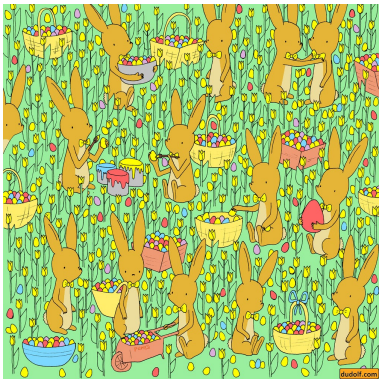
BÁO CÁO GIỮA KÌ MÔN XỬ LÝ VÀ PHÂN TÍCH HÌNH ẢNH

Trịnh Minh Hiếu
MSSV 22022536

Tháng 3, 2025

1 Tổng quan.

Trong bài tập giữa kì này, em đã lựa chọn đề bài đếm đối tượng (counting), và chủ thể thực hành là ảnh "rabbit.jpeg". Về phương án đã thử nghiệm thành công, em đã áp dụng cách tiếp cận **template matching** với template được sử dụng là một phần crop từ ảnh đã xử lý, và những khu vực có độ tương đồng cao sẽ được gắn bounding box.



2 Quá trình thử nghiệm.

2.1 Ý tưởng 1.

Khi xem bức ảnh, em thấy những con thỏ không khác nhau quá nhiều, nên hướng tới một phương pháp đơn giản mà vẫn hiệu quả - template matching. Để thuật toán được chính xác và ít nhiễu thì phải có cách xử lý sao cho lọc được tất cả những thứ không quan trọng, và chỉ để lại thỏ. Vì vậy, phương án tiền xử lý ảnh ban đầu là *grayscale - blur - threshold - closing*. Sau khi áp dụng, ảnh gốc chỉ còn hai màu đen trắng, với những đốm trắng rải rác (hoa sáng màu) và những vùng trắng lớn (con thỏ).

Tiếp theo, em đã thử nhiều template khác nhau để thử xem đặc trưng nào của con thỏ cho ra kết quả ổn định nhất:

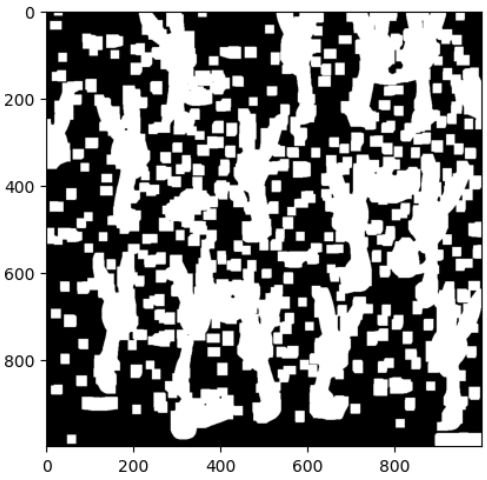
- **Template 1**: Đầu và tai thỏ - đặc trưng phổ biến nhất, nhưng lại cho ra kết quả không đúng, box không chính xác dù đã tinh chỉnh threshold. Nguyên nhân là do hướng nghiêng của đầu và hướng nghiêng của tai phân hóa, làm cho template khó tổng quát hóa.

- **Template 2**: Toàn bộ con thỏ từ đỉnh tai đến hết chân nhưng cho ra kết quả thấp hơn template 1 vì lí do tương tự lần thử trước.

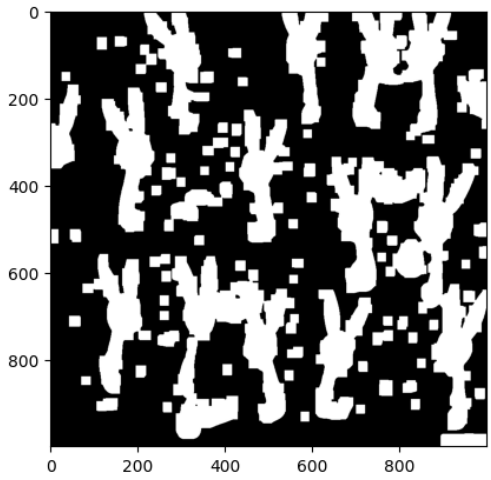
- **Template 3**: Chỉ có tai và phần đỉnh đầu (bán nguyệt). Phương án này làm cho kết quả đếm gần sát mức đúng ở 13 hoặc 15, nhưng không thể tinh chỉnh ngưỡng để về chính xác (quá nhạy), heatmap nhiễu mạnh và bounding box hỗn loạn.

