

Báo cáo bài toán sử dụng kỹ thuật xử lý ảnh để đếm vật thể trong ảnh

Nguyễn Minh Hường - 22022542

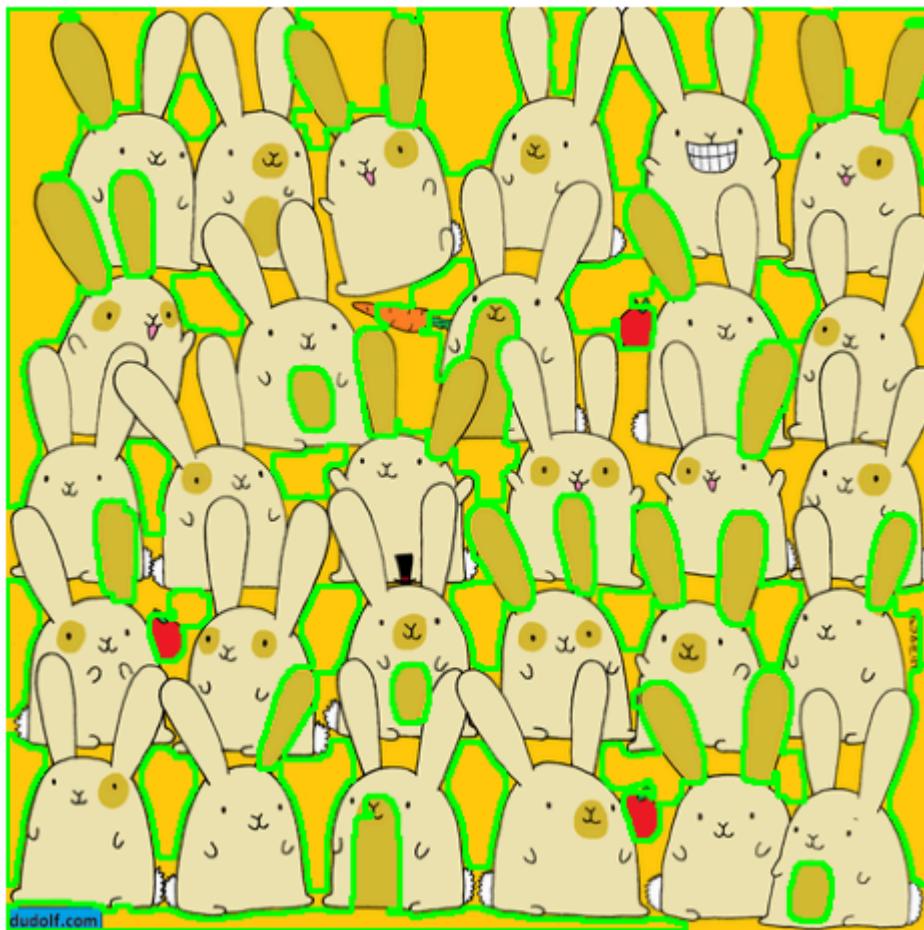
[Github\[link\]](#)

[Collab\[link\]](#)

