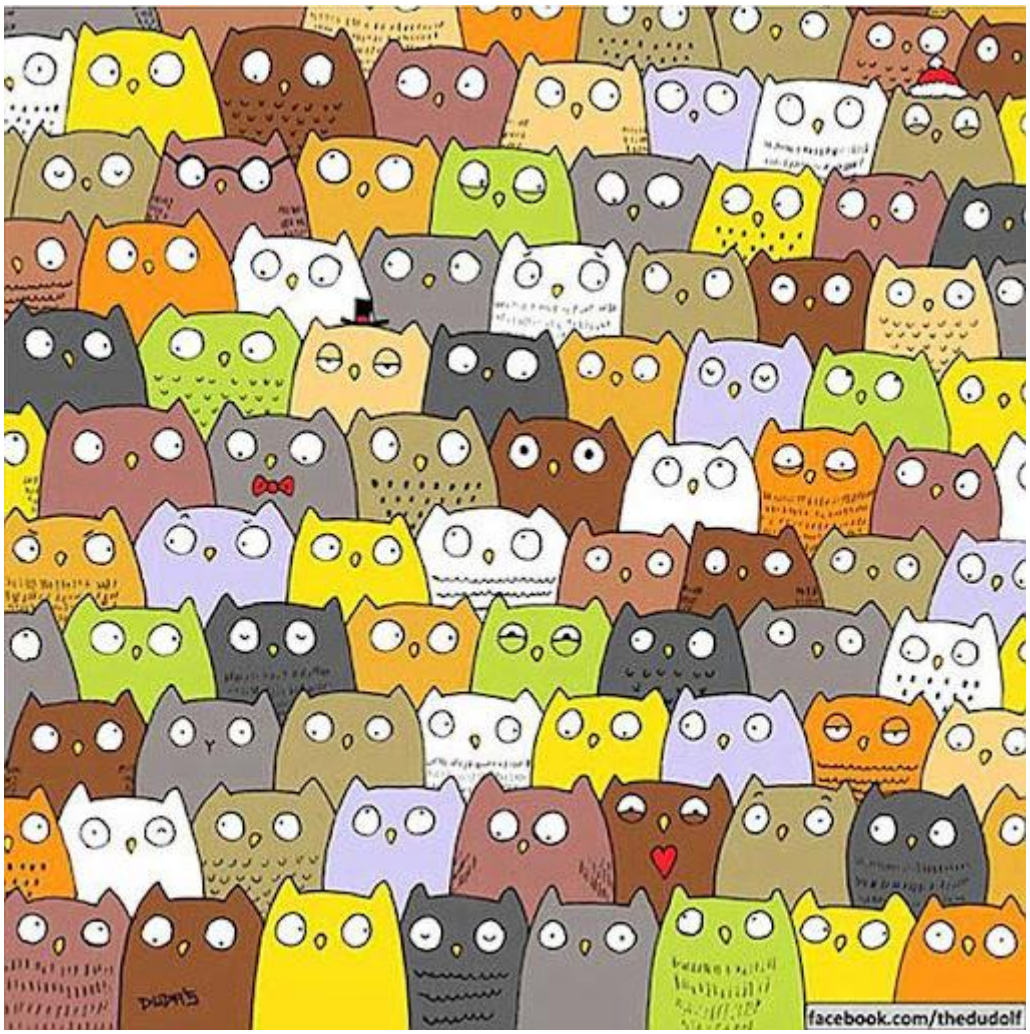
Báo cáo về bài toán đếm cú vật thể trong ảnh

Họ và tên sinh viên: Đinh Văn Sinh – 22022615

Link github: https://github.com/sinh2206/Midterm-ImageSet

1. Bài toán

Đếm số lượng mèo có trong ảnh sau:



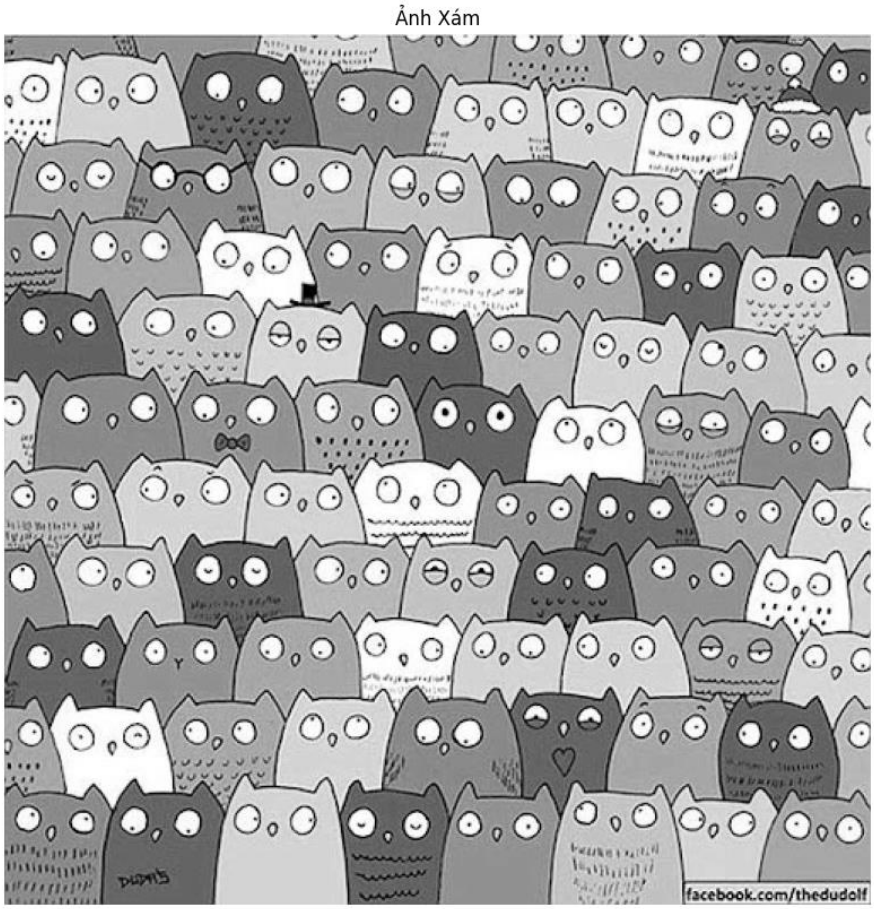
2. Mục tiêu

Đếm bằng mắt thường, chúng ta có thể đếm được khoảng 96 con cú (nếu coi những con chỉ xuất hiện bằng một vùng màu nhỏ ở góc bên trái phía trên và những con không xuất hiện đầy đủ là cú)

3. Các bước thực hiện

Mục tiêu của bài toán là tự động phát hiện và đếm số lượng các vật thể có trong ảnh thông qua các bước tiền xử lý và xử lý ảnh. Cụ thể, chúng ta sẽ sử dụng các kỹ thuật:

- Đọc ảnh và chuyển đổi sang grayscale.

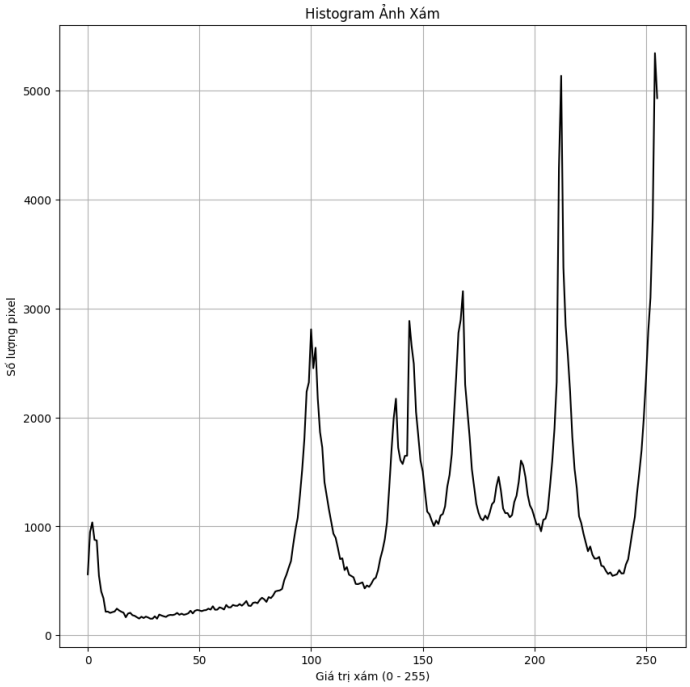


Ta có thể dễ dàng nhận thấy ảnh có sự phân bố màu không đều.

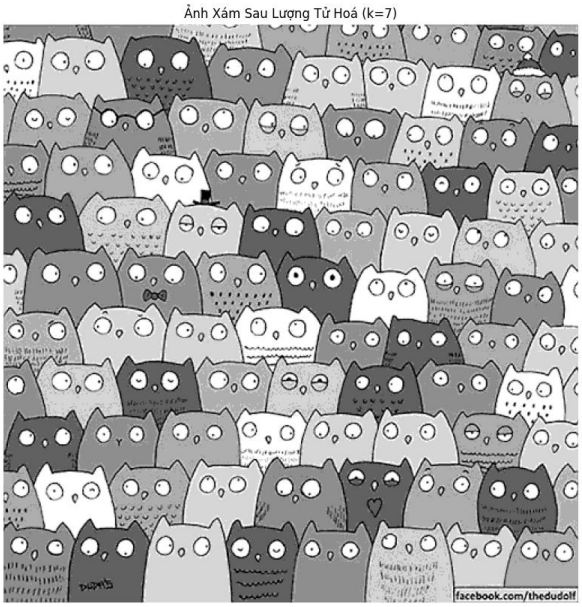
Để xử lý vấn đề này, chúng ta cần phải tìm cách lượng tử hóa (quantize) hình ảnh về một số lượng màu nhất định.

- Tính histogram để hiểu phân bố cường độ ảnh.

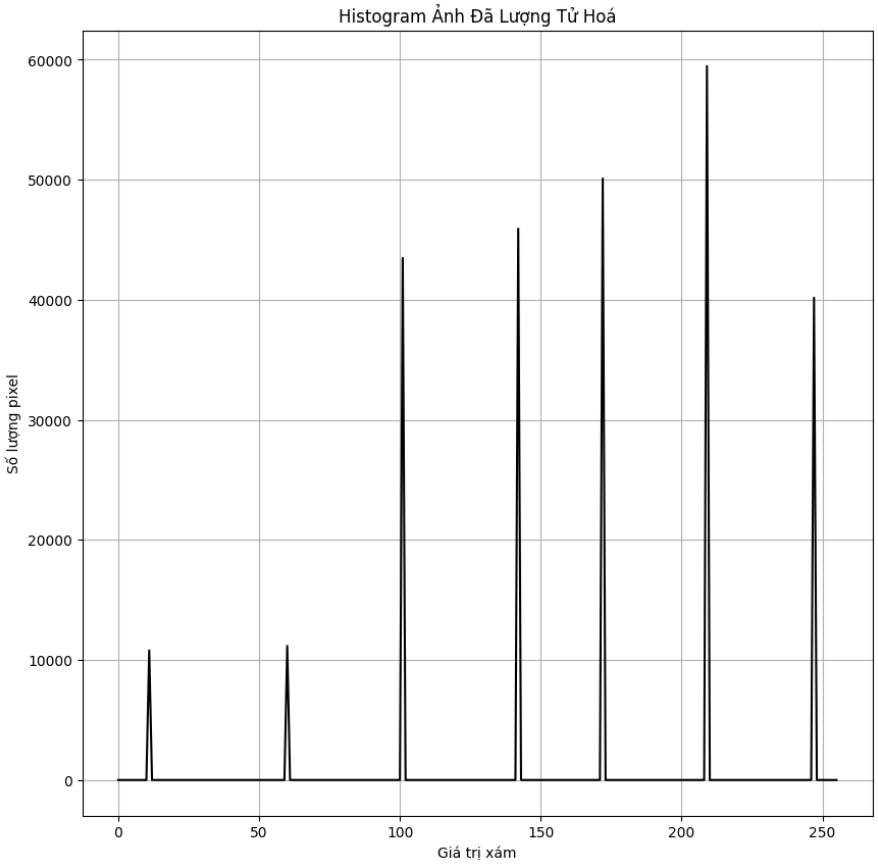
Chúng ta sẽ nhìn vào histogram của ảnh đen trắng để xem số lượng màu chúng ta nên quantize ảnh về:



- Lượng tử hoá (Quantization)
  Chúng ta sẽ nhìn vào histogram của ảnh đen trắng để xem số lượng màu chúng ta nên quantize ảnh về:



  Thay vì làm việc với 256 mức xám, sử dụng thuật toán K-Means để nhóm các pixel thành k cụm (ví dụ k=7). Điều này giúp giảm sự phức tạp và tập trung vào những giá trị xám chủ đạo trong ảnh.
- Tạo histogram cho ảnh sau lượng tử hoá
  Sử dụng thuật toán K-means để phân cụm các màu vào 7 cụm, sử dụng thuật khởi tạo k-means++ để đảm bảo tính ổn định của thuật toán phân cụm. Histogram của hình ảnh sau khi quantized như sau:



Dựa vào đây, chúng ta có thể chia thành 7 phần riêng biệt

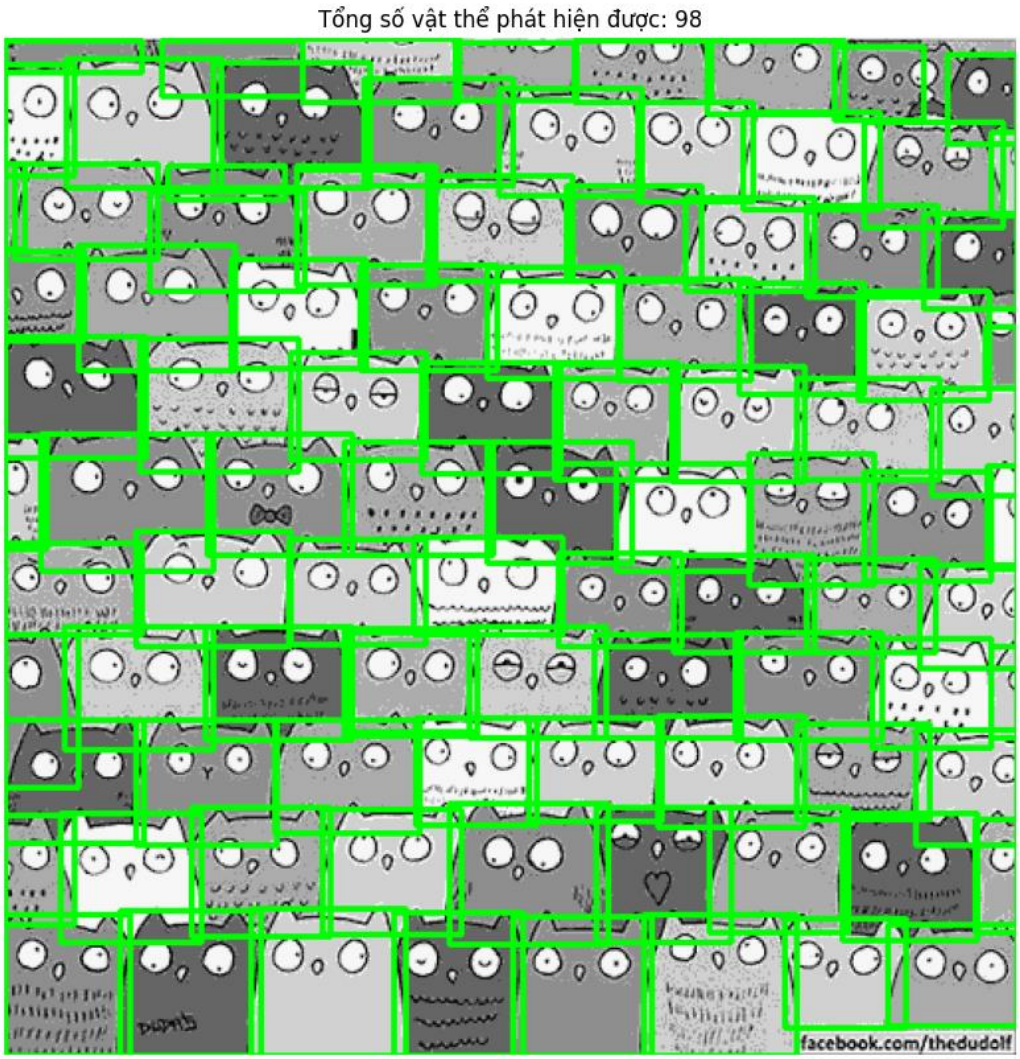
- Đếm và vẽ bounding box

```python
def count_objects_connected_components(quantized_img, min_size=200):
    """
    Duyệt qua từng mức xám trong ảnh đã lượng tử hoá.
    Tạo mask và tìm thành phần liên thông.
    Trả về danh sách (level, bounding_box, area).
    """
    unique_values = np.unique(quantized_img)
    objects_info = []

    for val in unique_values:
        # Tạo mask cho mức xám val
        mask = (quantized_img == val).astype(np.uint8)

        # Tìm các vùng liên thông kèm thống kê
        num_labels, labels, stats, centroids = cv2.connectedComponentsWithStats(
            mask,
            connectivity=8
        )

        # stats[i] = [x, y, width, height, area]
        for i in range(1, num_labels):  # bỏ qua i=0 (nền background)
            x, y, w, h, area = stats[i]
            if area >= min_size:
                # Lưu lại thông tin (mức xám, bounding box, diện tích)
                objects_info.append((val, (x, y, w, h), area))

    return objects_info

objects_info = count_objects_connected_components(quantized_gray, min_size=200)
```

Tiếp theo, ta sẽ đếm số lượng cú dựa trên màu bằng cách đếm các thành phần liên thông nhau, và lọc các thành phần liên thông dựa vào size. Chọn min_size bằng 200 bằng trực quan của ta có vẻ là hợp lí nhất

```
Tổng số vùng (>=200 pixel) tìm được: 98
- Object 0: Gray Level=101, Vị trí=(475,8), Kích thước=35x45, Diện tích=896
- Object 1: Gray Level=101, Vị trí=(105,12), Kích thước=79x63, Diện tích=2906
- Object 2: Gray Level=101, Vị trí=(463,88), Kích thước=47x48, Diện tích=1277
- Object 3: Gray Level=101, Vị trí=(370,123), Kích thước=63x56, Diện tích=1863
- Object 4: Gray Level=101, Vị trí=(0,150), Kích thước=71x61, Diện tích=2605
- Object 5: Gray Level=101, Vị trí=(210,163), Kích thước=70x56, Diện tích=2064
- Object 6: Gray Level=101, Vị trí=(243,203), Kích thước=73x61, Diện tích=2363
- Object 7: Gray Level=101, Vị trí=(337,260), Kích thước=71x49, Diện tích=1816
- Object 8: Gray Level=101, Vị trí=(101,296), Kích thước=74x57, Diện tích=2261
- Object 9: Gray Level=101, Vị trí=(297,300), Kích thước=74x52, Diện tích=2062
- Object 10: Gray Level=101, Vị trí=(0,343), Kích thước=70x62, Diện tích=2251
- Object 11: Gray Level=101, Vị trí=(298,387), Kích thước=68x68, Diện tích=2346
- Object 12: Gray Level=101, Vị trí=(422,390), Kích thước=68x64, Diện tích=2702
- Object 13: Gray Level=101, Vị trí=(58,439), Kích thước=71x73, Diện tích=3449
- Object 14: Gray Level=101, Vị trí=(196,440), Kích thước=65x72, Diện tích=2829
- Object 15: Gray Level=142, Vị trí=(0,1), Kích thước=69x16, Diện tích=583
- Object 16: Gray Level=142, Vị trí=(79,1), Kích thước=71x28, Diện tích=691
- Object 17: Gray Level=142, Vị trí=(418,4), Kích thước=45x37, Diện tích=680
- Object 18: Gray Level=142, Vị trí=(181,20), Kích thước=75x54, Diện tích=2329
- Object 19: Gray Level=142, Vị trí=(82,64), Kích thước=59x17, Diện tích=474
- Object 20: Gray Level=142, Vị trí=(283,74), Kích thước=68x48, Diện tích=1736
- Object 21: Gray Level=142, Vị trí=(73,77), Kích thước=78x50, Diện tích=1536
- Object 22: Gray Level=142, Vị trí=(404,84), Kích thước=66x46, Diện tích=1780
- Object 23: Gray Level=142, Vị trí=(0,107), Kích thước=40x45, Diện tích=781
...
- Object 94: Gray Level=247, Vị trí=(206,251), Kích thước=75x49, Diện tích=2140
- Object 95: Gray Level=247, Vị trí=(437,302), Kích thước=60x55, Diện tích=1694
- Object 96: Gray Level=247, Vị trí=(207,342), Kích thước=61x54, Diện tích=1719
- Object 97: Gray Level=247, Vị trí=(28,390), Kích thước=71x65, Diện tích=2542
```

Tìm được vật thể dao động trong khoảng 96 -> 98, khá phù hợp với trực quan của con người



Hạn chế:

- Nhạy cảm với nhiễu: Nếu ảnh có nhiều nhiễu hoặc đối tượng không rõ ràng, các vùng liên thông có thể bị phân chia sai.
- Phụ thuộc vào ngưỡng: Việc lựa chọn ngưỡng diện tích (min_size) và số lượng cụm K trong lượng tử hoá có thể ảnh hưởng đến kết quả.
- Trường hợp chồng lấn: Các đối tượng chồng lấn hoặc có ranh giới không rõ ràng có thể bị nhận diện không chính xác.