Báo cáo bài tập giữa kỳ xử lý ảnh

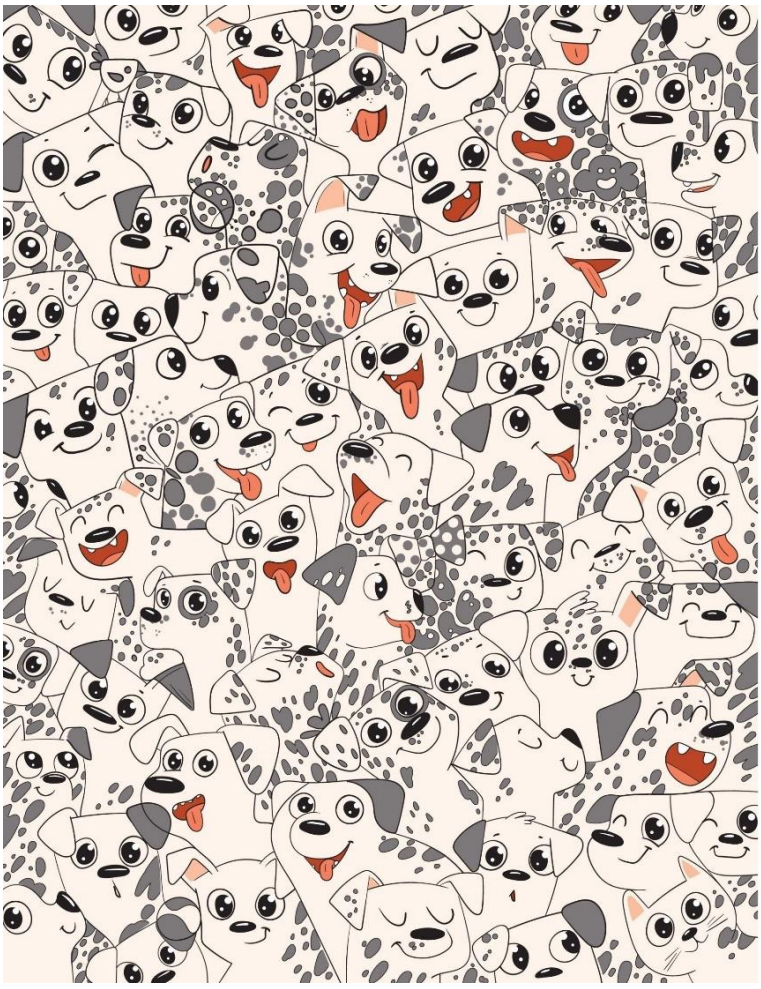
Họ và tên: Lê Thị An

Mã sinh viên: 19020204

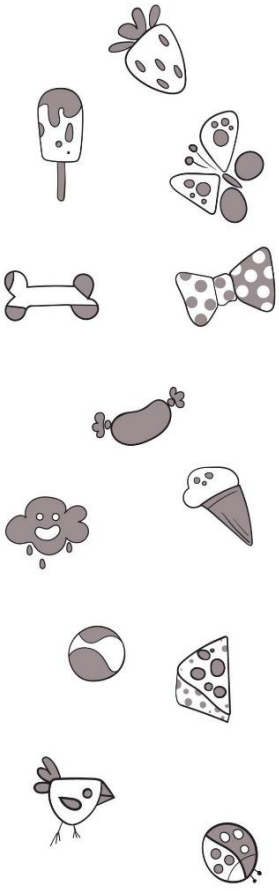
I. Bài toán: Finding

Cho một source image và objects như hình sau, cần xác định vị trí của objetcs trên source image

Souce Image



objects



1. Phương pháp giải quyết bài toán: Sử dụng multi-scale template matching

- Để thực hiện Template Matching chúng ta cần:
  + Source Image: Ảnh chứa objects cần tìm
  + Template Image: lần lượt là các object dùng để tìm kiếm trong Source Image.
- Template Image sẽ được "slide" từ trái sang phải, từ trên xuống dưới trong Source Image và tại mỗi vị trí, một số liệu sẽ được tính toán để thể hiện mức độ phù hợp của Template.
- Khi sửa dụng template matching để phát hiện đối tượng trong ảnh thì template phải đảm bảo giống với object trong ảnh cả về kích thước độ nghiêng, ... Nếu có sự khác biệt quá lớn sẽ không thể phát hiện được object đó
- Như bài toán đã nêu trên thì kích thước của object được cung cấp dùng làm template có kích thước khác với kích thước của object cần tìm trong source image (cụ thể là kích thước của template lớn hơn kích thước của object trong source image)

  ➔ Sử dụng image pyramid , tiến hành resize template thành nhiều kích thước khác nhau, sau đó matching với source image. Cụ thể:

  + Lặp lại template ở nhiều tỷ lệ (làm cho template nhỏ dần).

