**Báo Cáo Week 5**

**Họ và tên: Nguyễn Mạnh Cường - 22022516**

FINDING(Hoàn thành)

## **1. Mục tiêu**

* **Xác định vị trí các đối tượng trong hình ảnh** một cách chính xác.
* **Làm nổi bật vị trí của các đối tượng** bằng cách sử dụng bounding boxes.



Các phần dưới đây sẽ xét cho ảnh FIND 15 HIDDEN OBJECT, ý tưởng FIND 12 HIDDEN OBJECT là y hệt, chỉ khác là FIND 12 HIDDEN OBJECT phải xử lý thêm trường hợp : Khi lấy mẫu vật thể cần **xử lý vật thể có khoảng trống.**

## **2. Đặc điểm của ảnh đầu vào**

1. **Cấu trúc ảnh:**
   * Ảnh chứa các mẫu vật thể nằm ở **bên phải ảnh gốc**, và ảnh cần nhận diện nằm ở **bên trái ảnh gốc** với tỷ lệ **0.45:0.55** giữa hai phần.
2. **Tỷ lệ kích thước:**
   * Tỷ lệ của (vật thể cần nhận diện trong phần bên trái / mẫu vật thể bên phải) ~ **[0.6 ; 0.7]**
3. **Nền và màu sắc:**
   * Vật thể cần nhận diện thường được bao quanh bởi **nền trùng màu**, điều này gây khó khăn trong việc phân biệt và tách đối tượng khỏi nền.

## **3. Lấy mẫu đối tượng**

### **3.1 Các vấn đề trong lấy mẫu đối tượng:**

#### **Phương pháp cắt ảnh mẫu hiệu quả:**

* **Vấn đề**: Cần cắt ảnh mẫu chính xác để phục vụ quá trình nhận diện.
* **Giải pháp**:
  + Bỏ nền trắng đối với các ảnh có nhiều màu sắc, giúp dễ nhận diện đối tượng hơn.
  + Lưu ảnh dưới định dạng **PNG hoặc JPEG** với **mức độ trong suốt (opacity) khoảng 95%**, duy trì chất lượng tốt nhất và giúp các thuật toán xử lý dễ dàng hơn.

#### **Xử lý nhiều mẫu đối tượng:**

* **Vấn đề**: Khi có nhiều mẫu đối tượng, làm sao để giữ lại đúng các đối tượng mong muốn?
* **Giải pháp**:
  + Sử dụng kỹ thuật **thresholding** để chuyển ảnh thành ảnh nhị phân, tách biệt rõ các đối tượng với nền.
  + Kết hợp kiểm tra thủ công nhằm loại bỏ các đối tượng không mong muốn trong quá trình xử lý.
  + **Kỹ thuật**: Sử dụng hàm cv2.threshold để chuyển ảnh thành ảnh nhị phân, và cv2.findContours để tìm đường viền (**contours**) quanh các đối tượng.

#### **Điều chỉnh threshold:**

* **Vấn đề**: Nếu threshold quá thấp, có thể bỏ sót các chi tiết quan trọng hoặc giữ lại những vùng không mong muốn.
* **Giải pháp**:
  + Điều chỉnh giá trị **threshold lên mức 245** để tách biệt rõ ràng hơn giữa đối tượng và nền.
  + **Kỹ thuật**: Sử dụng tham số threshold=245 trong hàm cv2.threshold để làm rõ sự khác biệt giữa nền và đối tượng.

#### **Xử lý vấn đề trong bounding box 1 vật dính thêm phần của vật thể khác**:

* **Vấn đề**: Bounding box có thể bao gồm thêm phần của các vật thể gần kề, làm sai lệch kết quả.
* **Giải pháp**:
  + Thay vì dùng bounding box, sử dụng **contours** để khoanh vùng chính xác đối tượng.
  + **Kỹ thuật**: Sử dụng hàm cv2.drawContours để vẽ đường viền quanh đối tượng, chính xác hơn việc sử dụng bounding box (cv2.rectangle).

