ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO BÀI TẬP

XỬ LÝ DỮ LIỆU VÀ GIẢI THUẬT

CHO GAME TÌM ĐIỂM KHÁC BIỆT

Môn học: Xử lý ảnh

Giảng viên: Nguyễn Thị Ngọc Diệp

Họ và tên: Nguyễn Đức Hoàng Long - 20021386

HÀ NỘI 04/2023

Mục lục

Link source code: https://github.com/LnG-a/mySTDGame

1. Giới thiệu:

"Spot the Differences" là một trò chơi trực tuyến hoặc bản in trên giấy, trong đó người chơi cần tìm ra các điểm khác nhau giữa hai hình ảnh tương tự. Trò chơi này thường được chơi như một hoạt động giải trí hoặc làm tăng khả năng quan sát và tập trung. Các hình ảnh có thể được thể hiện dưới nhiều dạng, bao gồm tranh vẽ, ảnh chụp hoặc hoạt hình. Trong mỗi cặp hình ảnh, có thể có từ vài đến nhiều điểm khác nhau, và người chơi cần phát hiện chúng để chiến thắng.

Trong dự án này, ta sẽ thực hiện xử lý các bức ảnh làm đầu vào cho trò chơi nêu trên. Việc xử lý sẽ được tự động hóa bằng chương trình viết bằng mã python thay vì xử lý thủ công bằng các công cụ edit như paint hay photoshop. Đồng thời

2. Tạo dữ liệu cho game:

Đầu tiên chúng ta cần tạo dữ liệu cho game. Đầu vào của chương trình tạo ảnh là một bức ảnh và đầu ra là một bức ảnh mới với một vài điểm khác biệt so với bức ảnh đầu.

```
def level1(RES):
    # import file
    resource = cv2.imread('images/resource'+RES+'.jpg', 1)
    copy = resource.copy()
```

Ảnh 2.1. Đọc file ảnh gốc

Ở đây, ta mặc định các bức ảnh được đặt tên “resource”+số_thứ_tự.jpg, do đó khi chạy chương trình (ví dụ: level1.py), chương trình sẽ tạo ra 6 bản sao của 6 bức ảnh gốc và tự động lưu thành file ảnh có tên “copy”+số_thứ_tự+“level”+level.jpg

```
if __name__ == '__main__':
    for i in range(1, 7):
        level1(str(i))
```

Ảnh 2.2. Hàm thực thi khi chạy chương trình

Ta cũng xây dựng 3 chương trình tạo ảnh khác nhau được lưu trong file level1.py, level2.py và level3.py. Chi tiết sẽ được nêu dưới đây

2.1. Level 1: Đổi màu các khu vực ngẫu nhiên

Đầu tiên ta tạo ra các hình vuông ngẫu nhiên có cạnh dài 40px. Sử dụng hàm random_square() để tạo ra các hình vuông. Lưu ý hàm trả về một mảng của các điểm ứng với góc trái trên của mỗi hình vuông.

```
def random_square(resource):
    point_array = np.empty((0, 2))
    count = 0
    while (count < 3):
        skip = False
        x = random.randint(0, resource.shape[0]-SQUARE_SIZE)
        y = random.randint(0, resource.shape[1]-SQUARE_SIZE)
        for i in range(len(point_array)):
            if ((x < point_array[i][0]+SQUARE_SIZE and x > point_array[i]
            [0]-SQUARE_SIZE) and (y < point_array[i][1]+SQUARE_SIZE and y
            > point_array[i][1]-SQUARE_SIZE)):
                skip = True

        if not skip:
            new_array = np.array([[x, y]])
            point_array = np.concatenate((point_array, new_array), axis=0)
            count = count + 1

    return point_array
```

Ảnh 2.3. Hàm random_square

Sau khi có danh sách các điểm, ta sử dụng hàm add_color(x, y, copy). Đầu vào nhận vào tọa độ của góc trái trên hình vuông (x,y) và file ảnh cần chỉnh sửa (copy).

```
def add_color(x, y, copy):
    for i in range(x, x+SQUARE_SIZE):
        for j in range(y, y+SQUARE_SIZE):
            copy[i][j][0] = min(255, copy[i][j][0]+25)
            copy[i][j][1] = min(255, copy[i][j][1]+25)
            copy[i][j][2] = min(255, copy[i][j][2]+25)
```

Ảnh 2.4. Hàm add_color

Chúng ta dùng hàm min ở đây mục đích để giá trị B, G, R của color không vượt quá 255. Nếu vượt quá 255, màu sẽ mặc định sẽ đếm lại từ 0 và thường sẽ dẫn

đến việc sắc độ bị thay đổi quá nhiều, từ đó người chơi sẽ rất dễ dàng nhận ra điểm khác biệt.

