ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO BÀI TẬP

XỬ LÝ ẢNH

Project: Spot the Differences Game and Creator Data

Giảng viên:
TS. Nguyễn Thị Ngọc Diệp
Lớp: INT3404E 21

Sinh viên:
Nguyễn Đình Thành Đạt - 21021476

HÀ NỘI – 2023

MỤC LỤC

Source code: Github

1.Giới thiệu:

Spot the Difference là một trò chơi giải đố tìm điểm khác biệt giữa hai hình ảnh giống nhau. Trò chơi bao gồm hai bức tranh được hiển thị cùng một lúc và yêu cầu người chơi tìm ra và đánh dấu các điểm khác nhau giữa hai bức tranh đó. Thường thì trò chơi này sẽ có thời gian giới hạn, và người chơi sẽ phải tìm ra tất cả các điểm khác nhau trước khi thời gian kết thúc.

Spot the Difference là một trò chơi giải đố đơn giản và phổ biến, được chơi trên nhiều nền tảng khác nhau, từ sách tập hình, tạp chí, đến các trò chơi trực tuyến và ứng dụng trên điện thoại di động. Trò chơi này không chỉ giúp người chơi giải trí và giảm căng thẳng mà còn giúp cải thiện khả năng quan sát, tập trung và tư duy logic.

1.1.Mục đích

Để tạo ra dữ liệu cho trò chơi một cách thủ công thì thường khá tốn thời gian và công sức, đòi hỏi nhiều kỹ năng để từ bức ảnh ban đầu ta có thể thiết kế ra bức ảnh thứ hai với một vài điểm khác biệt. Sau khi trải qua giai đoạn tạo dữ liệu cho trò chơi, việc tiếp theo chính là làm sao để có thể tìm ra được những khu vực có điểm khác biệt ấy.

Báo cáo này gồm hai mục đích chính. Thứ nhất, từ một bức ảnh đầu vào, ta có thể tạo ra dữ liệu cho trò chơi với nhiều độ khó khác nhau bằng cách áp dụng các thuật toán xử lý ảnh như: đổi màu vùng ảnh, đổi màu cạnh, đổi màu vật thể, … Thứ hai, sau khi tạo ra được dữ liệu, ta sử dụng dữ liệu ấy để giải trò chơi với các thuật toán xử lý ảnh.

2. Task 1: Make “Spot the Differences” game data

2.1. Vấn đề

Như đã nêu ở trên, vấn đề của nhiệm vụ thứ nhất chính là ta có thể tự sinh ra ảnh thứ hai với một vài điểm khác biệt so với ảnh ban đầu. Ta cần thiết kế thuật toán sao cho khi áp dụng với bất cứ bức ảnh nào đều sẽ tự động sinh ra được ảnh thứ hai mà không cần xử lý cho từng trường hợp riêng biệt. Do kết cấu, đặc điểm của từng bức ảnh là khác nhau, việc áp dụng một thuật toán cho tất cả có thể khiến ta nhận được đầu ra không đạt mong muốn. Tuy nhiên, các thuật toán được áp dụng sao cho có thể có độ hiệu quả nhất định.

2.2. Các phương pháp

2.2.1. Đổi màu ngẫu nhiên các vùng ảnh

Cách đơn giản nhất ta có thể áp dụng đó chính là đổi màu các vùng ngẫu nhiên. Các vùng này là các hình chữ nhật có kích thước và màu sắc khác nhau, được sinh ra bằng thuật toán ngẫu nhiên và không bị trùng lặp. Bằng cách này, cách vùng được lựa chọn sẽ không chồng chất lên nhau mà được phân tán ra toàn bộ ảnh.

- Thuật toán sinh ngẫu nhiên:

```python
def generate_rects(img, num_of_rects, min_h=MIN_H, max_h=MAX_H, min_w=MIN_W, max_w=MAX_W):
    '''
    This function generate random rectangles
    :param img: input image
    :param num_of_rects: how many rects to generate
    :param min_h: minimum height of rect
    :param max_h: maximum height of rect
    :param min_w: minimum width of rect
    :param max_w: maximum width of rect
    :return: a list of random rects
    '''
    rects = []
    shape = img.shape
    height, width = shape[0], shape[1]
    while len(rects) < num_of_rects:
        x = random.randint(0, width - max_w)
        y = random.randint(0, height - max_h)
        w = random.randint(min_w, max_w)
        h = random.randint(min_h, max_h)

        if not overlap_rect(x, y, w, h, rects):
            rects.append((x, y, x + w, y + h))

    return rects
```

