Họ và tên: Nguyễn Trường Huy

MSSV: 22022509

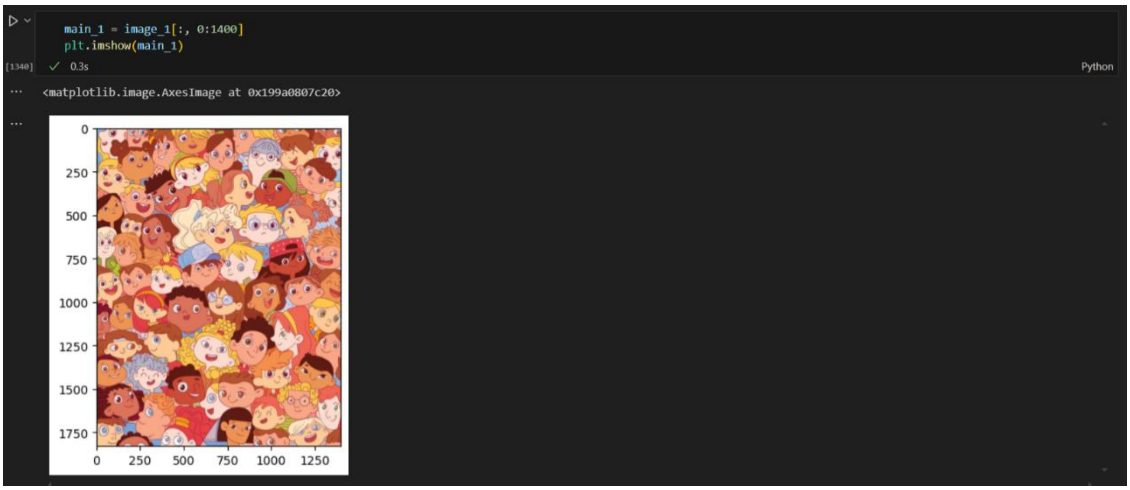
BÁO CÁO BÀI TẬP

FIND & COUNT

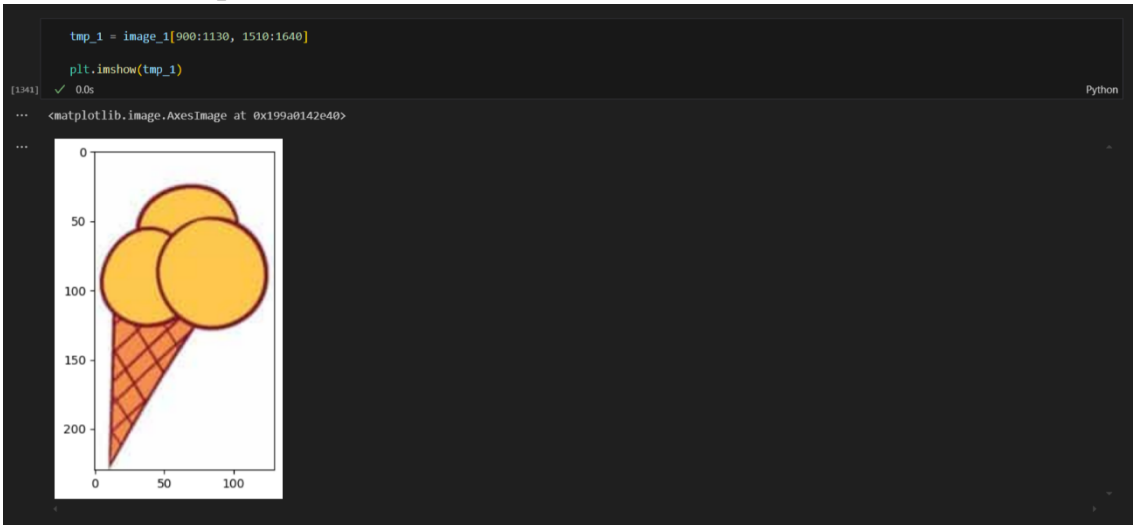
A. FINDING

1. Chuẩn bị đầu vào

- Vùng ảnh tìm kiếm:



- Ảnh template:



2. Finding function

```python
def finding(main, tmp, ratio, threshold, new_color, canny=False):
    # resize lại template
    tmp = cv2.resize(tmp, None, fx = ratio, fy = ratio, interpolation=cv2.INTER_AREA)

    # convert sang ảnh xám
    tmp_gray = cv2.cvtColor(tmp, cv2.COLOR_RGB2GRAY)
    main_gray = cv2.cvtColor(main, cv2.COLOR_RGB2GRAY)
    # giới hạn độ sáng
    tmp_gray[tmp_gray > threshold] = new_color

    h, w = tmp.shape[:2]

    # Áp dụng Canny để phát hiện các cạnh
    if canny:
        main_gray = cv2.Canny(main_gray, 50, 150)
        tmp_gray = cv2.Canny(tmp_gray, 50, 150)

    # hàm matching
    result = cv2.matchTemplate(main_gray, tmp_gray, cv2.TM_CCOEFF_NORMED)

    # vẽ khung matching
    min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(result)
    top_left = max_loc
    bottom_right = (top_left[0] + w, top_left[1] + h)
    copy = main.copy()
    cv2.rectangle(copy, top_left, bottom_right, (0, 0, 255), 20)
    plt.imshow(copy)
```

Generate docs (Ctrl+.)

