Nguyễn Văn Hải: 20020397

Link github: link

Chương trình viết bằng py, có sử dụng một số thư viện của py trong quá trình viết.

Sau khi clone code về chạy chương trình chính (midterm.py)

* import thư viện

```
import tkinter as tk
from tkinter.filedialog import askopenfilename
import cv2
import numpy as np
import imutils
from PIL import ImageTk, Image, ImageDraw,ImageFont
```

*Load ảnh từ máy tính:

```
def loadOriginalImage():
    # Hiển thị hộp thoại mở tệp tin
    file_path = askopenfilename(filetypes=[("Image files", "*.jpg;*.jpeg;*.png")])

    # Kiểm tra xem người dùng có chọn ảnh hay không
    if file_path:
        # Đọc ảnh từ đường dẫn được chọn
        image = cv2.imread(file_path)
```

* Thực hiện việc thay đổi ảnh:

```
def doChange(image, cnt1, cnt2):
    img1 = changeAllBlack(image.copy(), (35,35,35))
    img2 = changeContourColor(image.copy(), cnt1, cnt1, -70)
    img3 = changeContourColor(image.copy(), resizeContour(cnt2), resizeContour(cnt2), -50)
    img_result1 = resultForChangeAllBlack(img1.copy(), (255,0,0))
    img_result2 = resultForChangeContourColor(img2.copy(), cnt1)
    img_result3 = resultForChangeContourColor(img3.copy(), resizeContour(cnt2))
```

* Tast 1:

- Sử dụng 2 thuật toán là lọc màu theo ngưỡng và tìm viền bằng canny để xây dựng 3 mức độ xử lý.

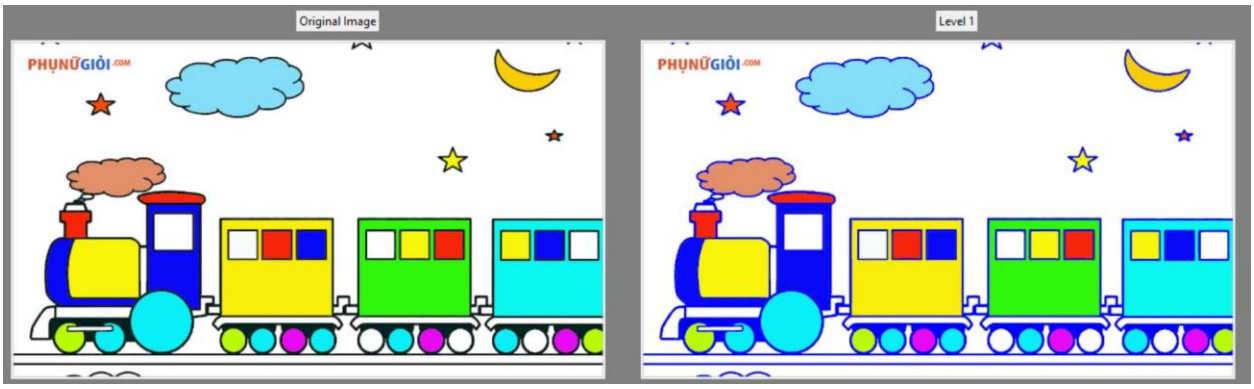
- Mức 1:

+ Sử dụng thuật toán lọc màu theo ngưỡng để lọc màu đen ra khỏi ảnh, sau đó thay thế bằng 1 màu bất kì.

```python
def changeAllBlack(image, color):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    # Xác định ngưỡng màu để phân đoạn
    lower = np.array([0, 0, 0])  # giá trị ngưỡng thấp cho H, S, V
    upper = np.array([250, 255, 100])  # giá trị ngưỡng cao cho H, S, V
    mask = cv2.inRange(hsv, lower, upper)  # tạo mặt nạ

    # Áp dụng mặt nạ để lấy vật thể và đổi màu sắc
    result = image.copy()
    result[mask != 0] = color  # đổi màu thành màu xám
    return result
```

