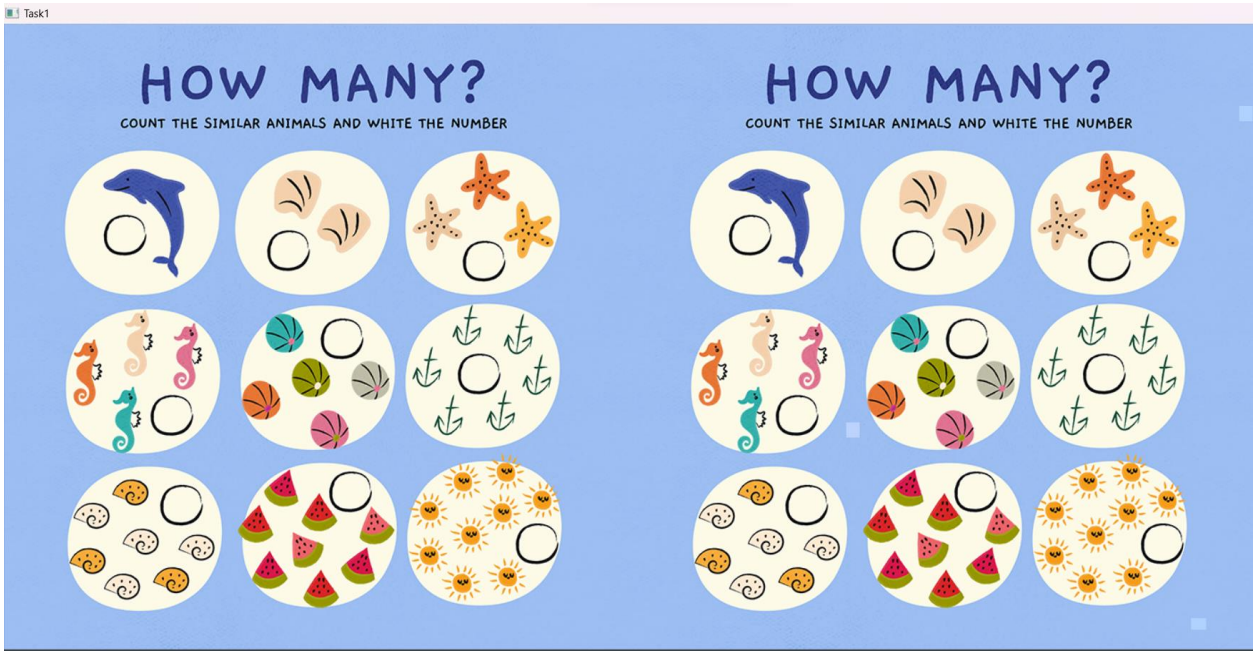
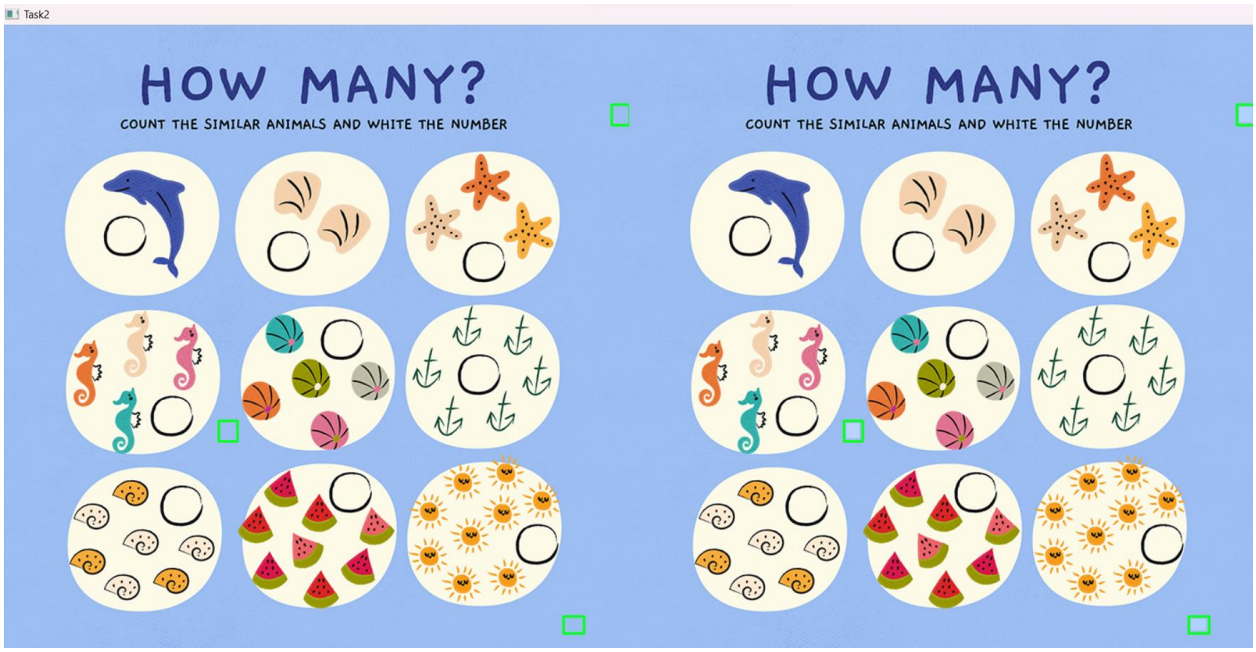
Make “spot the difference” game data

