ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Báo cáo Dự án giữa kỳ

Tạo và tìm điểm khác nhau giữa các ảnh

Sinh viên: Trương Tấn Thành

MSV: 21020095

Mục lục

I. Giới thiệu

1. Phát biểu bài toán

- Đề bài *Make "spot the difference" game data*. Tạo ra một chương trình có hai chức năng chính:

  - Một là *Make_data*: Sinh ra một bức ảnh từ một ảnh có sẵn bằng cách thay thế các vật thể, màu sắc theo mức độ phức tạp khác nhau.
  - Hai là *Detect_difference*: Chương trình tự động nhận diện các điểm khác nhau của hai bức ảnh và hiển thị các điểm khác nhau đó cho người dùng.

2. Ý tưởng thực hiện

- Phần *Make_data* sẽ sử dụng thuật toán `cv2.Canny` để lập ra các danh sách cạnh. Sau đó sẽ chọn các objects để tô màu khác vào. Việc chọn các objects từ to đến nhỏ và việc tô các màu sắc từ dễ đến khó nhận biết phụ thuộc vào mức độ khó của level.

- Phần phát hiện các điểm khác nhau giữa hai ảnh sử dụng OpenCV, Scikit-image và Python để tính toán tập các tọa độ có sự khác nhau sau đó khoanh vùng lại. Cuối cùng là highlight các phần đó cho người dùng thấy được.

II. Mô tả chi tiết dự án

1. Tạo ra bức ảnh khác từ một ảnh đầu vào

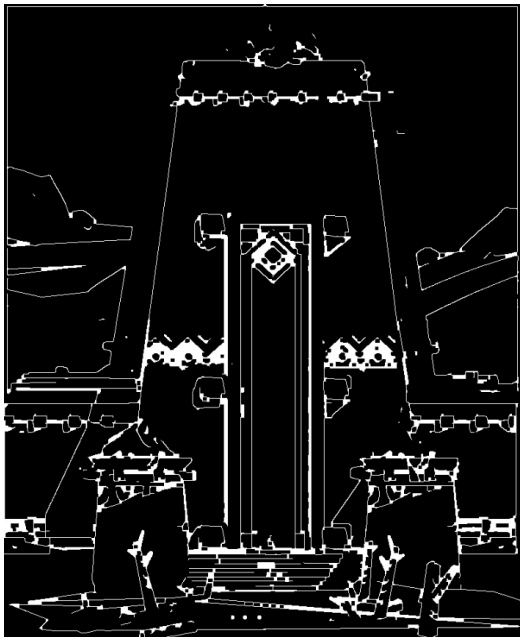
1.1. Tiền xử lý

- Hàm `edgePreprocess(img)` có chức năng chuyển hình ảnh đầu vào sang hình ảnh chỉ chứa danh sách các cạnh của các objects.

- Đầu tiên sử dụng `cv2.Candy()` để phát hiện các cạnh.
- Tiếp theo làm mịn ảnh sử dụng hai phép biến đổi là *erode* và *dilate*.

```
def edgePreprocess(img):
    imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    Temp = cv2.Canny(imgray, 150, 200)
    #normalize by smoothing and erosion + dilation
    Temp = cv2.dilate(Temp, None)
    Temp = cv2.erode(Temp, None)
    kernel_c = np.ones((5, 5), np.uint8)
    kernel_o = np.ones((1, 1), np.uint8)
    Temp = cv2.morphologyEx(Temp, cv2.MORPH_CLOSE, kernel_c)
    Temp = cv2.morphologyEx(Temp, cv2.MORPH_OPEN, kernel_o)
    return Temp
```