+ Thuật toán được viết ở hàm changeAllBlack(image, color): image là ảnh gốc truyền vào, color là màu dùng để thay thế màu đen

Example 1: (input bên trái, ouput bên phải)



Example 2: (input bên trái, ouput bên phải)

(trong ví dụ em đã truyền vào màu thay thế khá khác biệt nên dễ dàng nhận ra)

- Mức 2:

+ Sử dụng thuật toán Canny edge detection để tìm viền và lấy ra các vùng vật thể trong ảnh. Dùng findContours trong openCV để lấy ra các vùng đó. (được viết khi load ảnh gốc lên trong hàm loadOriginalImage()):

```
# Đọc ảnh từ đường dẫn được chọn
image = cv2.imread(file_path)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Làm mịn ảnh để loại bỏ nhiễu
blur = cv2.GaussianBlur(gray, (9, 9), 0)

# Tìm vật thể trong ảnh bằng phương pháp Canny edge detection
edges = cv2.Canny(blur, 0, 100)

# Tìm các vật thể liên thông trong ảnh
contours = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
contours = imutils.grab_contours(contours)
if len(contours) != 0:

    # Sắp xếp các contour theo diện tích giảm dần:
    area_cnt = [cv2.contourArea(cnt) for cnt in contours]
    area_sort = np.argsort(area_cnt)[::-1]

    cnt1 = contours[area_sort[0]]
    if len(contours) == 1:
        cnt2 = cnt1
    else:
        cnt2 = contours[area_sort[1]]
    doChange(image, cnt1, cnt2)
```

- Sau khi đọc ảnh, ảnh được chuyển qua màu xám, được làm mịn bởi thuật toán GaussianBlur(gray, (9, 9), 0) của OpenCv.
- Sau đó áp dụng thuật toán canny edge detection lên ảnh đã được làm mờ để tìm ra viền của ảnh.
- Dùng thuật toán findContours của OpenCV để tìm kiếm các vật thể là vùng liên thông trong ảnh.
- Sau đó lấy ra 2 vùng lớn nhất để thực hiện sự thay đổi trên chúng.

+ Sau đó từ vùng sẽ được lấy ra để thay đổi màu sắc tạo ra mức độ tiếp theo của bài toán.

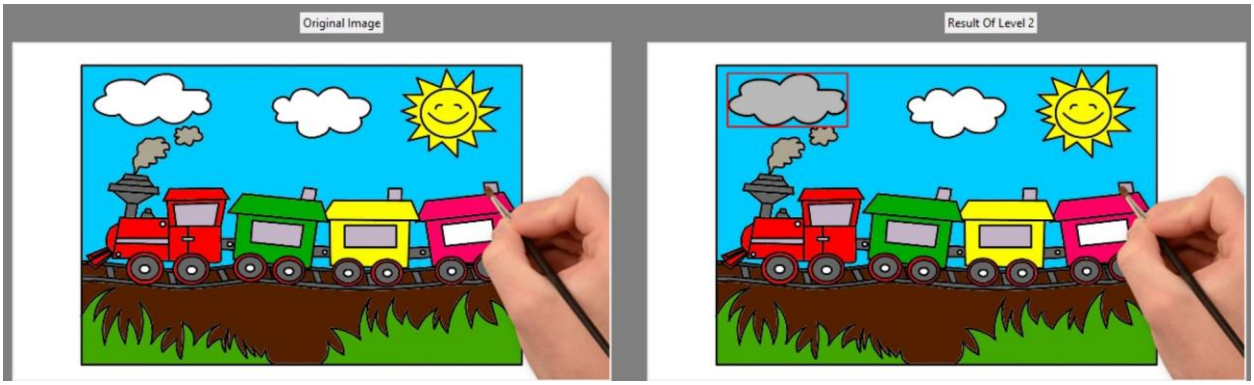
+ Màu sắc ở đây được là tất cả các điểm ảnh được duyệt qua và được tăng (giảm) độ sáng của màu sắc đó.