Level 1: Task 1 + Task 2

Mô tả: Vẽ ngẫu nhiên 1 hình chữ nhật nhỏ tại vị trí ngẫu nhiên trong ảnh, màu sắc hình chữ nhật có nét tương đồng với màu sắc xung quanh nó



Task1



Task2

Cách làm:

1. Người dùng chọn tệp hình ảnh (.jpg, .jpeg hoặc .png) bằng hộp thoại tệp
2. Đọc hình ảnh đã chọn và tạo một bản sao
3. Tạo 3 hình chữ nhật ngẫu nhiên với vị trí, kích thước và màu sắc ngẫu nhiên, được vẽ trên hình ảnh đã sao chép
4. Lưu hình ảnh đã sửa đổi với tên "_output_level1" được thêm vào tệp của nó trong cùng thư mục với hình ảnh gốc
5. Chuyển đổi hình ảnh sang thang độ xám
6. Tính toán chỉ số tương đồng về cấu trúc (SSIM) giữa ảnh gốc và ảnh đã sửa đổi.
7. Tính toán hình ảnh khác biệt giữa hình ảnh gốc và hình ảnh đã sửa đổi, đồng thời đặt ngưỡng cho hình ảnh đó để thu được hình ảnh nhị phân về sự khác biệt
8. Tìm các đường viền trong ảnh nhị phân và vẽ các hình chữ nhật xung quanh các vùng khác biệt được phát hiện trong ảnh gốc.
9. Hiển thị chỉ mục SSIM và hình ảnh đã sửa đổi với các điểm khác biệt được vẽ xung quanh.

Yêu cầu nhập một số thư viện:

1. skipage.metrics.structural_similarity từ thư viện skipage để tính chỉ số SSIM.
2. Tk và askopenfilename từ mô-đun tkinter.filedialog để mở hộp thoại tệp và cho phép người dùng chọn tệp hình ảnh.
3. cv2 từ thư viện opencv-python để đọc, sửa đổi và ghi các tệp hình ảnh, tính toán hình ảnh khác biệt và phát hiện các đường viền.
4. numpy để tạo số nguyên ngẫu nhiên và thực hiện các thao tác trên mảng
5. PIL.Image từ thư viện Gối để trích xuất tên tệp và đường dẫn thư mục của hình ảnh đầu vào.