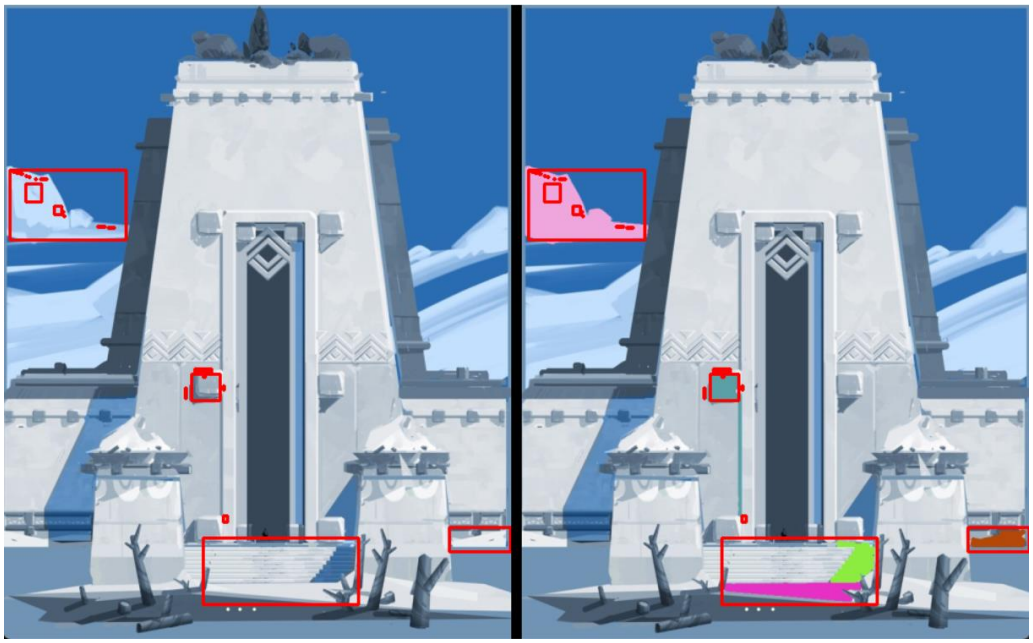
- Dưới dây là hình ảnh nhận được sau khi chạy hàm `edgePreprocess(img).`



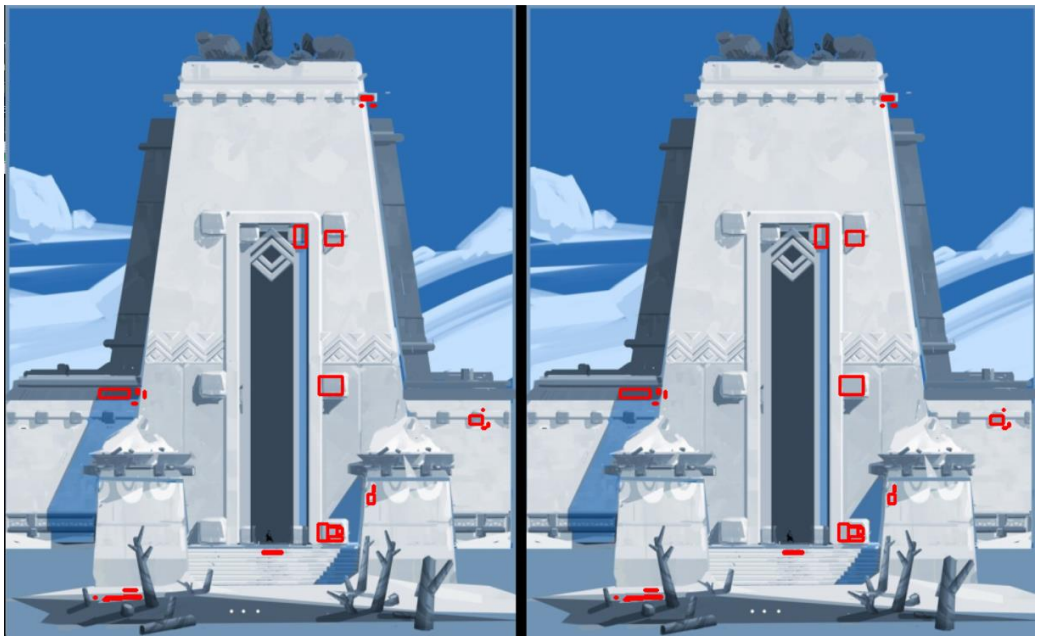
1.2. Thay đổi màu sắc các objects ngẫu nhiên

- Việc thay đổi các objects trong ảnh theo kích thước cũng như độ khó nhận diện của màu tùy theo việc cài đặt độ khó của level: easy, medium và hard.

