COSC 4372 Medical Imaging Mammography Project

ClearView Imaging Members: Andrew Guzman, Jasmine Garcia, Abiha Fatima

Introduction

Breast cancer is a significant concern for women worldwide, with early detection playing a crucial role in improving survival rates. Advancements in medical imaging have enhanced the ability to detect and treat breast cancer early, saving lives. However, the development and testing of imaging techniques often involve expensive, time-consuming processes and require state-of-the-art equipment.

The objective of this project that we were tasked with is to develop software that simulates a conventional film-based X-ray machine using a graphical user interface (GUI). The software allows users to modify acquisition parameters, view reconstructed images, and perform basic analysis. It incorporates a 2D phantom for validating algorithms and analyzing parameter effects, a 3D phantom designed to mimic anatomical structures like a breast for mammography imaging, and finally, a function to produce a 2D mammogram from the 3D phantom using user-defined parameters.

To learn and provide future success, Simulation, and Modeling offer promising alternatives for developing medical imaging techniques. Simulation allows replicated real-world imaging systems without the need for physical patients to test on. Modeling complements this by creating realistic representations of anatomical structures, such as a breast phantom or cancerous lesion, to evaluate imaging techniques. Both Simulation and Modeling can form the foundation of a Digital Twin, which is a virtual replica of a physical system that copies a specific behavior. By creating a digital twin, medical imaging systems can be developed and analyzed in a more cost-effective, flexible, and experimental manner, paving the way for improved efficiency and innovation in healthcare.

In the competitive medical imaging market for mammography X-rays, major players such as Canon Medical Systems, Siemens Healthineers, Planmed Oy, Konica Minolta, Metaltronica SpA, Philips, Fujifilm, and Carestream Health dominate the field. These companies compete by focusing on factors like technological advancements in image quality, patient comfort, radiation dose optimization, comprehensive service support, and cost-effectiveness of their mammography machines. Their goal is to capture a larger share of the growing market for breast cancer screening and diagnosis through mammograms. Despite the advancements offered by these industry leaders, many of their solutions come with significant barriers, including high costs, complexity, and limited accessibility for smaller institutions or researchers. Additionally, most commercial systems are aimed toward clinical application rather than research, experimentation, or education, making it difficult for students and researchers to test novel imaging concepts.

To address a solution to the competitive market, this mammography project proposes a MATLAB-based solution that leverages the concept of a digital twin to simulate a conventional

mammography system. This system provides an accessible, cost-effective, and flexible platform for exploring mammographic imaging. The customizable 3D phantom model's anatomical structures, such as a breast and lesion, while integrating key parameters like beam energy, X-ray cone angle, and distances between the X-ray source, phantom, and film. By enabling users to modify these parameters and visualize the results dynamically, this tool serves as a valuable resource for optimizing imaging systems, studying imaging effects, and improving lesion detectability, all without the need for physical equipment or costly commercial software.

Methods

The Mammography project that we created used MATLAB to be able to generate the code. The simulation involves 2 phantoms, as highlighted before, a 2D phantom for validating algorithms and analyzing the effects of parameter changes, and a 3D phantom designed to mimic anatomical structures, such as a breast and lesion. Here is an in-depth overview of the code used for both phantoms and the method behind creating this project:

3D Phantom

1) Parameters Used:

- Beam Energy (15,19,26 keV): Controls the attenuation coefficients (μ) of the breast and lesion tissues.
- X-Ray Cone Angle (40°-60°): Defines the angle of the X-ray cone, and influences the beam spread.
- Phantom-to-Film Distance (0-60 cm): Positions the phantom relative to the film.
- Source-to-Phantom Distance (0-60 cm): Positions the phantom relative to the source.

```
beamEnergy = 15; % Beam energy in keV (15,19,26)
xrayAngle = 40; % X-ray cone angle in degrees (40-60)
phantomToFilm = 60; % Distance of the film from the phantom in cm (0-60)
phantomToSource = 60; % Distance of the x-ray source from the phantom in cm (0-60)
```

2) Phantom and Environment Initialization:

Since this is a 3D phantom, this is the area showing the dimensions and positioning of the phantom (breast and lesion) and surrounding elements (wall and film).

Variables:

- phantomCenter: This is the center of the breast sphere (outer).
- radiusBreast and radiusLesion: This is the radius of the breast and lesion spheres.
- wallHeight, wallWidth, and wallThickness: The dimensions of the simulated wall.

```
wallHeight = 120;
wallWidth = 120;
wallThickness = 10;
phantomCenter = [60, 60, 70];
radiusBreast = 20;
radiusLesion = 8;
filmSize = [wallWidth, wallHeight];
```

3) Attenuation Coefficients Based on Beam Energy:

The getAttenuationCoefficients function dynamically calculates the linear attenuation coefficients (μ) for the breast and lesion tissues based on the beam energy. The purpose is to reduce attenuation when higher beam energy is used, affecting image contrast.

```
% Get μ Values Based on Beam Energy
[muBreast, muLesion] = getAttenuationCoefficients(beamEnergy);
```

4) Grid and Distance Calculations:

This part of the code creates a 3D grid while using the function ndgrid, and calculates the distance from the phantom center to each point in the grid. It is used to determine whether a grid point is within the breast or lesion region.

```
wall3D = zeros(wallHeight, wallWidth, wallThickness);

[x, y, z] = ndgrid(1:wallWidth, 1:wallHeight, 1:wallThickness + 100);
distanceBreast = sqrt((x-phantomCenter(1)).^2 + (y-phantomCenter(2)).^2 + (z-phantomCenter(3)).^2);
distanceLesion = sqrt((x-phantomCenter(1)).^2 + (y-phantomCenter(2)).^2 + (z-phantomCenter(3)).^2);
```