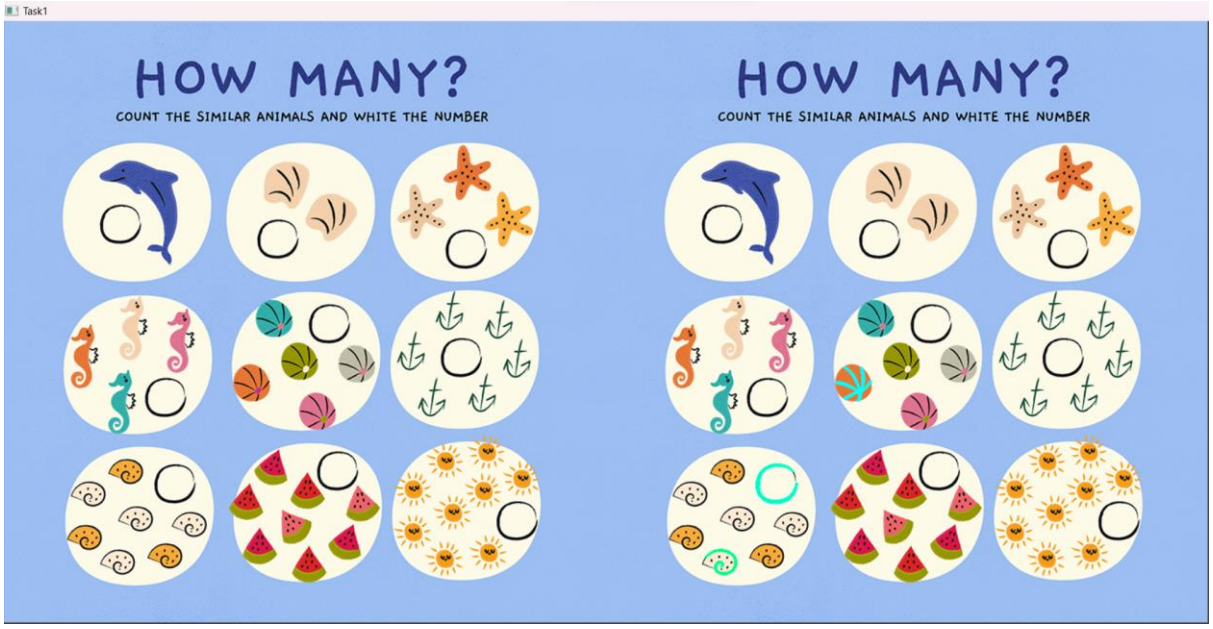
```
from skimage.metrics import structural_similarity
from tkinter import Tk
from tkinter.filedialog import askopenfilename
import cv2
import numpy as np
from PIL import Image
import os

# Đọc file Image_input
root = Tk()
root.withdraw()
file_path = askopenfilename(filetypes=[("Image_input",
".jpg;.jpeg;*.png")])

# Kiểm tra
```

```
if file_path:
    # Đọc ảnh từ đường dẫn
    img_origin = cv2.imread(file_path)
    img = img_origin.copy()

    #Tạo 3 hình chữ nhật random từng vị trí
    num_rectangles = 3
    for i in range(num_rectangles):
        # chọn vị trí vẽ hình chữ nhật
        height, width = img.shape[:2]
        w = np.random.randint(16, width // 8)
        h = np.random.randint(16, height // 8)
        x = np.random.randint(w // 8, width - w // 8)
        y = np.random.randint(h // 8, height - h // 8)
        # Chọn màu hình chữ nhật bằng cách lấy màu trung bình
quanh vị trí
        roi = img[y - h // 2:y + h // 2, x - w // 2:x + w //
2]
        avg_color = np.mean(roi, axis=(0, 1))
        # Cộng thêm màu để tránh trường hợp vẽ trùng với màu
nền
        background_color = [0, 0, 0]  # Giả sử màu nền là đen
        threshold = 30  # Khoảng ngưỡng để xác định sự khác
biệt giữa màu trung bình và màu nền
        if np.sum(np.abs(avg_color - background_color)) >
threshold:
            color = avg_color + [20, 20, 20]
        else:
            color = [255, 255, 255]  # Nếu màu trung bình
không khác biệt đủ, sử dụng màu trắng thay thế
        # Vẽ hình chữ nhật
        cv2.rectangle(img, (x - w//8, y - h//8), (x + w//8, y
+ h//8), color, -1)
        x = np.clip(x, w // 8, width - w // 8)
        y = np.clip(y, h // 8, height - h // 8)

    # Xuất hình ảnh
    # Lấy tên file và đường dẫn đến thư mục chứa file
    file_name = os.path.basename(file_path)
    file_dir = os.path.dirname(file_path)

    # Tạo tên file output: "image_imput_output_level1.png"
    file_name_out = file_name.split(".")[0] +
"_output_level1.png"

    # Tạo đường dẫn để lưa file
    file_path_out = os.path.join(file_dir, file_name_out)

    # Lưu ảnh đầu ra
    cv2.imwrite(file_path_out, img)
```