  + Áp dụng template matching và theo dõi đối sánh có hệ số tương quan lớn nhất (cùng với tọa độ x, y của vùng có hệ số tương quan lớn nhất).

  + Sau khi lặp qua tất cả các scale, lấy vùng có hệ số tương quan lớn nhất và sử dụng vùng đó làm vùng "phù hợp".

- Sau khi đã tiến hành resize template thành nhiều kích thước khác nhau rồi tiến hành matching với source image thì có một số trường hợp không thể xác định được object.

  ➔ Tiến hành detect edges cả template và souce image trước khi matching. Áp dụng template matching bằng cách sử dụng các cạnh thay vì sử dụng ảnh thô thì độ chính xác khi detect object cao hơn.

2. Cài đặt

Xử lí ảnh đầu vào:

- Đọc Source image và template
- Chuyển source image và template sang grayscale.
- Tiến hành detect egdes cả template và source image.

Template matching:

sử dụng **cv2.matchTemplate(image, template, method) với:**

- image: source imagecó chứa đối tượng muốn phát hiện
- template: Mẫu của đối tượng
- Method: Phương pháp dùng để matching. OpenCv cung cấp các method sau:
  - **cv2.TM_SQDIFF** : Sum of Squared Differences (SSD)
  - **cv2.TM_SQDIFF_NORMED**: Sum of Squared Differences Normalized (SSD)
  - **cv2.TM_CCOEFF** : Correlation Coefficient Definition
  - **cv2.TM_CCOEFF_NORMED**: Correlation Coefficient Normalized
  - **cv2.TM_CCORR** : Cross-Correlation
  - **cv2.TM_CCORR_NORMED**: Normalized Cross Correlation

Cụ thể trong bài: sử dụng method **cv2.TM_CCOEFF_NORMED**

Detect cạnh: dùng cv2.Canny(image, threshold1, threshold2)

Các tham số cần thiết là:

- image: Nguồn / Ảnh đầu vào.
- threshold1: Đây là giá trị ngưỡng Cao của intensity gradient.
- threshold2: Đây là giá trị ngưỡng Thấp của intensity gradient.

Detect object:

Sử dụng hàm **cv2.minMaxLoc,** kết quả trả về gồm có:

- minimum correlation value

+ maximum correlation value
+ (x, y)-coordinate of the minimum value
+ (x, y)-coordinate of the maximum value

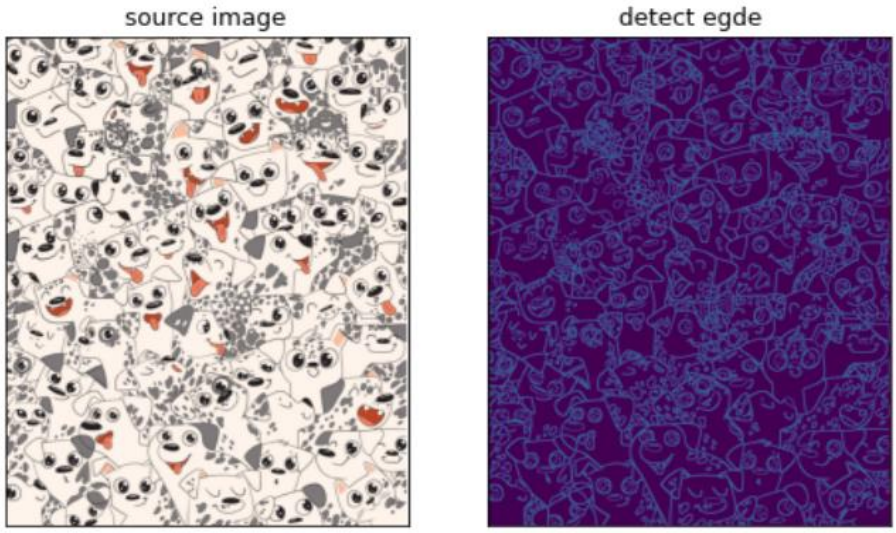
Tùy theo method được lựa chọn ở trên, ta sẽ quyết định sử dụng minimum hay maximum value để khoanh vùng đối tượng.

Do trong bài toán này method được sử dụng là: **cv2.TM_CCOEFF_NORMED** nên giá trị được chọn là **maximum**

3. Mã nguồn cụ thể và kết quả thu được.

+ **Đọc souce image, detect cạnh của source image.**

```
#read image
image= cv2.imread('cho.jpg')

#convert color image to grayscale
img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
img = image.copy()

#detect egde.
i = cv2.Canny(img, 50, 200)

#show
plt.figure(figsize=(8,6))
plt.subplot(121),plt.imshow(image[:,:,::-1])
plt.title('source image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(i)
plt.title('detect egde'), plt.xticks([]), plt.yticks([])
plt.show()
```



+ **Hàm đọc, detect cạnh template**

```
#read template with grayscale and dectect egde of template.
def read_template(template_name):
    template = cv2.imread(template_name, 0)
    t = cv2.Canny(template, 50, 200)
    return t
```

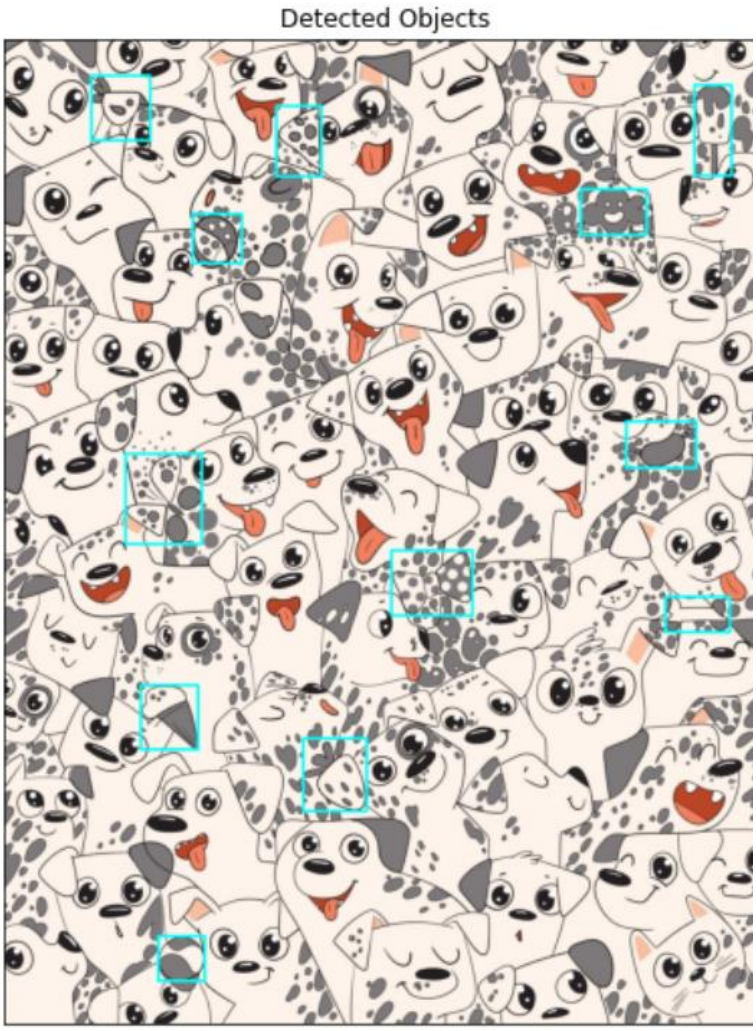
+ **Hàm finding object.**

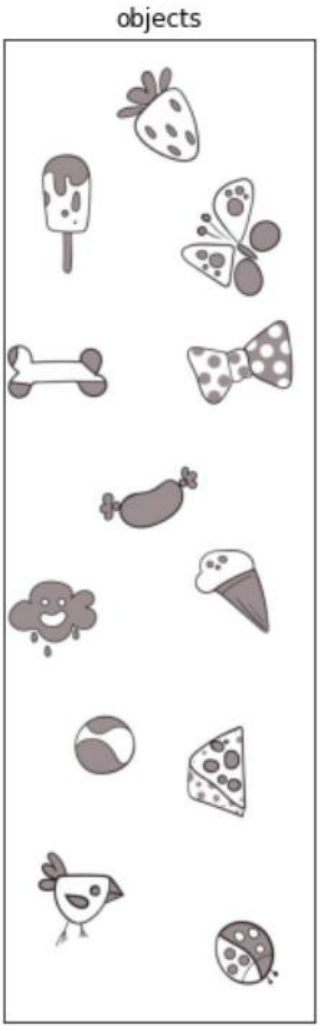
```
def match_template(t, i, img_copy, method):
    found = None
    w, h = t.shape[::-1]
    for scale in np.linspace(0.2, 1.0, 20)[::-1]:

        #resize image according to the scale
        resized = imutils.resize(t, width = int(t.shape[1]*scale))

        # ratio of the resizing
        r = t.shape[1] / float(resized.shape[1])

        # Apply template Matching
        res = cv2.matchTemplate(i,resized,method)
        min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)

        if found is None or max_val > found[0]:
            found = (max_val, max_loc, r)

    ## draw a bounding box around the detected region
    (max_val, max_loc, r) = found
    (startX, startY) = (int(max_loc[0]), int(max_loc[1]))
    (endX, endY) = (int((max_loc[0] + w/r)), int((max_loc[1] + h/r)))
    cv2.rectangle(img, (startX, startY), (endX, endY), (255, 255, 0), 5)
```

