Introduction:

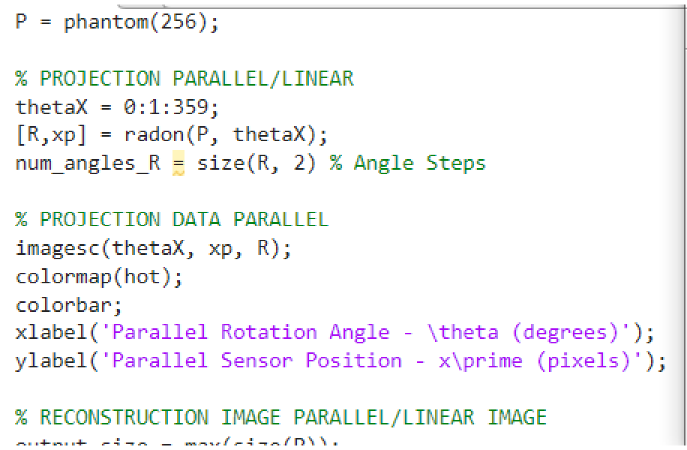
In our work, we present a simulation of a CT Scanner, built in with a simple functioning Graphical User Interface using MATLAB and the add-on, App Developer. We use two test phantoms with simple shapes along with MATLAB's base phantom(), which produces the Shepp-Logan Phantom for our performance test. MATLAB had many incredible tools we were able to utilize to easily showcase what we wanted, with minimal effort on our part.

Objectives:

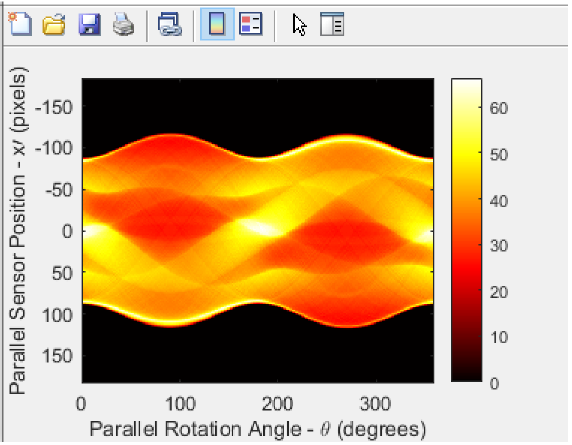
- **Phantom Generation:** Two simple test Phantoms for proof of concept with semi-customizable parts. Shepp-Logan phantom is included.
- **Graphical User Interface:** House all relevant graphing and images of our functions, allowing simple modularity of Phantoms and editing of all variables of data.
- **Signal Intensity:** On any degree of scan, shows the frequency of particular colored pixel values.
- **Image Reconstruction:** Using the radon and plot data, generate a new image entirely from function data.
- **Modularity:** Able to change the characteristics of our functions, such as, Distance, Angle-Step Rotation, etc.

Primary Functions

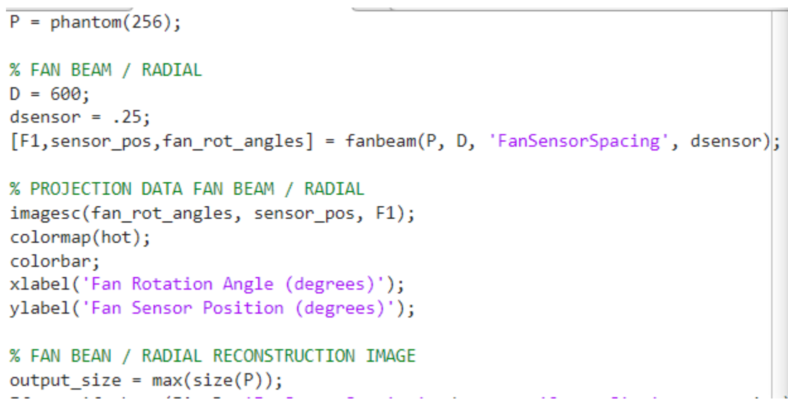
Projection Data (Parallel & Radial):



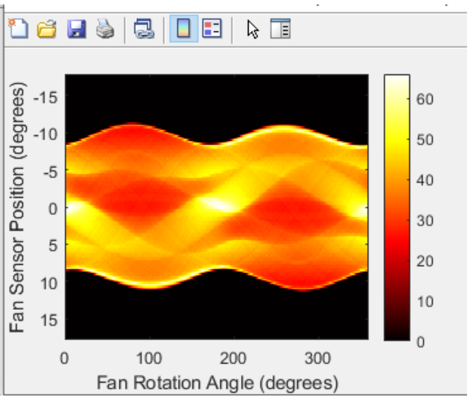
```
P = phantom(256);

% PROJECTION PARALLEL/LINEAR
thetaX = 0:1:359;
[R,xp] = radon(P, thetaX);
num_angles_R = size(R, 2) % Angle Steps

% PROJECTION DATA PARALLEL
imagesc(thetaX, xp, R);
colormap(hot);
colorbar;
xlabel('Parallel Rotation Angle - \theta (degrees)');
ylabel('Parallel Sensor Position - x\prime (pixels)');

% RECONSTRUCTION IMAGE PARALLEL/LINEAR IMAGE
output_size = max(size(R));
```



With Phantom() as the base, we first define Theta angles.
Using **X:Y:Z**, we can define, **X, beginning of the <u>vector</u>, <u>Y, step amount, and Z, end of the vector.</u>** This means we can establish **<u>Rotation Step Size</u>**, or the step angle in which happens between each "scan". And with Matlab's **radon()** function, we can generate plot data for a projection in which the function **imagesc()**, allows us to visualize Rotation Angle and Parallel Sensors.