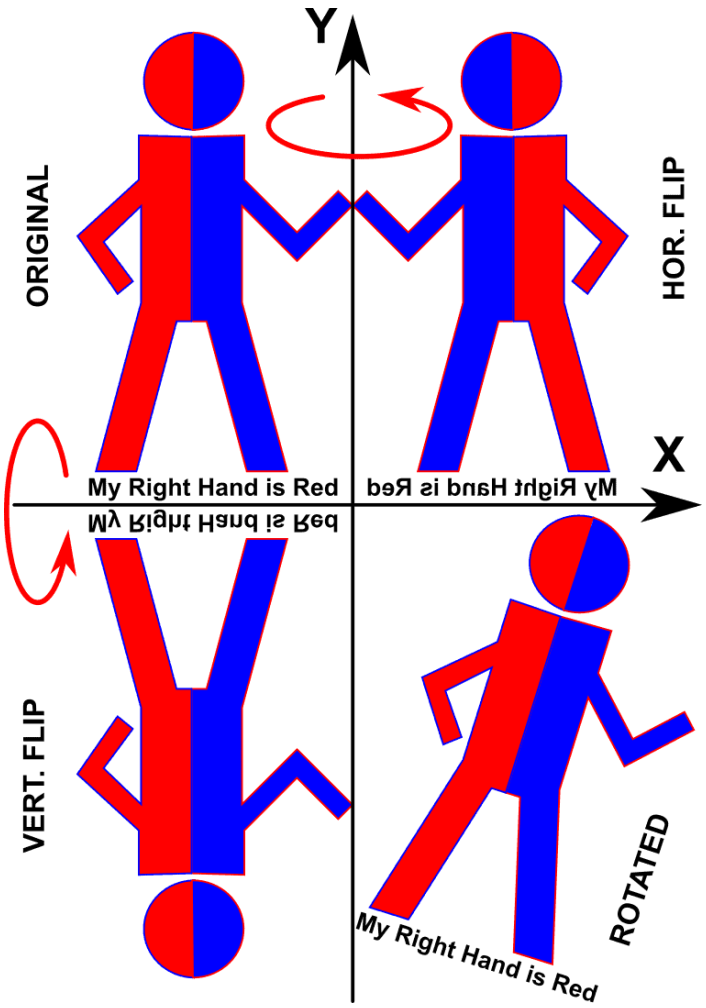
Ảnh 2.5. Kết quả khi không sử dụng hàm min

Ảnh 2.6. Kết quả khi sử dụng min (resource4.jpg)

2.2. Level 2: Flip ảnh ở các khu vực ngẫu nhiên

Sử dụng hàm random_square() đã có từ trước, ta lấy ra được một số hình vuông. Bước tiếp theo, ta có thực hiện tùy ý các thao tác thay đổi ảnh như xoay trái,

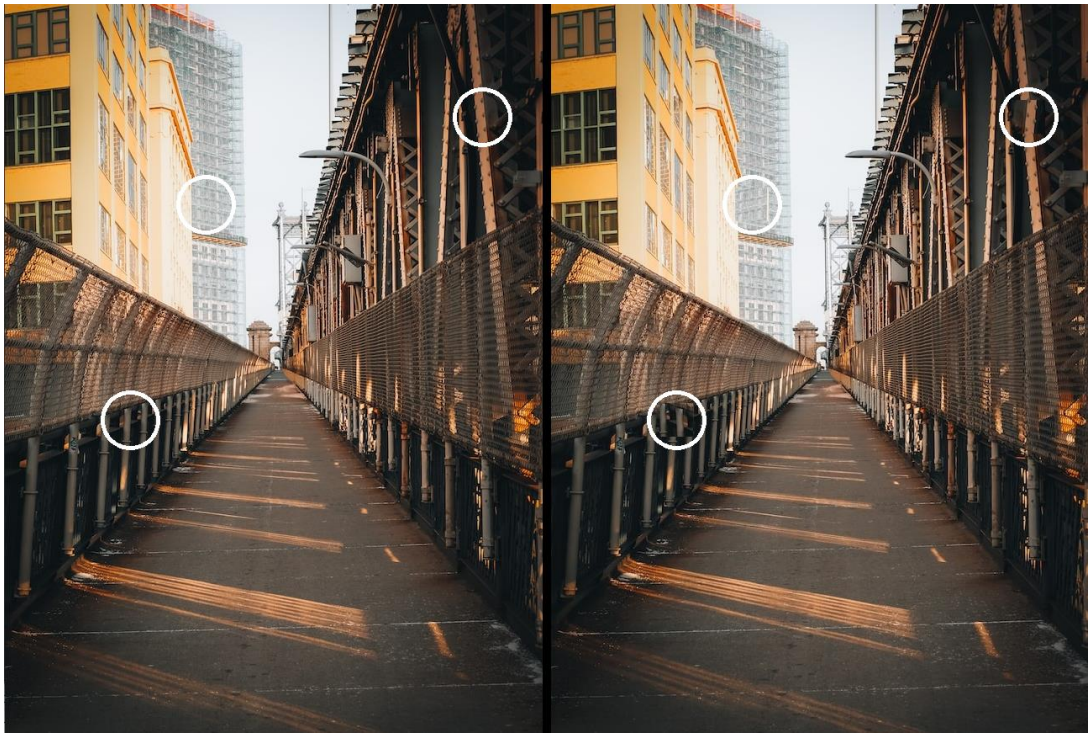
xoay phải 90º, flip vertical, flip horizontal, flip theo đường chéo,… Ở đây, ta chọn flip horizontal làm ví dụ tiêu biểu.