Hình 2.1. *Hàm generate_rects trong create_data/utils.py*

Thuật toán trên sẽ sinh ra một số lượng nhất định hình chữ nhật đã được định. Mỗi hình chữ nhật được sinh ra bằng cách random tọa độ (x, y) là điểm trên cùng bên trái của hình chữ nhật cùng với chiều dài (h), chiều rộng (w) sao cho hình chữ nhật có thể lọt vừa vào trong bức ảnh. Thêm vào đó, mỗi hình chữ nhật khi sinh ra sẽ có hàm để kiểm tra có trùng lặp hay không (hình này đè lên hình khác), nếu không bị trùng lặp thì hình chữ nhật đó được tính và thêm vào danh sách. Hàm trên sau khi chạy sẽ trả về một danh sách hình chữ nhật.

- Hàm kiểm tra trùng lặp:

```
32  def overlap_rect(x, y, w, h, rects):
33      '''
34      Check if the input (x, y, w, h) are overlap with rect in rects
35      :param x: x-coordinate
36      :param y: y coordinate
37      :param w: width of the rect
38      :param h: height of the rect
39      :param rects: a list of rectangle
40      :return: if the new rect are not overlap with any rect in rects, return False
41      '''
42      if len(rects) == 0:
43          return False
44      for rect in rects:
45          if (x + w < rect[0] or y + h < rect[1]
46                  or rect[2] < x or rect[3] < y):
47              return False
48
49      return True
```

Hình 2.2. *Hàm overlap_rect trong create_data/utils.py*

Hàm này kiểm tra hình chữ nhật vừa mới được sinh ra có bị trùng lặp hay không bằng cách so sánh tọa độ của nó với mỗi hình chữ nhật đã được sinh ra trước đó. Để kiểm tra, ta sẽ có 4 trường hợp nếu không bị trùng lặp: hình mới nằm hẳn bên trái hoặc bên phải, nằm hẳn bên trên hoặc bên dưới hình cũ.

- Hàm vẽ các hình chữ nhật được sinh ra lên trên ảnh bằng màu ngẫu nhiên:

```python
def draw_random_rectangle(img, num_changes):
    '''
    Draw ramdom rects into the input image with random color
    :param img: input image
    :param num_changes: number of rects to draw
    :return: a modified image
    '''
    copy_img = img.copy()
    rects = generate_rects(img, num_changes)

    # draw rect to img
    for rect in rects:
        # random color
        color = generate_color()

        cv.rectangle(copy_img, (rect[0], rect[1]), (rect[2], rect[3]), color[::-1], cv.FILLED)

    return copy_img
```

Hình 2.3. *Hàm draw_random_rectangle() trong create_data/modify_img.py*

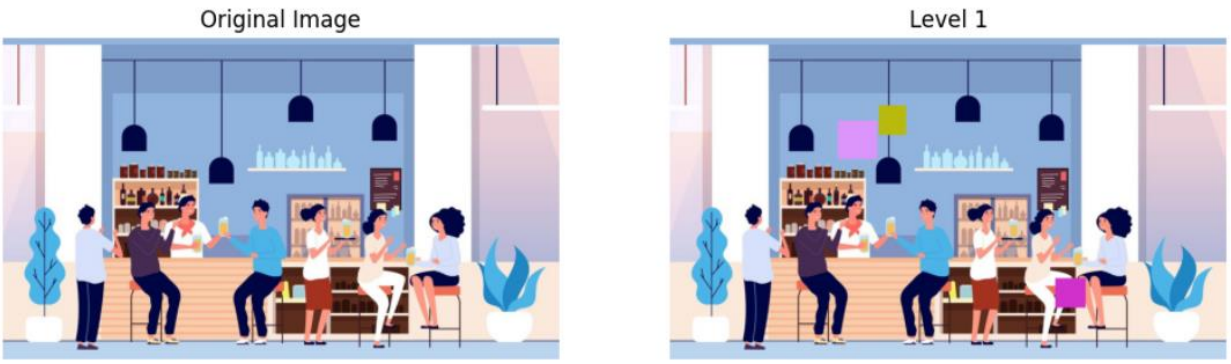
Với mỗi hình chữ nhật trong danh sách, ta sẽ vẽ nó lên bức ảnh với màu ngẫu nhiên bằng hàm cv.rectangle(). Hàm trả lại bức ảnh đã được sau khi được vẽ màu.

