BÁO CÁO BÀI TẬP FIND & COUNT SỬ DỤNG TEMPLATE MATCHING

Tags

Bài toán

Sử dụng Template Matching để đếm về tìm kiếm các vật thể, đối tượng trong hình ảnh

Kỹ thuật sử dụng

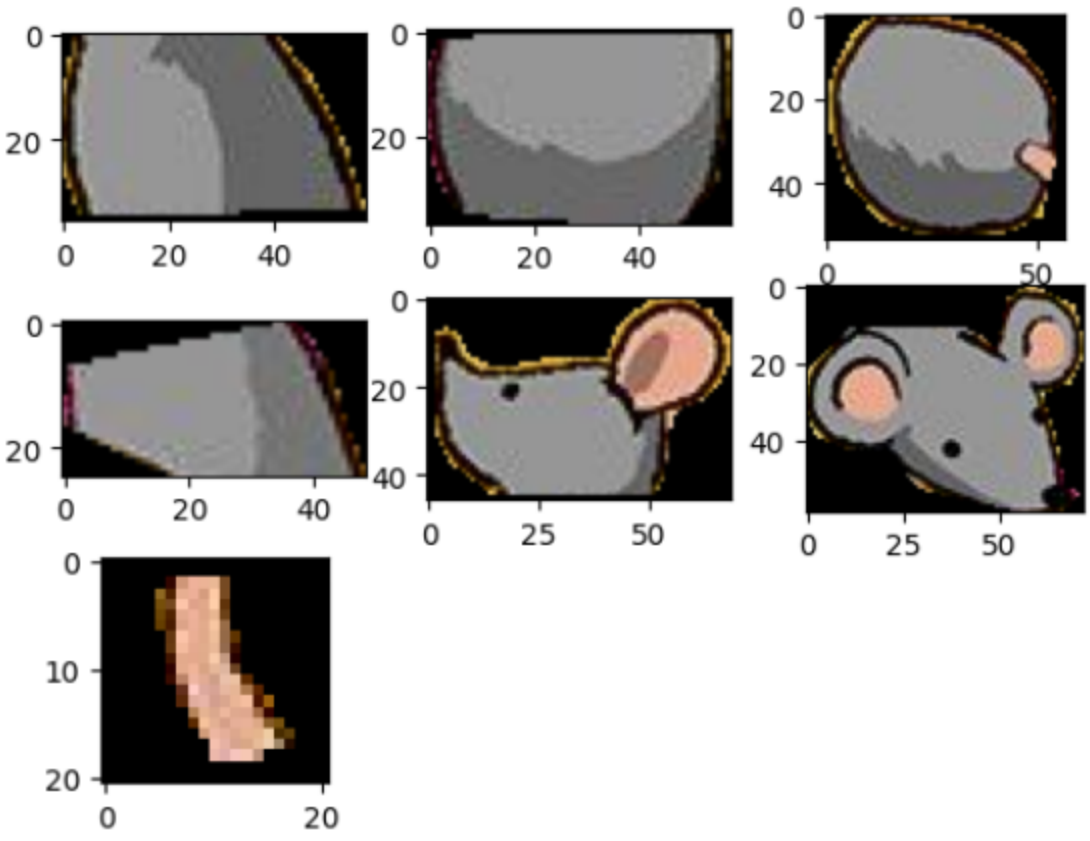
- Template Matching: cv2.matchTemplate (normalized correlation coefficient)
- Edge Detection: cv2.canny
- Filtering: cv2.GaussianBlur, cv2.threshold

Counting

Quy trình chung

1. Load các ảnh, template và chuyển về dạng gray
2. Processing
    a. Edge detection , convert to binary image
    b. flip, rotate template, resize
3. Template matching
4. Reduce bounding box