Ảnh 2.7 Ví dụ của một số cách biến đổi

```
def flip_horizontal(x, y, resource, copy):
    for i in range(x, x+SQUARE_SIZE):
        for j in range(y, y+SQUARE_SIZE):
            copy[i][j] = resource[i][y+SQUARE_SIZE-j+y]
```

Ảnh 2.8: Hàm flip horizontal



Ảnh 2.9. Kết quả level2 (resource6.jpg)

2.3. Level 3: Thay đổi màu vật thể

Ở level3, ta tiến hành tìm một số vật thể trong hình và biến đổi màu của chúng. Ta sử dụng Canny và findContours để tìm các vùng viền bao quanh vật thể. Tuy nhiên trước đó ta sẽ blur để giảm các chi tiết, tránh việc có quá nhiều viền dẫn đến nhiễu khi tô màu vào các vùng này.

Ảnh 2.10. So sánh giữa việc không dùng blur(bên trái) và dùng blur(bên phải)

Sau đó ta chọn một số vùng thỏa mãn các điều kiện về diện tích và diện tích chữ nhật bao quanh để tránh việc vùng bị quá nhỏ, quá to, hoặc thỏa mãn về diện tích nhưng lại trải trong một khu vực quá dài:

```
# choose contours to modify
choices = []
for contour in contours:
    area = cv2.contourArea(contour)
    x, y, w, h = cv2.boundingRect(contour)
    if (250 < area < 1000 and w*h < 2500):

        choices.append(contour)

choices = random.sample(choices, min(3, len(choices)))
```

Ảnh 2.11. Chọn vùng để tô màu

Bước cuối cùng ta sẽ tô màu, sử dụng cách thức tô ở level1, ta lấy giá trị trung bình vùng cần tô, rồi + thêm một lượng cho mỗi giá trị B, G, R, ta sẽ được một vùng có màu khá tương đầu với màu cũ.

Ảnh 2.11. Kết quả level3 (resource3.jpg)

2.4. Nhận xét

Các cách thức biến đổi ảnh ở cả 3 level đều có những ưu nhược điểm riêng. Level 1 là cách biến đổi đơn giản nhất, do đó có thể áp dụng lên bất kì loại ảnh nào,

tuy nhiên hiệu quả không thật sự cao, dễ bị phát hiện, hoặc đôi khi là quá khó để phát hiện, đến mức gần như không thể phát hiện. Ví dụ như random một hình vuông ở một vùng màu trắng, thuật toán gần như sẽ không thay đổi điều gì.

Level 2 bộc lộ điểm yếu khi sử dụng để xử lí các ảnh ,đặc biệt là tranh vẽ 2D, có vùng background rộng lớn, ví dụ như khi chọn một hình vuông ở một vùng trời toàn màu xanh, kết quả chương trình sẽ không thay đổi ảnh. Tuy nhiên, level 2 lại khá phù hợp với những hình có nhiều chi tiết rắc rối, cùng tone màu, và ảnh thực tế.

Level 3 mặc dù có thuật toán phức tạp nhất, tuy nhiên lại khó áp dụng cho ảnh thực tế, hoặc phải config nhiều những thông số để cho ra kết quả tốt. Sức mạnh thật sự của level 3 nằm ở việc xử lý các ảnh hoạt hình có nét vẽ rõ ràng, không bị đứt đoạn, và có thể tạo ra những màn chơi với độ khó vô cùng cao.

3. Giải game:

Ở phần này, ta sẽ tạo một chương trình solution.py có đầu vào là 2 ảnh và đầu ra là một ảnh ghép bởi 2 ảnh đầu vào kèm với những hình tròn khoanh vùng những điểm khác biệt.