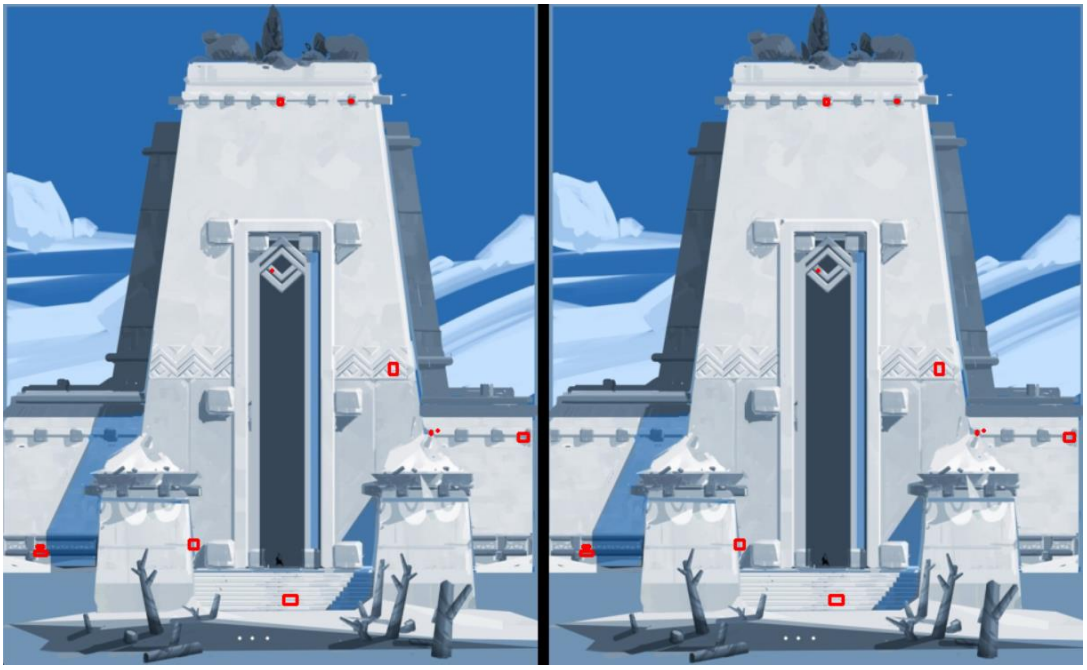
- Level dễ (easy): Objects được đổ màu ngẫu nhiên, có kích thước vừa phải.



- Level trung bình (medium): Các objects kích thước nhỏ được đổ màu theo màu nền xung quanh objects nên rất khó nhận ra bằng mắt thường.



- Level khó (hard): Các objects có kích thước rất nhỏ được chọn để đổ màu theo màu môi trường xung quanh. Rất khó để nhận diện bằng mắt thường.



- Đoạn code sau thể hiện việc lựa chọn một objects ngẫu nhiên tạm gọi là borderItem để thực hiện việc tô màu:

```
for boderItem in borderList:
        area = boderItem[1]
        al = np.random.randint(1, 100)
        if minRange <= area <= maxRange and al % 3 == 0:
            x, y, w, h = cv2.boundingRect(boderItem[0])
            R, G, B = colorArea(x, y, x+w, y+h, test)
            # print(str(R) + " " + str(G) + " " + str(B))

            if level == 'easy':
                new_color = (np.random.uniform(0, 255), np.random.uniform(0, 255), np.rand
            elif level == 'medium':
                new_color = (int(R), int(G), int(B))
            elif level == 'hard':
                new_color = (int(R), int(G), int(B))

            cv2.fillPoly(img, [boderItem[0]], new_color)
            numOfDiff = numOfDiff - 1
            if (numOfDiff == 0):
                break
```

- Đối với mức level '*easy*' thì các màu được tô ngẫu nhiên bằng hàm `np.random.uniform()`
- Còn với mức level '*medium*' và '*hard*' việc chọn các tham số để tô màu do hàm `coloArea(x1, y1, x2, y2, img)` đảm nhiệm:

```python
#calculate the average color in an area of an object
def colorArea(x1,y1,x2,y2, img):
    # Extract the pixels within the rectangular area
    roi = img[y1:y2, x1:x2]

    # Calculate the mean value of the pixel values for each color channel (R, G, B)
    mean_color = np.mean(roi, axis=(0, 1)).astype(int)

    # Print and return the mean color value as a tuple (R, G, B)
    # print(f"Mean color value: {mean_color}")
    mean_color = tuple(mean_color)
    return mean_color
```