5) Tissue Properties:

The following section assigns attenuation coefficients (μ) to the breast and lesion regions. The phantom is split vertically, with one side solid and the other transparent. We split the sphere because we wanted to be able to see both the outer and inner layers of the phantoms. When adjusting the beam energy, the user can see noticeable differences between the outer layer of the phantom and the cancerous lesion that is inside. So, this gives a better representation of what is going on instead of leaving the user to interpret the changes.

```
% Create 3D Phantom with Vertical Split
phantom3D = zeros(size(distanceBreast));

% Left Side: Solid Breast
phantom3D(distanceBreast <= radiusBreast & x <= phantomCenter(1)) = muBreast;

% Right Side: Transparent Breast with Lesion
phantom3D(distanceBreast <= radiusBreast & x > phantomCenter(1)) = muBreast / 2;
phantom3D(distanceLesion <= radiusLesion & x > phantomCenter(1)) = muLesion;
```

6) Visual:

The following code is responsible for rendering the phantom, wall, film, and X-ray cone in 3D. The components that are involved are:

- Wall: This is displayed by utilizing the isosurface and patch functions.
- Breast Sphere: Is displayed with it being split.
- Lesion Sphere: Shown on the transparent side of the breast sphere (right).
- Film: The flat plane underneath the phantom.
- X-Ray Cone: Simulated using the cylinder function.

```
% Wall (background)
[xWall, yWall, zWall] = ndgrid(1:wallWidth, 1:wallHeight, 1:wallThickness);
isosurface(xWall, yWall, zWall, wall3D, 0.1);
p = patch(isosurface(xWall, yWall, zWall, wall3D, 0.1));
set(p, 'FaceColor', [0.9, 0.9, 0.9], 'EdgeColor', 'none');
```

```
% Breast Sphere (Split)
breastSurf = isosurface(x, y, z, distanceBreast, radiusBreast);
```

```
% Lesion Sphere (Visible on Right Side Only)
lesionSurf = isosurface(x, y, z, distanceLesion, radiusLesion);
pLesion = patch(lesionSurf);
set(pLesion, 'FaceColor', [muLesion, muLesion, muLesion], 'EdgeColor', 'none', 'FaceAlpha', 1.0);
```

```
% Film (visualized as a flat plane)
[xFilm, yFilm] = meshgrid(1:filmSize(1), 1:filmSize(2));
zFilm = ones(size(xFilm)) * (phantomCenter(3) - phantomToFilm);
surf(xFilm, yFilm, zFilm, 'FaceColor', [0.3, 0.3, 0.3], 'EdgeColor', 'none');
```

```
% X-Ray Cone (Touches Film as Base)
coneBaseRadius = tan(deg2rad(xrayAngle/2)) * (phantomToSource + phantomToFilm);
[xCone, yCone, zCone] = cylinder([0, coneBaseRadius], 50);
zCone = -zCone * (phantomToSource + phantomToFilm) + xraySourceZ;
xCone = xCone + phantomCenter(1);
yCone = yCone + phantomCenter(2);
surf(xCone, yCone, zCone, 'FaceAlpha', 0.1, 'EdgeColor', 'none', 'FaceColor', [0.5, 0.5, 0.5]);
```

7) Extracting Vertices and Faces

The purpose of this section of code is to split the phantom into solid and transparent halves by manipulating the vertices and faces of the isosurface. The code splits the vertices into left and right halves based on the X-coordinate (phantomCenter(1)). The faces are updated to make sure they match the new vertex indices for the left and right sides.

- Vertices: The points defining the surface of the isosurface.
- Faces: The triangular patches connecting these vertices.

```
% Extract Vertices and Faces
vertices = breastSurf.vertices;
faces = breastSurf.faces;
```

```
% Split Vertices and Update Faces
% Left Side (Solid)
leftVerticesIdx = find(vertices(:, 1) <= phantomCenter(1));
leftVertices = vertices(leftVerticesIdx, :);
[~, newIdx] = ismember(faces, leftVerticesIdx);
leftFaces = newIdx(all(newIdx > 0, 2), :);
pLeft = patch('Vertices', leftVertices, 'Faces', leftFaces, 'FaceColor', [muBreast, muBreast, muBreast], ...
    'EdgeColor', 'none', 'FaceAlpha', 1.0);
```

```
% Right Side (Transparent with Lesion)
rightVerticesIdx = find(vertices(:, 1) > phantomCenter(1));
rightVertices = vertices(rightVerticesIdx, :);
[~, newIdx] = ismember(faces, rightVerticesIdx);
rightFaces = newIdx(all(newIdx > 0, 2), :);
pRight = patch('Vertices', rightVertices, 'Faces', rightFaces, 'FaceColor', [muBreast, muBreast, muBreast], ...
    'EdgeColor', 'none', 'FaceAlpha', 0.2);
```

8) Adjusting View and Lighting

The purpose of the lighting is to enhance visualization by setting the viewing angle, ensuring realistic lighting, and improving depth perception.

