ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO GIỮA KỲ

MÔN HỌC XỬ LÝ ẢNH

Họ và tên: Cao Việt Hoàng

Ngày sinh: 04/10/2001

MSSV: 19020819

Lớp học phần: INT3404 1

Hà Nội, năm 2023

Em dùng Google Colab (Colaboratory) để hoàn thành các tasks.

Task 1: Make "spot the difference" game data

Ảnh gốc:

LEVEL 1: THÊM 1 VÀI QUẢ BÓNG ĐẶC BIỆT KHÁC VÀO HÌNH ẢNH

1. *Import packages cần thiết*

```
import cv2 import
numpy as np
from google.colab.patches import cv2_imshow
```

2. *Input hình ảnh gốc, và các hình ảnh muốn thêm vào*

```
img=cv2.imread("/content/colorball.jpg")
add_img1=cv2.imread("/content/add_b_150.jpg")
add_img2=cv2.imread("/content/smile_ball.jpg")
cv2_imshow(img);
cv2_imshow(add_img1)
cv2_imshow(add_img2)
```

3. *Sử dụng phương thức* ***shape*** *trong thư viện* ***numpy*** *để trả về các kích thước của ảnh cần chèn*

```
# Lấy ra các thông số kích thước hình dạng của các ảnh thêm vào
rows1,cols1,channels1=add_img1.shape
rows2,cols2,channels2=add_img2.shape
```

4. *Xác định vị trí cần chèn các ảnh*

```
# Xác định vị trí cần chèn các ảnh
roi1=img[50:(50+rows1),50:(50+cols1)]
roi2=img[150:(150+rows2),300:(300+cols2)]
```

5. *Convert các ảnh cần chèn sang GRAY*

```
# Convert ảnh cần chèn sang GRAY
add_Gray1 = cv2.cvtColor(add_img1,cv2.COLOR_BGR2GRAY)
add_Gray2 = cv2.cvtColor(add_img2,cv2.COLOR_BGR2GRAY)
```

6. *Threshold các ảnh GRAY: sử dụng THRESH_BINARY*

```
# threshold: THRESH_BINARY
ret1,mask1 = cv2.threshold(add_Gray1,220,255,cv2.THRESH_BINARY)
ret2,mask2 = cv2.threshold(add_Gray2,220,255,cv2.THRESH_BINARY)
```

7. *Trích xuất bitwise các phần cần thiết (inverseMask, background, frontImage)*

```
# trích xuất các phần cần thiết
inverseMask1 = cv2.bitwise_not(mask1)
background1 = cv2.bitwise_and(roi1,roi1,mask=mask1)
frontImage1 = cv2.bitwise_and(add_img1,add_img1,mask=inverseMask1)
inverseMask2 = cv2.bitwise_not(mask2)
background2 = cv2.bitwise_and(roi2,roi2,mask=mask2)
frontImage2 = cv2.bitwise_and(add_img2,add_img2,mask=inverseMask2)
```

8. Thêm các phần đã trích xuất vào ảnh ban đầu đúng vị trí mong muốn

```
# thêm các phần đã trích xuất vào ảnh ban đầu đúng vị trí mong muốn
dest1=cv2.add(background1,frontImage1)
dest2=cv2.add(background2,frontImage2)
img[50:(50+rows1),50:(50+cols1)]=dest1
img[150:(150+rows2),300:(300+cols2)]=dest2
```

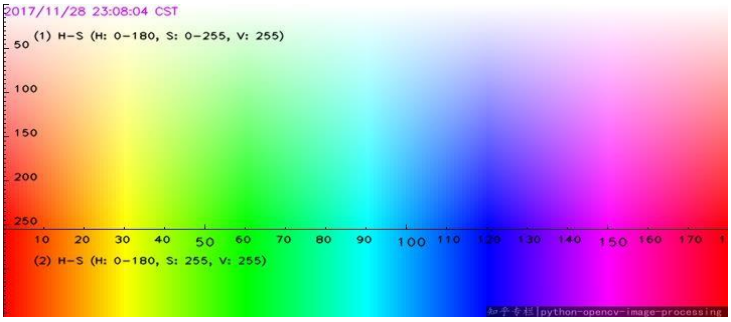
9. Output kết quả

```
# hiển thị kết quả
cv2_imshow(img)
```

