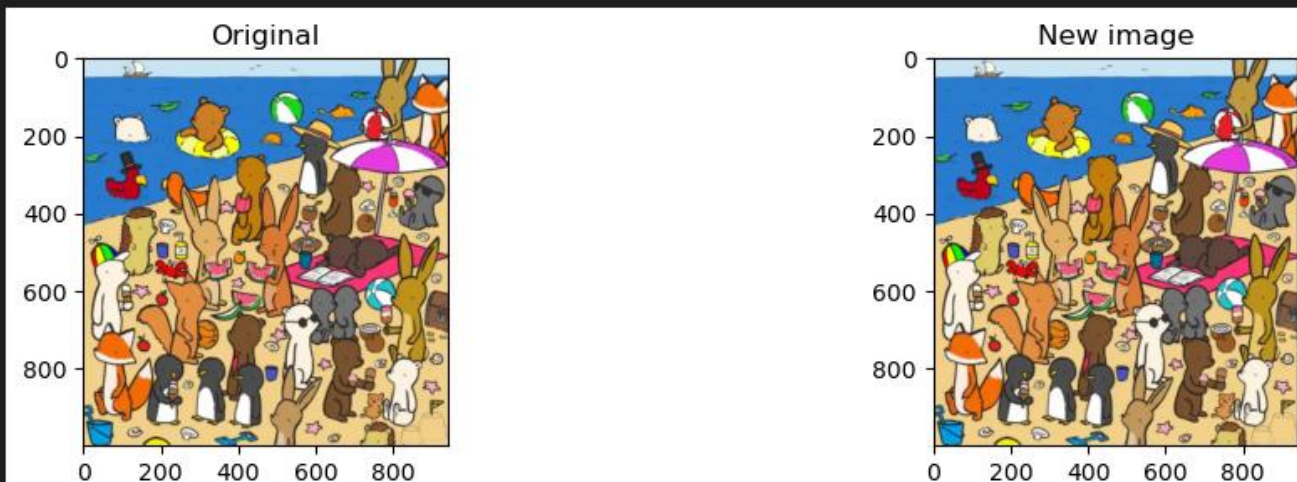
Hàm cho ra ảnh thay đổi ở một số chỉ tiết nhất định với màu gần giống với màu gốc

```
pic4 = getDifferentPicture4(image, 2)

plt.figure(figsize = (12, 3))
plt.subplot(121),plt.imshow(image[...,:-1]),plt.title('Original')
plt.subplot(122),plt.imshow(pic4[...,:-1]),plt.title('New image')
```

✓ 0.4s

```
(<AxesSubplot:title={'center':'New image'}>,
<matplotlib.image.AxesImage at 0x284403157c0>,
Text(0.5, 1.0, 'New image'))
```



Hình 2.4. Đầu ra với Level 4

Tại đây ta khó nhìn thấy sự khác biệt, cụ thể hơn trường hợp này ta sẽ xét ở phần sau.

3. Xử lý sự khác biệt (Difference Detection):

Có nhiều cách tính sự khác biệt giữa 2 bức ảnh, cách đơn giản nhất là ta trừ trực tiếp hai bức ảnh cho nhau sau khi chuyển sang dạng grayscale, vẽ các bounding box của contour lúc đó. Một cách làm khác là ta sử dụng SSIM như dưới đây

```
def checkDifference(img1, img2):
    i1 = img1.copy()
    i2 = img2.copy()
    no_of_differences = 0

    gray1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
    gray2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

    #Compute Structural Similarity Index(SSIM)
    (score, diff) = structural_similarity(gray1, gray2, full=True)
    diff = (diff * 255).astype("uint8")
    print("SSIM: {}".format(score))

    thresh = cv2.threshold(diff, 100, 128, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU) [1]
    cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cnts = imutils.grab_contours(cnts)

    for c in cnts:
        (x, y, w, h) = cv2.boundingRect(c)

        areaBoundingBox = w*h
        if (areaBoundingBox >= 1000 and areaBoundingBox <= 100000):

            no_of_differences += 1
            cv2.rectangle(i1, (x, y), (x + w, y + h), (0, 0, 255), 2)
            cv2.rectangle(i2, (x, y), (x + w, y + h), (0, 0, 255), 2)

    print("Number of differences: ", no_of_differences)

    plt.subplot(121); plt.imshow(i1[:, :, :-1]); plt.axis("off"); plt.title('Original detection')
    plt.subplot(122); plt.imshow(i2[:, :, :-1]); plt.axis("off"); plt.title('New image detection')
```