- gouraud: Is a lighting effect that smoothes surface shading.
- camlight: Makes sure proper illumination/light on the objects.
- view: This is a function that sets a 3D perspective.

```
% Adjust View and Lighting
view(3);
axis equal;
grid on;
xlabel('X-axis');
ylabel('Y-axis');
zlabel('Z-axis');
title(['3D Wall with Vertically Split Phantom (Beam Energy: ', num2str(beamEnergy), ' keV)']);
lighting gouraud;
camlight;
```

9) X-Ray Source

The purpose of the code is to simulate the X-ray source and its interaction with the phantom. The source is found above the phantom which will then be able to send the x-ray beam energy to the phantom. The source is represented as a red marker (plot3) at a specific position. The X-ray cone is visualized as a cylinder with a variable base radius (in gray). The cone base radius depends on the X-ray angle and distances to the phantom and film. Then, plot3 highlights the source position, and surf renders the cone.

```
% X-Ray Source (Adjust for Distance)
xraySourceZ = phantomCenter(3) + phantomToSource;
plot3(phantomCenter(1), phantomCenter(2), xraySourceZ, 'ro', 'MarkerSize', 10, 'MarkerFaceColor', [0.5, 0.5, 0.5]);

% X-Ray Cone (Touches Film as Base)
coneBaseRadius = tan(deg2rad(xrayAngle/2)) * (phantomToSource + phantomToFilm);
[xCone, yCone, zCone] = cylinder([0, coneBaseRadius], 50);
zCone = -zCone * (phantomToSource + phantomToFilm) + xraySourceZ;
xCone = xCone + phantomCenter(1);
yCone = yCone + phantomCenter(2);
surf(xCone, yCone, zCone, 'FaceAlpha', 0.1, 'EdgeColor', 'none', 'FaceColor', [0.5, 0.5, 0.5]);
```

10) generateMammogram Function Call

This part of the code extends into creating the 2D image of the 3D phantom. It creates a 2D X-ray projection (mammogram) from the volumetric data of the phantom and dynamically adjusts based on the phantom properties and parameters provided. It works by the following inputs:

- phantom3D: 3D array representing the volumetric phantom (with attenuation coefficients for breast and lesion tissues).
- beamEnergy: X-ray beam energy in keV.
- phantomCenter: Coordinates of the phantom's center in 3D space.
- phantomToSource: Distance from the X-ray source to the phantom.
- phantomToFilm: Distance from the phantom to the imaging film.
- xrayAngle: Cone angle of the X-ray beam.

```
generateMammogram(phantom3D, beamEnergy, phantomCenter, phantomToSource, phantomToFilm, xrayAngle);
```

11) Function to Get μ Values Based on Beam Energy

Lastly, this is where the code determines tissue attenuation coefficients based on beam energy, affecting signal intensity and contrast. The following function maps specific μ values for breast and lesion tissues based on the user-selected beam energy that is input. The function ensures that attenuation properties dynamically adjust based on beam energy, therefore, enabling accurate simulation of tissue interactions with X-rays.

```
    % Function to Get Intensity values (I = I0 * exp(-attenuation coefficient * thickness)
    function [breastIntensity, lesionIntensity] = getAttenuationCoefficients(beamEnergy)
        if beamEnergy == 15
            breastIntensity = 15 * exp(-0.794 * 40);
            lesionIntensity = 15 * exp(-1.608 * 16);
        elseif beamEnergy == 19
            breastIntensity = 19 * exp(-0.488 * 40);
            lesionIntensity = 19 * exp(-0.920 * 16);
        elseif beamEnergy == 26
            breastIntensity = 26 * exp(-0.303 * 40);
            lesionIntensity = 26 * exp(-0.483 * 16);
        else
            error('Beam energy must be 15, 19, or 26 keV.');
        end
    end
end
```

2D Phantom

1) Parameters Used:

- Beam Energy (15,19,26 keV): Controls the attenuation coefficients (μ) of the breast and lesion tissues.
- X-Ray Cone Angle (40°-60°): Defines the angle of the X-ray cone, and influences the beam spread.
- Phantom-to-Film Distance (0-60 cm): Positions the phantom relative to the film.
- Source-to-Phantom Distance (0-60 cm): Positions the phantom relative to the source.

```
beamEnergy = 15; % Beam energy in keV (15,19,26)
xrayAngle = 40; % X-ray cone angle in degrees (40-60)
phantomToFilm = 60; % Distance of the film from the phantom in cm (0-60)
phantomToSource = 60; % Distance of the x-ray source from the phantom in cm (0-60)
```

2) Phantom and Geometry Initialization

The following code is used to define the geometry of the phantom and the X-ray beam.

- gridSize: The size of the 2D grid.
- phantomCenter: The center of the breast phantom in 2D.
- radiusBreast & radiusLesion: Radius of the outer breast and inner lesion circles.

```
gridSize = 120;
phantomCenter = [60, 70];
radiusBreast = 20;
radiusLesion = 8;
```

3) Beam Geometry and Film Initialization

The purpose of the code is to simulate the x-ray cone and film placement. The cone base width depends on the X-ray angle and the source-to-phantom distance. The film is positioned directly below the phantom.

```
% Beam Geometry (Cone)
coneBaseWidth = 2 * (tan(deg2rad(xrayAngle / 2)) * phantomToSource);
coneBaseX = [xraySource(1) - coneBaseWidth/2, xraySource(1) + coneBaseWidth/2];
coneBaseZ = [phantomCenter(2) - phantomToFilm, phantomCenter(2) - phantomToFilm];

% Film
filmX = [coneBaseX(1), coneBaseX(2)];
filmZ = [coneBaseZ(1), coneBaseZ(1)];
```

4) Visualization

The purpose of this section of the code is to render the x-ray cone, breast phantom, cancer lesion, and film.

- X-Ray Cone: Be able to visualize the X-ray beam cone.
- Breast Phantom: Used to render the outer breast phantom as a vertically split circle.
- Cancer Lesion: Used to render the lesion (inner circle) on the transparent half of the phantom.
- Film: Represent the imaging film as a horizontal line below the phantom.

```
% X-Ray Cone
fill([xraySource(1), coneBaseX(1), coneBaseX(2)], ...
     [xraySource(2), coneBaseZ(1), coneBaseZ(2)], ...
     [0.8, 0.8, 0.8], 'FaceAlpha', 0.2, 'EdgeColor', 'none');
```

```
% Breast Circle (Vertically Split)
theta = linspace(0, 2*pi, 100);
xBreast = phantomCenter(1) + radiusBreast * cos(theta);
zBreast = phantomCenter(2) + radiusBreast * sin(theta);

% Left Half: Solid Outer Sphere
fill(xBreast(xBreast <= phantomCenter(1)), zBreast(xBreast <= phantomCenter(1)), ...
     [muBreast, muBreast, muBreast], 'EdgeColor', 'none');

% Right Half: Transparent Outer Sphere with Lesion
fill(xBreast(xBreast > phantomCenter(1)), zBreast(xBreast > phantomCenter(1)), ...
     [muBreast, muBreast, muBreast], 'FaceAlpha', 0.2, 'EdgeColor', 'none');
```

```
% Lesion Circle: Inner Sphere on Right Side Only
theta = linspace(0, 2*pi, 100);
xLesion = phantomCenter(1) + radiusLesion * cos(theta);
zLesion = phantomCenter(2) + radiusLesion * sin(theta);
fill(xLesion(xLesion > phantomCenter(1)), zLesion(xLesion > phantomCenter(1)), ...
     [muLesion, muLesion, muLesion], 'EdgeColor', 'none');
```

```
% Film
plot(filmX, filmZ, 'k-', 'LineWidth', 3);
```

5) Adjusting Visualization

The code will configure axis limits and labels for a clear 2D visualization.

```
% Adjust Visualization
axis equal;
xlim([0, gridSize]);
ylim([0, gridSize + 20]);
axis on;
title(['2D Phantom with Vertically Split Sphere (Beam Energy: ', num2str(beamEnergy), ' keV)']);
```

6) Function to Get μ Values Based on Beam Energy

The following code will calculate attenuation coefficients based on beam energy.

```
% Get μ Values Based on Beam Energy
[muBreast, muLesion] = getAttenuationCoefficients(beamEnergy);
```

```
    % Function to Get μ Values Based on Beam Energy
    function [muBreast, muLesion] = getAttenuationCoefficients(beamEnergy)
        if beamEnergy == 20
            muBreast = 0.6;
            muLesion = 0.8;
        elseif beamEnergy == 30
            muBreast = 0.4;
            muLesion = 0.6;
        elseif beamEnergy == 40
            muBreast = 0.2;
            muLesion = 0.4;
        else
            error('Beam energy must be 20, 30, or 40 keV.');
        end
    end
end
```

7) X-Ray Source Position

The X-ray source is simulated as a point above the phantom, from which the cone-shaped beam originates. Its position is determined based on the phantom's center and the distance to the source. The X-coordinate (60) aligns the source with the center of the phantom. The Z-coordinate is calculated by adding phantomToSource to the phantom's center position along the Z-axis, placing the source above the phantom.

```
% X-Ray Source Position
xraySource = [60, phantomCenter(2) + phantomToSource];
```

8) X-Ray Source

The purpose is to display the X-ray source to provide visual context for the phantom and film.

```
% X-Ray Source
plot(xraySource(1), xraySource(2), 'ko', 'MarkerSize', 8, 'MarkerFaceColor', [0.5, 0.5, 0.5]);
```

2D Image from 3D Phantom

1) generateMammogram Function

The function simulates a 2D X-ray image (mammogram) by projecting the 3D phantom onto a 2D plane. This process mimics the behavior of a real mammography system, where X-rays pass through the breast to create an image on the film. It is important because it includes:

- Projection of the 3D Phantom: Converts the volumetric phantom into a 2D intensity map based on attenuation.
- Simulation of the X-Ray Cone: Ensures only pixels within the X-ray cone contribute to the mammogram.
- Visualization: Displays the resulting 2D mammogram with intensity values that simulate real X-ray imaging.

```
function generateMammogram(phantom3D, beamEnergy, phantomCenter, phantomToSource, phantomToFilm, xrayAngle)
```

2) Phantom Dimensions

The following code retrieves the dimensions of the 3D phantom, which determine the size of the film and the projection.

```
% Get dimensions of the phantom
[height, width, depth] = size(phantom3D);
```

3) X-Ray Source and Cone Geometry

For the following section, each is used to:

- X-Ray Source Position: Positioned above the phantom at a distance of phantomToSource.
- Cone Radius: This calculates the radius of the X-ray cone at the phantom, defining the imaging system's field of view.

```
% Define the X-ray source position
xraySource = [phantomCenter(1), phantomCenter(2), phantomCenter(3) + phantomToSource];

% Calculate the cone's base radius at the phantom
coneRadiusAtPhantom = tan(deg2rad(xrayAngle / 2)) * phantomToSource;
```

4) Film Initialization

The following code creates a blank 2D projection (film) to store the accumulated attenuation values for each pixel.

```
% Initialize the film (2D projection)
filmProjection = zeros(height, width);
```

5) Pixel-by-Pixel Evaluation

The following section involves 4 important functions:

- Pixel Position: Calculates the 3D coordinates of each pixel on the film.
- Vector to Pixel: Determines whether the pixel lies within the X-ray cone by projecting its position onto the X-Y plane and comparing it to the cone radius.
- Z-Axis Integration: This function accumulates attenuation values along the Z-axis for pixels within the cone. It models the cumulative effect of X-ray absorption as the rays pass through the phantom.
- Outside the Cone: Pixels outside the X-ray cone are assigned NaN to represent no signal.

```
% Loop over each pixel on the film and check if it's within the cone's scope
for i = 1:height
    for j = 1:width
        % Calculate the position of the current pixel on the film
        pixelPosition = [i, j, phantomCenter(3) - phantomToFilm];

        % Calculate the vector from the X-ray source to the pixel
        vectorToPixel = pixelPosition - xraySource;

        % Project the vector onto the X-Y plane
        distanceToCenter = sqrt(vectorToPixel(1)^2 + vectorToPixel(2)^2);

        % Check if the pixel is within the X-ray cone
        if distanceToCenter <= coneRadiusAtPhantom
            % Integrate attenuation along the Z-axis through the phantom
            for z = 1:depth
                filmProjection(i, j) = filmProjection(i, j) + phantom3D(i, j, z);
            end
        else
            % Outside the phantom, set the projection value to 0
            filmProjection(i, j) = NaN; % Represent as no signal
        end
    end
end
```

6) Signal Intensity Conversion

This section converts accumulated attenuation to simulated X-ray intensity using $I = e^{-\mu x}$. The Pixels outside the cone are set to a neutral value (0.5) to represent the background signal.

```
% Convert attenuation to intensity
filmImage = exp(filmProjection); % Simulated X-ray intensity
filmImage(isnan(filmImage)) = 0.5; % Set the background signal to 0
```

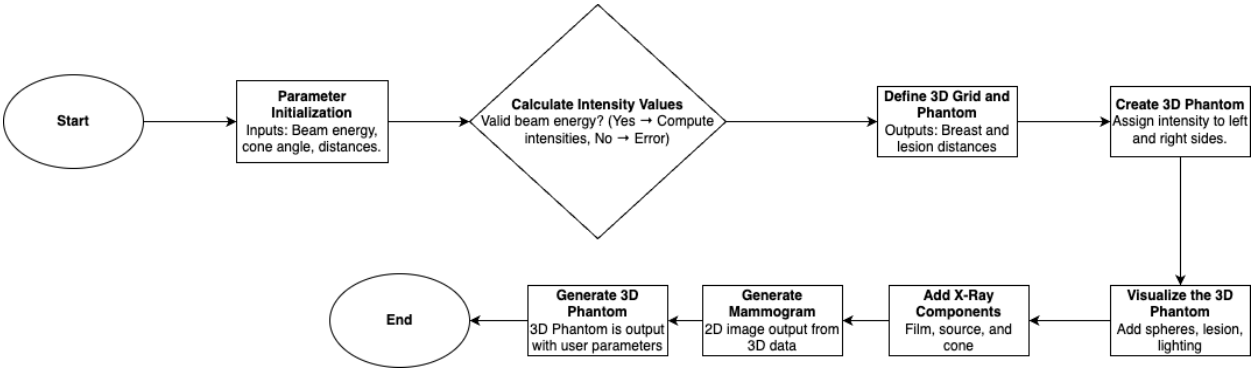
7) Visualization

The final section of the code displays the final 2D mammogram image using a grayscale colormap. The intensity values visually represent the X-ray transmission through the phantom.

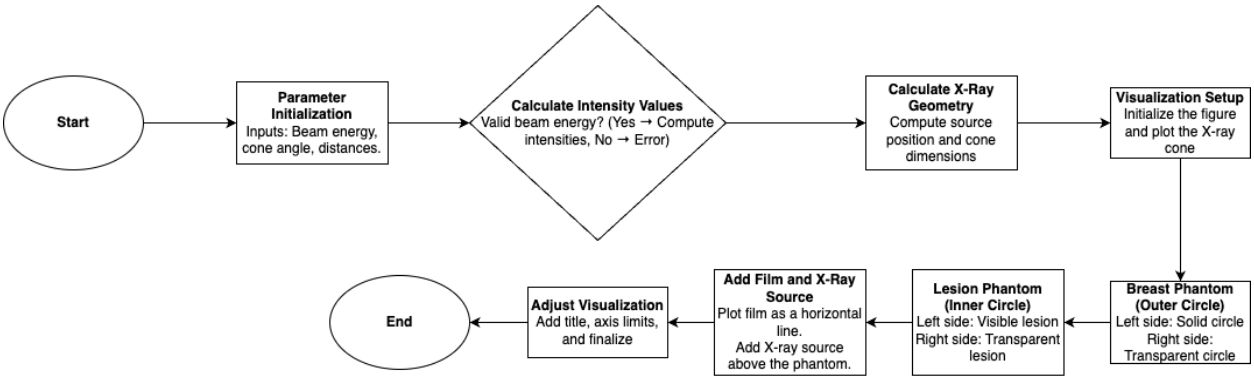
```
    % Display
    figure;
    imagesc(filmImage);
    colormap(gray);
    colorbar;
    title(['Simulated Mammogram within X-Ray Cone (Beam Energy: ', num2str(beamEnergy), ' keV)']);
    xlabel('X-axis (pixels)');
    ylabel('Y-axis (pixels)');
    axis equal;
    axis tight;
end
```

Flowchart

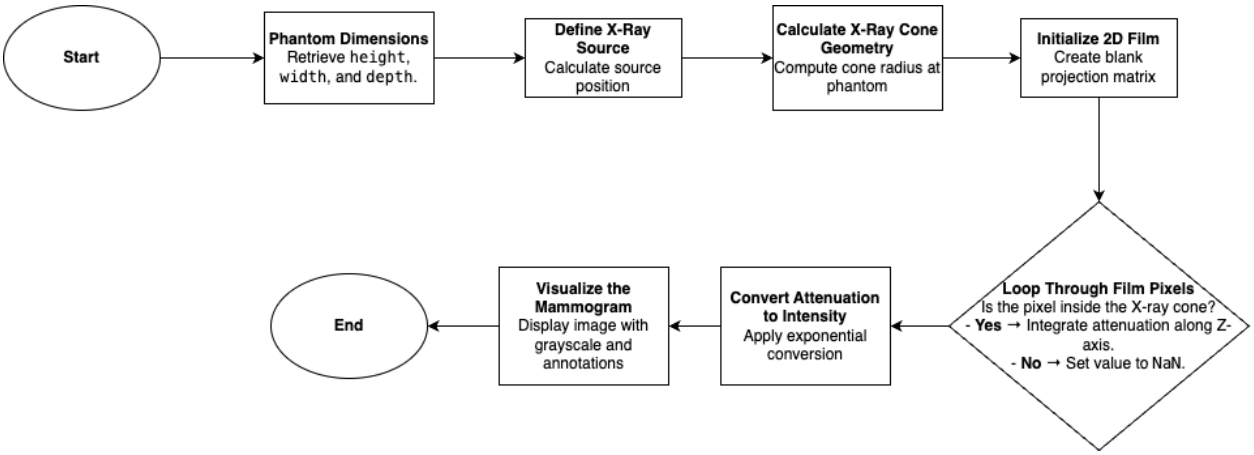
3D Phantom



2D Phantom



2D Image from 3D Phantom

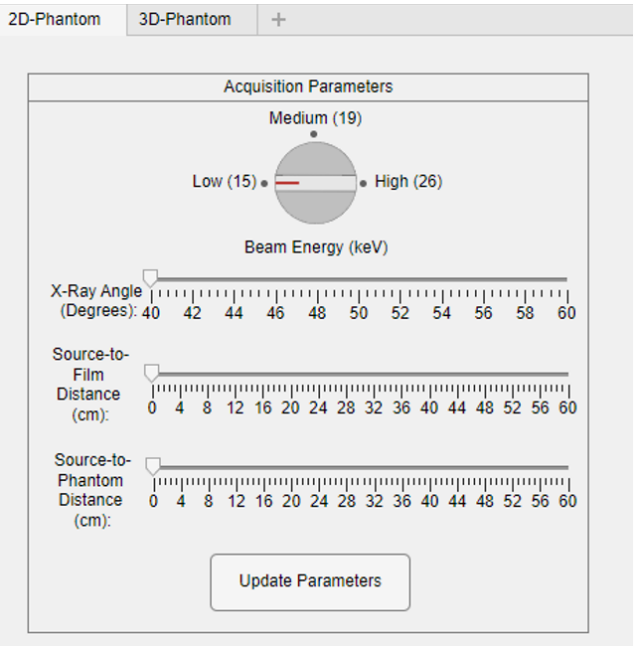


GUI

The creation of the GUI involved using Appdesigner in MATLAB. It is an interactive development environment for designing an app layout and programming its behavior. The GUI involved using the 4 parameters asked in the project PDF, being:

- Beam Energy
- X-Ray Angle
- Source-to-Film Distance
- Source-to-Phantom Distance

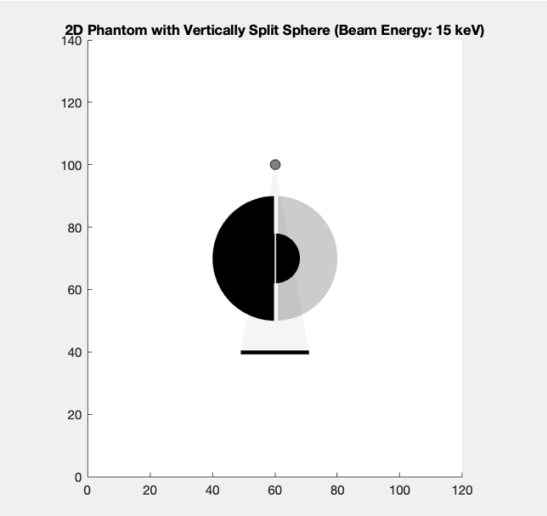
This was created to integrate both the 2D and 3D phantoms. Having components like Knobs, Sliders, and Buttons helped develop the GUI for the user.

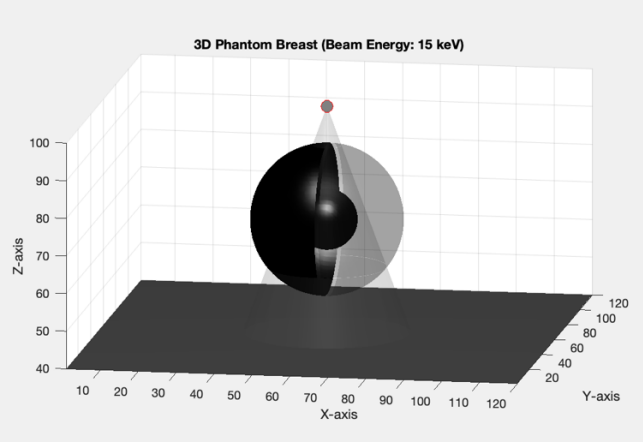


Results

This section presents the results of the Mammography simulation, including visual outputs, quantitative evaluations, and answers to the project-specific questions. The performance of the 2D and 3D phantoms under different parameter configurations is analyzed, and errors are documented. Additionally, key metrics such as contrast and signal intensity are quantified.

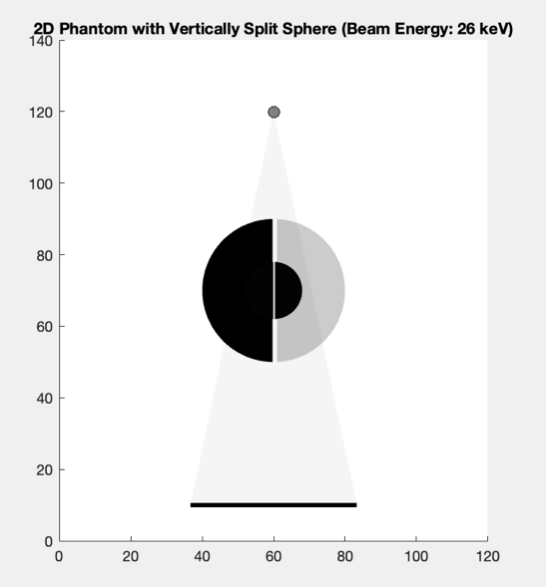
Visual Outputs

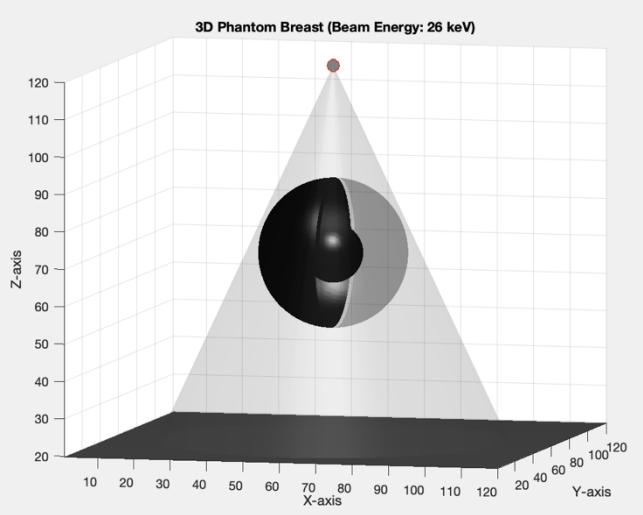




The following 2 images show the 2D and 3D outputs of the program. The parameters for both:
Beam Energy: 15 keV (low)
X-Ray Angle: 40 degrees
Source-to-Film Distance: 30 cm
Source-to-Phantom Distance: 30 cm

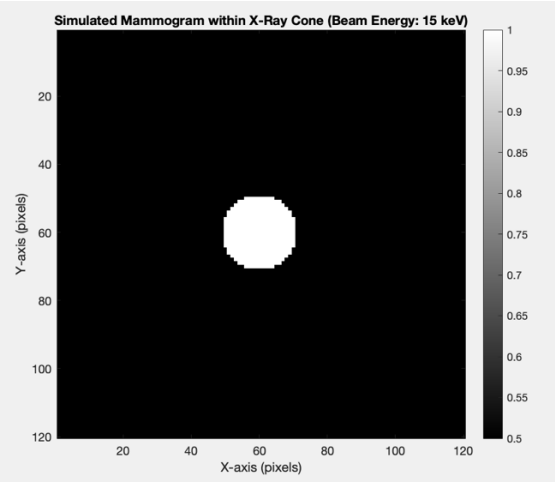
The outputs for the 2D and 3D phantoms at 15 keV showcase how low-energy X-rays impact Mammographic Imaging. In the 2D phantom, the solid left side demonstrates how breast tissue heavily attenuates X-rays, making it appear opaque, while the transparent right side highlights the cancerous lesion. The enhanced contrast between the lesion and surrounding tissue is a direct result of low beam energy, which increases attenuation differences and makes the lesion more visible. The 3D phantom provides similar results, with the solid left side blocking the visibility of the lesion and the transparent right side revealing it. These observations align with the theory that low-energy X-rays enhance contrast by exploiting differences in tissue attenuation.

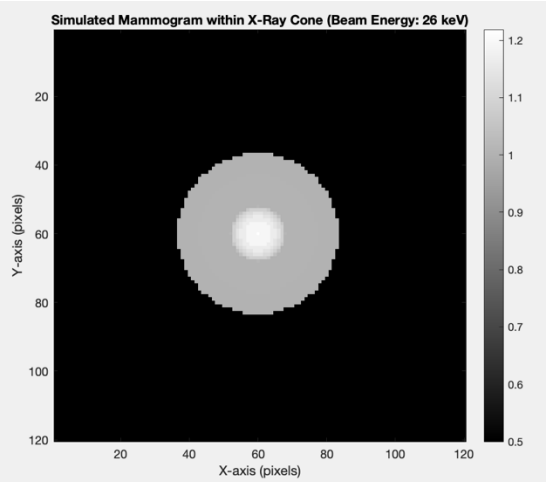




The following 2 images show the 2D and 3D outputs of the program. The parameters for both:
Beam Energy: 26 keV (high)
X-Ray Angle: 50 degrees
Source-to-Film Distance: 60 cm
Source-to-Phantom Distance: 50 cm

With the newly adjusted parameters, there is a visual difference in the output compared to the previous two images. The same statements from the previous two images can also be said for the new pair, but there is also the adjusted distance between the film, phantoms, and source. As each are increased, the amount of area that is covered also increases. The adjusted X-ray angle seems to cover more area of the phantom. If the user would like, they can adjust the angle so that it mainly focuses on the cancerous lesion.





The two images represent simulated mammograms from the two 3D phantom that were previously created using different beam energy levels. In the first image, where the beam energy is set to 15 keV, the cancerous lesion appears prominently as a bright white circular region against a black background, while the outer tissue is not visible, highlighting the lesion's contrast relative to the surroundings. This high contrast is a result of the low beam energy, which enhances attenuation differences between the lesion and surrounding tissues. The second image generated with a higher beam energy of 26 keV, revealing both the lesion and the surrounding tissue. The lesion remains visible but is less distinct compared to the first image, as the higher beam energy reduces the contrast. Additionally, the outer breast tissue appears in a gradient of gray, indicating the broader attenuation range captured at this energy level. These outputs demonstrate how increasing beam energy decreases the contrast of the lesion while providing more detail about the surrounding tissue, reflecting the trade-off between contrast and visibility in X-ray imaging.

Project Questions

1) What is the contrast of the cancerous lesion relative to the rest of the Tissue?

The contrast between the cancerous lesion and the surrounding tissue is determined by their respective attenuation coefficients. At lower beam energies (e.g., 20 keV), the attenuation difference is more pronounced, leading to higher contrast. The lesion appears more distinct because X-rays are attenuated more effectively in the lesion than in the surrounding breast tissue. Conversely, at higher beam energies (e.g., 40 keV), the attenuation differences are reduced, resulting in lower contrast and making the lesion less distinguishable.

2) Can you see the cancer clearly?

The lesion can be seen clearly under optimized conditions. Specifically:

- At low beam energy (e.g., 20 keV), the lesion is visible due to enhanced contrast between the lesion and the surrounding tissue.
- The transparency on the right side of the phantom in both 2D and 3D images allows for a clear view of the lesion.
- However, at higher beam energies, the lesion’s visibility decreases as tissue contrast diminishes.

3) Now compress the breast by reducing the width of your phantom. What is the effect of this? Quantify the effect you may have

- **Effect**: Compressing the breast reduces its thickness, leading to:
    - **Improved lesion detectability**: Compression reduces tissue overlap, decreasing the mammogram's complexity and enhancing the lesion's clarity.
    - **Increased contrast-to-noise ratio (CNR)**: By reducing scatter radiation, compression enhances the differentiation between the lesion and the surrounding tissue.
    - **Reduced attenuation**: Thinner tissues absorb fewer X-rays, resulting in a more uniform image with clearer lesion boundaries.

- **Quantification**: The simulation allows qualitative observation of these effects but does not provide metrics such as:
    - Signal-to-noise ratio (SNR).
    - CNR improvements before and after compression.
    - Reduction in total attenuation values or specific intensity changes.

Without these metrics, the simulation lacks scientific rigor and fails to substantiate the clinical benefit of compression, making the findings incomplete.

Errors

- Visualization Inconsistencies
    - **Issue**: The split visualization of the phantom may lead to misinterpretation if the rendered lesion is partially obscured.
    - **Cause**: Imperfect alignment or rendering of solid and transparent halves.
    - **Impact**: Misrepresentation of lesion detectability in the simulation.

- Parameter Sensitivity
    - **Issue**: Extreme or invalid parameter values (e.g., unrealistic beam energies or distances) may produce erroneous results.
    - **Cause**: Insufficient input validation in the MATLAB code or GUI.
    - **Impact**: Non-physiological outputs or crashes during simulation.

- Attenuation Coefficient Limitations
    - **Issue**: Simplified attenuation coefficients may not fully represent the heterogeneity of real breast tissue.
    - **Cause**: Static mapping of coefficients to specific beam energies without accounting for tissue variations.
    - **Impact**: Reduced simulation realism.

- GUI Usability
    - **Issue**: The GUI lacks error handling for invalid inputs.
    - **Cause**: Limited testing of edge cases in parameter inputs.
    - **Impact**: Users may encounter unexpected behavior or crashes.

- Compression Simulation Challenges
    - **Issue**: Compression effects may not fully account for tissue scattering and edge effects.
    - **Cause**: Simplified geometry and material assumptions in the phantom model.
    - **Impact**: Inaccurate representation of the benefits of breast compression.

Resources

The following project was created with the help of various resources. Some provided help with the code used in the making of the phantoms while others provided in-depth information about questions being asked in the PDF and Mammography overall:

- https://radiologykey.com/breast-imaging-mammography/
- https://radiologykey.com/2-mammography/
- https://radiologykey.com/x-ray-imaging-mammography/
- https://radiologykey.com/mammography-equipment-and-basic-physics/
- https://www.mathworks.com/matlabcentral/fileexchange/30207-cone-beam-ct-simulation
- https://tomroelandts.com/articles/astra-toolbox-tutorial-reconstruction-from-projection-images-part-1
- https://blogs.mathworks.com/graphics/2015/08/06/transparency-in-3d/
- https://github.com/MichaelBehr/Xray_Sim
- https://www.emergenresearch.com/blog/top-10-leading-companies-in-the-breast-imaging-market-in-2023#:~:text=Canon%20Medical%20Systems%20Corporation%20is%20a%20Japan,diagnoses%2C%20improve%20treatment%20and%20enhance%20patient%20care
- https://www.mathworks.com/help/matlab/visualize/displaying-complex-three-dimensional-objects.html?
- https://web.mat.upc.edu/toni.susin/files/3dplot.pdf?
- https://www.mathworks.com/help/matlab/ref/ndgrid.html
- https://www.researchgate.net/publication/45629831_Measurement_of_the_linear_attenuation_coefficients_of_breast_tissues_by_synchrotron_radiation_computed_tomography