Level 2: Thay đổi các quả bóng màu vàng thành màu cam

Thay đổi màu của một số đối tượng trong ảnh, sử dụng không gian màu của HSV (HSV Color Spaces)

HSV Color Spaces



1. *Import các packages cần thiết*

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
```

2. *Input hình ảnh gốc*

```
image = cv2.imread("/content/colorball.jpg")
```

3. *Chuyển đổi BGR sang HSV*

```
hsv=cv2.cvtColor(image,cv2.COLOR_BGR2HSV)
```

4. *Xác định phạm vi màu (vàng) cần thay đổi*

```
# xác định phạm vi màu vàng trong không gian màu HSV
yellow_lo=np.array([20, 100, 100])
yellow_hi=np.array([30, 255, 255])
mask=cv2.inRange(hsv,yellow_lo,yellow_hi)
```

5. *Chuyển sang màu khác (cam) trong HSV*

```
# tính toán chuyển sang màu khác (cam)
h, s, v = cv2.split(hsv)
h = np.mod(h + 170, 180)
s = np.clip(s - 100, 0, 255)
v = np.clip(v, 0, 255)
hsv = cv2.merge([h, s, v])
```

6. *Convert từ HSV trở lại BGR*

```
# convert từ HSV sang BGR
bgr = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
```

7. *Trích xuất các phần cần thiết cho kết quả*

```
# trích xuất các phần cần thiết cho kết quả
inv_mask = cv2.bitwise_not(mask)
result = cv2.bitwise_or(cv2.bitwise_and(image, image, mask=inv_mask),
 cv2.bitwise_and(bgr, bgr, mask=mask))
```

8. *Hiển thị kết quả*

```
cv2_imshow(result)
```

Level 3: Thay đổi các quả bóng màu xanh lá cây thành màu hồng nhũ, thêm 1 vài quả bóng đặc biệt vào ảnh

1. *Import các packages cần thiết*

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
```

2. *Input hình ảnh gốc, và các hình ảnh muốn thêm vào*

```
img = cv2.imread("/content/colorball.jpg")
add_img1=cv2.imread("/content/add_b_150.jpg")
add_img2=cv2.imread("/content/smile_ball.jpg")
```

3. *Chuyển đổi BGR sang HSV*

```
hsv=cv2.cvtColor(image,cv2.COLOR_BGR2HSV)
```

5. *Xác định phạm vi màu cần thay đổi*

```
# xác định phạm vi của màu xanh lá cây
green_lower = np.array([36,0,0])
green_upper = np.array([86,255,255])
mask = cv2.inRange(hsv, green_lower, green_upper) #green
```

4. *Chuyển sang màu khác (màu hồng nhũ) trong HSV*

```
# tính toán chuyển sang màu hồng nhũ
h, s, v = cv2.split(hsv)
h = np.mod(h + 100, 180)
s = np.clip(s - 100, 0, 255)
v = np.clip(v, 0, 255)
hsv = cv2.merge([h, s, v])
```

6. *Convert từ HSV trở lại BGR*

```
# convert từ HSV sang BGR
bgr = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
```

7. *Trích xuất các phần cần thiết cho kết quả*

```
# trích xuất các phần cần thiết cho kết quả
inv_mask = cv2.bitwise_not(mask)
result = cv2.bitwise_or(cv2.bitwise_and(image, image, mask=inv_mask),
 cv2.bitwise_and(bgr, bgr, mask=mask))
```

8. *Hiển thị hình ảnh sau khi đổi màu*

```
cv2_imshow(result)
```

9. *Sử dụng phương thức* ***shape*** *trong thư viện* ***numpy*** *để trả về các kích thước của ảnh cần chèn*

```
# Lấy ra các thông số kích thước hình dạng của các ảnh thêm vào
rows1,cols1,channels1=add_img1.shape
rows2,cols2,channels2=add_img2.shape
```

10. *Xác định vị trí cần chèn các ảnh*

```
# Xác định vị trí cần chèn các ảnh
roi1=result[200:(200+rows1),50:(50+cols1)]
roi2=result[150:(150+rows2),300:(300+cols2)]
```