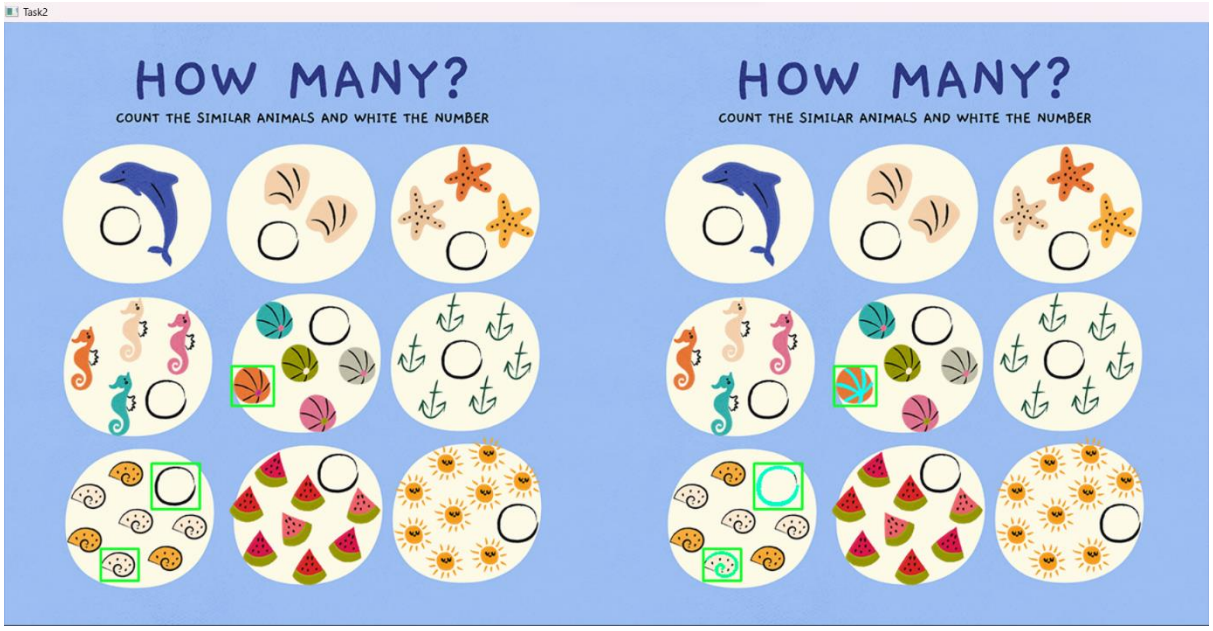
```
    #Đọc ảnh before và ảnh after
    before = cv2.imread(file_path)
    after = cv2.imread(file_name_out)

    #Chuyển đổi hình ảnh sang thang độ xám
    before_gray = cv2.cvtColor(before, cv2.COLOR_BGR2GRAY)
    after_gray = cv2.cvtColor(after, cv2.COLOR_BGR2GRAY)

    #Tính toán SSIM giữa hai hình ảnh
    (score, diff) = structural_similarity(before_gray,
after_gray, full=True)
    print("Image similarity", score)

    #Hình ảnh khác biệt chứa sự khác biệt hình ảnh thực tế
giữa hai hình ảnh
    #và được biểu diễn dưới dạng kiểu dữ liệu dấu phẩy động
trong phạm vi [0,1]
    #vì vậy ta phải chuyển đổi mảng thành số nguyên không dấu
8 bit trong phạm vi
    #[0,255] trước khi ta có thể sử dụng nó với OpenCV
    diff = (diff * 255).astype("uint8")

    #Ngưỡng hình ảnh khác biệt, tiếp theo là tìm đường viền để
    #có được các vùng của hai hình ảnh đầu vào khác nhau
    thresh = cv2.threshold(diff, 0, 255, cv2.THRESH_BINARY_INV
| cv2.THRESH_OTSU)[1]
    contours = cv2.findContours(thresh.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    contours = contours[0] if len(contours) == 2 else
contours[1]

    #Vẽ hình chữ nhật bao quanh vùng khác biệt
    for c in contours:
        area = cv2.contourArea(c)
        if area > 40:
            x, y, w, h = cv2.boundingRect(c)
            cv2.rectangle(before, (x, y), (x + w, y + h), (36,
255, 12), 2)
            cv2.rectangle(after, (x, y), (x + w, y + h), (36,
255, 12), 2)

    # Hiển thị hình ảnh
    result1 = np.hstack((img_origin, img))
    cv2.imshow("Task1", result1)
    result2 = np.hstack((before, after))
    cv2.imshow("Task2", result2)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Level 2: Task1 + Task2

Mô tả: Đổi màu đối tượng ngẫu nhiên trong ảnh, màu sắc của đối tượng là màu trung bình xung quanh.



Task1



Task2

Cách làm:

1. Người dùng chọn tệp hình ảnh (.jpg, .jpeg hoặc .png) bằng hộp thoại tệp
2. Đọc hình ảnh đã chọn và tạo một bản sao
3. Áp dụng phát hiện cạnh Canny và phát hiện đường viền cho hình ảnh đầu vào để phát hiện các đối tượng.
4. Chọn ba đối tượng ngẫu nhiên có diện tích từ 200 đến 700 pixel.
5. Thay đổi màu của các đối tượng được chọn thành màu trung bình của đối tượng.
6. Lưu hình ảnh đầu ra với tên tệp giống với hình ảnh đầu vào nhưng có thêm "_output_level2.png" vào cuối.
7. Chuyển đổi hình ảnh sang thang độ xám
8. Tính toán chỉ số tương đồng về cấu trúc (SSIM) giữa ảnh gốc và ảnh đã sửa đổi.
9. Tính toán hình ảnh khác biệt giữa hình ảnh gốc và hình ảnh đã sửa đổi, đồng thời đặt ngưỡng cho hình ảnh đó để thu được hình ảnh nhị phân về sự khác biệt
10. Tìm các đường viền trong ảnh nhị phân và vẽ các hình chữ nhật xung quanh các vùng khác biệt được phát hiện trong ảnh gốc.
11. Hiển thị chỉ mục SSIM và hình ảnh đã sửa đổi với các điểm khác biệt được vẽ xung quanh.

Yêu cầu nhập một số thư viện:

1. skipage.metrics.structural_similarity từ thư viện skipage để tính chỉ số SSIM.
2. Tk và askopenfilename từ mô-đun tkinter.filedialog để mở hộp thoại tệp và cho phép người dùng chọn tệp hình ảnh.
3. cv2 từ thư viện opencv-python để đọc, sửa đổi và ghi các tệp hình ảnh, tính toán hình ảnh khác biệt và phát hiện các đường viền.
4. numpy để tạo số nguyên ngẫu nhiên và thực hiện các thao tác trên mảng
5. PIL.Image từ thư viện Gối để trích xuất tên tệp và đường dẫn thư mục của hình ảnh đầu vào.