+ Hàm changeContourColor(image, n_contour, contour, change) thực hiện việc thay đổi màu sắc:

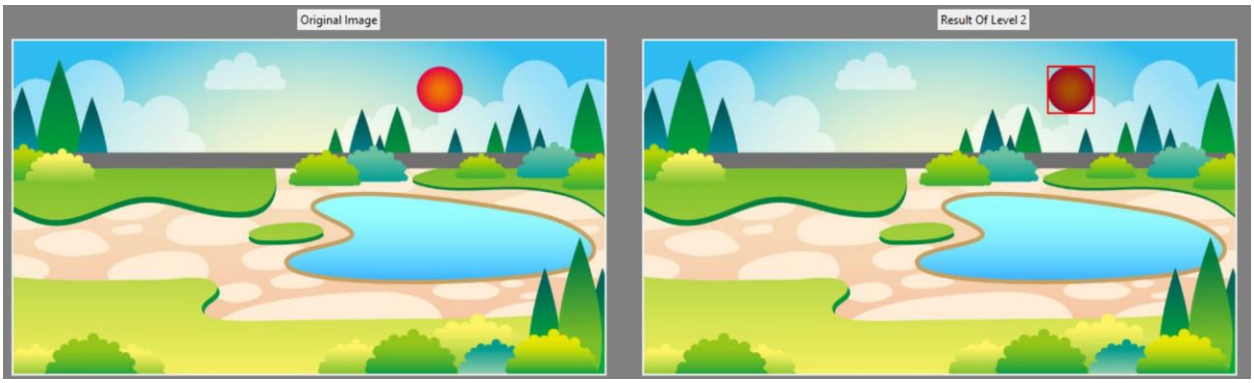
```python
def changeContourColor(image, n_contour, contour, change):

    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # Khởi tạo contour mask zero
    mask = np.zeros(gray.shape, np.uint8)
    n_mask = np.zeros(gray.shape, np.uint8)
    # Vẽ contour trên back ground mask zero
    cv2.drawContours(mask,[contour],0,255,-1)
    cv2.drawContours(n_mask,[n_contour], 0, 255, -1)

    # dùng hàm cv2.findNoneZero() lấy ra toàn bộ các points có giá trị khác 0 là các điểm ở trong contour
    pixel = cv2.findNonZero(mask)
    n_pixel = cv2.findNonZero(n_mask)
    # chuyển ảnh từ BGR sang HSV
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    # lặp qua tất cả điểm ở trong contour, lấy màu của điểm đó (dạng HSV) rồi thay đổi 1 chút.(change)
    for i in range(len(n_pixel)):
        h, s, v = hsv[pixel[i][0][1], pixel[i][0][0]]
        hsv[n_pixel[i][0][1], n_pixel[i][0][0]] = np.clip([h,s,v + change], 0, 255)

    # chuyển ảnh về dạng BRG rồi return lại ảnh
    return cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
```

- Tham số truyền vào là ảnh gốc (image) , 1 contour mới (n_contour) là vùng contour sau khi thực hiện phép biến đổi (thu nhỏ, dịch chuyển) contour ban đầu, 1 contour ban đầu (contour) và 1 số nguyên là độ lệch màu của màu cũ so với màu mới (change).
- Hàm sẽ chuyển ảnh sang màu xám, sau đó tạo 2 mask có màu đen trên ảnh xám ban đầu.
- Sau đó sẽ vẽ vùng contour và n_contour với màu trắng trên mask và n_mask đó.
- Sau đó dùng hàm findNonZero(mask) thuộc OpenCV để tìm ra tất cả các điểm có giá trị khá 0 (khác màu đen) trên mask và n_mask. => Ta được tập hợp tất cả các điểm ở trong vật thể đó.
- Sau đó chuyển ảnh sang dạng hsv để dễ dàng xử lý.
- Duyệt qua tất cả điểm ảnh trong n_contour (n_pixel), lấy giá trị màu ở vùng contour cũ (pixel) rồi gán lại giá trị màu đó với 1 chút thay đổi cho vùng n_contour.
- Cuối cùng convert lại thành dạng BGR rồi return ảnh đã thay đổi.