- **Template 4**: Chỉ một bên tai thỏ (đặc trưng có dáng parabol hơi méo). Phương án này làm cho có những con thỏ được nhận cả 2 tai, có con không được nhận do dáng tai khác nhau. Có những trường hợp phần thân con thỏ được nhận do đều có dáng thuôn dài.

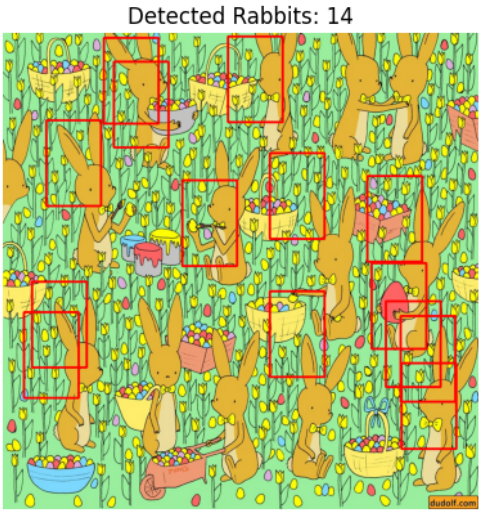
Nhận thấy vẫn còn nhiều nhiễu, em đã thay *close* thành *open* và tăng kích thước của kernel lên 20x20 để loại bỏ nhiễu nhưng không làm ảnh hưởng tới thỏ (hình 1b), thử lại template số 2 và template mới là *con thỏ bị crop bên góc trái*. Kết quả đưa ra số đúng là 14, nhưng vẫn còn nhiều box sai, và nhiều con thỏ không được nhận diện (hình 1c, 1d). Đáng chú ý nhất, đây là lần đầu tiên con thỏ bên trái được nhận diện.



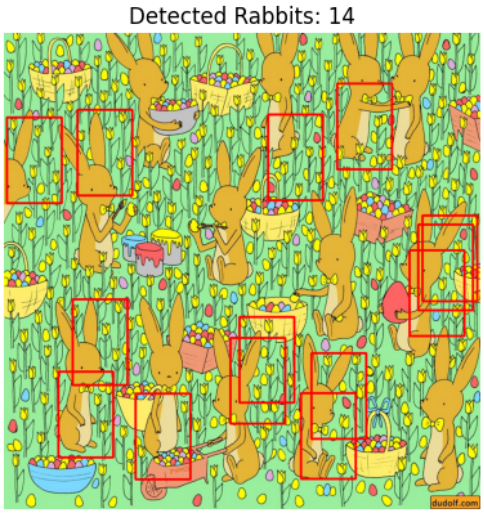
(a) Tiền xử lý còn nhiễu.



(b) Tiền xử lý đã loại bớt nhiễu.



(c) Template 2.



(d) Template mới.

Hình 1

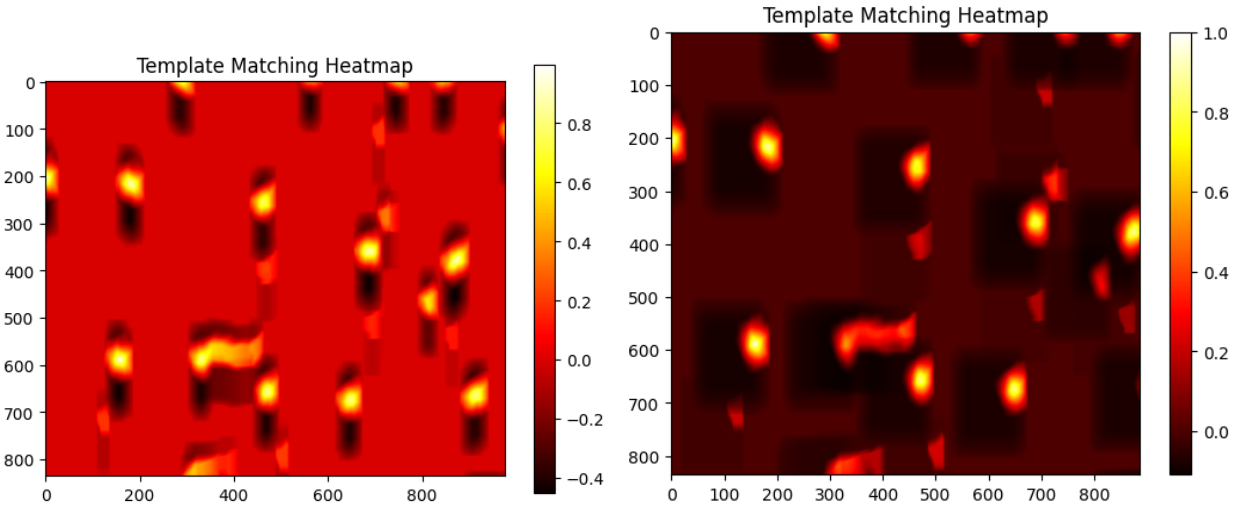
2.2 Ý tưởng 2.