```
from skimage.metrics import structural_similarity
import cv2
import numpy as np
import random
from tkinter import Tk
from tkinter.filedialog import askopenfilename
```

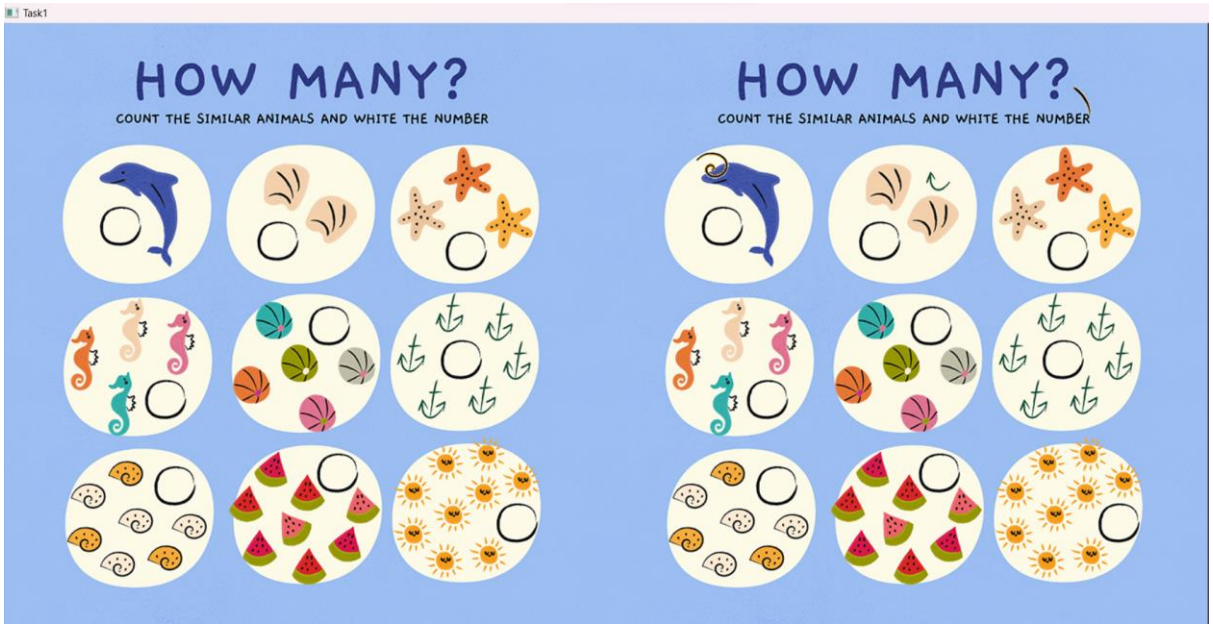
```
import os

#Đọc file ảnh Image_input
root = Tk()
root.withdraw()
file_path = askopenfilename(filetypes=[("Image_input",
".jpg;.jpeg;*.png")])

# Kiểm tra
if file_path:
    # Đọc ảnh từ đường dẫn
    img_origin = cv2.imread(file_path)
    img = img_origin.copy()

    # Chuyển đổi sang ảnh xám
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Bộ lọc canny để phát hiện cạnh
    edges = cv2.Canny(img, 50, 130)