Example 1: (input bên trái, output bên phải)



Example 2: (input bên trái, output bên phải)



(trong ví dụ em đã đổi truyền vào độ thay đổi màu khá lớn nên dễ dàng nhận ra)

- Mức 3:

+ Từ mức độ 2, vùng contour được chọn sẽ được thu nhỏ theo tỉ lệ (hàm resizeContour(contour)) và thay đổi màu sắc tạo ra mức độ khó hơn cho bài toán.

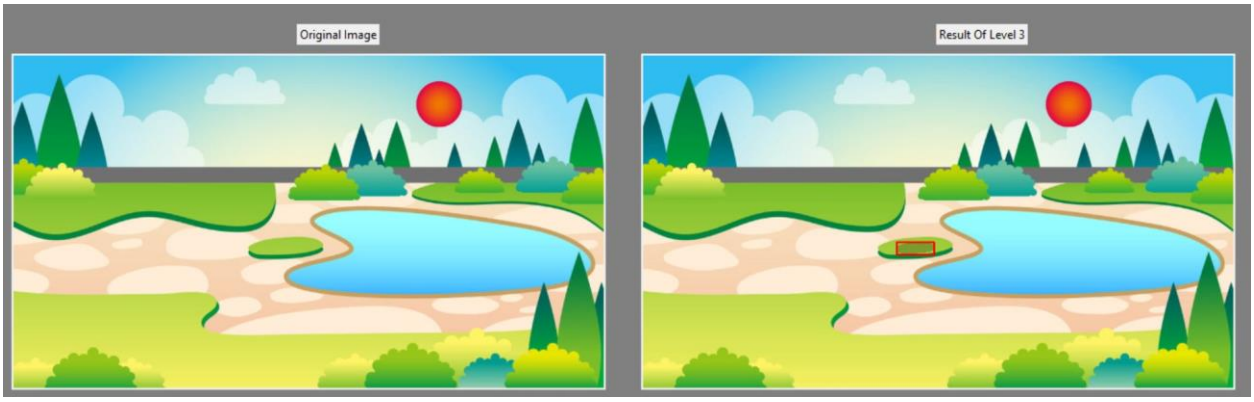
```
# Lấy contour đầu tiên và tính toán hình dạng của nó
M = cv2.moments(contour)
cx = int(M['m10'] / M['m00']) #tổng tọa độ x chia cho diện tích
cy = int(M['m01'] / M['m00']) #tổng tọa độ y chia cho diện tích

# Tính toán tỉ lệ thu nhỏ và tạo ma trận biến đổi
scale_percent = 0.5 # tỉ lệ thu nhỏ
M = cv2.getRotationMatrix2D((cx, cy), 0, scale_percent)

# Áp dụng ma trận biến đổi để thu nhỏ contour
contour_transformed = cv2.transform(contour, M)
return contour_transformed
```

+ Màu sắc ở đây được là tất cả các điểm ảnh được duyệt qua và được tăng (giảm) độ sáng của màu sắc đó.

+ Có thể dịch chuyển cả contour được chọn nhưng dễ gây khó khăn cho người chơi nên em chỉ thu nhỏ lại và đặt trong lòng vật thể ban đầu. (trong bài không chuyển vị trí mà chỉ thay đổi tỉ lệ và màu sắc)

```
# Chuyển contour đến vị trí mới
M = np.float32([[1, 0, -50], [0, 1, 10]])  # Tạo ma trận chuyển đổi 2D
# Dùng hàm transform để chuyển đổi tọa độ của các điểm trong contour
contour_transformed = cv2.transform(contour, M)
```

Example 1: (input bên trái, output bên phải)



Example 2: (input bên trái, output bên phải)

(trong ví dụ em đã truyền vào độ thay đổi màu khá lớn nên dễ dàng nhận ra)

* Tast 2:

- Kết quả được show ngay ở trong chương trình khi xem xong 3 mức độ xử lý.