Load ảnh và template

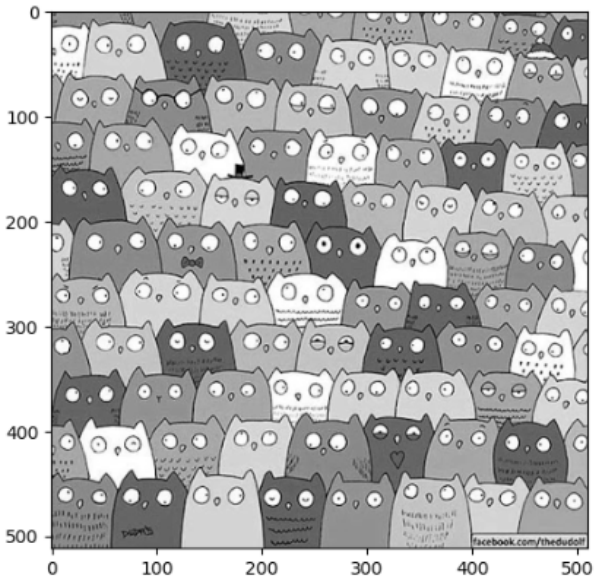


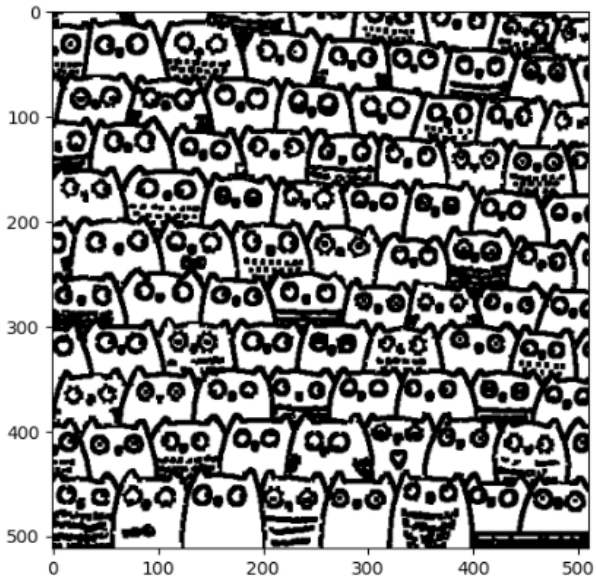
Xử lý ảnh và template

Edge detection, convert to binary image

sử dụng cv2.canny nhằm phát hiện các cạnh

sử dụng cv2.GaussianBlur và cv2.threshold để làm mượt các cạnh



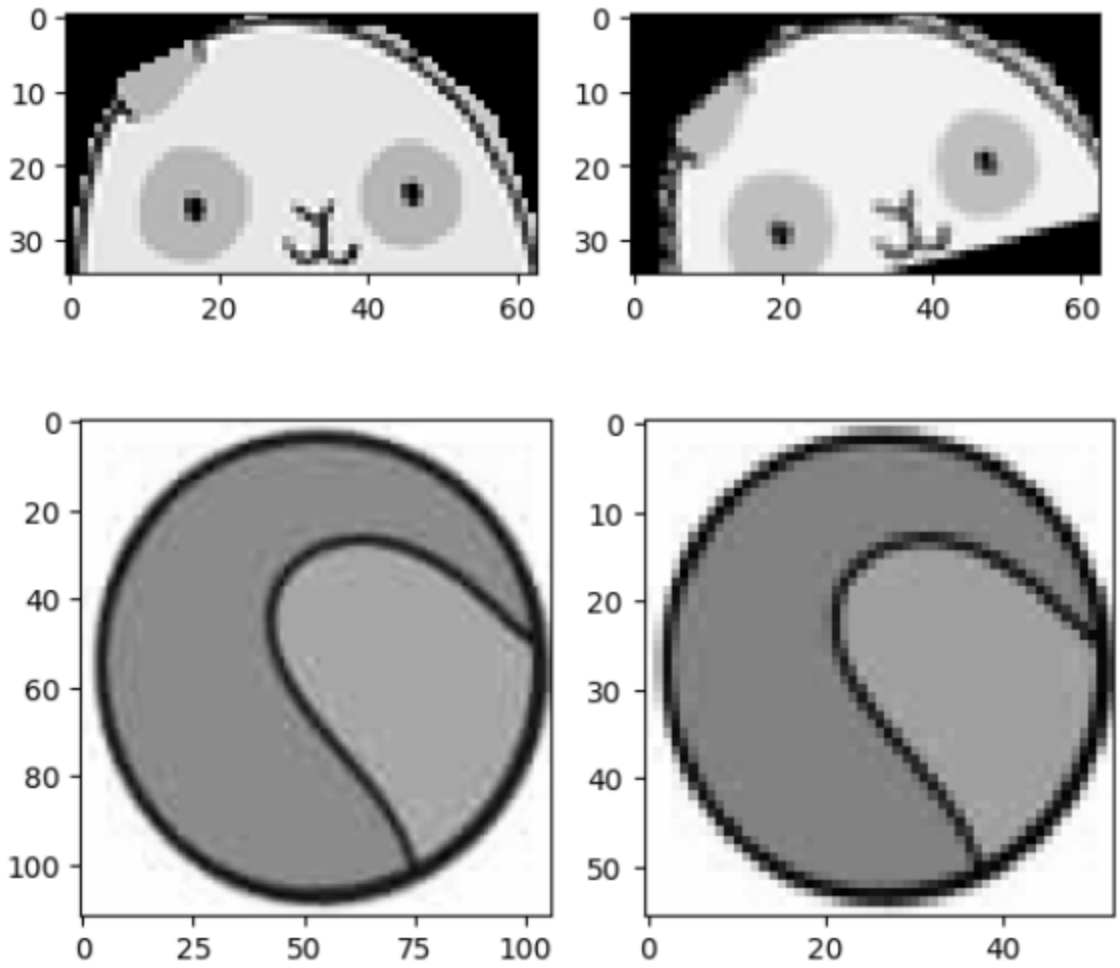


sử dụng cv2.canny ⇒ loại bỏ các màu giúp việc đếm dễ dàng hơn

```
def eliminate_color(image: np.array):
    """
    Input: gray image

    Output: binary image

    Process:
    detect edge using canny edge detection
    revert the color of the edge
    thresholding
```

```
    """
    edges = cv2.Canny(image, 127, 255)
    edges = cv2.GaussianBlur(255 - edges, (3, 3), 0)