Hàm `colorArea()` có đầu vào là vị trí của đỉnh hình chữ nhật chứa objects cần tô màu lại và đầu ra là mã màu trung bình của vùng đó. Bước một là trích xuất các pixel trong khu vực hình chữ nhật bằng cách sử dụng kỹ thuật cắt mảng NumPy. Sau đó, tính giá trị trung bình của các giá trị pixel cho từng kênh màu (R, G, B) bằng cách sử dụng hàm mean() của NumPy dọc theo các trục axis=(0, 1). Cuối cùng, in và trả về giá trị màu trung bình dưới dạng một bộ (R, G, B).

```
176 160 142
172 147 122
178 161 144
191 178 163
215 208 201
178 162 145
193 177 161
219 211 204
188 174 158
180 167 153
```

(Danh sách các bộ màu RGB)

1.3. Xuất ảnh mới tạo ra

- Ảnh gốc và ảnh sau khi được chỉnh sửa sẽ cùng được lưu lại:

```python
# Export ouput image
    cv2.imwrite('./Difference/Source.jpg', test)
    cv2.imwrite('./Difference/Replaced.jpg', img)
```

- `Source.jpg` : Là ảnh gốc chưa qua chỉnh sửa.

- `Replaced.jpg` : Là ảnh đã được chỉnh sửa bằng cách biến đổi một số objects từ ảnh đầu vào.

2. Detect và hiển thị các điểm khác nhau giữa 2 ảnh

2.1. Tiền xử lý

- Load hai ảnh được sinh ra từ quá trình *Make_data*. Sau đó resize các ảnh về kích thước hợp lý và chuyển ảnh về thang độ xám, thuận tiện cho các thao tác sau này:

```
# load the two input images
imgA = cv2.imread('./Difference/Source.jpg')
imgB = cv2.imread('./Difference/Replaced.jpg')

# resize image
imgA = imutils.resize(imgA, height=600)
imgB = imutils.resize(imgB, height=600)
img_height = imgA.shape[0]

# convert the images to grayscale
grayA = cv2.cvtColor(imgA, cv2.COLOR_BGR2GRAY)
grayB = cv2.cvtColor(imgB, cv2.COLOR_BGR2GRAY)
```

2.2. Tính toán độ khác nhau giữa hai hình ảnh

- Sử dụng hàm `compare_ssim` từ thư viện Scikit-image, chúng ta tính được 2 giá trị là score và ảnh thể hiện sự khác biệt, diff

- `score` : Thể hiện chỉ số tương đồng cấu trúc giữa hai ảnh đầu vào. Giá trị này có thể nằm trong khoảng [-1, 1]

```
SSIM: 0.9996912796906771
```

`score` càng gần 1 thì mức độ giống nhau của 2 ảnh càng lớn, đồng nghĩa với việc phát hiện bằng mắt thường càng khó.

- `diff` : Chứa sự khác nhau giữa hình ảnh ban đầu và hình ảnh được sinh ra sau quá trình *Make_data*

```
# compute the Structural Similarity Index (SSIM) between the two
# images, ensuring that the difference image is returned
(score, diff) = compare_ssim(grayA, grayB, full=True)
diff = (diff * 255).astype("uint8")
print("SSIM: {}".format(score))
```

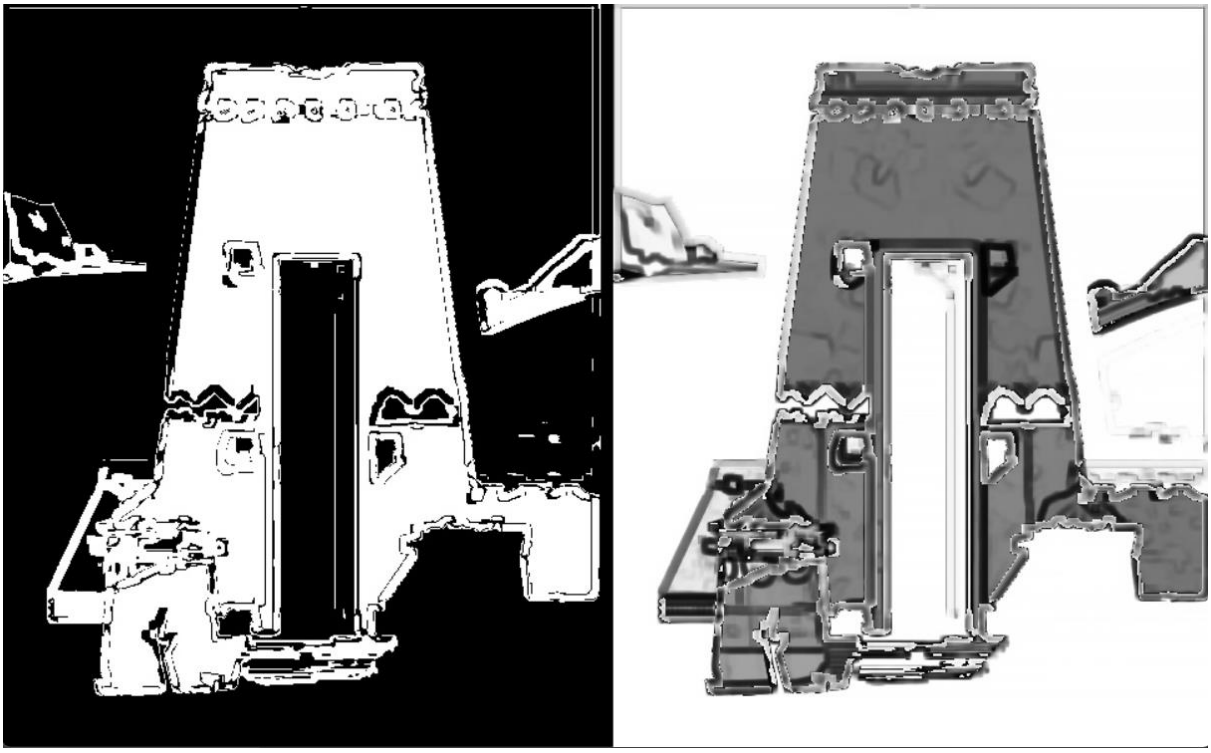
2.3. Threshold để thể hiện rõ hơn sự khác biệt và tìm đường viền khác nhau

- Áp dụng đồng thời `cv2.THRESH_BINARY_INV` và `cv2.THRESH_OTSU` để cho ra những điểm khác biệt cụ thể trong ảnh

- Tiếp theo đó là tìm danh sách cạnh của `thresh` , sau đó lưu vào một list là `cnts`

```
# threshold the difference image, followed by finding contours to
# obtain the regions of the two input images that differ
thresh = cv2.threshold(diff, 0, 255,
  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)

cv2.imwrite('./Difference/Thresh.jpg', thresh)
cv2.imwrite('./Difference/Diff.jpg', diff)
```



Kết quả sau khi threshold.

2.4. Khoanh vùng các điểm khác nhau

- Chúng ta đã có danh sách cạnh được lưu trong `cnts` . Tiếp theo sẽ lặp qua các cạnh trong `cnts` , tính toán phần giới hạn xung quanh vùng khác nhau sử dụng hàm `cv2.boundingRect` . Lưu trữ các tọa độ (x, y) để phục vụ việc vẽ phần viền giới hạn của những vùng khác nhau sau này.

- Sử dụng các giá trị (x, y) vừa lưu được để vẽ một hình chữ nhật màu đỏ trên mỗi hình ảnh với `cv2.rectangle`

```
# loop over the contours
for c in cnts:
  # compute the bounding box of the contour and then draw the
  # bounding box on both input images to represent where the two
  # images differ
  (x, y, w, h) = cv2.boundingRect(c)
  cv2.rectangle(imgA, (x, y), (x + w, y + h), (0, 0, 255), 2)
  cv2.rectangle(imgB, (x, y), (x + w, y + h), (0, 0, 255), 2)
```

2.5. Hiển thị các ảnh đã được khoanh vùng lên màn hình

- Sử dụng hàm `np.hstack()` để ghép các cặp ảnh rồi dùng `cv2.imshow()` để hiển thị ảnh lên màn hình.

```
source = np.hstack((imgA, x, imgB))
difference = np.hstack((imgThresh, x, imgDiff))
cv2.imshow("Difference", difference)
cv2.imshow("Source", source)
```