#### **Xử lý vật thể có khoảng trống**:

* **Vấn đề**: Một số vật thể có khoảng trống giữa các bộ phận(như con bướm), dẫn đến việc chúng bị tách thành nhiều phần khi sử dụng contour.
* **Giải pháp**:
  + Sử dụng phép toán **giãn nở (dilation)** để lấp đầy các khoảng trống và làm mịn đối tượng.
  + **Kỹ thuật**: Sử dụng hàm cv2.dilate để giãn nở các biên cạnh, giúp nối các vùng rời rạc thành một đối tượng khép kín.

### **3.2 Quy trình lấy mẫu vật thể hoàn chỉnh**

* Đọc ảnh đầu vào (cv2.imread)
* Tạo thư mục đầu ra (os.makedirs)
* Chuyển đổi sang ảnh xám (cv2.cvtColor)
* Giảm nhiễu ảnh (cv2.GaussianBlur)
* Tạo ảnh nhị phân (cv2.threshold)
* Tìm đường viền vật thể (cv2.findContours)
* Lọc vật thể theo diện tích (cv2.contourArea)
* Tạo mask và tách vật thể (cv2.drawContours, cv2.bitwise\_and)
* Cắt ảnh chứa vật thể (cv2.boundingRect)
* Hiển thị và lưu ảnh đã tách (cv2\_imshow, cv2.imwrite)



## **4. Quy trình tìm kiếm vật thể**

### **4.1 Các phương pháp đã thử nghiệm:**

#### **Template Matching và Multi-scale Template Matching**

* **Đặc điểm**:
  + **Kích thước ảnh gốc**: 1415x1829.
  + **Kích thước ảnh mẫu**: Đa dạng (201x234, 126x209, 244x322,...).
* **Giải pháp**: Sử dụng phương pháp **template matching** với đa tỷ lệ (**multi-scale template matching**) để so khớp mẫu với các tỷ lệ khác nhau của đối tượng → không thành công.

#### **Sử dụng đặc trưng không phụ thuộc tỷ lệ (ORB)**

* **Thử nghiệm với ORB (Oriented FAST and Rotated BRIEF)**: Dù tăng số lượng đặc trưng (**nfeatures**) lên 1500, kết quả vẫn chưa cải thiện đáng kể (FAIL).

#### **Chia ảnh làm 2 phần và so sánh:**

* **Phương pháp**: Chia ảnh theo tỷ lệ **0.45 (trái) : 0.55 (phải)**, tách vật thể từ phần trái và so sánh với phần phải.
* **Vấn đề**: Khó áp dụng **multi-template matching** do diện tích bounding box của các vật thể khác nhau quá nhiều

#### **Phương pháp cuối cùng (SUCCESS):**

* **Áp dụng edge detection**: Phát hiện biên cạnh kết hợp với **resize** với tỷ lệ từ **0.4 đến 0.9** đã cho kết quả thành công.
* **Template matching** được áp dụng sau khi phát hiện biên cạnh giúp cải thiện độ chính xác.

### **4.2 Quy trình hoàn chỉnh:**

#### **1. Tiền xử lý ảnh chính:**

* **Đọc ảnh**: Sử dụng hàm cv2.imread.
* **Chuyển đổi sang ảnh xám**: Sử dụng cv2.cvtColor.
* **Phát hiện biên cạnh**: Sử dụng hàm cv2.Canny để phát hiện biên cạnh, giúp tách đối tượng khỏi nền.

#### **2. Chuẩn bị ảnh mẫu:**

* **Đọc ảnh mẫu**: Sử dụng hàm cv2.imread từ thư mục chứa mẫu.
* **Chuyển đổi sang ảnh xám**: Sử dụng cv2.cvtColor để chuyển mẫu sang ảnh xám.

#### **3. So khớp mẫu đa tỷ lệ:**

* **Thay đổi kích thước ảnh mẫu**: Sử dụng hàm cv2.resize để thử nhiều tỷ lệ từ 0.4 đến 0.9.
* **Phát hiện biên cạnh của mẫu**: Sử dụng hàm cv2.Canny để phát hiện biên cạnh ảnh mẫu.
* **So khớp mẫu**: Sử dụng hàm cv2.matchTemplate để tìm sự tương đồng giữa mẫu và ảnh chính.