    _, binary_img = cv2.threshold(edges, 240, 255, cv2.THRESH_BI

    return binary_img
```

Flip, Rotate và Resize



Template matching

Dùng các công cụ xử lý đã được xây dựng (Flip, Rotate, Resize), thực hiện thay đổi các template

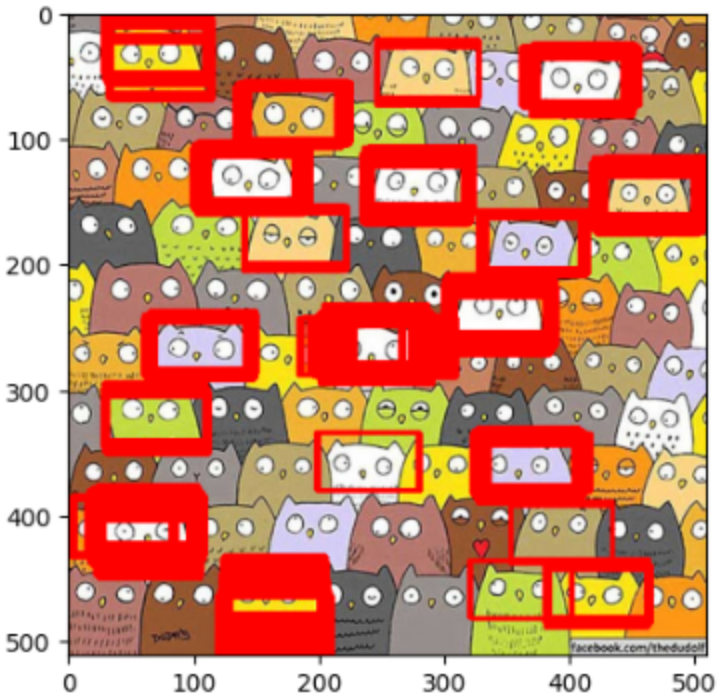
Sử dụng Normalized Cross Correlation Coefficient để phát hiện đối tượng

```
res = cv2.matchTemplate(image_gray, templates_gray[1], cv2.TM_C
threshold = sorted(res.flatten())[int(0.99 * len(res.flatten()))
loc = np.where(res >= threshold)
x, y = loc[::-1]
res_image = image_rgb.copy()
w, h = templates_gray[0].shape[::-1]
for bb in zip(*loc[::-1]):
    cv2.rectangle(res_image, bb, (bb[0] + w, bb[1] +h), (0,0,255
plt.imshow(res_image[:, :, ::-1])
```