11. *Convert các ảnh cần chèn sang GRAY*

```
 # convert các ảnh cần chèn sang GRAY
add_Gray1 = cv2.cvtColor(add_img1,cv2.COLOR_BGR2GRAY)
add_Gray2 = cv2.cvtColor(add_img2,cv2.COLOR_BGR2GRAY)
```

12. *Threshold: sử dụng THRESH_BINARY*

```
# threshold: THRESH_BINARY
ret1,mask1 = cv2.threshold(add_Gray1,220,255,cv2.THRESH_BINARY)
ret2,mask2 = cv2.threshold(add_Gray2,220,255,cv2.THRESH_BINARY)
```

13. *Trích xuất bitwise các phần cần thiết (inverseMask, background, frontImage)*

```
# trích xuất các phần cần thiết
inverseMask1 = cv2.bitwise_not(mask1)
background1 = cv2.bitwise_and(roi1,roi1,mask=mask1)
frontImage1 = cv2.bitwise_and(add_img1,add_img1,mask=inverseMask1)
inverseMask2 = cv2.bitwise_not(mask2)
background2 = cv2.bitwise_and(roi2,roi2,mask=mask2)
frontImage2 = cv2.bitwise_and(add_img2,add_img2,mask=inverseMask2)
```

14. *Thêm các phần đã trích xuất vào ảnh ban đầu đúng vị trí mong muốn*

```
# thêm các phần đã trích xuất vào ảnh ban đầu đúng vị trí mong muốn
dest1=cv2.add(background1,frontImage1)
dest2=cv2.add(background2,frontImage2)
result[200:(200+rows1),50:(50+cols1)]=dest1
result[150:(150+rows2),300:(300+cols2)]=dest2
```

15. *Output kết quả*

```
# hiển thị kết quả
cv2_imshow(result)
```

Task 2: Solve game Spot the differences

1. *Import các packages cần thiết*

```
from skimage.metrics import structural_similarity
from google.colab.patches import cv2_imshow
import numpy as np
import imutils
import cv2
```

2. *Input 2 hình ảnh khác nhau, convert 2 hình ảnh sang Gray*

```
image_one = cv2.imread("/content/colorball.jpg")
image_two = cv2.imread("/content/level3_output.jpg")
gray1 = cv2.cvtColor(image_one, cv2.COLOR_BGR2GRAY)
gray2 = cv2.cvtColor(image_two, cv2.COLOR_BGR2GRAY)
```

3. *Tính toán Structural Similarity Index Measurement (SSIM) giữa hai hình ảnh, trả về hình ảnh khác biệt*

```
# Compute the Structural Similarity Index (SSIM) between the two
images, ensuring that the difference image is returned
(score, diff) = structural_similarity(gray1,gray2,full=True)
diff = (diff*255).astype("uint8")
```

4. *Threshold: THRESH_BINARY_INV, THRESH_OTSU*

```
# threshold the difference image
thresh = cv2.threshold(diff,0,128, cv2.THRESH_BINARY_INV | cv2.THRESH
_OTSU) [1]
```

5. *Sử dụng findContours để tìm các khu vực*

```
# find contours to get the regions of the two input images that
differ
cnts = cv2.findContours(thresh.copy(),cv2.RETR_EXTERNAL, cv2.CHAIN_AP
PROX_SIMPLE)
cnts = imutils.grab_contours(cnts)
```

6. *Dùng vòng lặp với các contours được tìm thấy:*

Tính toán bounding của contour; xác định tọa độ tâm, bán kính; sau đó vẽ vòng tròn để chỉ ra sự khác biệt.

```
# 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)
    x1=int(x+w/2)
    y1=int(y+h/2)
    r=int(1/2*np.sqrt(w*w+h*h))
    if cv2.contourArea(c) >50:
      cv2.circle(image_two,(x1,y1),r,(0,0,255),2)
```

7. *Hiển thị hình ảnh output*

```
# show the origin image and the output image
cv2_imshow(image_one)
cv2_imshow(image_two)
```

Level 1:

Level 2:

Level 3: