**Nội dung báo cáo Project 2**

1. **Thông tin sinh viên**

| Mã sinh viên | Họ và tên đệm | Tên |
| --- | --- | --- |
| 21115991 | Võ Công | Đoàn |
| 21120671 | Nguyễn Thành | Trung |
| 21064051 | Nguyễn Hùng | Anh |

1. **Thống kê mức độ hoàn thành**

| STT | Các chức năng | Mức độ hoàn thành | Sinh viên thực hiện | | |
| --- | --- | --- | --- | --- | --- |
| 1 | Xử lý x-quang xương người | 100% | Nguyễn Hùng Anh  Võ Công Đoàn  Nguyễn Thành Trung | | |
| 2 | T-Rex | 100% | Nguyễn Hùng Anh | | |
| 3 | Xúc xắc | 100% | Nguyễn Thành Trung  Võ Công Đoàn | | |

1. **Phân tích mô tả thuật toán**

**a. Xử lý x-quang xương người:**

**-** Ý tưởng:

1. Xây dựng hàm tính tích chập
2. Xây dựng thêm hàm Laplacian filter
3. Thực hiện tính toán Laplacian
4. Dùng hàm để trừ 2 hình ảnh với nhau
5. Thực hiện làm sắc nét ảnh
6. Xây dựng hàm cộng 2 hình ảnh với nhau
7. Tạo hàm tính sobel filter
8. Thực hiện tính sobel filter
9. Tạo hàm average filter
10. Thực hiện tính average filter
11. Tạo hàm để nhân 2 ma trận
12. Nhân 2 ma trận với nhau
13. Thực hiện cộng cộng 2 hình ảnh
14. Áp dụng biến đổi power law
15. In ra hình ảnh đã biến đổi

- Mô tả thuật toán:

1. Hàm tính tích chập:

- Lấy kích thước của kernel

- Dùng cv2.copyMakeBorder để thêm các giá trị pixel ở biên ngoài của ảnh

- Lấy kích thước và tính kích thước mới cho ảnh

- Tạo mảng mới cho ảnh sau tích chập

- Dùng for để thực hiện phép tính tích chập

1. Hàm laplacian filter:

- Tạo kernel

- Thêm padding

- Sử dụng convolution2d để thực hiện phép tích chập giữa ảnh và kernel laplacian

- Trả về ảnh sau khi lọc

1. Hàm thực hiện trừ 2 hình ảnh với nhau:

- Lấy kích thước của ảnh

- Tạo mảng để chứa kết quả

- Dùng vòng lặp for để thực hiện phép trừ

- Trả về ảnh kết quả

- Thực hiện làm sắc nét ảnh

1. Hàm thực hiện cộng 2 hình ảnh với nhau:

- Lấy kích thước của ảnh

- Tạo mảng để chứa kết quả

- Dùng vòng lặp for để thực hiện phép cộng

- Trả về ảnh kết quả

1. Hàm tính sobel filter:

- Tạo kernel x và y cho sobel

- Dùng convolution2d để áp dụng kernel sobel lên ảnh

- Dùng add images để cộng 2 ảnh lại

- Tiếp tục dùng add\_images để cộng với ảnh gốc

- Trả về ảnh kết quả

- Thực hiện phép tính sobel

1. Hàm tính average filter:

- Tạo kernel

- Dùng convolution2d để thực hiện phép tích chập giữa ảnh và kernel

- Trả về ảnh kết quả

- Thực hiện phép tính average filter

1. Hàm nhân 2 ma trận:

- Lấy kích thước của ảnh

- Tạo mảng rỗng cho kết quả

- Dùng vòng lặp for để thực hiện phép nhân

- Gán giá trị vào ảnh kết quả

- Trả về ảnh kết quả

- Thực hiện nhân 2 ma trận với nhau

1. Hàm power law:

- Lấy giá trị gamma = 0.5

- Lấy ảnh nhân lũy thừa với gamma

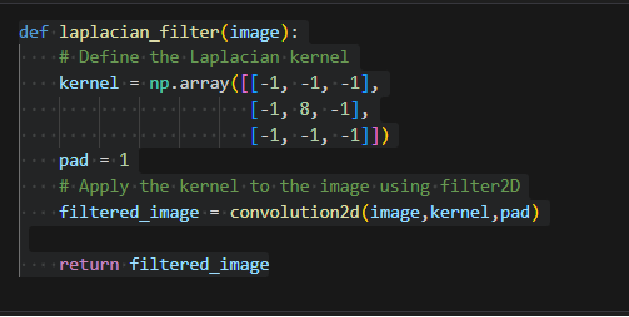
- Trả về kết quả

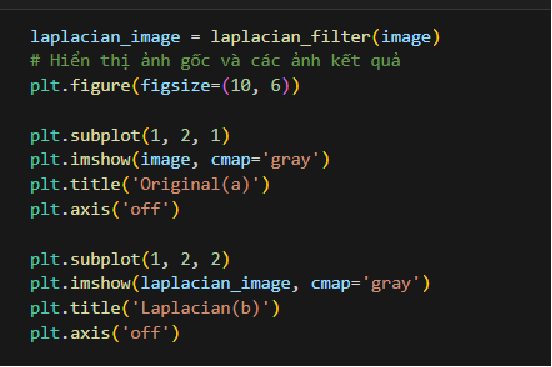
- Áp dụng hàm vào ảnh

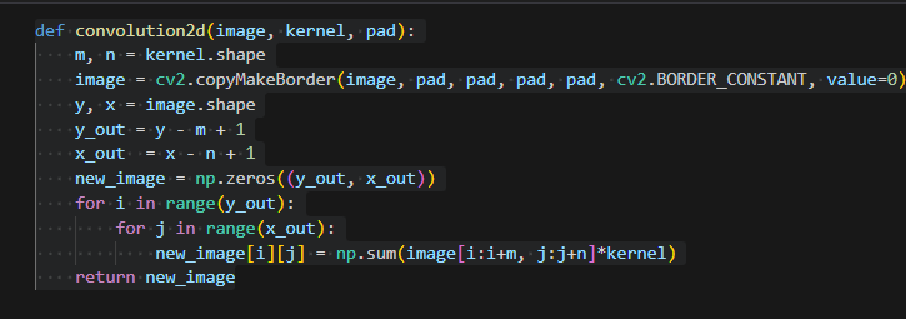
- In ra ảnh gốc và ảnh vừa biến đổi để so sánh

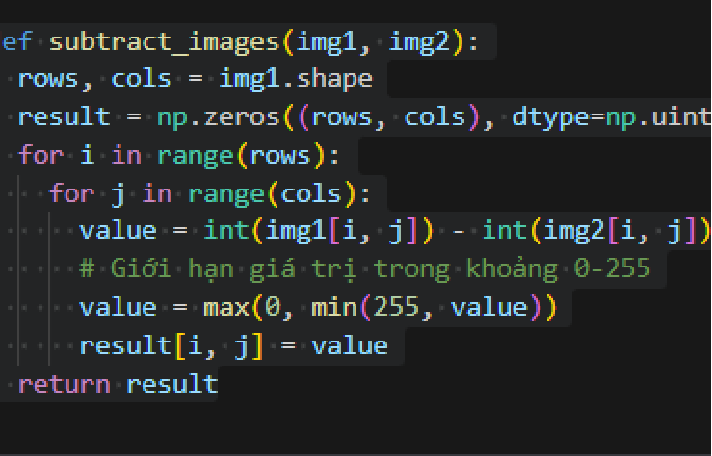
- Code:

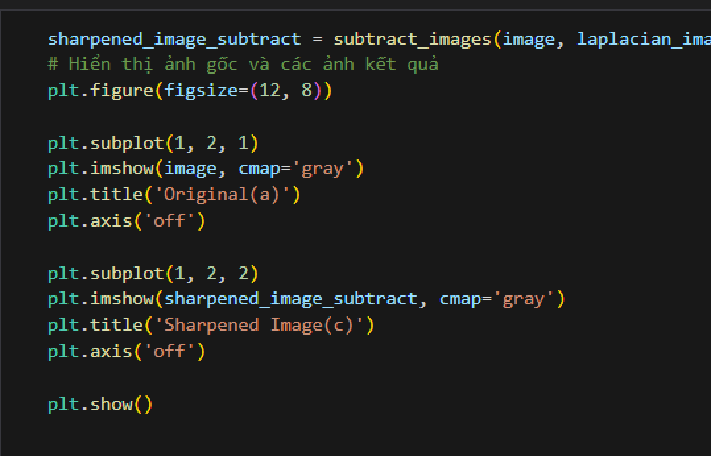


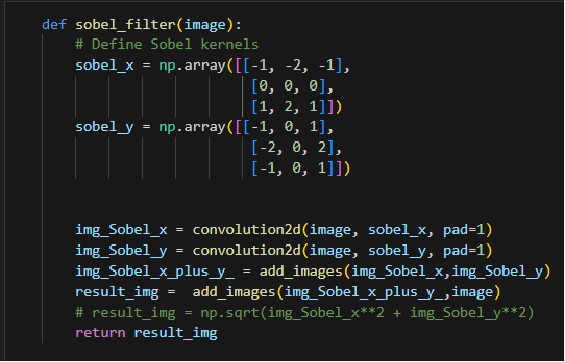


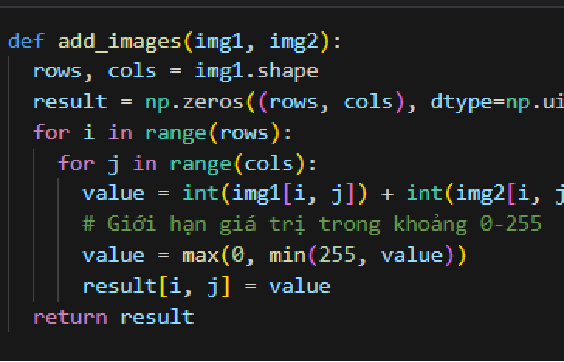


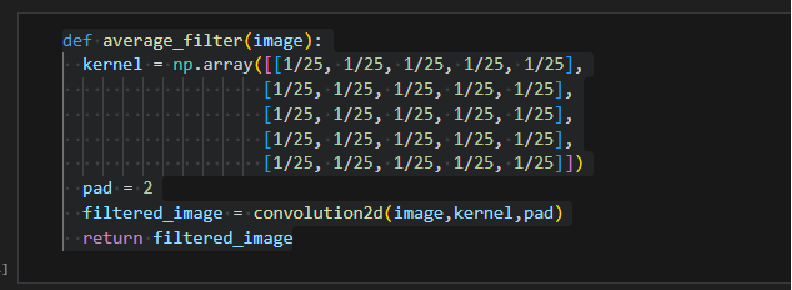


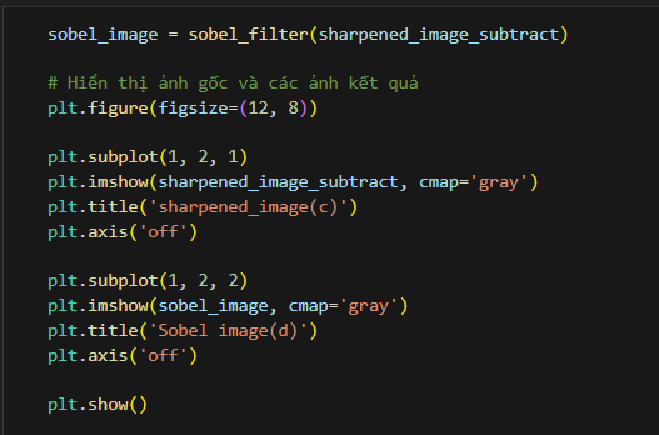


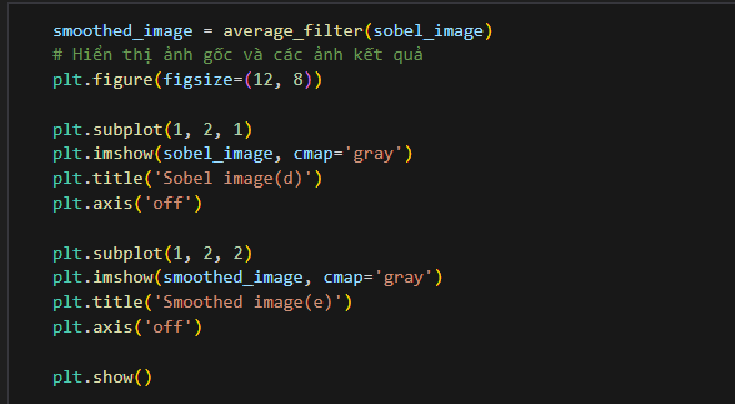




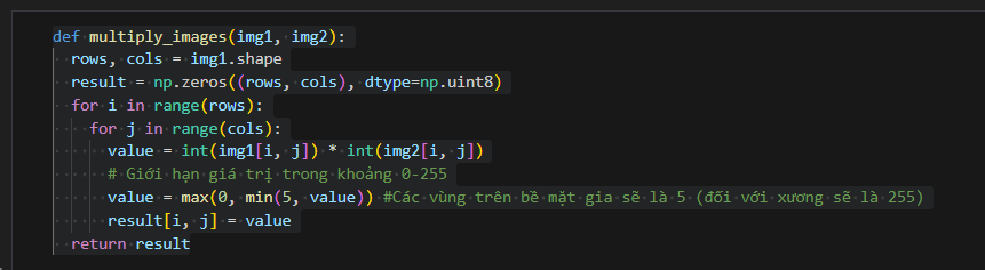


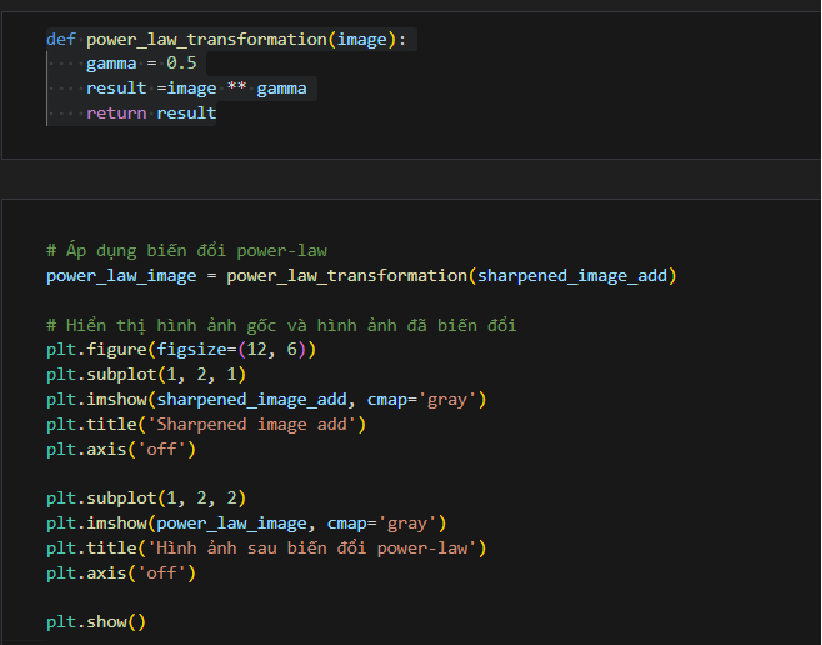


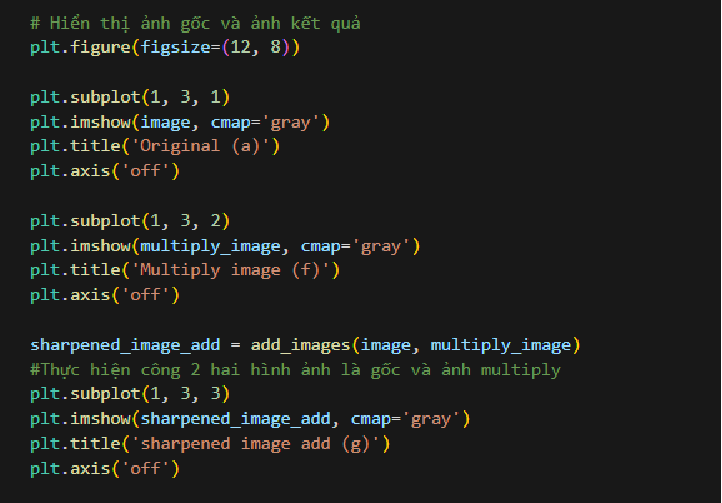


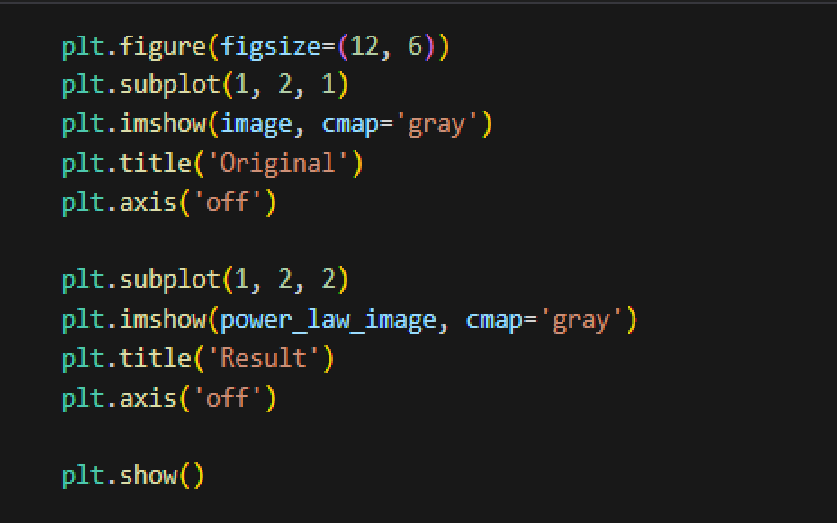




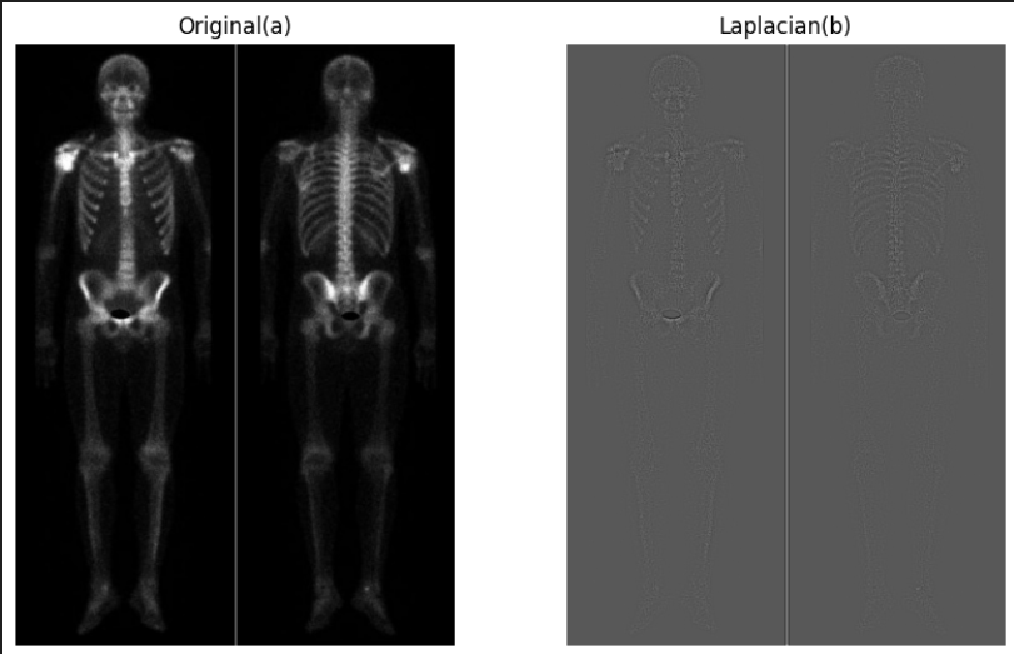


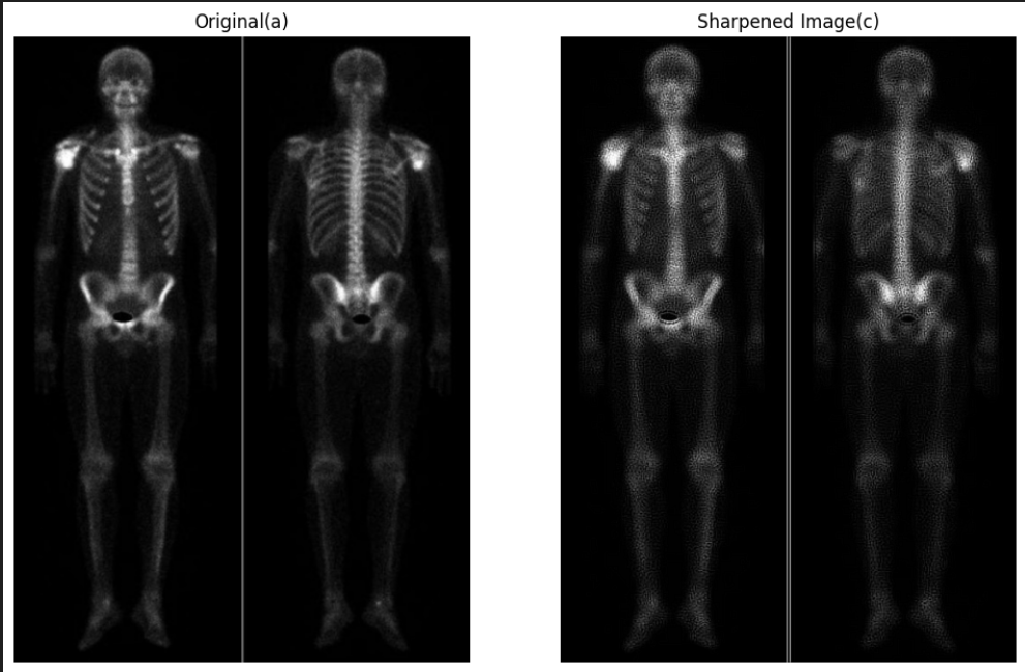


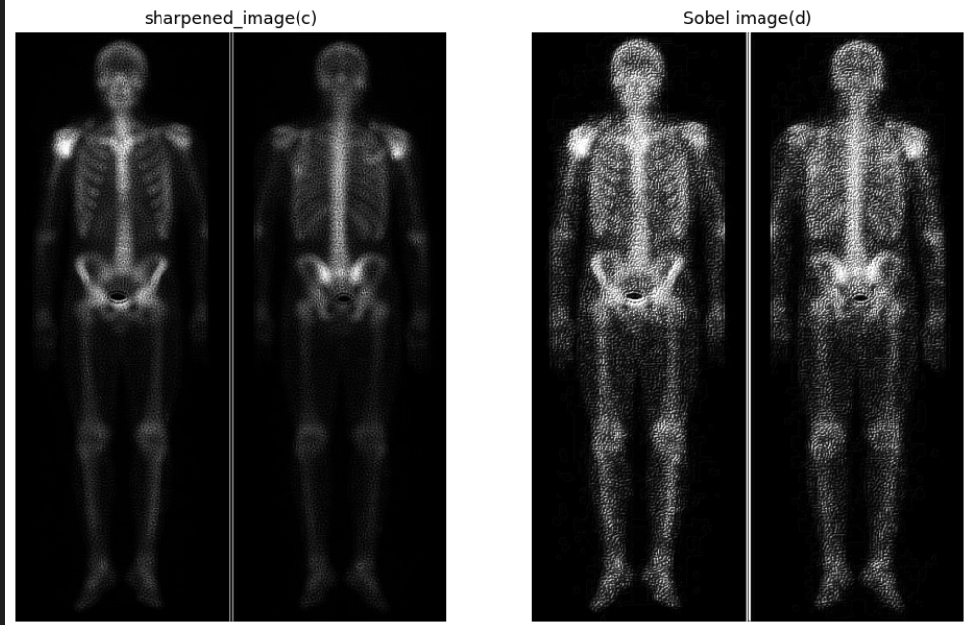


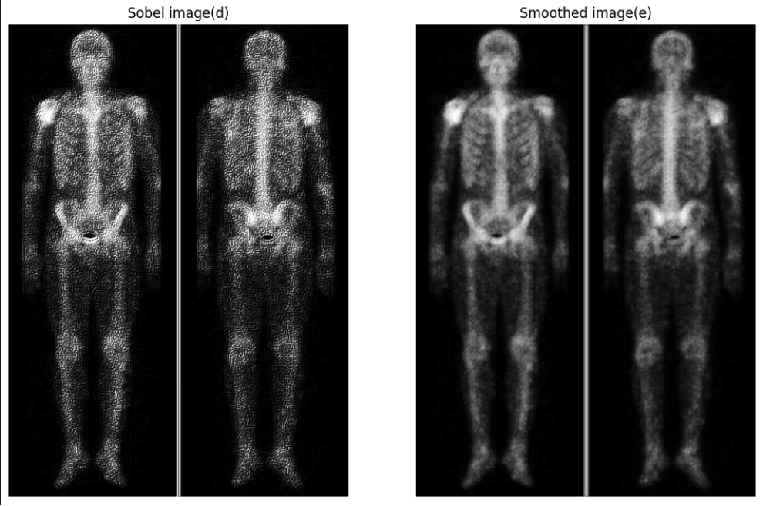


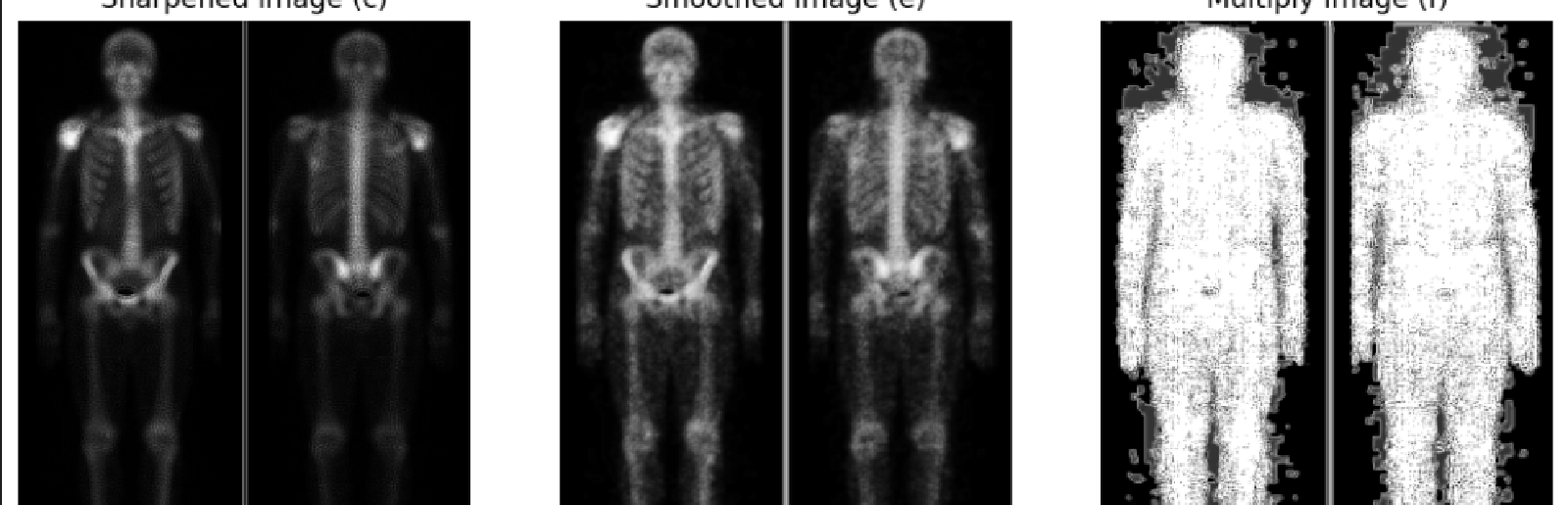
- Kết quả:

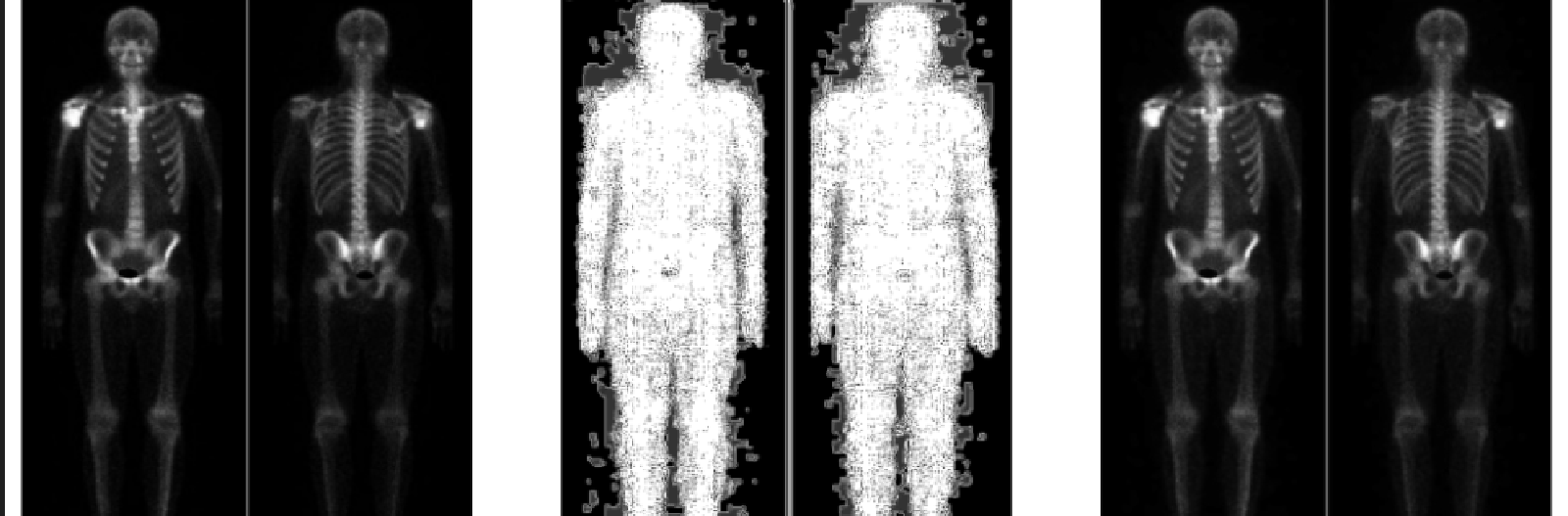


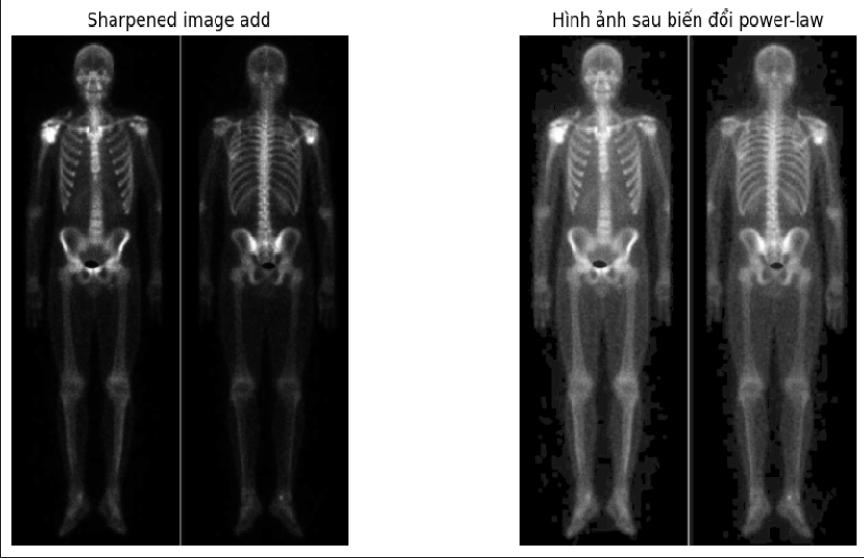


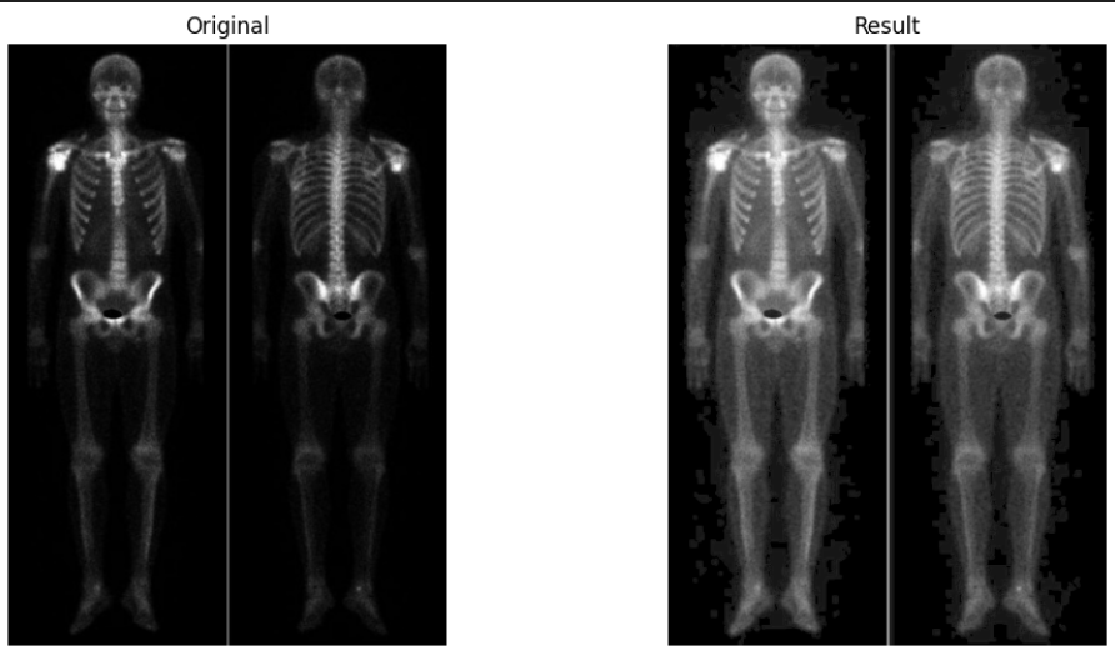












**b. T-rex:**

- Ý tưởng:

Mục tiêu là phát hiện các đối tượng trong ảnh, cụ thể là xác định các đường viền của xúc sắc trong ảnh bằng cách sử dụng các phép biến đổi hình học cơ bản như co (erosion) và giãn (dilation), sau đó xác định các đường viền bằng các phương pháp tìm kiếm khác nhau.

- Mô tả thuật toán

1. Ngưỡng hóa (Thresholding):

Ý tưởng: Chuyển đổi ảnh màu hoặc ảnh xám thành ảnh nhị phân (đen trắng) bằng cách sử dụng một ngưỡng xác định. Các pixel có giá trị lớn hơn ngưỡng sẽ trở thành trắng (255), và các pixel nhỏ hơn sẽ trở thành đen (0).

Hàm thực hiện: threshold(image, threshold=127).

Quy trình:

Tạo một ảnh nhị phân có cùng kích thước với ảnh gốc, khởi tạo tất cả pixel là 0.

Duyệt qua từng pixel trong ảnh gốc. Nếu pixel lớn hơn ngưỡng, gán giá trị 255 cho pixel tương ứng trong ảnh nhị phân.

1. Co (Erosion):

Ý tưởng: Xóa bỏ các pixel biên (border pixels) của các đối tượng, làm nhỏ kích thước của chúng.

Hàm thực hiện: erode(image, kernel\_size).

Quy trình:

Tạo một kernel hình vuông với kích thước kernel\_size.

Duyệt qua từng pixel trong ảnh (bỏ qua vùng biên tương ứng với kích thước của kernel).

Tại mỗi pixel, lấy vùng lân cận và gán pixel hiện tại bằng giá trị nhỏ nhất trong vùng lân cận.

1. Giãn (Dilation):

Ý tưởng: Mở rộng các pixel sáng, làm tăng kích thước của các đối tượng trong ảnh.

Hàm thực hiện: dilate(image, kernel\_size).

Quy trình:

Tạo một kernel hình vuông với kích thước kernel\_size.

Duyệt qua từng pixel trong ảnh (bỏ qua vùng biên).

Tại mỗi pixel, lấy vùng lân cận và gán pixel hiện tại bằng giá trị lớn nhất trong vùng lân cận.

1. Tìm kiếm đường viền (Contours):

Ý tưởng: Phát hiện các đường biên của các đối tượng trong ảnh nhờ vào việc so sánh giữa ảnh nhị phân và ảnh đã trải qua các phép co và giãn.

Hàm thực hiện: findContours\_RETR\_EXTERNAL(binary\_color, dilate\_color, erode\_color) và findContours\_RETR\_LIST(dilate\_color, erode\_color).

Quy trình:

Duyệt qua từng pixel của ảnh nhị phân và ảnh đã giãn nở.

Nếu một pixel là đen trong ảnh nhị phân nhưng trắng trong ảnh giãn nở, đánh dấu pixel đó là xanh lá cây trong ảnh kết quả.

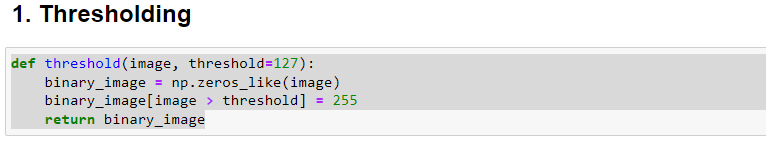
Nếu pixel trắng trong cả hai ảnh, giữ nguyên màu trắng.

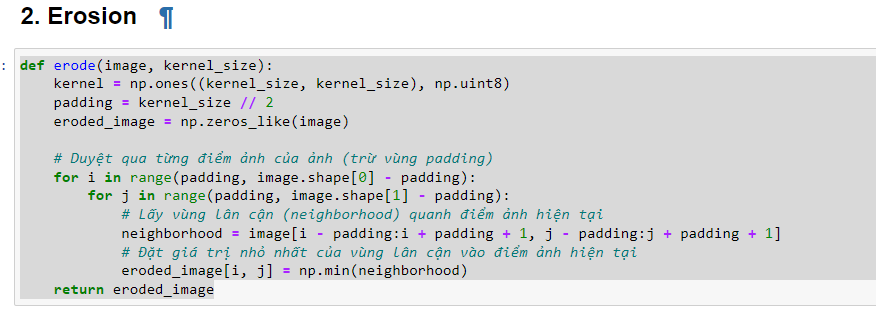
Sử dụng kết quả từ phép co để làm sạch các pixel không cần thiết.

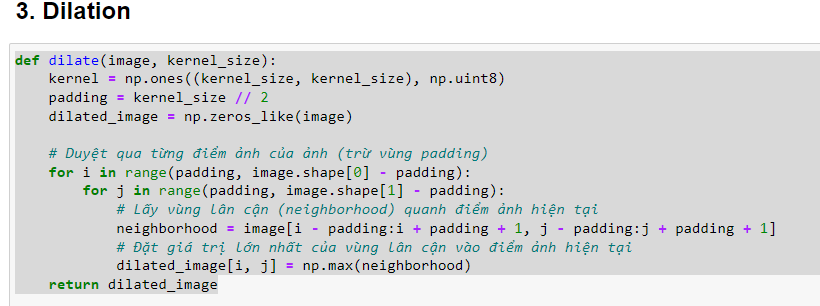
1. Hiển thị kết quả:

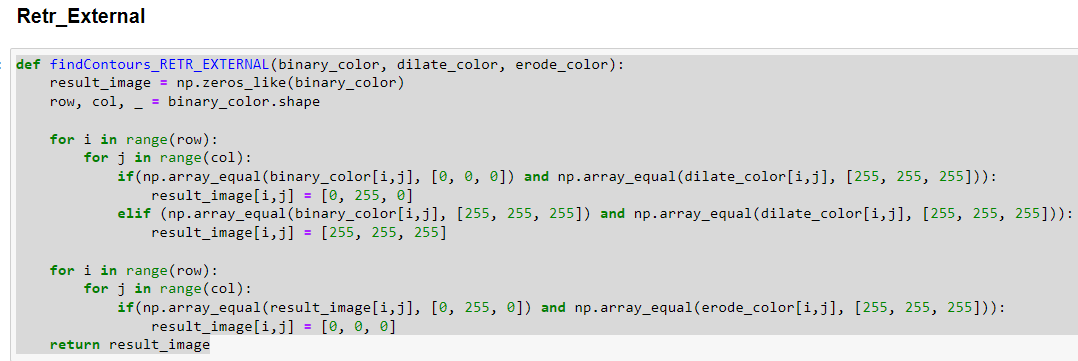
Sử dụng matplotlib để hiển thị ảnh gốc, ảnh nhị phân, ảnh sau khi giãn, ảnh sau khi co và ảnh kết quả với các đường viền đã xác định.

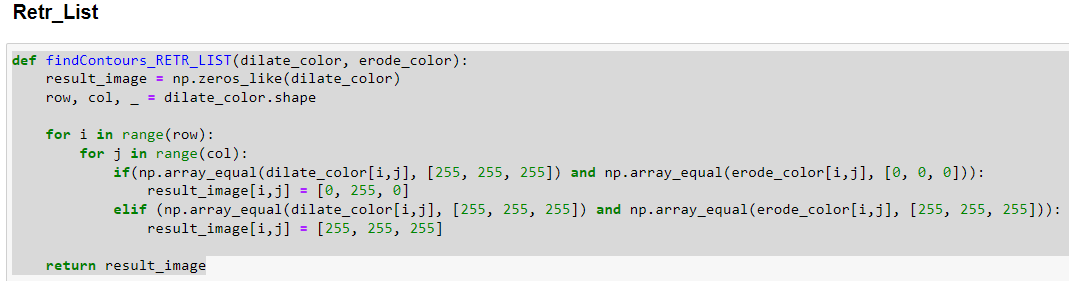
- Code

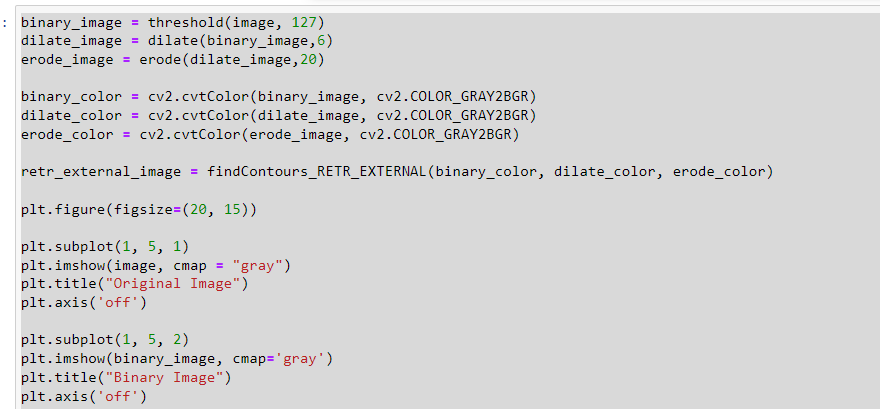


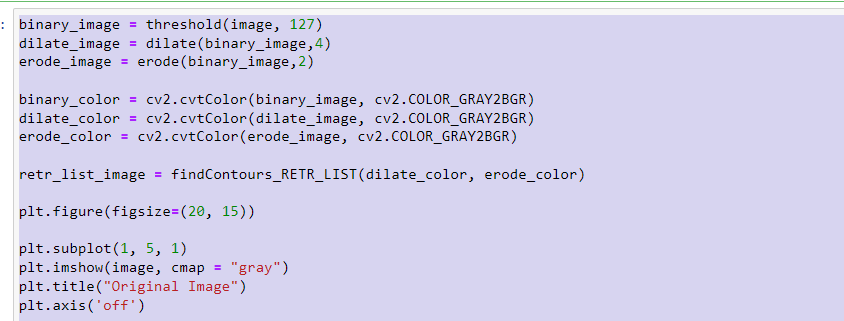




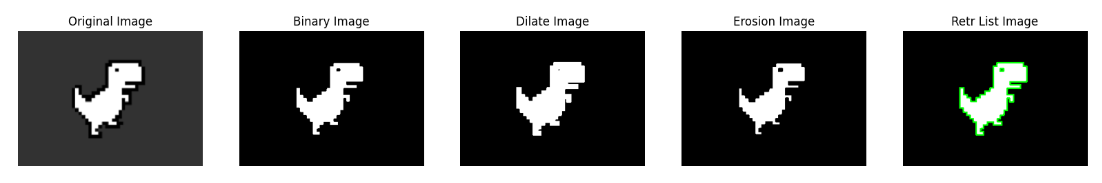


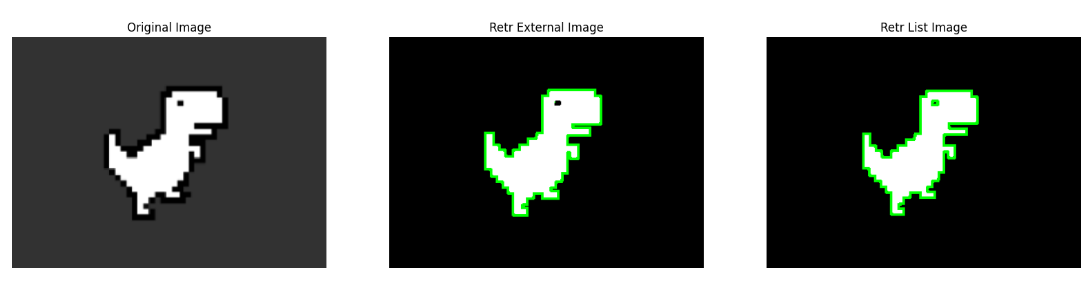


****

****

**-Kết quả**

****

****

**c. Xúc xắc:**

- Ý tưởng:

1. Đọc và tiền xử lý ảnh:

Đọc ảnh bằng OpenCV.