Giảm số lượng bounding box

Nhận thấy có nhiều hình ảnh (đối tượng) ko phát hiện được, đi kèm với đó sự trùng lặp của các bounding box

⇒ phải kết hợp nhiều template và loại bỏ các bounding box trùng nhau



Thuật toán:

- Tạo 1 list các bounding box (bboxes) với các phần tử có dạng (score, x, y, w, h)

score: giá trị ở trong matching map

x, y: tọa độ

w, h: kích thước bounding box

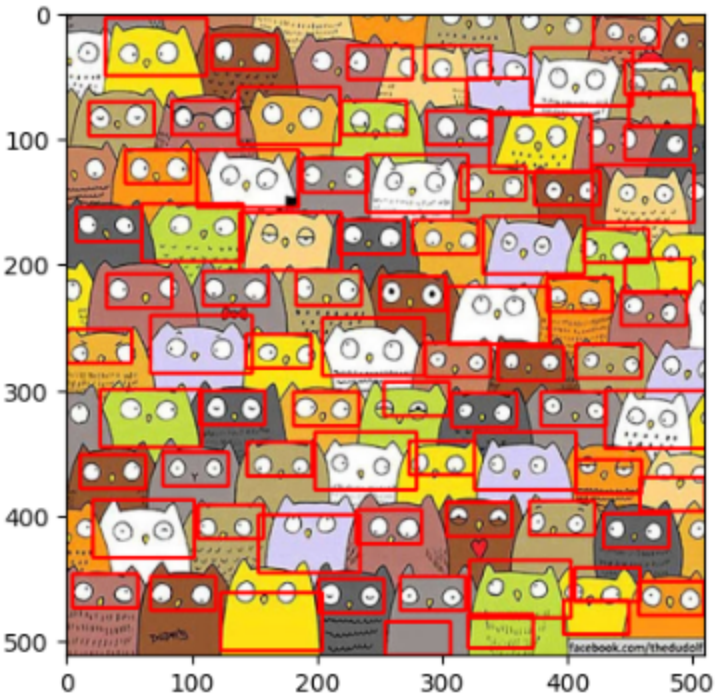
- Thực hiện tính toán lại score của các bounding box: Do khi hình ảnh được chuyển về dạng binary (chỉ chứa các điểm ảnh có giá trị 0 và 256), việc sử dụng template matching để tìm kiếm vật thể gây ra việc phát hiện 1 số đối tượng có màu khác. Vì vậy cần thực hiện tính toán để giảm score của các đối tượng bị lệch đã được nêu trên

Lấy trung bình score và SSD (sum of squared differences)

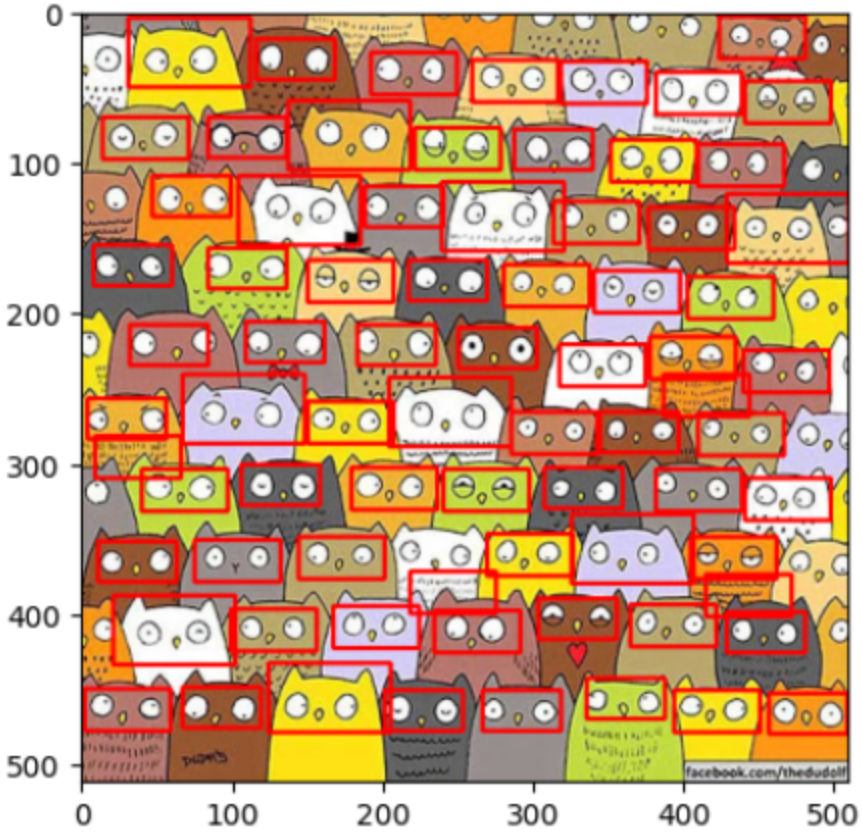
- Sắp xếp bboxes, lấy phần trăm hoặc số lượng phần tử nhất định (ví dụ: lấy 100 bounding box có chỉ số cáo nhất)
- Lấy dần các bounding box có điểm số cao, khi xét 1 bounding box mới sẽ kiểm tra xem có bị trùng lặp với các bounding box đã được xác định