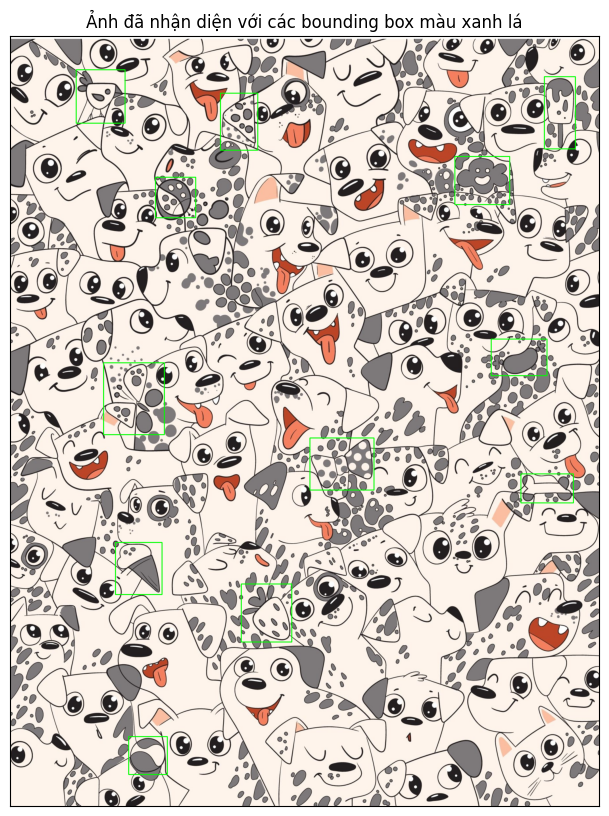
#### **4. Đánh giá kết quả:**

* **Tìm vị trí khớp tốt nhất**: Sử dụng cv2.minMaxLoc để tìm vị trí có độ khớp tốt nhất.
* **Cập nhật kết quả khớp tốt nhất**: Nếu tìm được vị trí tốt hơn, kết quả sẽ được cập nhật.

#### **5. Hiển thị kết quả:**

* **Vẽ bounding box**: Sử dụng cv2.rectangle để vẽ khung giới hạn quanh vị trí nhận diện.
* **Chuyển đổi màu**: Sử dụng hàm cv2.cvtColor để chuyển ảnh sang dạng màu nhằm hiển thị rõ ràng.
* **Hiển thị kết quả**: Sử dụng thư viện matplotlib.pyplot để hiển thị ảnh kết quả sau khi nhận diện.

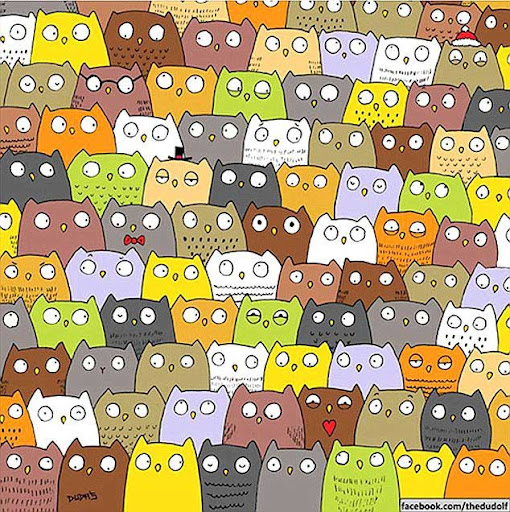
# **5. Kết quả(Đã nhận diện được hết các vật thể)**