✓ 0.0s Python

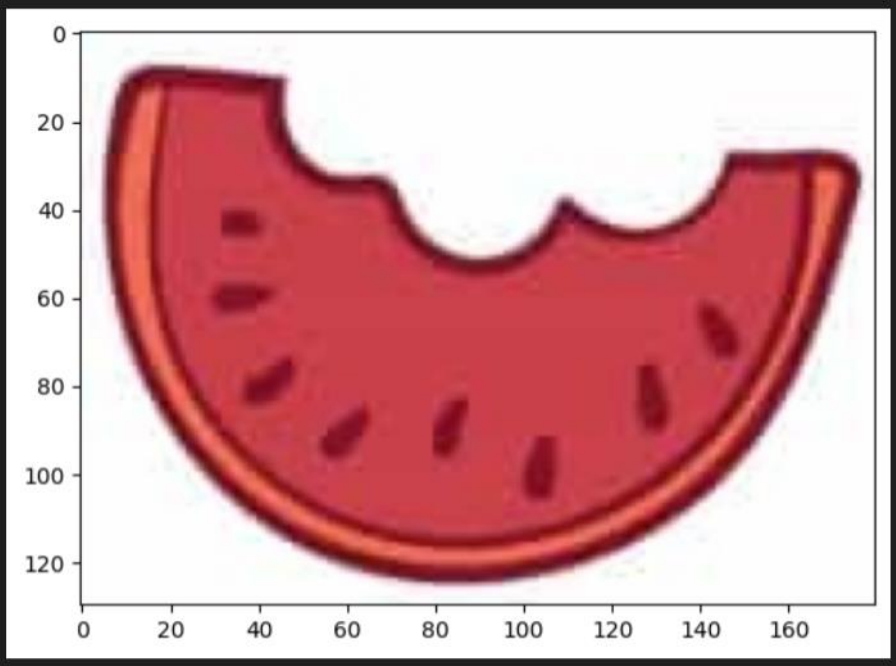
- Resize lại ảnh template để phù hợp với kích thước của nó trong ảnh tìm kiếm
- Chuyển đổi ảnh tìm kiếm và template về ảnh xám và kiểm soát độ sáng để cải thiện độ chính xác
- Sử dụng thêm bộ lọc Canny để phát hiện các cạnh tốt hơn
- Dùng cv2.matchTemplate() để thực hiện việc matching. Ở đây sử dụng phương pháp cv2.TM_CCOEFF_NORMED (Normalized Coefficient Matching)
- Lấy tọa độ của vật thể đã matching và vẽ khung bao quanh vật thể

3. Gọi hàm matching

```
tmp_6 = image_1[1220:1350, 1840:2020]
plt.imshow(tmp_6)
```

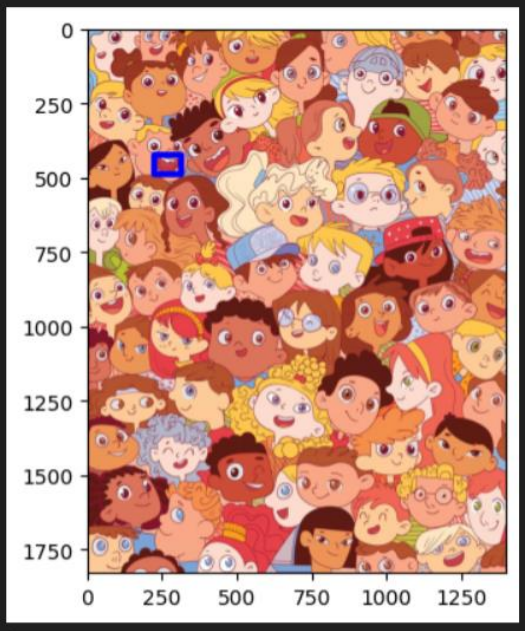
[1351] ✓ 0.1s

```
<matplotlib.image.AxesImage at 0x199a0961820>
```




```
finding(main_1, tmp_6, 0.5, 300, 200, canny=True)
```