- Hàm sinh ra màu ngẫu nhiên:

```python
def generate_color():
    '''
    Generate random RGB color
    :return: a tuple of (red, green, blue) value
    '''
    r = random.randrange(255)
    g = random.randrange(255)
    b = random.randrange(255)
    return r, g, b
    # return (255, 0, 0)
```

Hình 2.4. *Hàm generate_color() trong create_data/utils.py*

Hàm sinh ra màu ngẫu nhiên trong không gian màu RGB.



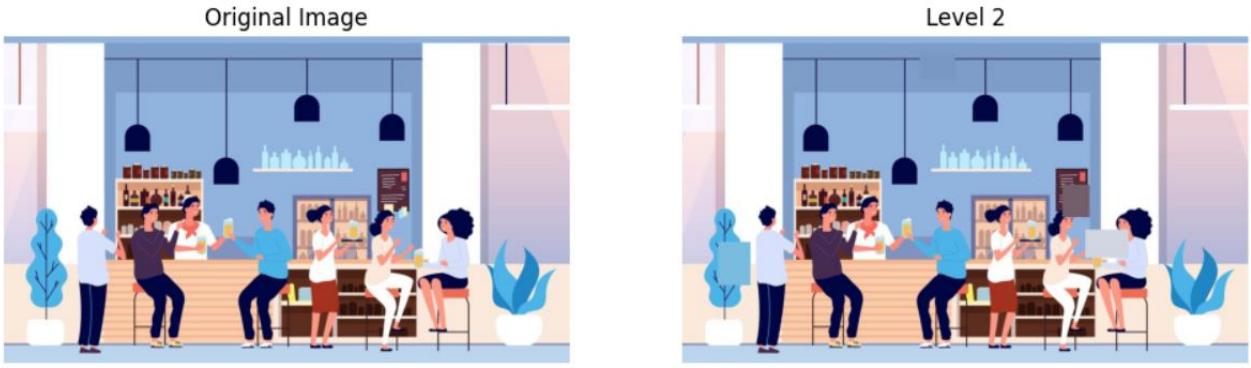
Hình 2.5. Ảnh gốc và ảnh sau khi vẽ màu ngẫu nhiên

- Ngoài vẽ màu ngẫu nhiên, ta còn có thể lấy màu chủ đạo của vùng được vẽ để khiến nó khó nhận ra hơn.

```
29  def draw_avg_rectangle(img, num_changes):
30      '''
31      Draw ramdom rects into the input image with the average color of the area
32      :param img: input image
33      :param num_changes: number of rects to draw
34      :return: a modified image
35      '''
36      copy_img = img.copy()
37      rects = generate_rects(img, num_changes)
38
39      # draw rect to img
40      for rect in rects:
41          # get the avg color of the area
42          xA, yA = rect[0], rect[1]
43          xB, yB = rect[2], rect[3]
44          avg_color_row = np.average(img[yA:yB, xA:xB], axis=0)
45          avg_color = np.average(avg_color_row, axis=0)
46
47          cv.rectangle(copy_img, (xA, yA), (xB, yB), avg_color, cv.FILLED)
48
49      return copy_img
50
```

Hình 2.6. *Hàm draw_avg_rectangle() trong create_data/modify_img.py*

Ta vẽ tương tự như ở trên nhưng khác ở cách lấy màu.