    # Chuyển đổi sang ảnh nhị phân
    thresh = cv2.threshold(gray, 127, 255,
cv2.THRESH_BINARY)[1]

    # Hàm contour để tìm các đối tượng
    contours, hierarchy = cv2.findContours(thresh,
cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    # Lưu các contour có diện tích 300 đến 100 vào danh sách
objects
    objects = []
    for contour in contours:
        area = cv2.contourArea(contour)
        if area > 200 and area < 700:
            objects.append(contour)

    # Nếu không tìm thấy đối tượng nào có diện tích lớn hơn
200, thoát khỏi chương trình
    if not objects:
        exit()

    # Chọn ngẫu nhiên 3 đối tượng từ danh sách objects
    selected_contours = random.sample(objects, k=3)

    # Đổi màu bằng màu trung bình cho các đối tượng đã chọn
    for contour in selected_contours:
        mean_bgr = cv2.mean(contour)
        cv2.drawContours(img, [contour], -1, mean_bgr, -2)

    # Xuất hình ảnh
    # Lấy tên file và đường dẫn đến thư mục chứa file
```

```
    file_name = os.path.basename(file_path)
    file_dir = os.path.dirname(file_path)

    # Tạo tên file output: "image_imput_output_level2.png"
    file_name_out = file_name.split(".")[0] +
"_output_level2.png"

    # Tạo đường dẫn để lưa file
    file_path_out = os.path.join(file_dir, file_name_out)

    # Lưu ảnh đầu ra
    cv2.imwrite(file_path_out, img)

    # Đọc ảnh before và ảnh after
    before = cv2.imread(file_path)
    after = cv2.imread(file_name_out)

    # Chuyển đổi hình ảnh sang thang độ xám
    before_gray = cv2.cvtColor(before, cv2.COLOR_BGR2GRAY)
    after_gray = cv2.cvtColor(after, cv2.COLOR_BGR2GRAY)

    # Tính toán SSIM giữa hai hình ảnh
    (score, diff) = structural_similarity(before_gray,
after_gray, full=True)
    print("Image similarity", score)

    # Hình ảnh khác biệt chứa sự khác biệt hình ảnh thực tế
giữa hai hình ảnh
    # và được biểu diễn dưới dạng kiểu dữ liệu dấu phẩy động
trong phạm vi [0,1]
    # vì vậy ta phải chuyển đổi mảng thành số nguyên không dấu
8 bit trong phạm vi
    # [0,255] trước khi ta có thể sử dụng nó với OpenCV
    diff = (diff * 255).astype("uint8")

    # Ngưỡng hình ảnh khác biệt, tiếp theo là tìm đường viền
để
    # có được các vùng của hai hình ảnh đầu vào khác nhau
    thresh = cv2.threshold(diff, 0, 255, cv2.THRESH_BINARY_INV
| cv2.THRESH_OTSU)[1]
    contours = cv2.findContours(thresh.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    contours = contours[0] if len(contours) == 2 else
contours[1]

    # Vẽ hình chữ nhật bao quanh vùng khác biệt
    for c in contours:
        area = cv2.contourArea(c)
        if area > 40:
            x, y, w, h = cv2.boundingRect(c)
            cv2.rectangle(before, (x, y), (x + w, y + h), (36,
```

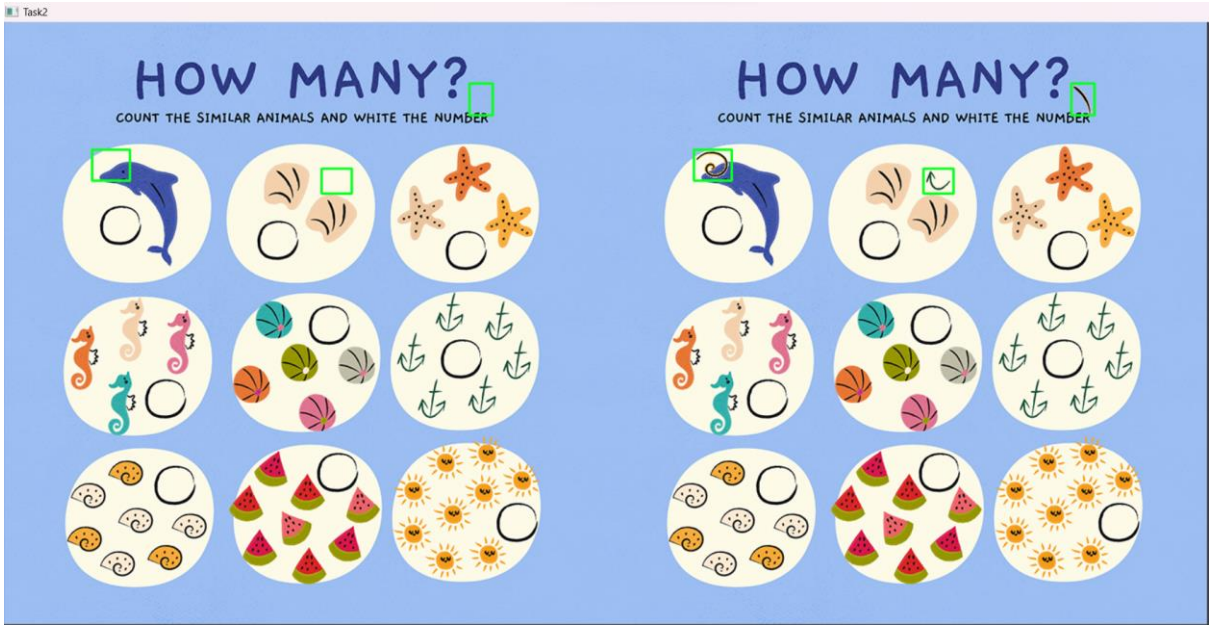
```
255, 12), 2)
            cv2.rectangle(after, (x, y), (x + w, y + h), (36,
255, 12), 2)

    # Hiển thị hình ảnh
    result1 = np.hstack((img_origin, img))
    cv2.imshow("Task1", result1)
    result2 = np.hstack((before, after))
    cv2.imshow("Task2", result2)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Level 3: Task 1 + Task 2

Mô tả: Đổi vị trí đối tượng copy đến vị trí ngẫu nhiên trong ảnh



Task1



Task2

Cách làm:

1. Người dùng chọn tệp hình ảnh (.jpg, .jpeg hoặc .png) bằng hộp thoại tệp
2. Đọc hình ảnh đã chọn và tạo một bản sao
3. Áp dụng phát hiện cạnh Canny và phát hiện đường viền cho hình ảnh đầu vào để phát hiện các đối tượng.
4. Chọn ba đối tượng ngẫu nhiên có diện tích từ 100 đến 700 pixel.
5. Tìm vị trí ngẫu nhiên trong ảnh mà không giao với bất kỳ cạnh nào
6. Copy đối tượng được chọn đến vị trí ngẫu nhiên tìm được
7. Lưu hình ảnh đầu ra với tên tệp giống với hình ảnh đầu vào nhưng có thêm "_output_level3.png" vào cuối.
8. Chuyển đổi hình ảnh sang thang độ xám
9. Tính toán chỉ số tương đồng về cấu trúc (SSIM) giữa ảnh gốc và ảnh đã sửa đổi.
10. Tính toán hình ảnh khác biệt giữa hình ảnh gốc và hình ảnh đã sửa đổi, đồng thời đặt ngưỡng cho hình ảnh đó để thu được hình ảnh nhị phân về sự khác biệt
11. Tìm các đường viền trong ảnh nhị phân và vẽ các hình chữ nhật xung quanh các vùng khác biệt được phát hiện trong ảnh gốc.
12. Hiển thị chỉ mục SSIM và hình ảnh đã sửa đổi với các điểm khác biệt được vẽ xung quanh.

Yêu cầu nhập một số thư viện:

1. skipage.metrics.structural_similarity từ thư viện skipage để tính chỉ số SSIM.
2. Tk và askopenfilename từ mô-đun tkinter.filedialog để mở hộp thoại tệp và cho phép người dùng chọn tệp hình ảnh.
3. cv2 từ thư viện opencv-python để đọc, sửa đổi và ghi các tệp hình ảnh, tính toán hình ảnh khác biệt và phát hiện các đường viền.
4. numpy để tạo số nguyên ngẫu nhiên và thực hiện các thao tác trên mảng
5. PIL.Image từ thư viện Gối để trích xuất tên tệp và đường dẫn thư mục của hình ảnh đầu vào.

```
from skimage.metrics import structural_similarity
import cv2
import numpy as np
import random
from tkinter import Tk
from tkinter.filedialog import askopenfilename
import os

# Đọc file Image_input
root = Tk()
```

```
root.withdraw()
file_path = askopenfilename(filetypes=[("Image_input",
".jpg;.jpeg;*.png")])

# Kiểm tra
if file_path:
    # Đọc ảnh từ đường dẫn được chọn
    img_origin = cv2.imread(file_path)
    img = img_origin.copy()

    # Chuyển đổi sang ảnh xám
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Bộ lọc canny để phát hiện cạnh
    edges = cv2.Canny(img, 50, 130)

    # Chuyển đổi ảnh sang ảnh nhị phân
    thresh = cv2.threshold(gray, 127, 255,
cv2.THRESH_BINARY)[1]

    # Áp dụng hàm tìm contour để tìm các đối tượng trong ảnh
    contours, hierarchy = cv2.findContours(thresh,
cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    # Lưu các contour có diện tích lớn hơn 100 vào danh sách
objects
    objects = []
    for contour in contours:
        area = cv2.contourArea(contour)
        if area > 100 and area < 700:
            objects.append(contour)

    # Nếu không tìm thấy đối tượng nào có diện tích lớn hơn
100, thoát khỏi chương trình
    if not objects:
        exit()

    # Lặp lại 3 lần để chọn và thay đổi màu cho 3 đối tượng
    for i in range(3):
        # Chọn ngẫu nhiên một đối tượng từ danh sách objects
        selected_contour = random.choice(objects)

        # Tìm vị trí ngẫu nhiên trong ảnh mà không giao với
bất kỳ cạnh nào
        rows, cols = img.shape[:2]
        mask = np.zeros((rows, cols), dtype=np.uint8)
        cv2.drawContours(mask, [selected_contour], -1, 255, -
1)
        rand_x, rand_y = None, None
        while rand_x is None or mask[rand_x, rand_y] == 255:
            rand_x = np.random.randint(0, rows)
```

```
            rand_y = np.random.randint(0, cols)

        # Copy đối tượng được chọn đến vị trí ngẫu nhiên tìm
được
        mask = np.zeros((rows, cols), dtype=np.uint8)
        cv2.drawContours(mask, [selected_contour], -1, 255, -
1)
        x, y, w, h = cv2.boundingRect(selected_contour)
        roi = img[y:y + h, x:x + w]
        mask_roi = mask[y:y + h, x:x + w]
        masked_img = cv2.bitwise_and(roi, roi, mask=mask_roi)
        new_x = rand_x - w // 2
        new_y = rand_y - h // 2
        for i in range(new_x, new_x + masked_img.shape[0]):
            for j in range(new_y, new_y +
masked_img.shape[1]):
                if i >= 0 and i < rows and j >= 0 and j < cols
and mask_roi[i - new_x, j - new_y] != 0:
                    img[i, j] = masked_img[i - new_x, j -
new_y]

    # Xuất hình ảnh
    # Lấy tên file và đường dẫn đến thư mục chứa file
    file_name = os.path.basename(file_path)
    file_dir = os.path.dirname(file_path)

    # Tạo tên file output: "image_imput_output_level2.png"
    file_name_out = file_name.split(".")[0] +
"_output_level2.png"

    # Tạo đường dẫn để lưa file
    file_path_out = os.path.join(file_dir, file_name_out)

    # Lưu ảnh đầu ra
    cv2.imwrite(file_path_out, img)

    # Đọc ảnh before và ảnh after
    before = cv2.imread(file_path)
    after = cv2.imread(file_name_out)

    # Chuyển đổi hình ảnh sang thang độ xám
    before_gray = cv2.cvtColor(before, cv2.COLOR_BGR2GRAY)
    after_gray = cv2.cvtColor(after, cv2.COLOR_BGR2GRAY)

    # Tính toán SSIM giữa hai hình ảnh
    (score, diff) = structural_similarity(before_gray,
after_gray, full=True)
    print("Image similarity", score)

    # Hình ảnh khác biệt chứa sự khác biệt hình ảnh thực tế
giữa hai hình ảnh
```

```
    # và được biểu diễn dưới dạng kiểu dữ liệu dấu phẩy động
trong phạm vi [0,1]
    # vì vậy ta phải chuyển đổi mảng thành số nguyên không dấu
8 bit trong phạm vi
    # [0,255] trước khi ta có thể sử dụng nó với OpenCV
    diff = (diff * 255).astype("uint8")

    # Ngưỡng hình ảnh khác biệt, tiếp theo là tìm đường viền
để
    # có được các vùng của hai hình ảnh đầu vào khác nhau
    thresh = cv2.threshold(diff, 0, 255, cv2.THRESH_BINARY_INV
| cv2.THRESH_OTSU)[1]
    contours = cv2.findContours(thresh.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    contours = contours[0] if len(contours) == 2 else
contours[1]

    # Vẽ hình chữ nhật bao quanh vùng khác biệt
    for c in contours:
        area = cv2.contourArea(c)
        if area > 40:
            x, y, w, h = cv2.boundingRect(c)
            cv2.rectangle(before, (x, y), (x + w, y + h), (36,
255, 12), 2)
            cv2.rectangle(after, (x, y), (x + w, y + h), (36,
255, 12), 2)

    # Hiển thị hình ảnh
    result1 = np.hstack((img_origin, img))
    cv2.imshow("Task1", result1)
    result2 = np.hstack((before, after))
    cv2.imshow("Task2", result2)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```