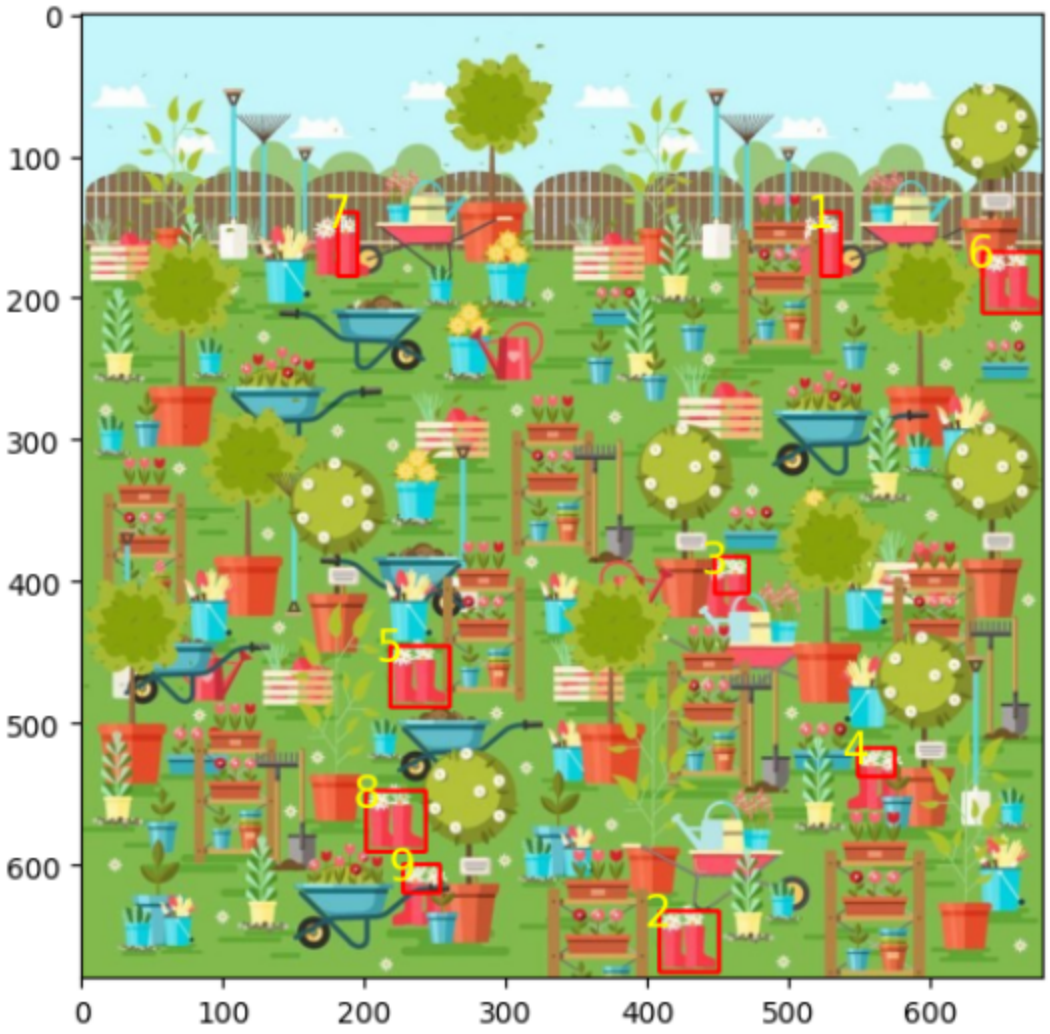
```
def check_overlap(box1, box2, overlap_threshold=0.5):
    """
    Input:
    box1: Tuple (x1, y1, w1, h1) of the first bounding box
    box2: Tuple (x2, y2, w2, h2) of the second bounding box
    """
    x1, y1, w1, h1 = box1
    x2, y2, w2, h2 = box2

    # Calculate the coordinates of the intersection rectangle
    x_left = max(x1, x2)
```

```
    y_top = max(y1, y2)
    x_right = min(x1 + w1, x2 + w2)
    y_bottom = min(y1 + h1, y2 + h2)

    # Check if there is an intersection
    if x_right < x_left or y_bottom < y_top:
        return False

    intersection_area = (x_right - x_left) * (y_bottom - y_top)

    # Calculate the area of the first bounding box
    box1_area = w1 * h1

    # Calculate the overlap ratio
    overlap_ratio = intersection_area / box1_area

    # Return True if the overlap ratio is greater than or equal
    return overlap_ratio >= overlap_threshold
```