Hình 2.7. *Ảnh gốc và ảnh sau khi vẽ màu*

2.2.2. Lật ngẫu nhiên các vùng ảnh:

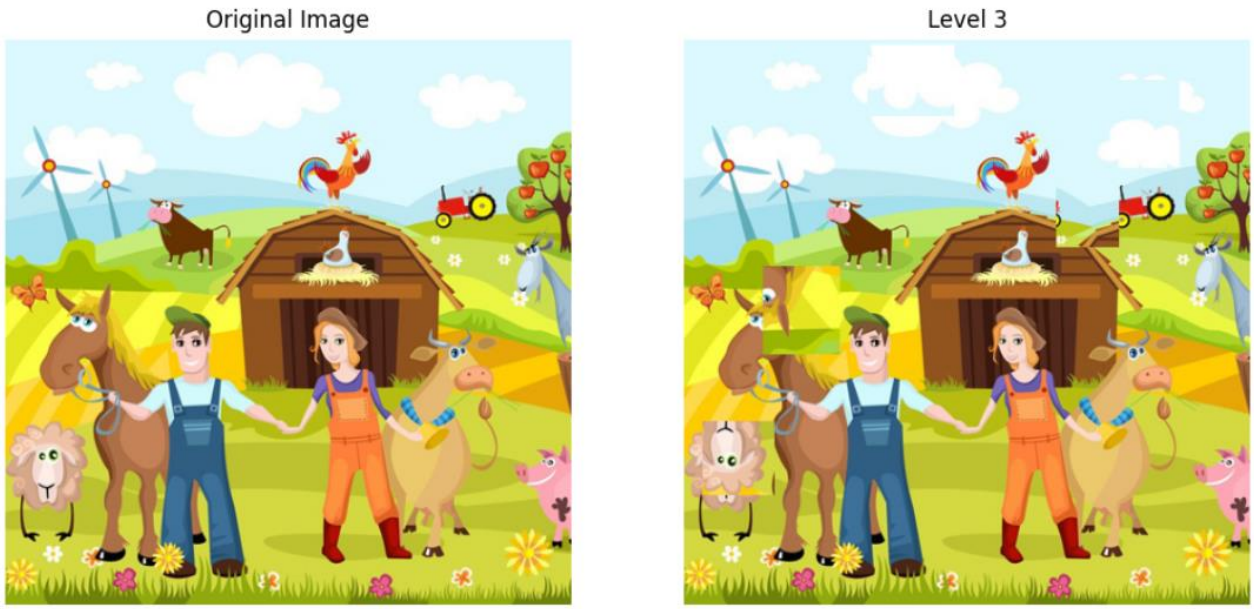
Cách tiếp theo chúng ta có thể áp dụng chính là chọn ngẫu nhiên vùng trong ảnh và tiến hành lật nó theo chiều ngang hoặc chiều dọc hoặc cả hai chiều.

- Hàm lật vùng ảnh:

```
52  def flip_rectangle(img, num_changes):
53      '''
54      Flip some area by vertically or horizontally or both
55      :param img: input image
56      :param num_changes: number of areas to flip
57      :return: a modified image
58      '''
59      copy_img = img.copy()
60      rects = generate_rects(img, num_changes)
61
62      # draw rect to img
63      for rect in rects:
64          # area to flip
65          xA, yA = rect[0], rect[1]
66          xB, yB = rect[2], rect[3]
67          area = img[yA:yB, xA:xB]
68
69          choices = [1, 0, -1]
70          # 1: flip horizontally
71          # 0: flip vertically
72          # -1: flip both horizontally and vertically
73
74          flip_direction = random.choice(choices)
75          copy_img[yA:yB, xA:xB] = cv.flip(area, flip_direction)
76
77      return copy_img
```

Hình 2.8. *Hàm flip_rectangle() trong create_data/modify_img.py*

Hàm nhận vào vùng ảnh cần lật là một hình chữ nhật, sau đó tiến hành lật theo chiều ngang hoặc chiều dọc hoặc cả hai với hàm cv.flip(). Hàm trả về ảnh với vùng ảnh đã được lật.



Hình 2.9. *Ảnh gốc và ảnh sau khi được lật vùng ngẫu nhiên*

2.2.3. Đổi màu viền của vật thể

Thuật toán tiếp theo để có thể tạo điểm khác biệt chính là đổi màu viền của những vật thể ngẫu nhiên trong ảnh.

- Hàm tìm đường viền của các vật thể trong ảnh:

```
def _find_contours(img, min_area, max_area):
    '''
    find the contour of the image, filter it by contour's area
    :param img: input image
    :param min_area: minimum contour's area
    :param max_area: maximum contour's area
    :return: a list of filtered contours
    '''
    copy_img = img.copy()

    gray_img = cv.cvtColor(copy_img, cv.COLOR_BGR2GRAY)

    # Detect edges with Canny
    canny_img = cv.Canny(gray_img, 30, 250)

    contours, hierarchy = cv.findContours(canny_img.copy(), cv.RETR_TREE, cv.CHAIN_APPROX_NONE)

    filtered_contours = []
    for contour in contours:
        if min_area < cv.contourArea(contour) < max_area:
            filtered_contours.append(contour)

    return filtered_contours
```