Chuyển đổi ảnh thành thang độ xám.

Làm mờ ảnh để giảm nhiễu bằng bộ lọc Gaussian hoặc bộ lọc trung vị.

1. Phát hiện cạnh và thresholding:

Dùng phương pháp Canny để phát hiện các cạnh.

Áp dụng threshold để phân vùng các phần sáng và tối trên ảnh.

1. Xử lý hình thái học (Morphology Operations):

Dùng phép đóng (closing) để lấp đầy các lỗ nhỏ trong vùng xác định.

Có thể áp dụng phép giãn hoặc co để tinh chỉnh hình dạng.

1. Phát hiện hình tròn (HoughCircles):

Dùng HoughCircles để phát hiện các hình tròn nhỏ trên mặt xúc xắc (đại diện cho các dấu chấm).

1. Đếm số chấm trên xúc xắc:

Đếm số lượng vòng tròn phát hiện được để xác định số mặt hiện trên xúc xắc.

- Mô tả thuật toán:

1. Tiền xử lý ảnh:

Đọc ảnh và chuyển đổi sang ảnh xám (cv2.cvtColor).

Áp dụng làm mờ ảnh (cv2.GaussianBlur).

1. Phát hiện cạnh:

Dùng cv2.Canny để phát hiện cạnh.

1. Áp dụng threshold và phép đóng:

Dùng cv2.threshold để phân ngưỡng ảnh.

Áp dụng cv2.morphologyEx với kernel để thực hiện phép đóng.

1. Phát hiện hình tròn với HoughCircles:

Dùng cv2.HoughCircles để tìm các hình tròn trên mặt xúc xắc.

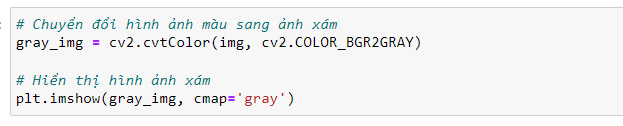
1. Hiển thị kết quả:

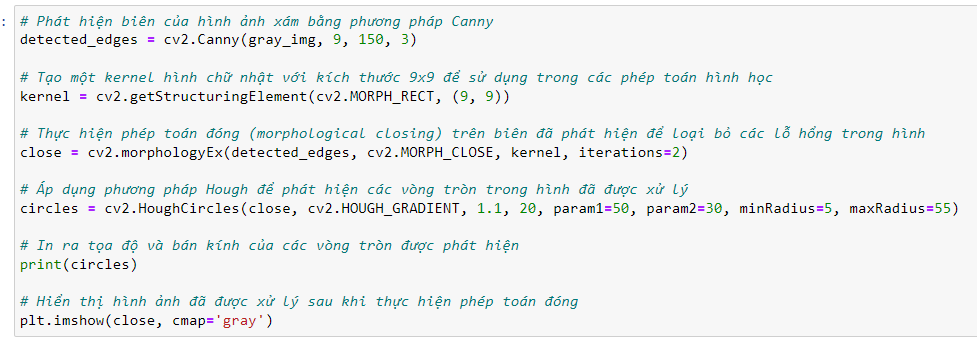
Vẽ các hình tròn tìm được lên ảnh gốc.

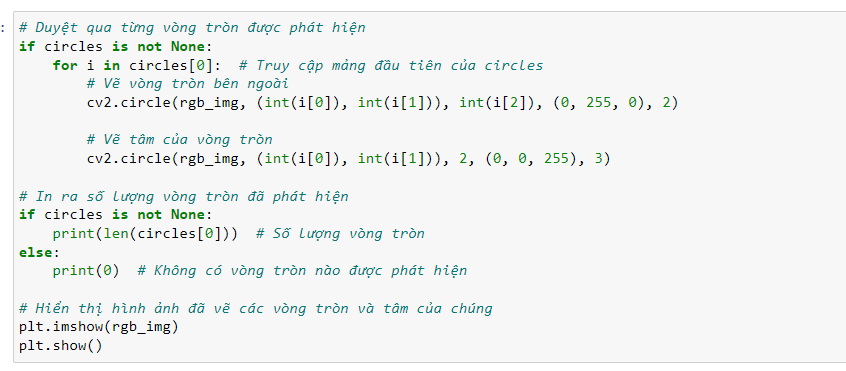
Hiển thị kết quả bằng cv2.imshow

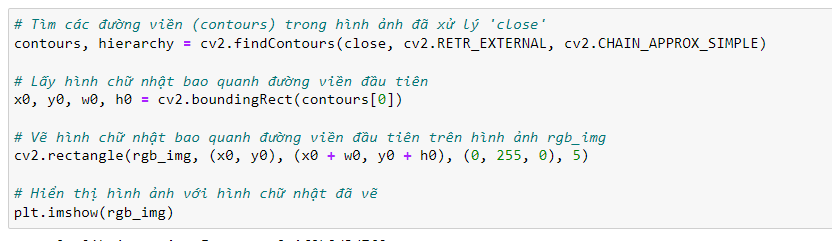
- Code:

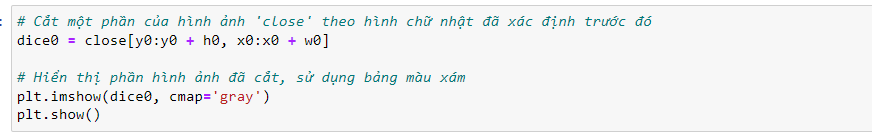


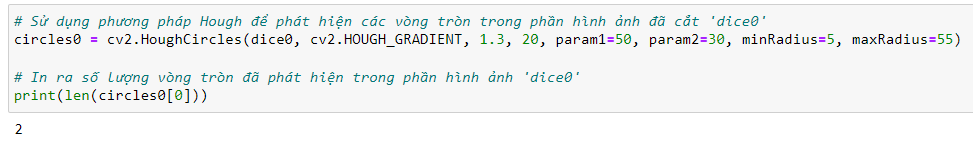












- Kết quả:

