COSC 4372 – Assignment 1

Name: Dan Kalhori

PSID: 2153552

9/10/2024

github link: https://github.com/DanKalh/4372-hw1-fall2024-DanKalh

I Problem

Q1. Modify the Shepp-Logan phantom: Start with the source code of the phantom() function in Matlab or other language of choice.

Q.1.1 Implement a function that takes the following parameters: Phantom Matrix Size (N), Number of ellipses (M), and for each ellipse, the center position (X, Y), rotation angle (ANGLE), size (width, height) and signal intensity (SI). The function generates a Shepp-Logan phantom using these input values. (40 pts)

Attached to end.

Q.1.2 What is the signal intensity of the background? (5 pts)

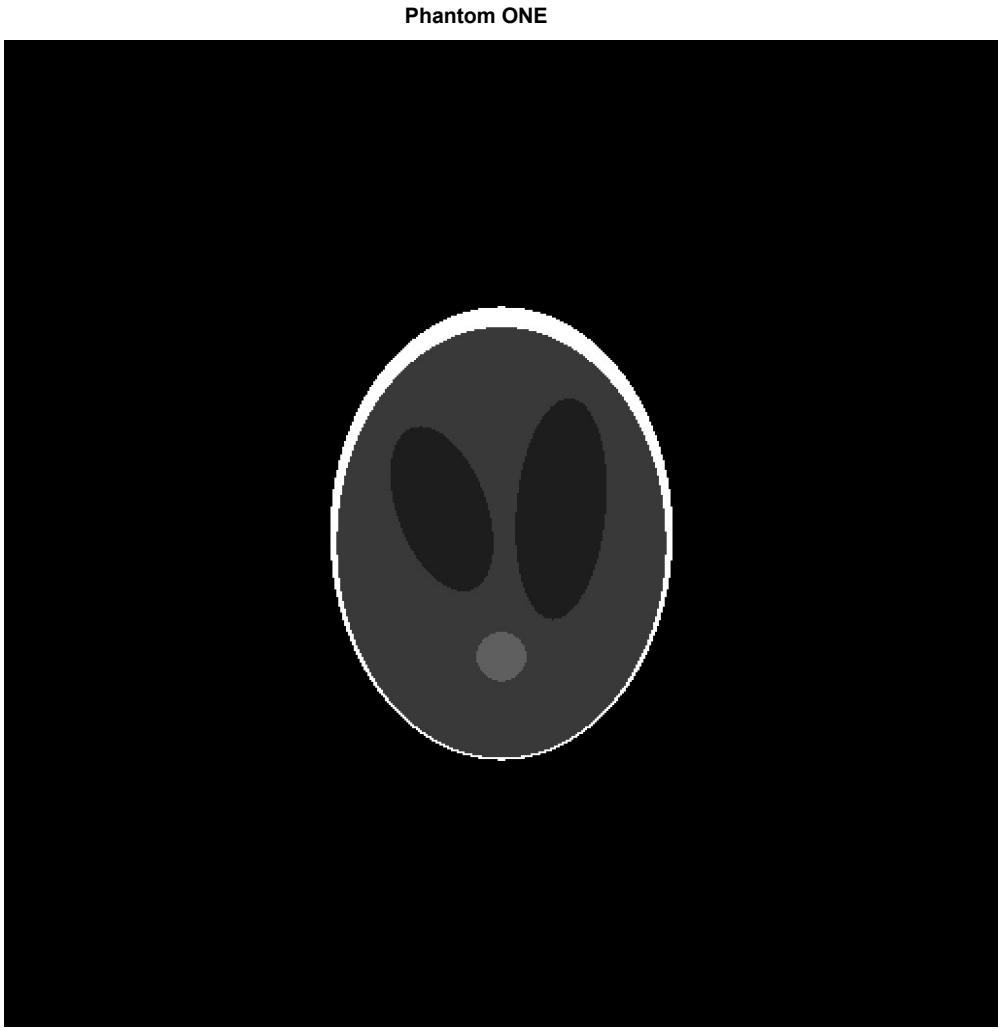
0.

Q.1.3 What would you expect the signal intensity of the background to be in the real world? (5pts)

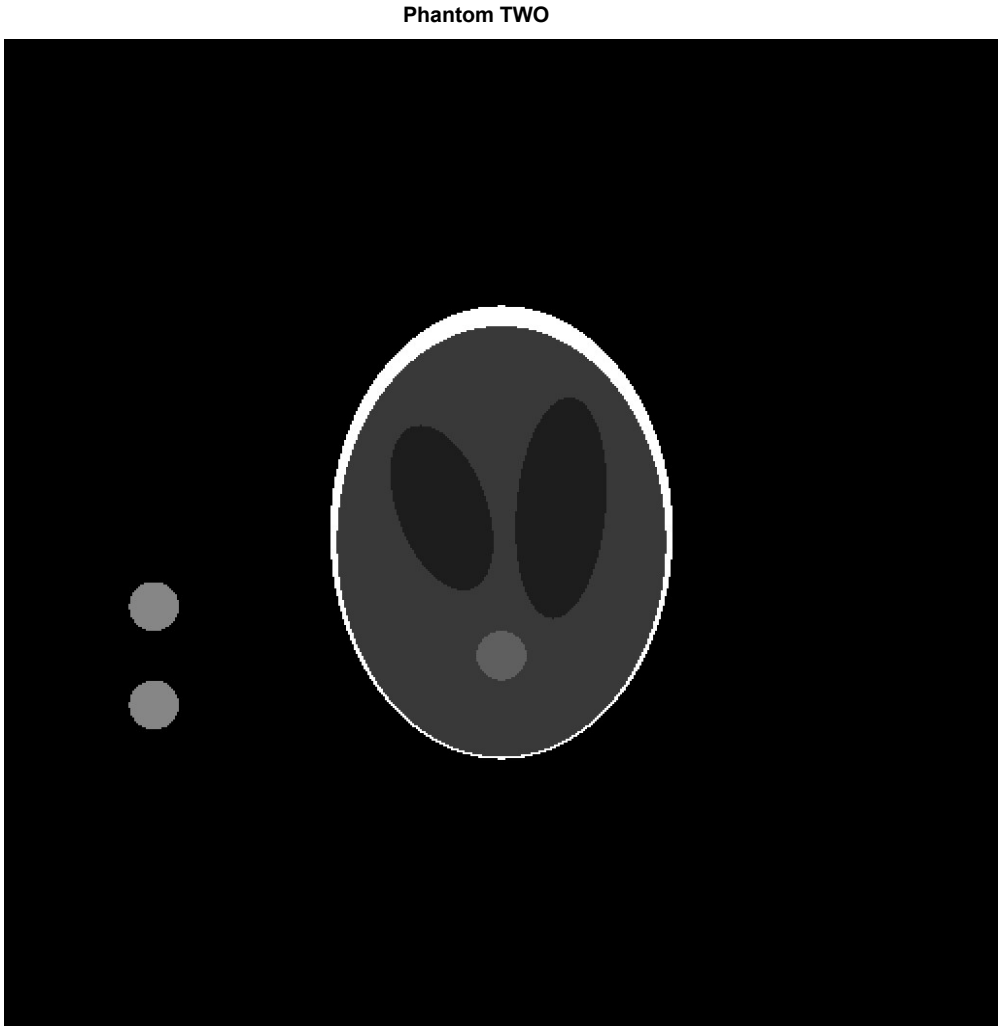
Very close to 0, but probably not actually 0.

Q2: Test the Code: Run the code you generated in Q1 to create three phantoms:

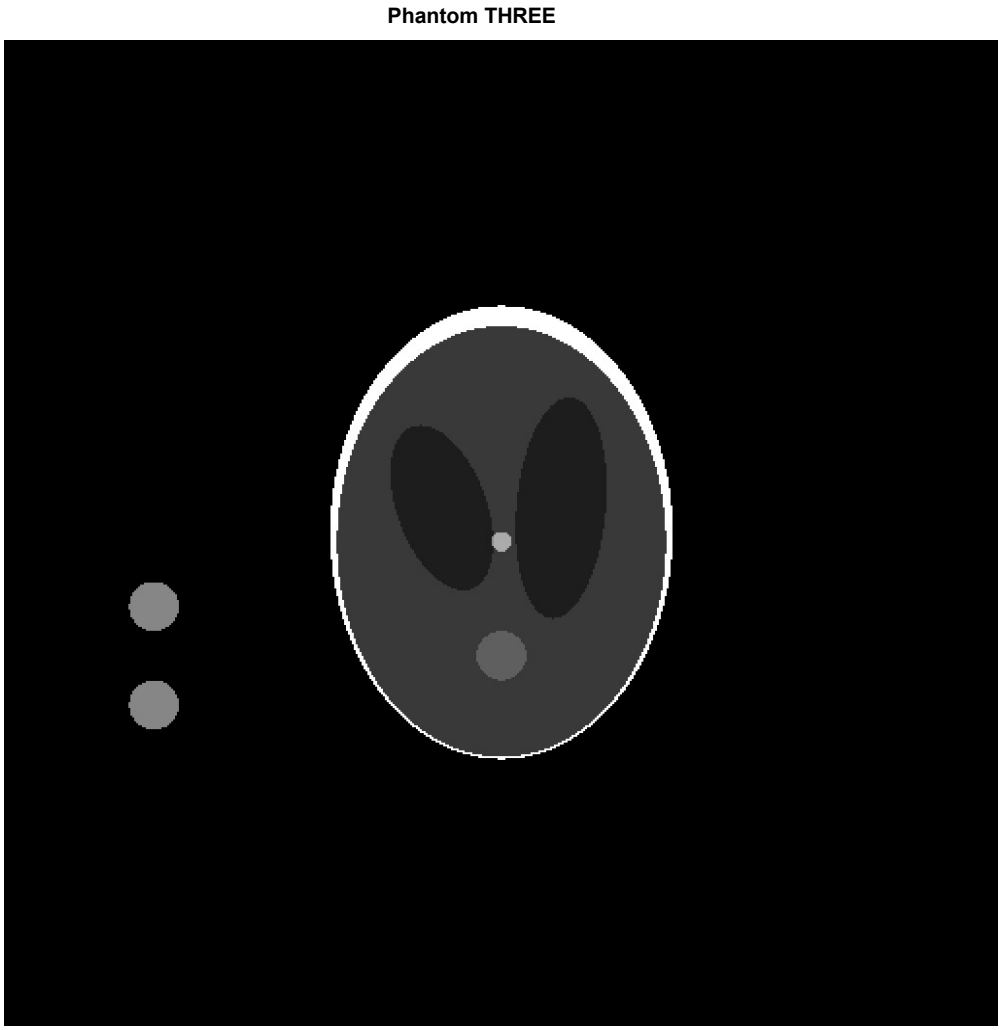
Q.2.1 One with no overlapping structures. As an example: make structure (1) smaller, move structure (2) positioned lower, etc. (10 pts)