Từ việc lần đầu nhận diện được con thỏ góc trái, em nhận ra là phải lấy template sao cho khớp với tất cả con thỏ, đặc biệt ưu tiên những con khó xác định. Vì vậy, phương án tối ưu là lấy template từ chú thỏ đó và áp dụng để tìm số còn lại. Hay nói ngắn gọn, template là vật bị crop vẫn có thể khớp với những vật tương đồng không bị crop, nhưng ngược lại thì không khả thi.

Sau khi đã quyết định được template, em đã thay đổi và cải tiến quá trình tiền xử lý sao cho có thể loại bỏ toàn bộ đốm trắng của hoa và các vùng sáng trung bình (tai và mình con thỏ) bằng cách thay phương pháp *open* bằng hai lần *erode* với các kích thước kernel khác nhau.

Pipeline giờ đây trở thành: **grayscale - threshold - pad (lề phải) - erode (kernel lớn) - erode (kernel trung) - template**.

- **Grayscale - Threshold**: Chuyển ảnh RGB thành dạng màu nhị phân để dễ thao tác.
- **Zero padding (lề phải)**: Trên ảnh sau khi xử lý có một số khu vực điển hình như [570:640,300:500] có vùng sáng dài do các vật thể lớn cùng màu bị dính vào nhau sau khi threshold. Để xử lý vấn đề này, em đã *lấy template sao cho có vùng đen xung quanh* để khi match template, kết quả sẽ thấp hơn ở các vùng trắng dính vào nhau. Tuy nhiên, điều đó lại dẫn đến việc con thỏ cuối cùng không được nhận do ở sát mép ảnh nên em đã thêm pad zero vào lề phải ảnh với kích thước bằng vùng đen trong template, khoảng 50 pixels.

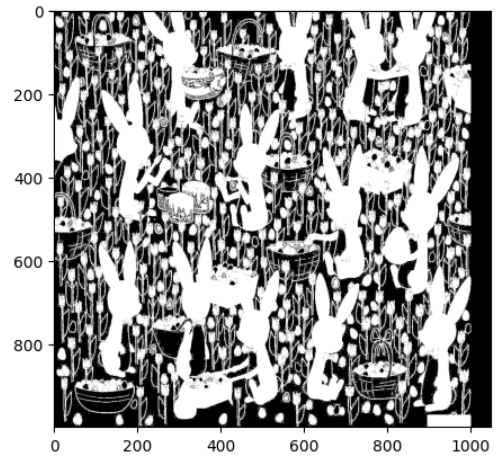


(a) Vùng sáng dài hoặc sát nhau gây ảnh hưởng kết quả.

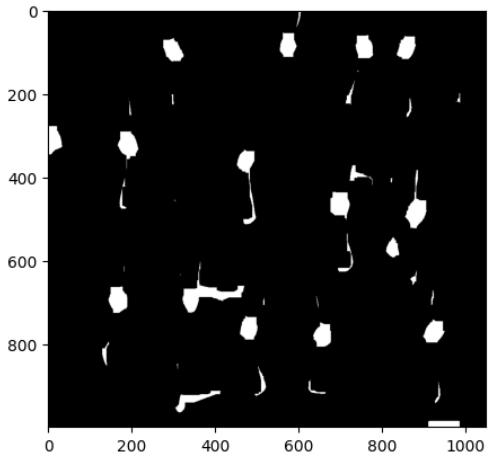
(b) Vùng sáng dài sau khi chỉnh template, nhưng bị mất con thỏ cuối.