Hình 2.10. *Hàm _find_contours() trong create_data/modify_img.py*

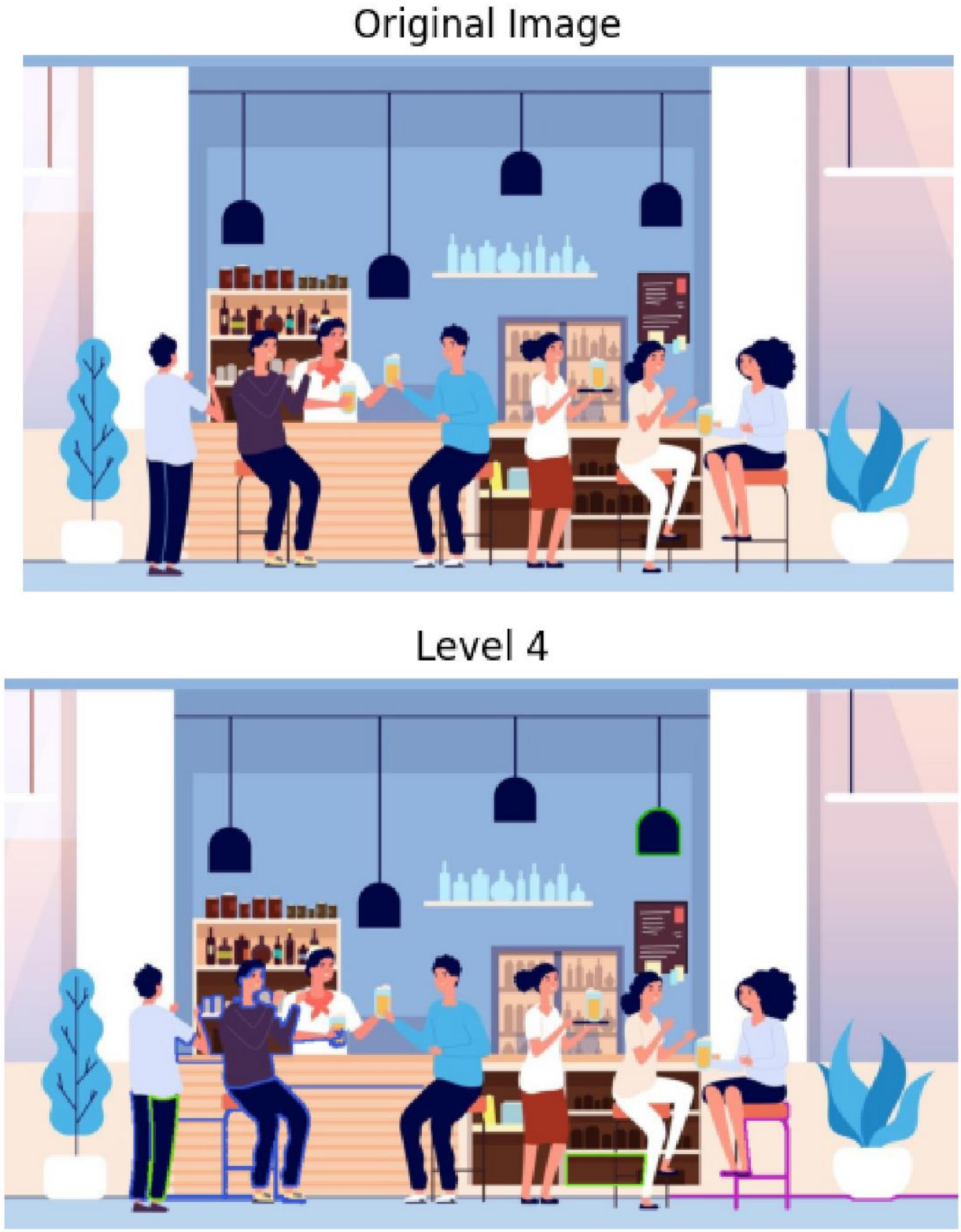
Hàm nhận nhận đầu vào là bức ảnh và minimum, maximum của đường viền đó (đường viền gồm bao nhiêu pixel). Đầu tiên, hình được chuyển sang thang xám bằng hàm cv.cvtColor với thông số cv.COLOR_BGR2GRAY. Sau đó, ta sử dụng hàm cv.Canny() để tìm cạnh của các vật trong ảnh. Tiếp theo là dùng hàm cv.findContours để tìm các đường viền của cạnh. Cuối cùng ta lọc các cạnh nằm trong khoảng yêu cầu với hàm tính diện dích đường viền cv.contourArea() và trả về danh sách các đường viền đã được lọc.