1. Cách tiếp cận đầu tiên

- Bước 1: Tải và hiển thị hình ảnh gốc
- Bước 2: Xác định một vùng quan tâm (ROI) trong hình ảnh, là khu vực chứa một con thỏ mẫu. Tọa độ và kích thước của vùng này được xác định thủ công ('x, y, w, h'). Vùng này được cắt ra từ hình ảnh gốc bằng cách sử dụng `image[y:y+h, x:x+w]` để tạo một mẫu (template) cho việc so khớp mẫu sau này. Bước này rất quan trọng vì nó cung cấp mẫu để tìm các con thỏ tương tự trong hình ảnh.
- Bước 3: Áp dụng kỹ thuật so khớp mẫu (Template Matching). Sử dụng hàm `template_matching` để tìm các vùng trong hình ảnh gốc khớp với mẫu thỏ đã cắt. Hàm này sử dụng `cv2.matchTemplate` với phương pháp `TM_CCOEFF_NORMED` để tính độ tương đồng giữa mẫu và các vùng trong hình ảnh. Nguồn: Các nguồn `down_threshold` và `up_threshold` được sử dụng để lọc các vị trí có độ tương đồng phù hợp. Kết quả trả về là danh sách các tọa độ (`locations`) của các vùng khớp với mẫu.
- Bước 4: Đánh dấu và che các vùng đã khớp. Sau khi tìm được các vị trí khớp, sử dụng hàm `draw` để vẽ các hình chữ nhật màu đen lên các vùng đã khớp, nhằm che chúng đi và tránh việc phát hiện trùng lặp. Điều này được thực hiện bằng cách sử dụng `cv2.rectangle` với tham số `thickness=-1` để tô kín hình chữ nhật. Bước này giúp đảm bảo rằng mỗi con thỏ chỉ được đếm một lần.
- Bước 5: Trả về kết quả
 - Hàm `process_match` được sử dụng để quản lý toàn bộ quy trình:
 - Cắt vùng ROI và gọi hàm `template_matching`.
 - Quản lý danh sách các vị trí đã phát hiện (`locations`) với tùy chọn `inheritance` để tích lũy các vị trí từ nhiều lần gọi hàm.
 - Trả về hình ảnh đã được đánh dấu và danh sách các vị trí khớp.
 - Kết quả cuối cùng là số lượng thỏ được đếm bằng cách lấy độ dài của `locations[0]`, tương ứng với số lượng vị trí khớp.

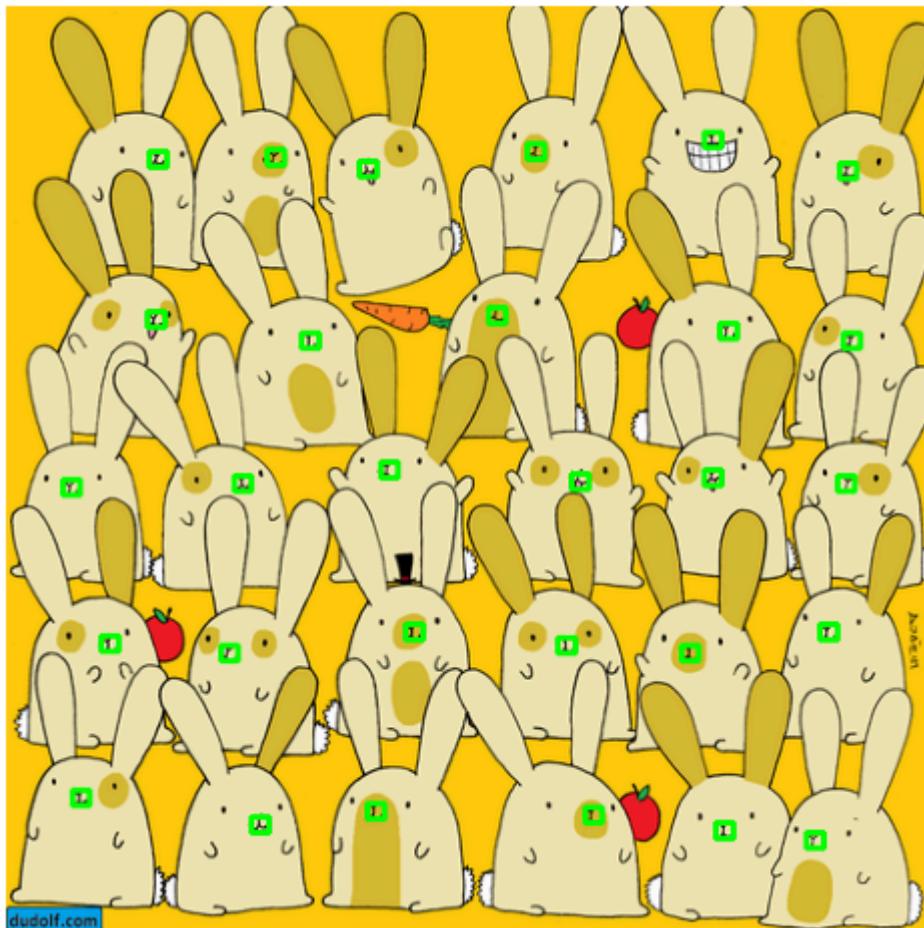
Step 5: Final Result - 31 Rabbits Detected