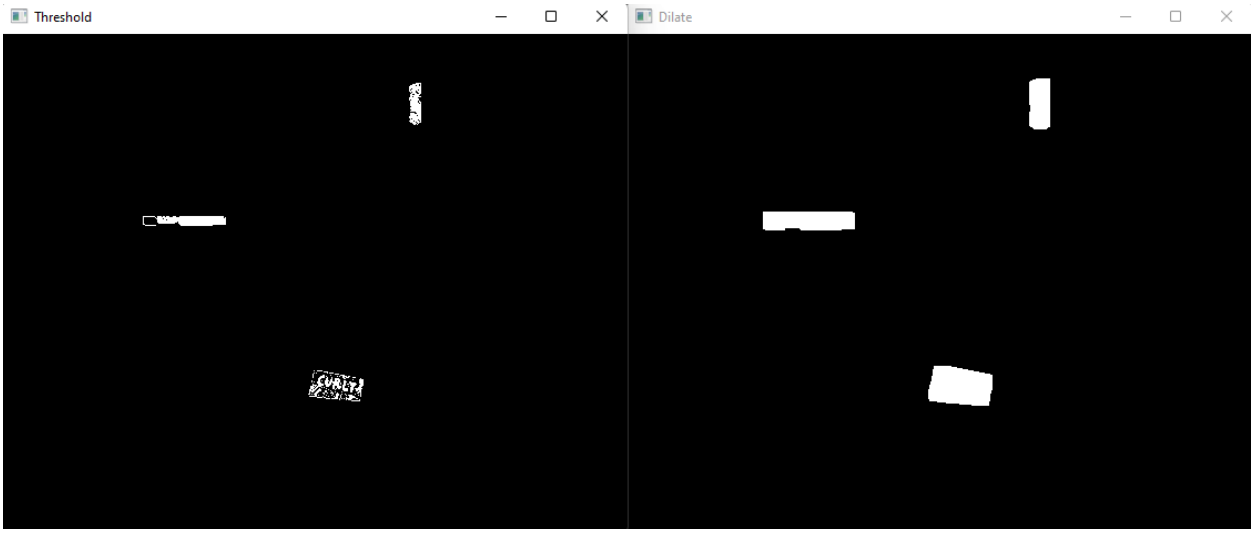
Đầu tiên ta sẽ import 2 ảnh, nhập 2 biến res và level để chọn ảnh và level cần giải. Sau đó ta biến 2 ảnh về grayscale đồng thời trừ 2 ảnh cho nhau kết quả sẽ cho ra một ảnh đen với những điểm trắng là những điểm khác biệt giưa 2 ảnh

```python
def solution():
    RES = str(3)
    LEVEL = str(3)
    # Load the two images
    img1 = cv2.imread('images/resource'+RES+'.jpg')
    img2 = cv2.imread('images/copy'+RES+'_level'+LEVEL+'.jpg')

    # Resize images if necessary
    img1 = resize_image(img1)
    img2 = resize_image(img2)

    img_height = img1.shape[0]

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

Ảnh 3.1. Trừ 2 ảnh cho nhau

Ảnh 3.2. Kết quả trừ 2 ảnh cho nhau

Sau đó ta dùng hàm threshold() của opencv2 để làm rõ các phần khác biệt này, và cuối cùng sử dụng dilate() cũng của opencv2 để hợp nhất các phần khác biệt ở gần nhau nhưng không liền với nhau (thực chất chỉ là một điểm khác biệt)



Ảnh 3.3. Sau khi được threshold (bên trái) và sau khi dilate (bên phải)

Cuối cùng ta sử dụng findContours() để khoanh vùng các điểm này sử dụng hàm boundingRect() để tìm hình chữ nhật bao quanh và một vài tính toán đơn giản để tính đường tròn bao quanh mỗi điểm khác biệt.

Ảnh 3.4. Kết quả

4. Kết luận

Trong nghiên cứu này, em đã tìm hiểu về cách tạo ra dữ liệu cho trò chơi Spot the Differences và cách xử lý để tìm điểm khác biệt giữa 2 bức ảnh. Em đã phát triển một vài phương pháp tạo ra dữ liệu bằng cách sử dụng các kỹ thuật xử lý ảnh và cố gắng tối ưu. Tuy nhiên do thời gian cũng như kiến thức có hạn, các thuật toán, phương pháp trên rõ ràng vẫn chưa được tối ưu hoàn toàn, nhưng vẫn có thể cải tiến để trở thành thuật toán phát triển cho một tựa game Spot the Differences trong tương lai.