- Hàm đổi màu các đường viền:

```
105  def change_edge_color(img, num_changes):
106      '''
107      Change random edge's color
108      :param img: input image
109      :param num_changes: number of edges to change color
110      :return: a modified image
111      '''
112      copy_img = img.copy()
113      contours = _find_contours(copy_img, MIN_AREA, MAX_AREA)
114
115      # Choose random contours to change
116      if len(contours) <= num_changes:
117          selected = list(range(len(contours)))
118      else:
119          selected = random.sample(range(len(contours)), num_changes)
120
121      print(len(selected))
122
123      for i in range(len(selected)):
124          # random color
125          color = generate_color()
126
127          cv.drawContours(copy_img, contours, selected[i], color[::-1], 2)
128
129      return copy_img
```

Hình 2.11. *Hàm change_edge_color() trong create_data/modify_img.py*

Ta sử dụng hàm _find_contours() ở trên để tìm các đường viền phù hợp, sau đó lựa chọn random trong số đó để tiến hành đổi màu. Với từng đường viền đã được chọn, ta tiến hành vẽ lên màu ngẫu nhiên cho nó. Màu sắc được chọn từ hàm generate_color đã nói ở trên.



Hình 2.12. *Ảnh gốc và ảnh sau khi được đổi màu viền*

2.2.4. Đổi màu sắc của vật thể

Cách tiếp theo mà ta có thể sử dụng đó là đổi màu của vật thể dựa vào đường viền tìm được với hàm _find_contours() đã nói ở trên.

- Hàm đổi màu vật thể ngẫu nhiên:

```
132  def change_object_color(img, num_changes):
133      '''
134      Change the color of an object (area) in the image
135      :param img: input image
136      :param num_changes: number of areas to change
137      :return: a modified image
138      '''
139      copy_img = img.copy()
140      contours = _find_contours(copy_img, MIN_POLY_AREA, MAX_POLY_AREA)
141
142      # Choose random contours to change
143      if len(contours) <= num_changes:
144          selected = list(range(len(contours)))
145      else:
146          selected = random.sample(range(len(contours)), num_changes)
147
148      print(len(selected))
149
150      for i in range(len(selected)):
151          # random color
152          color = generate_color()
153
154          # area = [contours[selected[i]]]
155          cv.fillPoly(copy_img, [contours[selected[i]]], color[::-1])
156
157      return copy_img
```

Hình 2.13. *Hàm change_object_color() trong create_data/modify_img.py*

Ta bắt đầu tương tự như đổi màu viền bằng cách sử dụng hàm _find_contours() với tham số MIN_POLY_AREA, MAX_POLY_AREA (minimum và maximum của diện tích vật thể mà đường viền bao quanh). Tiếp theo ta cũng lựa chọn các đường viền ngẫu nhiên để tiến hành vẽ màu lên vật thể được bao bởi đường viền đó. Với từng đường viền được lựa chọn, ta sử dụng hàm cv.fillPoly() để vẽ màu vào bên trong đường viền với màu sắc được sinh từ hàm generate_color() ở trên.

Hình 2.14. *Ảnh gốc và ảnh sau khi được đổi màu vật thể ngẫu nhiên*

3. Task 2: Solve game in task 1

Để phát hiện điểm khác biệt, đầu tiên ta có thể nghĩ tới trừ hai ảnh cho nhau, sau đó tiếp tục sử dụng các kỹ thuật xử lý ảnh để có thể xác định được vùng có điểm khác biệt.

- Hàm tìm điểm khác biệt:

```python
def spot_the_difference(original_img, modified_img):
    original_img_gray = cv.cvtColor(original_img, cv.COLOR_BGR2GRAY)
    modified_img_gray = cv.cvtColor(modified_img, cv.COLOR_BGR2GRAY)

    # Calc the difference
    diff = cv.absdiff(original_img_gray, modified_img_gray)

    # thresh to binary image
    thresh = cv.threshold(diff, 0, 255, cv.THRESH_BINARY + cv.THRESH_OTSU)

    # Dilate to reduce small noise
    kernel = cv.getStructuringElement(cv.MORPH_DILATE, (5, 5))
    dilate_img = cv.dilate(diff, kernel, iterations=2)

    # Find contour of the differences
    contours, hierarchy = cv.findContours(dilate_img, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)

    # Draw bounder around the contours
    for contour in contours:
        if cv.contourArea(contour) > MIN_DIFF_AREA:
            x, y, w, h = cv.boundingRect(contour)
            cv.rectangle(modified_img, (x, y), (x + w, y + h), SPOT_COLOR[::-1], 3)

    return modified_img
```

Hình 3.1. *Hàm spot_the_difference() trong spot_difference/spot_diff.py*

Đầu tiên ta sử dụng hàm cv.cvtColor() để chuyển hai ảnh đầu vào sang gray scale, ngay sau đó ta có thể trừ hai ảnh cho nhau bằng cách dùng hàm cv.asbdiff(). Khi đã có ảnh diff chứa những vùng khác biệt, ta sẽ threshold bằng hàm cv.threshold với cv.THRESH_BINARY (chuyển sang ảnh nhị phân) và cv.THRESH_OTSU (tự động tính thresh value) để biến nó thành ảnh nhị phân.