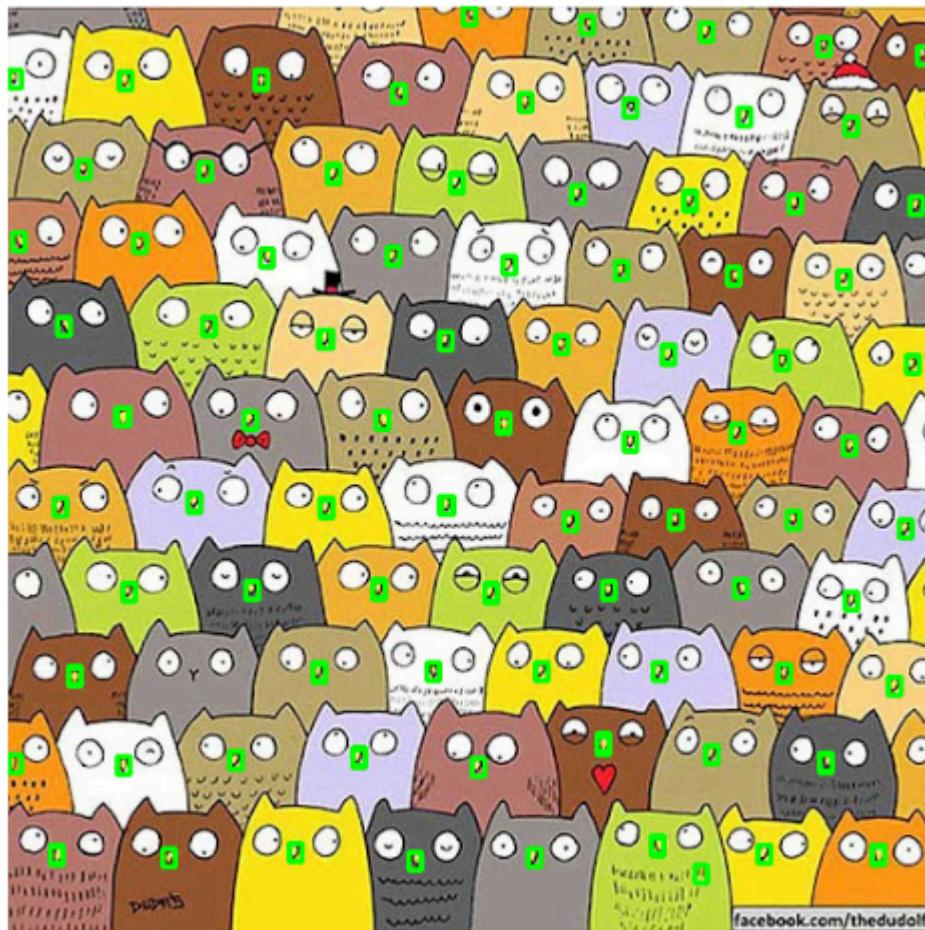
2. Cách 2

- Bước 1: Chuyển đổi ảnh sang ảnh xám và phát hiện cạnh:
 - Bắt đầu bằng cách chuyển đổi hình ảnh gốc sang ảnh xám bằng `cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)`.
 - Sau đó, áp dụng thuật toán phát hiện cạnh Canny (`cv2.Canny`) với các ngưỡng `low_threshold=50` và `high_threshold=150`. Bước này tạo ra một hình ảnh nhị phân chỉ chứa các cạnh, giúp làm nổi bật các đường viền của thỏ và giảm nhiễu từ nền.
- Bước 2: Áp dụng so khớp mẫu trên nhiều vùng ROI:
 - Dùng hàm `process_match` nhiều lần với các tọa độ ROI khác nhau ('x, y') và kích thước vùng ('w, h' hoặc 'w1, h1'). Ví dụ: `(x, y) = (422, 320)` với `w, h = 9, 8` và `down_threshold=0.71` ; `(x, y) = (183, 80)` với `w1, h1 = 9, 13` và `down_threshold=1` .
 - Sử dụng tùy chọn `rotate_temp=True` với các tham số `min_angle=-30`, `max_angle=30`, và `num_angles=51` để xoay mẫu ở nhiều góc độ, giúp phát hiện các con thỏ có hướng khác nhau.