Hình 2: Xử lý vùng sáng do thỏ và chậu hoa cùng màu.

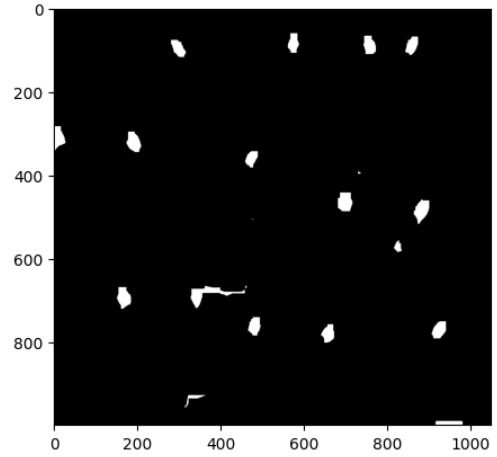
- **Erode với kernel lớn 30x30**: Loại bỏ hoa và các mảng sáng trung bình.
- **Erode với kernel trung 11x11**: Loại bỏ noise mà bước trước để lại.
- **Trích xuất template**: Con thỏ bị khuất bên trái.



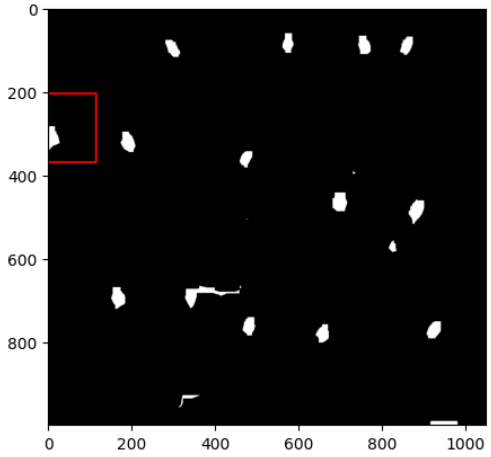
(a) Ảnh sau khi threshold tại 195 và pad bên phải 50 pixels.



(b) Ảnh sau khi erode với kernel lớn 30x30.



(c) Ảnh sau khi erode với kernel trung 11x11.

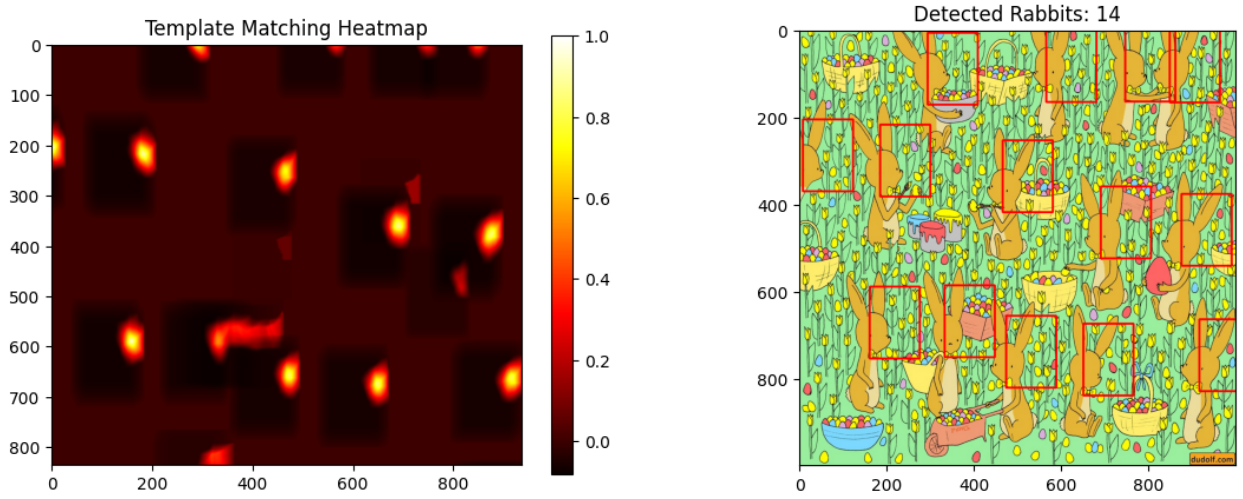


(d) Vị trí trích xuất template.

(e) Template được sử dụng.

Hình 3: Kết quả tiền xử lý.

Tiếp theo, em chạy thuật toán template matching với chủ thể là hình 3c, thu được một heatmap có độ phân hóa khá rõ ràng, có thể phân định bằng mắt thường. Cuối cùng, cần tinh chỉnh ngưỡng của để xác định đúng, đủ các vùng sáng và tạo bounding box để thể hiện rõ đối tượng đã tìm được, sử dụng non-maximum supression để tránh các box trùng lặp. Ở giá trị threshold là 0.42, thuật toán tìm ra 14 vị trí có độ tương đồng cao, nghĩa là đếm được 14 con thỏ.

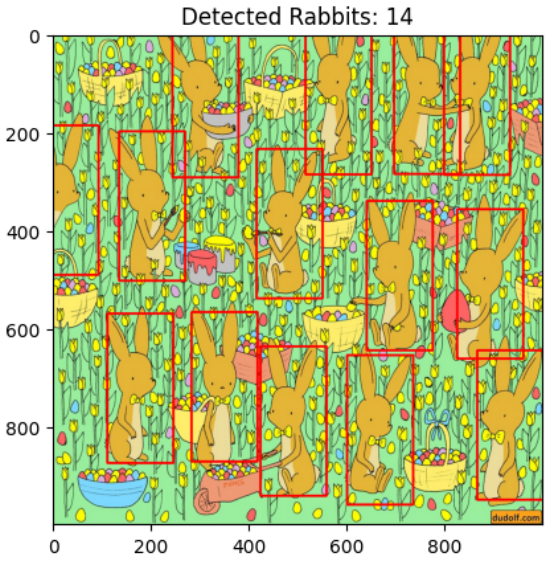


(a) Heatmap rõ ràng.

(b) Thuật toán tìm vị trí 14 con thỏ.

Hình 4: Kết quả chính xác nhưng box lệch.

Box ở kết quả dù đúng, nhưng lệch do cách chọn template. Cần điều chỉnh sao cho box có kích thước tương đương con thỏ, khoanh vùng chính xác, rõ ràng bằng cách tinh chỉnh độ lệch của bounding box.



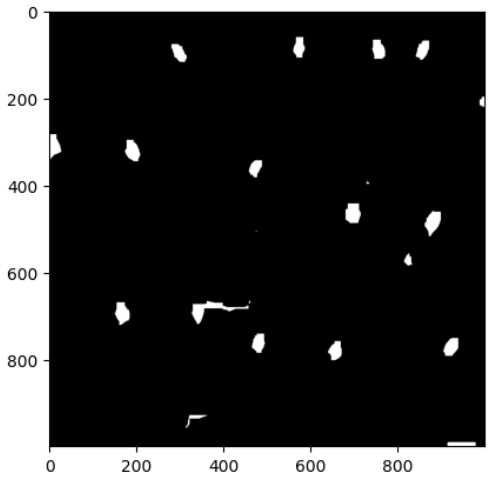
Hình 5: Vị trí box chính xác sau khi chỉnh.

3 Tối ưu.

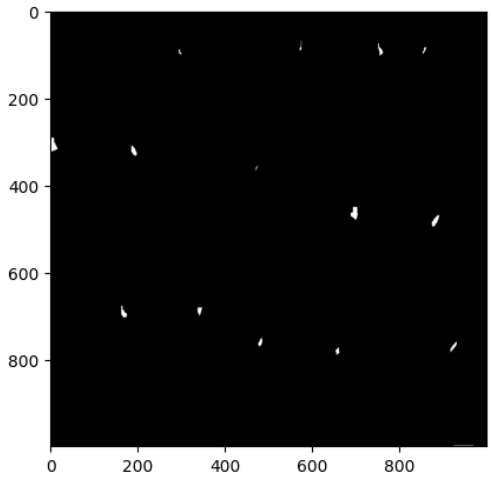
Kết quả thuật toán đã chính xác, nhưng phần tiền xử lý ảnh vẫn chưa triệt để và heatmap vẫn có nhiễu, nhạy cảm với threshold. Em đã thêm một bước *erode* nữa sau 2 bước đã có (chuỗi 3 *erode*) với kernel bé (5x5) và iteration = 4. Vì 3 lần *erode* đã ăn mòn đi rất nhiều những đặc trưng còn sót lại, em thực hiện *dilate* với kernel cỡ trung (11x11) để nhấn mạnh những mẩu sáng còn sót lại. Nhờ đó, vùng sáng dài gây nhiễu lúc trước đã biến mất, nghĩa là em có thể:

- Bỏ bước pad.
- Chọn template không cần vùng đen xung quanh nữa, nhưng vẫn phải là con thỏ bên trái với sự giải thích ở phần đầu ý tưởng 2.

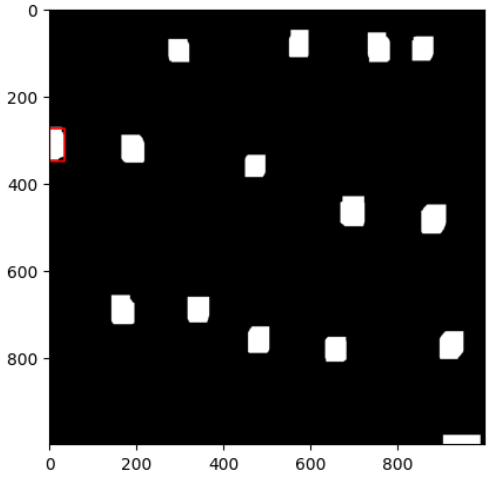
Heatmap giờ không còn nhiễu và ít nhạy với threshold (từ 0.1 - 0.45 đều ra kết quả đúng vị trí 14 thỏ). Sau khi chỉnh lại tọa độ bounding box do đổi template, kết quả cuối cùng không thay đổi so với hình 5.



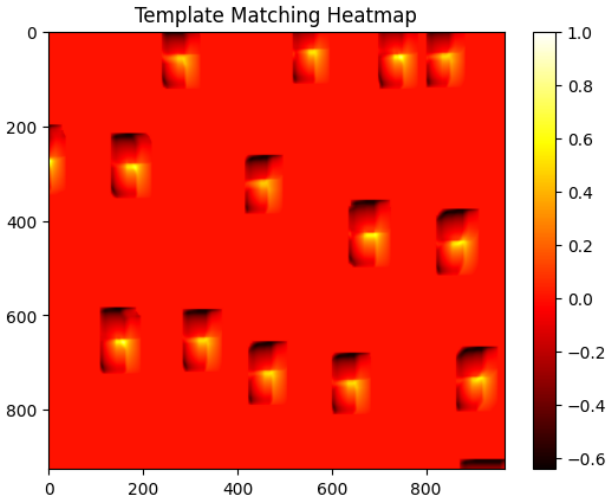
(a) Ảnh sau khi erode với kernel trung 11x11.



(b) Ảnh sau khi erode với kernel nhỏ, iterations = 4.



(c) Ảnh sau dilate, và vị trí lấy template.



(d) Heatmap.

Hình 6: Kết quả tiền xử lý đã tối ưu.

4 Tổng kết.

Phương pháp đếm đối tượng phù hợp cho bức ảnh "rabbit.jpeg" là **template matching** bởi vì hầu hết các đối tượng cần đếm đều cơ bản đồng nhất, với một số khác biệt về hướng đứng của đối tượng, hướng tai, kích thước và các bộ phận bị crop. Kết quả đếm chính xác đã củng cố cho tính đúng của nhận định về phương pháp. Qua nhiều lần thử, thì pipeline: **grayscale - threshold - erode - erode - erode - dilate - template** đã cho ra kết quả tối ưu để sử dụng template matching.

Trong quá trình làm bài tập này, công đoạn khó khăn nhất đối với em là lựa chọn ra đặc điểm chung của tất cả đối tượng và làm cách nào để trích xuất được nó bằng các phương pháp chỉnh ảnh. Để chọn được template thích hợp, em đã phải thử rất nhiều đặc trưng ở các vị trí khác nhau, và công đoạn này mang tính lặp lại với mỗi lần thay đổi, tối ưu cách xử lý ảnh. Chưa hết, việc tinh chỉnh ngưỡng của threshold, kích thước của các kernel cũng rất tốn thời gian trong những lần thử ý tưởng do cách tiếp cận là brute-force và quan sát kết quả. Sau khi tối ưu phương pháp, threshold đã không còn là vấn đề gây khó dễ, và kết quả các bước đã trở nên rõ ràng, cụ thể và triệt để.

Link GitHub: https://github.com/MinHius/Midterm-Digital-Image-Processing