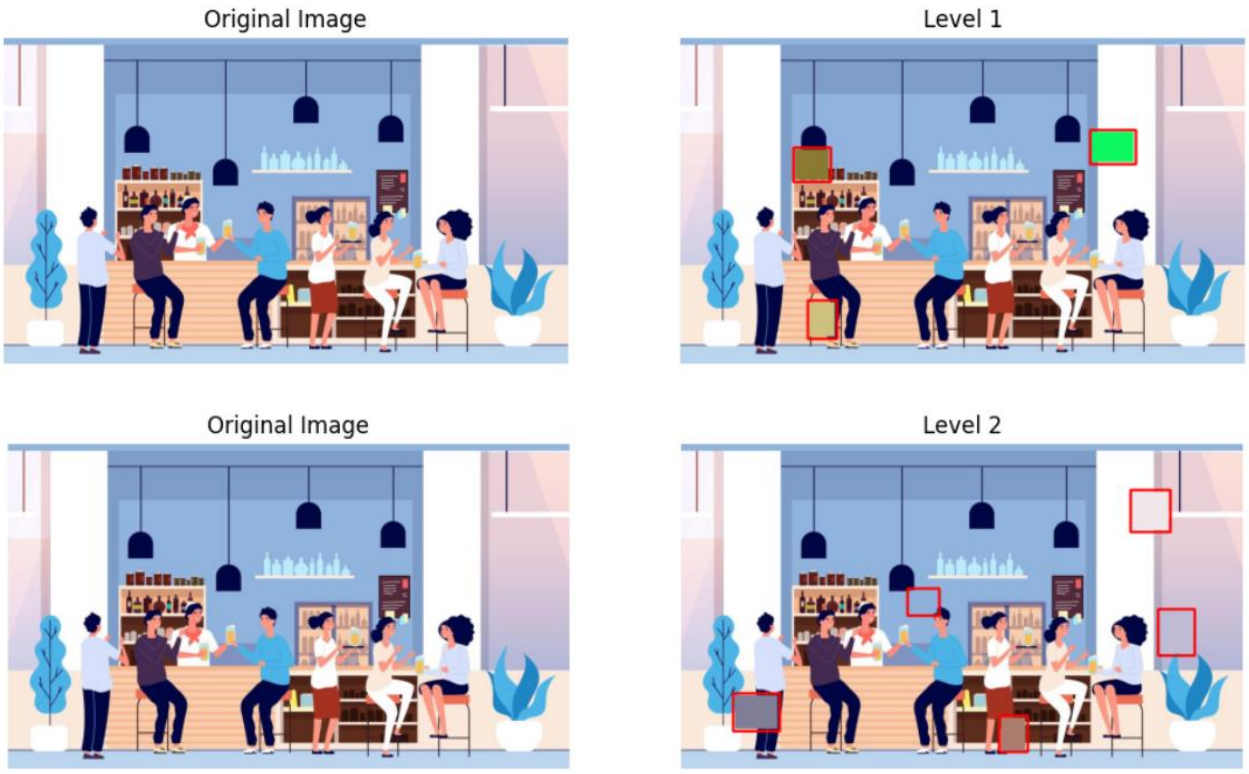
Tiếp theo, chúng ta tiến hành dilate ảnh nhị phân này để bỏ bớt đi các điểm nhiễu nhỏ như sau:

- Tạo kernel cho hàm dilate() bằng cv.getStructuringElement() với tham số cv.MORPH_DILATE và kernel size bằng 5.
- Tiến hành dilate ảnh nhị phân với hàm dilate() và thực hiện hai lần (iterations=2) để đảm bảo các điểm nhiễu được xóa bỏ.

Sau khi đã lọc điểm nhiễu, ta tìm các đường viền của vùng khác biệt bằng hàm findContours(). Với từng đường viền tìm được, ta tiến hành so sánh nó với kích cỡ của nhỏ nhất của vùng khác biệt để lọc ra sai số (có thể do các điểm nhiễu chưa được xóa đi hoàn toàn). Cuối cùng là vẽ hình chữ nhật để đánh dấu vùng khác biệt như sau:

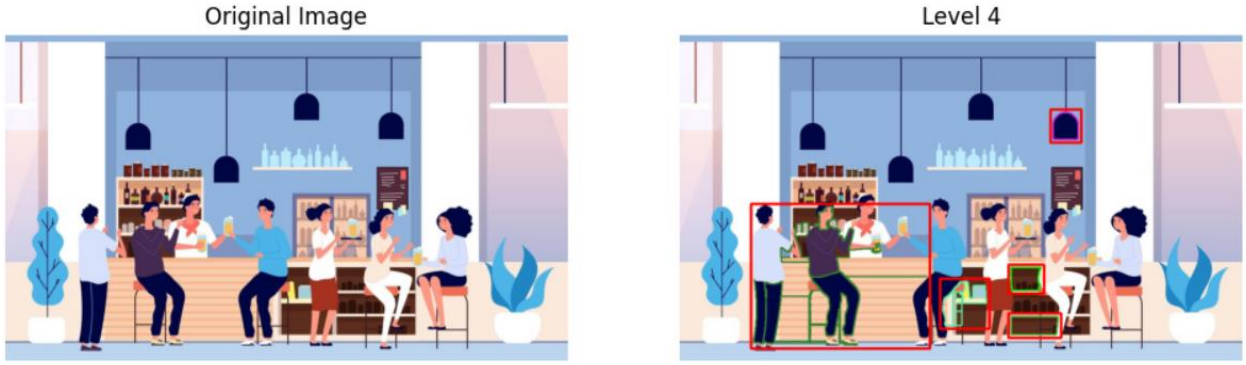
- Sử dụng hàm cv.boundingRect() để tìm hình chữ nhật bao quanh vùng khác biệt đó.
- Tiếp theo dùng hàm cv.rectangle để vẽ đánh dấu vùng khác biệt.

Sau đây là kết quả của hàm tìm khác biệt áp dụng cho từng mức độ thay đổi trong mục 2.2 ở trên.

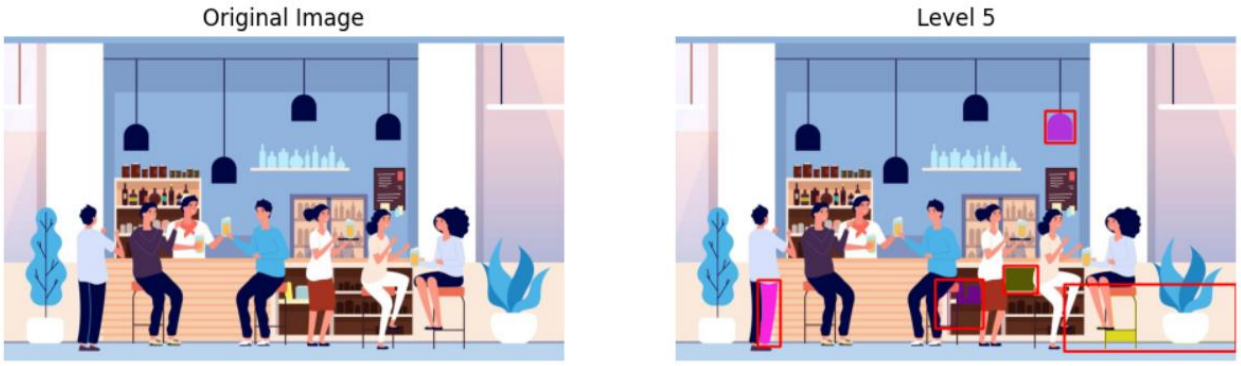


Hình 3.2. *Áp dụng tìm điểm khác biệt cho phương pháp đổi màu ở mục 2.2.1.*

Hình 3.3. *Áp dụng tìm điểm khác biệt cho phương pháp lật vùng ảnh ở mục 2.2.2.*



Hình 3.4. *Áp dụng tìm điểm khác biệt cho phương pháp đổi màu viền ở mục 2.2.3.*



Hình 3.5. *Áp dụng tìm điển khác biệt cho phương pháp đổi màu vật thể ở mục 2.2.4.*

4. Kết luận

Bản báo cáo đã đưa ra một số phương pháp áp dụng các thuật toán và kỹ thuật xử lý ảnh vào việc sinh ra dữ liệu và giải trò chơi “Spot the Differences”. Tuy nhiên, các phương pháp này vẫn còn có thể cải tiến để hiệu quả hơn và có thể sử dụng cho tập dữ liệu ảnh đầu vào đa dạng hơn. Qua bản báo cáo bài tập lớn này, tôi hiểu hơn về những kỹ thuật xử lý ảnh và cách đưa chúng áp dụng vào thực tiễn.

5. Nguồn tham khảo

- Slide và source code của lớp học.
- OpenCV documentation.
- Youtube video.