- Các ngưỡng `down_threshold` và `up_threshold` được điều chỉnh linh hoạt (ví dụ: từ 0.66 đến 1) để phù hợp với từng vùng ROI.
- Bước 3: Hiển thị và đếm kết quả: Sau khi gọi `process_match` nhiều lần, sử dụng hàm `draw` để vẽ các hình chữ nhật màu xanh lá cây (`color=(0, 255, 0)`, `thickness=2`) lên hình ảnh gốc tại các vị trí khớp (`locations`). Số lượng thỏ được tính bằng `len(locations[0])` và in ra kết quả: `Số lượng thỏ: 29`.



- Tương tự với ảnh đếm mèo nhưng sửa các tham số từng vùng
 - Dùng hàm process_match nhiều lần với các tọa độ ROI khác nhau (x, y) và kích thước vùng (w, h). Ví dụ: (x, y) = (61, 36) với w, h = 7, 10 và không sử dụng ngưỡng cụ thể; (x, y) = (27, 172) với w, h = 7, 10 và down_threshold=0.8. Các tọa độ khác như (x, y) = (325, 403), (203, 224), (212, 43), (340, 51), (0, 414), và (0, 35) cũng được sử dụng với cùng kích thước vùng w, h = 7, 10, trong đó các vị trí (0, 414) và (0, 35) có down_threshold=0.8.
 - Không sử dụng tùy chọn rotate_temp, do đó không áp dụng xoay mẫu để phát hiện các con mèo ở nhiều góc độ.
 - Các ngưỡng down_threshold được điều chỉnh linh hoạt (ví dụ: từ 0.8 đến giá trị mặc định) để phù hợp với từng vùng ROI, trong khi up_threshold không được chỉ định cụ thể và sử dụng giá trị mặc định.
 - Kết quả số lượng mèo: 105



- Với bài toán đếm đôi ứng:
 - Dùng hàm process_match nhiều lần với các tọa độ ROI khác nhau (x, y) và kích thước vùng (w, h). Ví dụ: $(x, y) = (205, 561)$ với $w, h = 28, 27$ và $\text{down_threshold}=0.54, \text{up_threshold}=1$; $(x, y) = (444, 400)$ với $w, h = 28, 27$ và $\text{down_threshold}=0.9, \text{up_threshold}=1$. Các tọa độ khác như $(x, y) = (541, 532)$ và $(228, 614)$ cũng được sử dụng với cùng kích thước vùng $w, h = 28, 27$ và $\text{down_threshold}=0.9, \text{up_threshold}=1$.
 - Không sử dụng tùy chọn rotate_temp, do đó không áp dụng xoay mẫu để phát hiện các đôi ứng ở nhiều góc độ.
 - Các ngưỡng down_threshold và up_threshold được điều chỉnh linh hoạt (ví dụ: từ 0.54 đến 0.9 cho down_threshold, và up_threshold luôn là 1) để phù hợp với từng vùng ROI.
 - Số lượng đôi ứng: 8



3. Sự khác biệt giữa phương pháp ở mục 1 và mục 2

Phương pháp ban đầu sử dụng so khớp mẫu (template matching) với một quy trình cơ bản, nhưng cách 2 đã cải tiến đáng kể. Dưới đây là các điểm khác biệt chính:

- Phát hiện cạnh (Canny Edge Detection): cách 2 đã thêm bước phát hiện cạnh bằng Canny trước khi áp dụng so khớp mẫu. Điều này giúp làm nổi bật các đường viền của thỏ và loại bỏ phần lớn nhiễu từ nền (ví dụ: màu sắc hoặc các chi tiết nhỏ). Hình ảnh cạnh nhị phân (`image_edges`) là đầu vào cho `process_match`, giúp tăng độ chính xác của việc so khớp mẫu.
- Sử dụng nhiều vùng ROI với tham số linh hoạt:
 - Cách 1 chỉ sử dụng một vùng ROI duy nhất để cắt mẫu và áp dụng so khớp mẫu trên toàn bộ hình ảnh. Điều này có thể bỏ sót các con thỏ có kích thước hoặc hướng khác nhau.
 - Cách 2 đã gọi `process_match` nhiều lần với các tọa độ ROI khác nhau (`x, y`) và kích thước vùng (`w, h` hoặc `w1, h1`) được điều chỉnh. Ví dụ: `(x, y) = (183, 80)` với `w1, h1 = 9, 13`; `(x, y) = (362, 240)` với `w1, h1 = 9, 12`. Điều này cho phép chọn nhiều mẫu đại diện cho các con thỏ có kích thước và hướng khác nhau, tăng khả năng phát hiện.
- Tinh chỉnh tham số so khớp mẫu:

- Cách 1 sử dụng các tham số mặc định hoặc cố định cho `down_threshold`, `up_threshold`, `min_angle`, `max_angle`, và `num_angles`, không điều chỉnh linh hoạt theo từng trường hợp.
- Cách 2 đã tinh chỉnh các tham số này cho từng lần gọi `process_match`. Ví dụ: `down_threshold` thay đổi từ 0.66 đến 1 tùy theo vùng ROI. `rotate_temp=True` với `min_angle=-30`, `max_angle=30`, và `num_angles=51` để xoay mẫu ở nhiều góc độ, giúp phát hiện các con thỏ có hướng khác nhau. Điều này làm tăng độ chính xác khi các con thỏ trong hình ảnh có thể không đồng nhất về kích thước hoặc góc xoay.

4. Tại sao cách 2 cho kết quả tốt hơn?

- Giảm nhiễu bằng phát hiện cạnh: Việc sử dụng Canny Edge Detection giúp loại bỏ các chi tiết không cần thiết trong hình ảnh (như màu sắc nền hoặc các vùng nhiễu nhỏ), chỉ giữ lại các đường viền quan trọng của thỏ. Điều này làm cho việc so khớp mẫu trở nên chính xác hơn, vì mẫu chỉ cần khớp với các đường viền thay vì toàn bộ vùng ảnh (có thể chứa nhiễu).
- Tăng khả năng phát hiện với nhiều mẫu: Bằng cách sử dụng nhiều vùng ROI với các tọa độ và kích thước khác nhau đã tạo ra nhiều mẫu đại diện cho các con thỏ có kích thước và hướng khác nhau. Điều này giúp phát hiện được nhiều con thỏ hơn, đặc biệt là những con có kích thước hoặc góc xoay không giống với mẫu ban đầu.
- Tinh chỉnh tham số linh hoạt:
 - Việc điều chỉnh `down_threshold` và `up_threshold` cho từng vùng ROI giúp kiểm soát độ nhạy của thuật toán so khớp mẫu. Ví dụ, với các vùng khó phát hiện hơn, `down_threshold` giảm xuống 0.66 để tăng khả năng phát hiện.
 - Tùy chọn `rotate_temp=True` với `num_angles=51` cho phép xoay mẫu ở nhiều góc độ, giúp phát hiện các con thỏ có hướng khác nhau (ví dụ: thỏ quay trái, quay phải, hoặc nghiêng).
- Tránh trùng lặp: Hàm `draw` trong `template_matching` được thiết kế để tô đen các vùng đã khớp (`thickness=-1`, `color=(0, 0, 0)`), giúp tránh việc đếm trùng các con thỏ. Bạn đã tận dụng cơ chế này để đảm bảo mỗi con thỏ chỉ được đếm một lần.

Kết quả 29 con thỏ cho thấy cách 2 hiệu quả hơn, đặc biệt trong trường hợp hình ảnh phức tạp với nhiều con thỏ có kích thước và hướng khác nhau. Tuy nhiên, để cải thiện thêm, có thể cân nhắc áp dụng Non-Maximum Suppression (NMS) để xử lý các trường hợp các vùng khớp bị chồng lấn, hoặc thử nghiệm với các kỹ thuật học sâu (như YOLO) nếu cần độ chính xác cao hơn.

5. Cải thiện hiệu suất nhiệm vụ đếm thỏ

Ban đầu, với phương pháp 2 đã đếm được 46 thỏ, nhưng kết quả thực tế chỉ có 14 thỏ, cho thấy có sự đếm thừa do các lỗi trong quá trình phát hiện.

Phương pháp ở mục 2:

- Dùng hàm `process_match` nhiều lần với các tọa độ ROI khác nhau (x, y) và kích thước vùng (w, h). Ví dụ: $(x, y) = (0, 276)$ với $w, h = 46, 57$ và $\text{down_threshold}=0.5$, $\text{up_threshold}=1$; $(x, y) = (448, 334)$ với $w, h = 46, 57$ và $\text{down_threshold}=0.6$, $\text{up_threshold}=1$.
- Không sử dụng tùy chọn `rotate_temp`, do đó không áp dụng xoay mẫu để phát hiện các con thỏ ở nhiều góc độ.
- Các ngưỡng `down_threshold` và `up_threshold` được điều chỉnh linh hoạt (ví dụ: từ 0.5 đến 0.6 cho `down_threshold`, và `up_threshold` luôn là 1) để phù hợp với từng vùng ROI.
- Ngoài ra, một hàm lọc `keep_coord` được áp dụng để giữ lại các tọa độ có $x > 100$, nhằm loại bỏ các tọa độ không mong muốn ở phía bên trái của ảnh.



Những thay đổi và cải tiến trong mã nguồn

5.1 Điều chỉnh ngưỡng phát hiện (threshold) trong hàm `process_match`

- Lúc đầu các ngưỡng `'down_threshold'` và `'up_threshold'` lần lượt là `'(0.5, 1)'` và `'(0.6, 1)'` cho hai vị trí mẫu (`template`) tại `'(0, 276)'` và `'(448, 334)'`.
- Trong mã cải tiến, các ngưỡng này được điều chỉnh thành `'(1, 1)'` và `'(0.28, 1)'`.

- Lý do: Việc điều chỉnh ngưỡng giúp kiểm soát độ nhạy của thuật toán phát hiện mẫu. Ngưỡng `down_threshold` thấp hơn (0.28) tại vị trí `(448, 334)` cho phép phát hiện các mẫu có độ tương đồng thấp hơn, nhưng vẫn đảm bảo tính chính xác khi kết hợp với các bước lọc sau. Ngưỡng `(1, 1)` tại vị trí `(0, 276)` yêu cầu độ tương đồng cao hơn, giảm thiểu các phát hiện sai lệch (false positives) tại khu vực này.

5.2. Thêm thuật toán Non-Maximum Suppression (NMS)

- Bổ sung hàm `nms` để loại bỏ các hộp giới hạn (bounding boxes) bị chồng lấn. Ngưỡng chồng lấn (`overlap_threshold=0.4`). Nó loại bỏ các hộp có độ chồng lấn lớn, chỉ giữ lại hộp đại diện nhất.
- Lý do cải tiến: Trong lần đầu, nhiều thỏ bị đếm trùng lặp do thuật toán phát hiện mẫu trả về nhiều hộp giới hạn tại cùng một vị trí (chồng lấn). Điều này dẫn đến số lượng thỏ được đếm là 46, cao hơn nhiều so với thực tế. NMS giúp loại bỏ các phát hiện trùng lặp bằng cách chỉ giữ lại hộp có độ tin cậy cao nhất trong các vùng chồng lấn, giảm số lượng thỏ đếm được xuống còn 14, khớp với số lượng thực tế.

5.3. Bỏ bộ lọc tọa độ tùy chọn (`keep_coord`)

- Bộ lọc `x > 100` có thể loại bỏ một số tọa độ hợp lệ của các thỏ nằm ở phía bên trái của ảnh (nếu `x` nhỏ hơn 100). Điều này không cần thiết sau khi áp dụng NMS, vì NMS đã xử lý hiệu quả các phát hiện trùng lặp. Việc bỏ bộ lọc này đảm bảo không bỏ sót các thỏ ở các vị trí khác nhau trong ảnh.

5.4 Kết quả và đánh giá.

- Đếm chính xác 14 thỏ, nhờ vào:
- Điều chỉnh ngưỡng phát hiện để cân bằng giữa độ nhạy và độ chính xác.
- Áp dụng NMS để loại bỏ các hộp giới hạn chồng lấn, giảm thiểu phát hiện sai.
- Bỏ bộ lọc tọa độ không cần thiết, tránh bỏ sót các thỏ ở các vị trí khác nhau.