- Với mức 1 thì kết quả được đưa ra là 1 dòng chữ, thêm vào đó là màu sắc được thay đổi rõ ràng hơn.

+ hàm resultForChangeAllBlack(image, color) sẽ nhận vào ảnh gốc và 1 màu sắc để thay đổi màu đen của ảnh gốc sang màu mới để người dùng dễ nhận ra điểm khác biệt hơn.

```
def resultForChangeAllBlack(image, color):
    image = changeAllBlack(image, color)
    cv2.putText(image, 'Result: All blacks are dimmed', (10, 25), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
    cv2.putText(image, 'This image changed black to blue', (10, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
    return image
```

- Với mức 2 và 3 thì kết quả sẽ được vẽ 1 hình chữ nhật bao quanh điểm khác biệt đó.

+ Hàm resultForChangeContourColor(image, contour) sẽ nhận vào ảnh sau biến đổi và 1 vùng contour là vùng thay đổi trong ảnh so vs ảnh gốc.

+ Sau đó hàm sẽ vẽ 1 hình chữ nhật bao quanh vùng contour đó và trả về hình ảnh sau khi vẽ.

```
def resultForChangeContourColor(image, contour):
    x, y, w, h = cv2.boundingRect(contour)
    cv2.rectangle(image, (x, y), (x+w, y+h), (0,0,255), 2)
    return image
```

-Trong bài có từng thử cách tìm kết quả bằng việc thực hiện thuật toán để tìm sự khác biệt giữa 2 ảnh nhưng sự thay đổi là quá nhỏ so với ảnh gốc nên việc tìm kiếm sự khác biệt giữa 2 ảnh đôi lúc bị nhầm lẫn. (code)

```
def findDifference(image1, image2):
    #Grayscale
    gray1 = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
    gray2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)

    #find the difference
    diff = cv2.absdiff(gray1, gray2)

    #Apply threshold
    thresh = cv2.threshold(diff, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]

    #Dilation
    kernel = np.ones((7,7), np.uint8)
    dilate = cv2.dilate(thresh, kernel, iterations=2)

    #Find contours
    contours = cv2.findContours(dilate.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    contours = imutils.grab_contours(contours)

    max_area = cv2.contourArea(contours[0])
    max_contour = contours[0]

    for cnt in contours:
        area = cv2.contourArea(cnt)
        if area > max_area:
            max_area = area
            max_contour = cnt

    x, y, w, h = cv2.boundingRect(max_contour)
    cv2.rectangle(image2, (x, y), (x+w, y+h), (0,0,255), 2)
    return image2
```

* Cách sử dụng:

- Chương trình viết bằng py, có sử dụng một số thư viện của py trong quá trình viết.

- Sau khi clone code về chạy chương trình chính (midterm.py)

  + Một màn hình tương tác hiện lên, người dùng có thể ấn nút "Load Image" để tải ảnh từ máy tính cá nhân. (Lưu ý: Ảnh phải là một trong các định dạng .jpg, .jpeg hoặc .png. Nếu không chương trình sẽ lỗi và ảnh không được load lên)

  + Sau khi ảnh được load thành công, ảnh gốc sẽ được show ở bên trái (có tiêu đề là "Original Image").

  + Người dùng ấn nút "View" để xem các mức độ thay đổi của hình ảnh sau khi xử lý. Ảnh được xử lý sẽ show ở bên phải màn hình với tiêu đề là các Level từ 1 đến 3.

  + Ấn "Next Level" để xem các level tiếp theo.

  + Sau khi xem 3 Level thì sẽ show ra các kết quả của từng ảnh.

  + Ấn nút "Next Result" để xem các kết quả tiếp theo của từng Level.

  + Sau khi xem 3 kết quả thì sẽ lại quay lại 3 ảnh được xử lý.

  + Trong quá trình sử dụng có thể thay đổi ảnh khác bằng cách lại nút "Load Image"