+ **Đọc vào các template, tiến hành xác định vị trí của các object.**

```
#read template from templates list, find object in source image.
templates=['1.jpg','2.jpg','3.jpg','4.jpg','5.jpg','6.jpg','7.jpg','8.jpg','9.jpg','10.jpg','11.jpg','12.jpg']
for temp in templates:
    name=''.join(str(temp))
    t= read_template(name)
    match_template(t,i,img,cv2.TM_CCOEFF_NORMED)
```

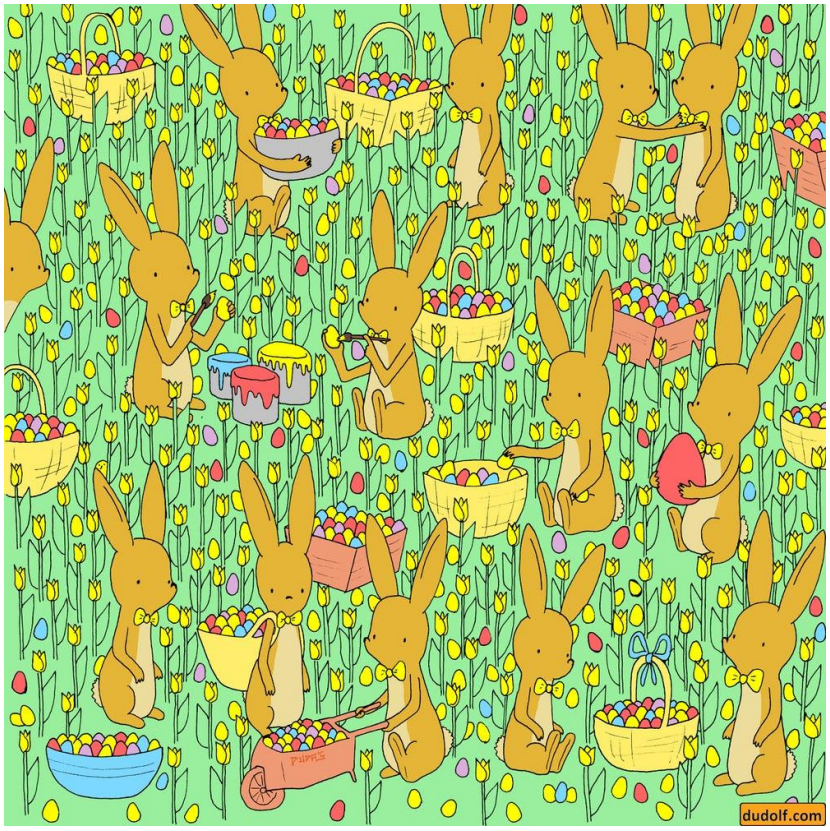
+ **Kết quả thu được: xác định được 12/12 object.**





II. Bài toán counting.

Cho ảnh sau, đếm xem trong ảnh có bao nhiêu con thỏ.



1. Phương pháp giải quyết bài toán

- Sử dụng template matching, với

    + Source image: ảnh đàn thỏ được cung cấp
    + Template: chọn mắt của con thỏ làm template

    + Method: **cv2.TM_CCOEFF_NORMED**

- sử dụng threshold để chọn nhiều đối tượng phù hợp, cụ thể chọn threshold = 0.85.
- để có thể đếm được có bao nhiêu con thỏ thì cần: Loại bỏ các điểm gần vị trí (x, y) là lần phát hiện đầu tiên của một đối tượng, chỉ giữ lại một điểm của một lần phát hiện đối tượng. Tức là một đối tượng sẽ chỉ được đánh dấu một lần duy nhất

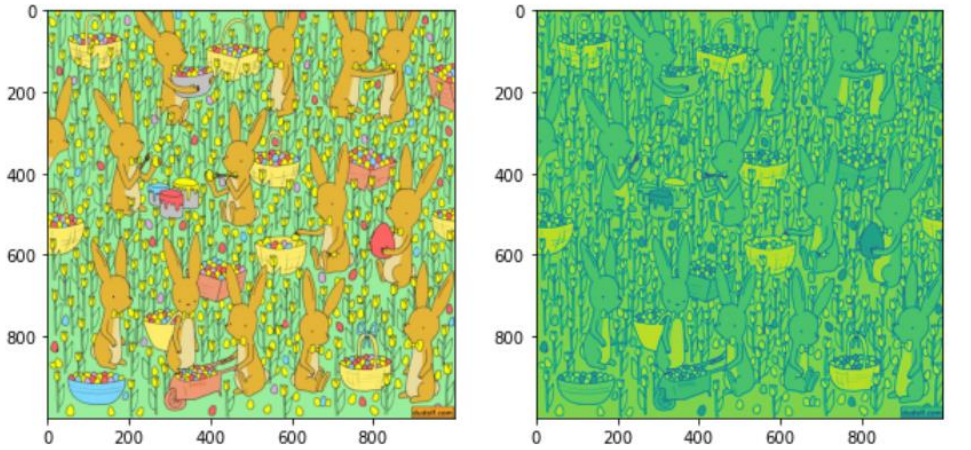
- dùng một mảng lưu lại toàn bộ đối tượng vừa được đánh dấu, kích thước của mảng là số lượng đối tượng cần đếm.

2. Cài đặt và kết quả thu được

Xử lí ảnh đầu vào:

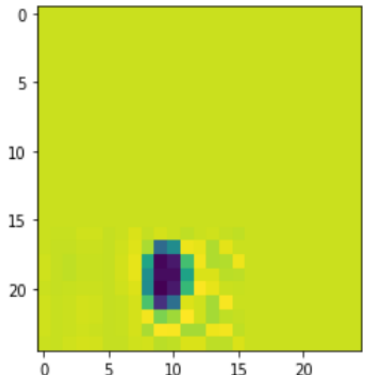
- Đọc Source image và template
- Chuyển source image và template sang grayscale

```
#read image
img_count = cv2.imread('rabbit.jpg')

#convert image into grayscale image
img_count_gray = cv2.cvtColor(img_count, cv2.COLOR_BGR2GRAY)

plt.figure(figsize=(10,10))
plt.subplot(121)
plt.imshow(img_count[:,:,::-1])
plt.subplot(122)
plt.imshow(img_count_gray)
plt.show()
```




```
#read the template, convert template to gray
template_count = cv2.imread('eye.jpg', 0)
plt.imshow(template_count)
```

```
<matplotlib.image.AxesImage at 0x1dfe56d1ee0>
```



- Hàm kiểm tra sự ‘giao nhau’ của các kết quả:

```
#kiểm tra sự giao nhau của các kết quả.
def intersected(bottom_left1, top_right1, bottom_left2, top_right2):
    if top_right1[0] < bottom_left2[0] or bottom_left1[0] > top_right2[0]:
        return 0
    if top_right1[1] < bottom_left2[1] or bottom_left1[1] > top_right2[1]:
        return 0
    return 1
```

- Áp dụng template matching, khoanh vùng đối tượng, đếm số lượng thỏ.

```
wt, ht = template_count.shape[::-1]

# Apply template Matching
res = cv2.matchTemplate(img_count_gray,template_count, cv2.TM_CCOEFF_NORMED)

#threshold
threshold = 0.85
loc = np.where(res >= threshold)

#để có thể đếm được có bao nhiêu con thỏ thì cần:
#Loại bỏ các điểm gần vị trí (x, y) là lần phát hiện đầu tiên của một đối tượng
#Chỉ giữ lại một điểm của một lần phát hiện đối tượng
#Tức là một đối tượng sẽ chỉ được đánh dấu một lần duy nhất
#làm như vậy với tất cả các vị trí trong loc.

#tạo một list chứa tất cả các kết quả phù hợp
#kiểm tra xem có điểm giao nhau giữa các kết quả trong danh sách hay không:
objects = []
for pt in zip( * loc[::-1]):
    is_one_object = 0
    for obj in objects:
        if intersected(obj, (obj[0] + wt, obj[1] + ht), pt, (pt[0] + wt, pt[1] + ht)):
            is_one_object = 1
            break
    if is_one_object == 0:
        objects.append(pt)
    rect = cv2.rectangle(image_copy, pt, (pt[0] + wt, pt[1] + ht), (255, 255, 0), 5)

#show result.
print("Số con thỏ có trong hình là: ",len(objects))
```

- **Kết quả thu được**

```
Số con thỏ có trong hình là:  14
```