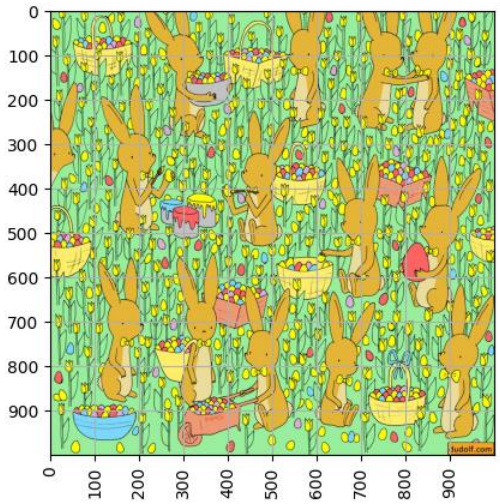
[1352] ✓ 0.4s

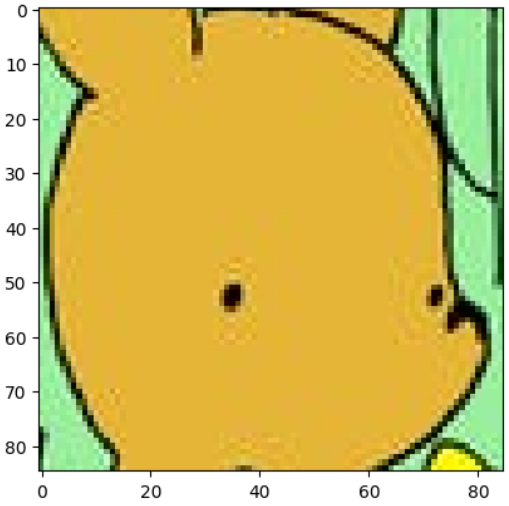


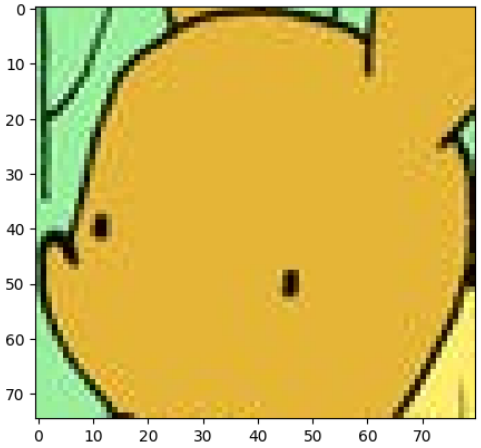
B. COUNTING

1. Chuẩn bị đầu vào

- Vật thể trong ảnh có sự khác nhau (VD: con thỏ quay sang phải/trái, các con cú có màu sắc khác nhau, đôi ủng bị che khuất,...)
- Cần chuẩn bị nhiều template hơn để khắc phục sự sai khác này
- VD:







2. Counting function

```python
def counting(image, temps, threshold, new_color, canny=False, match_thresh=0.8):
    boxes = list()
    for idx, tmp in enumerate(temps, 0):
    # save the image dimensions
        H, W = tmp.shape[:2]

    # Define a minimum threshold
        img_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        temp_gray = cv2.cvtColor(tmp, cv2.COLOR_BGR2GRAY)
        temp_gray[temp_gray > threshold] = new_color

        if canny:
            img_gray = cv2.Canny(img_gray, 50, 150)
            temp_gray = cv2.Canny(temp_gray, 50, 150)

        match = cv2.matchTemplate(image=img_gray, templ=temp_gray, method=cv2.TM_CCOEFF_NORMED)
        (y_points, x_points) = np.where(match >= match_thresh[idx])
        for (x, y) in zip(x_points, y_points):
            boxes.append((x, y, x + W, y + H))

    boxes = non_max_suppression(np.array(boxes))
    copy = image.copy()
    for (x1, y1, x2, y2) in boxes:
        cv2.rectangle(copy, (x1, y1), (x2, y2),
                      (0, 0, 0), 5)
    plt.imshow(copy)
    print(f'Number of object:{len(boxes)}')
```

✓ 0.0s Python

- Matching tương tự như ở phần 1
- Sau khi match, lấy ra vị trí của các vật thể có độ tương đồng ở một ngưỡng nhất định và vẽ khung bao quanh các vật thể
- Khi lấy vị trí sẽ xảy ra trường hợp các khung khác nhau bao quanh cùng một vật thể, nên ở đây dùng imutils.object_detection.non_max_suppression() để khắc phục

3. Gọi hàm count

```
counting(image_3, temps, 200, 200, match_thresh=[0.19, 0.179, 0.3])
```

[1505] ✓ 0.3s

```
Number of object:14
```