COUNTING

* Tiến độ hoàn thành: 1 hình do thiếu thời gian

## **1. Mục tiêu**

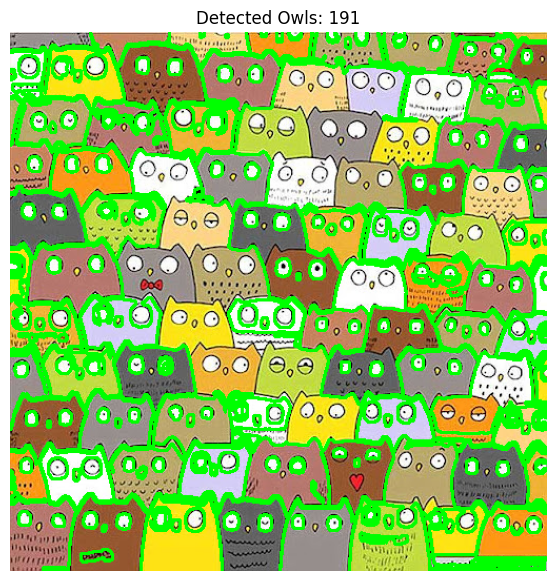
* **Xác định chính xác vị trí của các đối tượng (các con cú) trong hình ảnh** thông qua các phương pháp phát hiện đối tượng.
* **Làm nổi bật các đối tượng đã phát hiện bằng cách sử dụng bounding boxes** để đánh dấu vị trí của chúng trong ảnh.
* **Đếm tổng số các đối tượng** (các con cú) sau khi đã phát hiện và đánh dấu để cung cấp kết quả chính xác.(**Tổng là 96 con cú**).
* ****

## **2. Đặc điểm của ảnh đầu vào**

1. **Mẫu đối tượng**: Các con cú có thiết kế đơn giản, dễ nhận diện với mắt to màu trắng, màu sắc đa dạng và biểu cảm phong phú.
2. **Độ phân giải**: Ảnh vẽ tay, do đó độ sắc nét không cao.
3. **Biên ảnh:** Biên ảnh chứa khá nhiều con cú bị khuyết thiếu cơ thể(khó nhận dạng).

## **3. Quá trình xử lý các vấn đề khi đếm**

### **3.1 Thuật toán Edge Detection Canny không hiệu quả cho dù đã điều chỉnh một loạt tham số**

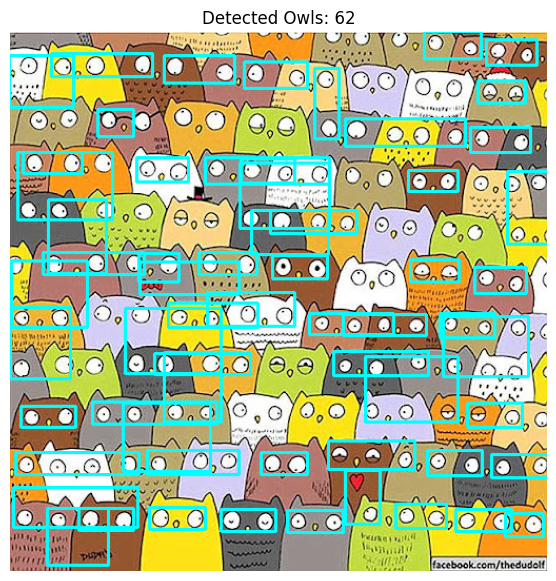


Sau khi thử nghiệm thì:



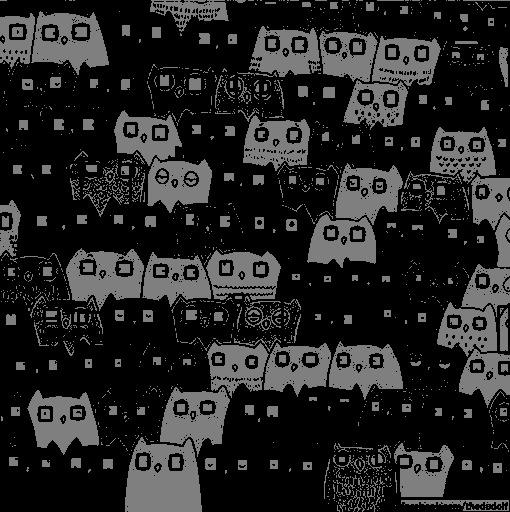
Kết quả khả thi nhất(**ảnh trên**) với bộ tham số {'blur': ((15, 15), 0), 'canny': (20, 60, 3), 'dilated': ((1, 1), 1), 'area\_coef': 0.02}

### **3.2 Kết hợp ý tưởng Edge detection + mắt là một hình kín và có diện tích < một ngưỡng (không thành công)**

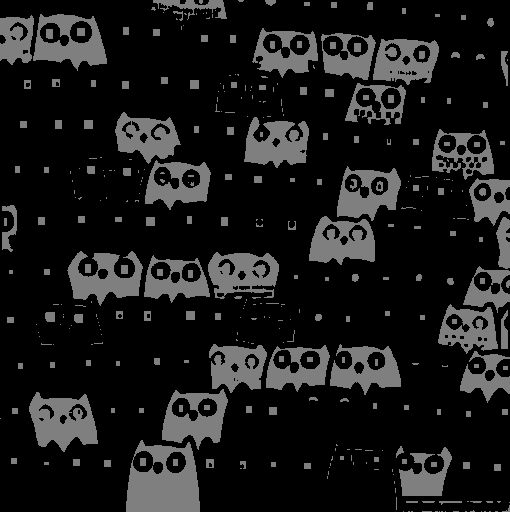
* ****

### **3.3 Phát hiện vấn đề khi lọc ảnh bằng các bộ lọc khác nhau(chất lượng ảnh không tốt)**

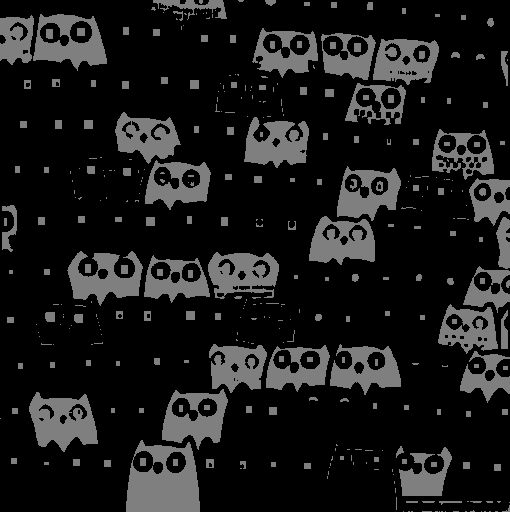
Dưới đây là một số ví dụ



Processed Original Image with Bounding Box (Threshold: 200)



Processed Mean Filtered Image with Bounding Box (Threshold: 200)



Processed Gaussian Filtered Image with Bounding Box (Threshold: 200)



Processed Median Filtered Image with Bounding Box (Threshold: 200)

### **3.4 Ý tưởng tối ưu: Con cú được xác định nếu có mắt màu trắng, mắt này được bao quanh bởi các pixel màu đen và diện tích của mắt nhỏ hơn một ngưỡng cho trước**

* Sử dụng ý tưởng này sẽ đếm kém hơn ở biên

## **4. Quy trình đếm**

**Đọc ảnh và chuyển thành ảnh xám**:

* Đầu tiên, hình ảnh gốc (mèo trong trường hợp này) được đọc và chuyển đổi thành ảnh xám để dễ dàng xử lý bằng các thuật toán nhị phân.
* Ví dụ: Nếu ảnh chứa nhiều mèo, chương trình sẽ chuyển đổi ảnh thành màu xám, làm đơn giản hóa các phép tính tiếp theo.

**Áp dụng ngưỡng (threshold)**:

* Sau khi chuyển sang ảnh xám, chương trình áp dụng ngưỡng để tìm các vùng trắng (các vùng có màu sáng). Các vùng trắng này đại diện cho các đối tượng trong ảnh mà ta cần tìm, chẳng hạn như mắt của con mèo.
* Ví dụ: Nếu vùng trắng là mắt mèo, thuật toán sẽ đánh dấu và tách biệt nó khỏi nền.

**Flood-fill và tìm các vùng trắng**:

* Hàm flood\_fill được sử dụng để tìm các vùng trắng liên tục và đánh dấu chúng. Khi tìm thấy một vùng trắng, nó sẽ tính toán diện tích và tạo một bounding box để bao quanh vùng đó.
* Ví dụ: Khi tìm thấy mắt của mèo (một vùng trắng), chương trình sẽ xác định tọa độ của mắt và diện tích của nó.

**Phát hiện cặp mắt**:

* Sau khi tìm ra tất cả các vùng trắng, chương trình sẽ kiểm tra khoảng cách giữa các vùng này. Nếu hai vùng trắng gần nhau và có cùng màu nền, chúng sẽ được xác định là một cặp mắt.
* Ví dụ: Nếu hai mắt mèo gần nhau, chương trình sẽ kết hợp chúng thành một cặp và bao quanh bằng một bounding box lớn.

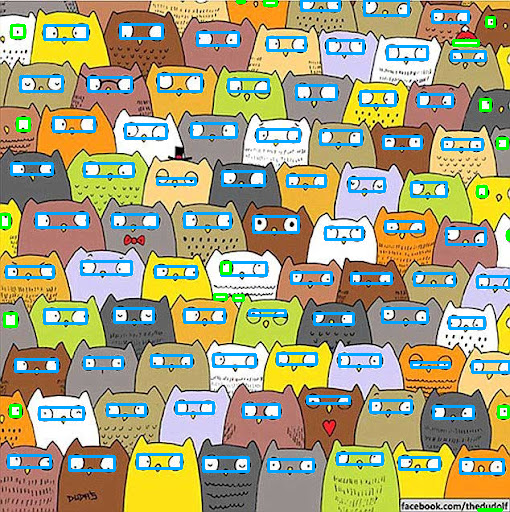
**Vẽ bounding boxes**:

* Chương trình sẽ vẽ bounding boxes cho các cặp mắt và mắt đơn lẻ (nếu chỉ tìm thấy một mắt). Các cặp mắt sẽ được đánh dấu bằng bounding box màu xanh biển, trong khi mắt đơn lẻ được đánh dấu bằng bounding box màu xanh lá cây.
* Ví dụ: Nếu phát hiện được một cặp mắt, chương trình sẽ vẽ một ô vuông màu xanh biển bao quanh cặp mắt này.

**Hiển thị kết quả**:

* Cuối cùng, ảnh kết quả với các bounding boxes sẽ được hiển thị, và chương trình sẽ in ra tổng số lượng cặp mắt và mắt đơn lẻ.

## **5. Kết quả(Đã nhận diện được hết các vật thể)**



Số lượng vùng trắng được tìm thấy: 169

Các cặp mắt: [(2, 3), (4, 5), (6, 7), (9, 12), (11, 13), (14, 15), (16, 17), (18, 19), (21, 22), (23, 24), (25, 29), (26, 27), (28, 30), (31, 32), (33, 34), (35, 36), (37, 38), (40, 41), (43, 45), (44, 46), (47, 48), (49, 50), (51, 52), (54, 55), (56, 58), (57, 59), (60, 61), (62, 63), (64, 66), (65, 67), (68, 69), (71, 74), (72, 77), (75, 76), (78, 79), (80, 81), (82, 83), (84, 86), (87, 88), (90, 89), (91, 93), (92, 94), (95, 98), (96, 99), (97, 100), (101, 102), (105, 108), (106, 114), (107, 110), (109, 111), (112, 113), (116, 117), (118, 119), (120, 122), (121, 128), (123, 125), (124, 127), (126, 129), (130, 131), (132, 134), (133, 135), (136, 137), (138, 141), (139, 140), (142, 146), (143, 144), (147, 151), (149, 150), (152, 154), (153, 156), (155, 160), (157, 161), (158, 162), (159, 163), (164, 166), (165, 167)]

Mắt đơn lẻ: [0, 1, 70, 39, 8, 73, 10, 42, 103, 104, 168, 145, 115, 20, 53, 85, 148, 89]

Số lượng bounding box màu xanh biển (cặp mắt): 76

Số lượng bounding box màu xanh lá (mắt đơn lẻ): 18

Số lượng bounding box màu xanh lá + màu xanh biển (mắt đơn lẻ): **94**

**Kết quả: 94 con cú**