Q.2.2 Start with the output of Q.2.1 (modified phantom) and add two circular structures outside the brain in the empty space. (10 pts)



Q.2.3 Create a phantom that is composed of three concentric circles. (10 pts)



II Method

We utilize the Shepp-Logan phantom concept, where ellipses are used to represent anatomical structures. Each ellipse has parameters for center position, rotation angle, size, and signal intensity. By modifying these parameters, we generate a series of progressively more complex phantom images, starting with basic structures and adding features like circular tumors and concentric ellipses.

III Implementation

I used MATLAB to implement a `modified_phantom` function that takes the matrix size, number of ellipses, center positions, rotation angles, sizes, and signal intensities as input parameters. We create three variations:

1. **Phantom One**: Basic structure with no overlapping ellipses.
2. **Phantom Two**: Builds on Phantom One by adding circular structures outside the brain.
3. **Phantom Three**: Creates a phantom with three concentric circles.

The code progressively adds new features and generates each image.

IV Results

The phantoms successfully represent very basic anatomical features using ellipses. The first phantom shows a basic non-overlapping structure, the second adds external circular features(for reference as mentioned in class), and the third creates a set of concentric circles although kind of used throughout to for a “head”. These phantoms can be used for testing imaging algorithms.

Modified Phantom:

```
function phantom_image = dynamic_phantom(N, M, center_positions, rotation_angles, sizes,
signal_intensities)
    % modified_phantom: Generate a custom phantom with dynamic number of ellipses.
    %   Inputs:
    %       N               - Phantom matrix size (NxN).
    %       M               - Number of ellipses.
    %       center_positions - Mx2 matrix, where each row is [X, Y] for ellipse center.
    %       rotation_angles  - Mx1 vector, each element is the rotation angle (in degrees).
    %       sizes            - Mx2 matrix, where each row is [width, height] of the ellipse.
    %       signal_intensities - Mx1 vector, signal intensity of each ellipse.
    %
    %   Outputs:
    %       phantom_image    - The generated phantom image.

    % Initialize the phantom matrix to zeros
    phantom_image = zeros(N);

    % Generate x and y axis grids that span [-1, 1]
    x_axis = ((0:N-1) - (N-1)/2) / ((N-1)/2);
    x_grid = repmat(x_axis, N, 1);   % X coordinates
    y_grid = rot90(x_grid);          % Y coordinates

    % Loop through each ellipse and add it to the phantom
    for k = 1:M
        % Extract ellipse properties for the current ellipse
        center_x = center_positions(k, 1);
        center_y = center_positions(k, 2);
        rotation_angle = rotation_angles(k);
        semi_axis_a = sizes(k, 1) / 2;  % Horizontal semi-axis (width / 2)
        semi_axis_b = sizes(k, 2) / 2;  % Vertical semi-axis (height / 2)
        signal_intensity = signal_intensities(k);

        % Convert rotation angle to radians
        rotation_angle_rad = rotation_angle * pi / 180;

        % Shift the x and y grids by the ellipse center position
        x_shifted = x_grid - center_x;
        y_shifted = y_grid - center_y;

        % Rotate the coordinates based on the rotation angle
        x_rotated = x_shifted * cos(rotation_angle_rad) + y_shifted * sin(rotation_angle_rad);
        y_rotated = -x_shifted * sin(rotation_angle_rad) + y_shifted * cos(rotation_angle_rad);

        % Equation of the ellipse (normalized form)
        ellipse_equation = (x_rotated.^2) / (semi_axis_a^2) + (y_rotated.^2) / (semi_axis_b^2);

        % Find pixels inside the ellipse (where equation is <= 1)
```

```
        ellipse_mask = ellipse_equation <= 1;

        % Add the ellipse to the phantom with the specified signal intensity
        phantom_image(ellipse_mask) = phantom_image(ellipse_mask) + signal_intensity;
    end
end
```