Hình 2.5.1 Hàm Check và đánh dấu sự khác biệt giữa hai bức ảnh

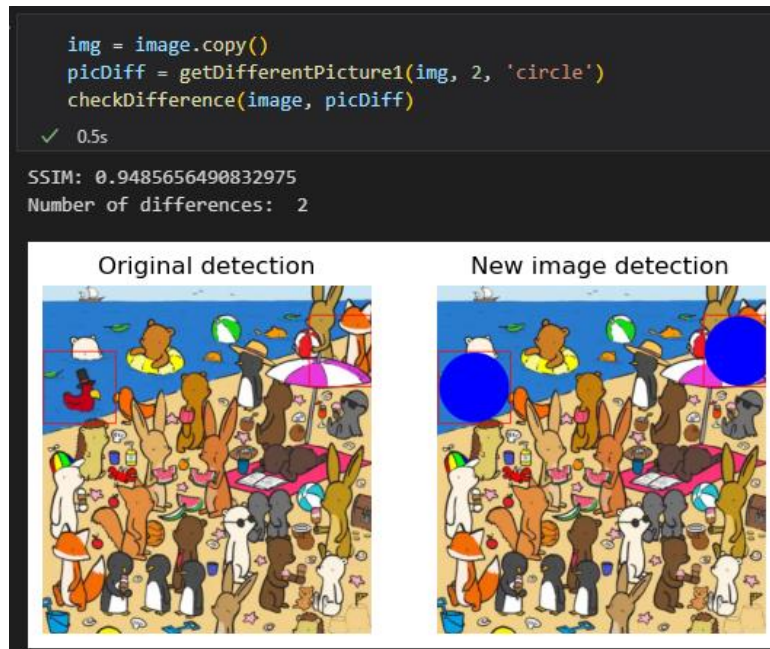
Các bước được thực hiện như sau:

- Copy hai ảnh đang xét sang img1, img2 (*tránh làm việc trực tiếp với ảnh đang xét*)
- Chuyển 2 ảnh đó sang grayscale
- Tính score và diff của structural_similarity giữa hai ảnh
 - + Score để tính mức độ tương đồng giữa hai ảnh
 - + Diff chứa thông tin khác biệt giữa hai ảnh, đang ở dưới dạng số chấm động trong khoảng [0, 1] → Phải chuyển chúng về số không âm trong khoảng [0, 255] như ảnh bình thường để xử lý
- Threshold lại ảnh diff theo mức tùy chỉnh để lọc
- Tìm các contour của ảnh vừa tạo
- Vẽ các bounding box của các contour trong đó, với mỗi cái hợp lệ tăng số lượng num_of_diff lên 1

Dưới là kết quả của các Level kể trên:



Hình 2.5.2 Kết quả Level 1 (trường hợp hình vuông)



Hình 2.5.3 Kết quả Level 1 (trường hợp hình tròn)



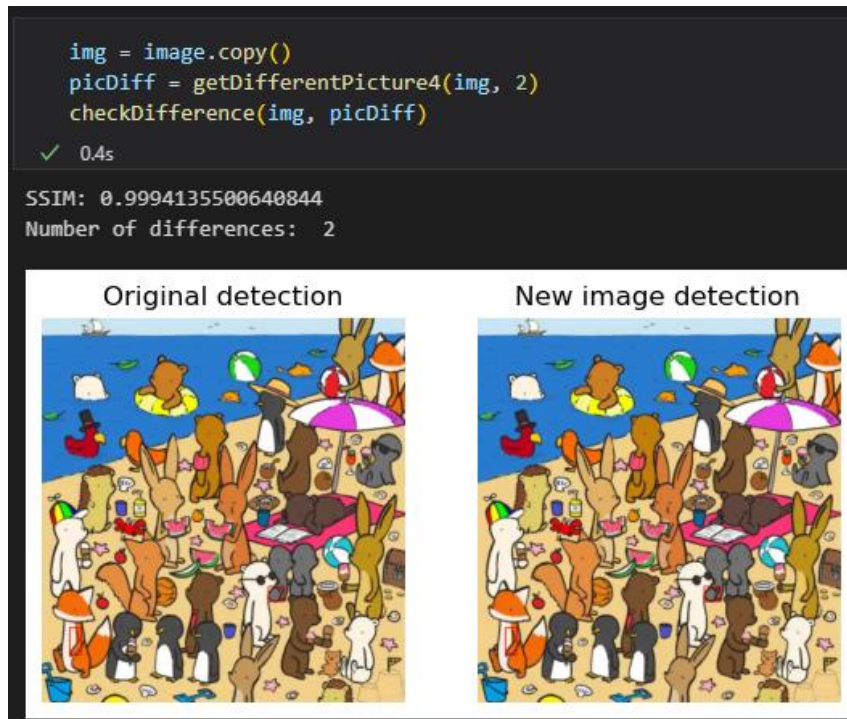
Hình 2.5.4 Kết quả Level 2



Hình 2.5.5 Kết quả Level 3



Hình 2.5.6 Kết quả Level 4 (với hình dễ xét hơn)



Hình 2.5.7 Kết quả Level 4 (với hình gốc)

III. Đánh giá:

Việc xây dựng ảnh và tìm kiếm sự khác biệt về cơ bản đã đạt đúng chỉ tiêu cũng như mục đích. Dưới đây là một số điểm em tự đánh giá quá trình thử nghiệm:

- Tại sao là *maxNumOfDif* mà không phải *numOfDif*?

→ Do ở tất cả các Level trên, việc sinh thành phần hầu như là ngẫu nhiên, nên có thể xảy ra việc contour nằm trong hoặc rất gần với contour khác, dẫn tới chỉ detect ra 1 sự khác biệt. Mặt khác, con số giới hạn diện tích *areaBoundingBox* có thể chưa phù hợp dẫn tới có thể bỏ sót một số chi tiết đã thay đổi. (thực tế đã thử và có thể số detection nhỏ hơn *maxNumOfDif*)

- Tại sao sử dụng *SSIM* thay vì trừ trực tiếp hai ảnh grayscale?

→ Theo tìm hiểu, đơn giản vì *SSIM* cho độ chính xác cao hơn. Khi ta trừ trực tiếp hai ảnh, các yếu tố như độ sáng, độ tương phản có thể ảnh hưởng tới ảnh và ảnh grayscale sẽ xét, ảnh hưởng đến kết quả phép trừ. *SSIM* tính tới các thông tin có cấu trúc (structural information), ví dụ như độ chói sáng, độ tương phản, cấu trúc,... của ảnh nên cho ra một kết quả hợp với góc độ con người hơn.

- Chưa xét đến trường hợp ảnh phức tạp, chi tiết hơn, ở 3 Level trên có thể không ảnh hưởng nhiều nhưng ở Level 4 việc detect trở nên khó khăn

→ Cần những cách xử lý khác với những loại ảnh đặc biệt hơn

IV. Tổng kết:

Báo cáo trên đã nêu ra một số phương pháp tạo ra dữ liệu ảnh cho trò chơi Spot the Difference cũng như tìm ra sự khác biệt giữa hai ảnh. Tuy nhiên, các phương pháp trên vẫn cần có nhiều yếu tố cần sửa và cải thiện để trở nên tối ưu và chính xác hơn mới có thể áp dụng vào các trường hợp thực tế.

Tổng kết lại, quá trình thực hiện báo cáo lần này đã giúp em tích lũy thêm nhiều kiến thức bổ ích về môn Xử lý ảnh cũng như biết thêm một số công cụ, cách ứng dụng và xử lý ảnh trong các trường hợp khác nhau.

V. Tham khảo:

- Slide và Code bài giảng
- <https://pyimagesearch.com/2016/02/01/opencv-center-of-contour/>
- <https://pyimagesearch.com/2017/06/19/image-difference-with-opencv-and-python/>
- https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html
- https://en.wikipedia.org/wiki/Structural_similarity
- <https://stackoverflow.com/questions/40895785/using-opencv-to-overlay-transparent-image-onto-another-image>
- <https://www.tutorialspoint.com/how-to-find-the-bounding-rectangle-of-an-image-contour-in-opencv-python>