Kết quả


0
100
200
300
400
500
0
100
200
300
400
500
facebook.com/thedudolf



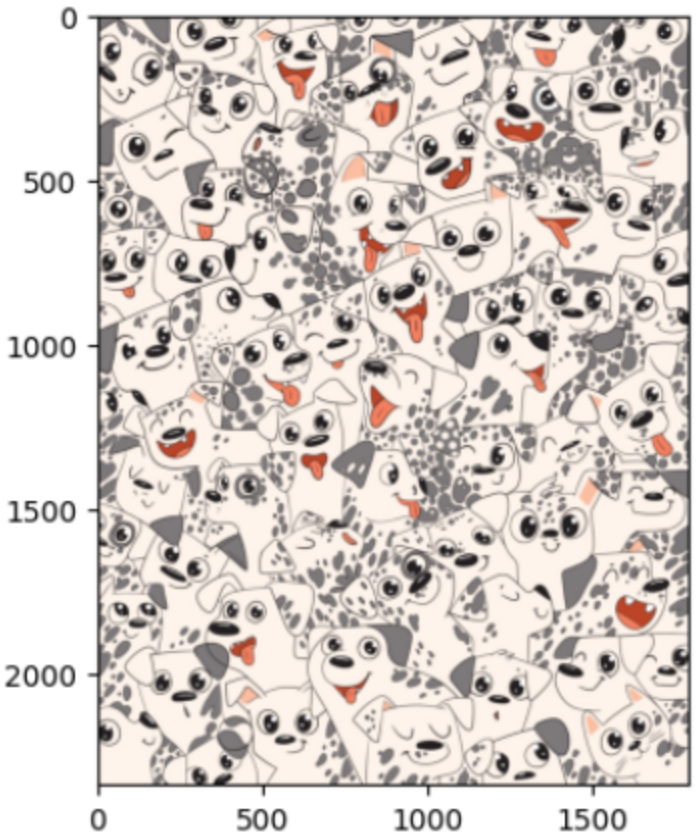
Finding

Thuật toán

1. Load hình ảnh và các template
2. Với mỗi loại template
    - Thực hiện resize template theo tỷ lệ từ 0.5 đến 1 (0.5, 0.55, ...1)
    - Sử dụng Template Matching với phương pháp normalized cross correlation
    - Tìm kiếm tỷ lệ thu được điểm số cao nhất
    - Đánh giấu vị trí trong hình ảnh tương đương với điểm số cao nhất đó

3. Vẽ hình ảnh với các bounding box của các vật thể cần tìm kiếm

Kết quả




1
7
8
6
4
10
5
3
2
9
11
0