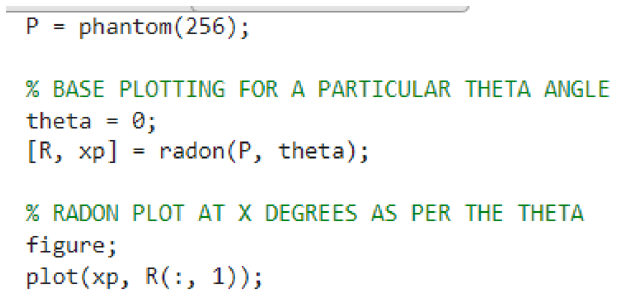


```
P = phantom(256);

% FAN BEAM / RADIAL
D = 600;
dsensor = .25;
[F1,sensor_pos,fan_rot_angles] = fanbeam(P, D, 'FanSensorSpacing', dsensor);

% PROJECTION DATA FAN BEAM / RADIAL
imagesc(fan_rot_angles, sensor_pos, F1);
colormap(hot);
colorbar;
xlabel('Fan Rotation Angle (degrees)');
ylabel('Fan Sensor Position (degrees)');

% FAN BEAM / RADIAL RECONSTRUCTION IMAGE
output_size = max(size(P));
```

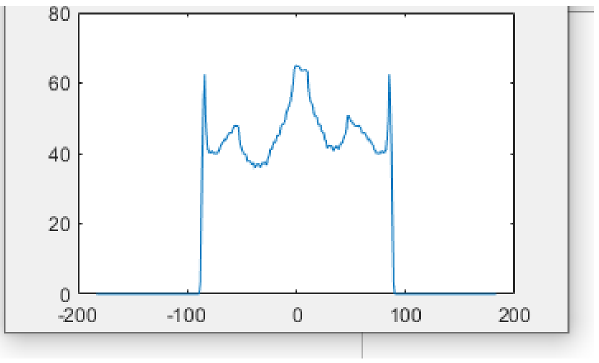


For Radial version, we first establish **<u>Detectors and Sensor Distance</u>**, which are noted by **D and dsensor** respectively. Matlab's **fanbeam()** function, imagesc() will give us similar projections.

Projection Plotting:

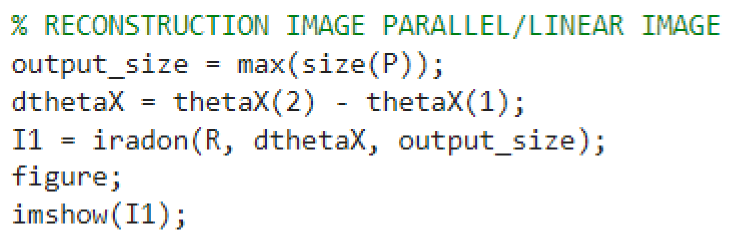


```
P = phantom(256);

% BASE PLOTTING FOR A PARTICULAR THETA ANGLE
theta = 0;
[R, xp] = radon(P, theta);

% RADON PLOT AT X DEGREES AS PER THE THETA
figure;
plot(xp, R(:, 1));
```

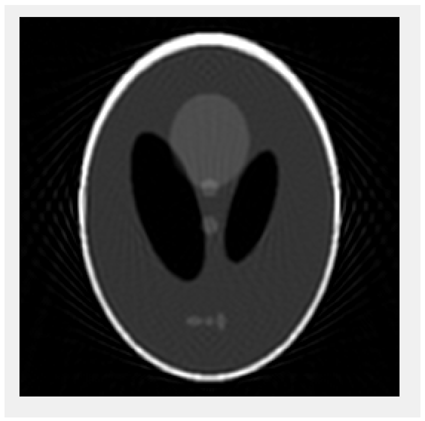


Using the previous radon() function and establishing **<u>Degrees</u>**, marked as **theta**, we can modulate degrees to any number, which will show us plot intensity values of pixel positions.

Parallel Reconstruction (Parallel/Linear):

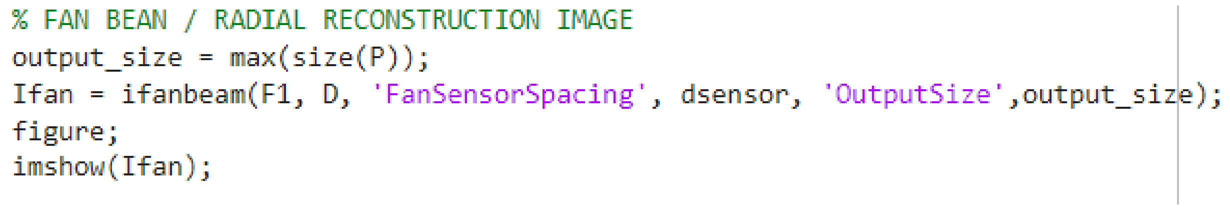


```
% RECONSTRUCTION IMAGE PARALLEL/LINEAR IMAGE
output_size = max(size(P));
dthetaX = thetaX(2) - thetaX(1);
I1 = iradon(R, dthetaX, output_size);
figure;
imshow(I1);
```



Matlab's **iradon()** function allows us to easily reconstruct our Shepp-Logan phantom() using our desired **vector** of theta degrees. Which will be our Parallel Reconstruction exclusive variable.

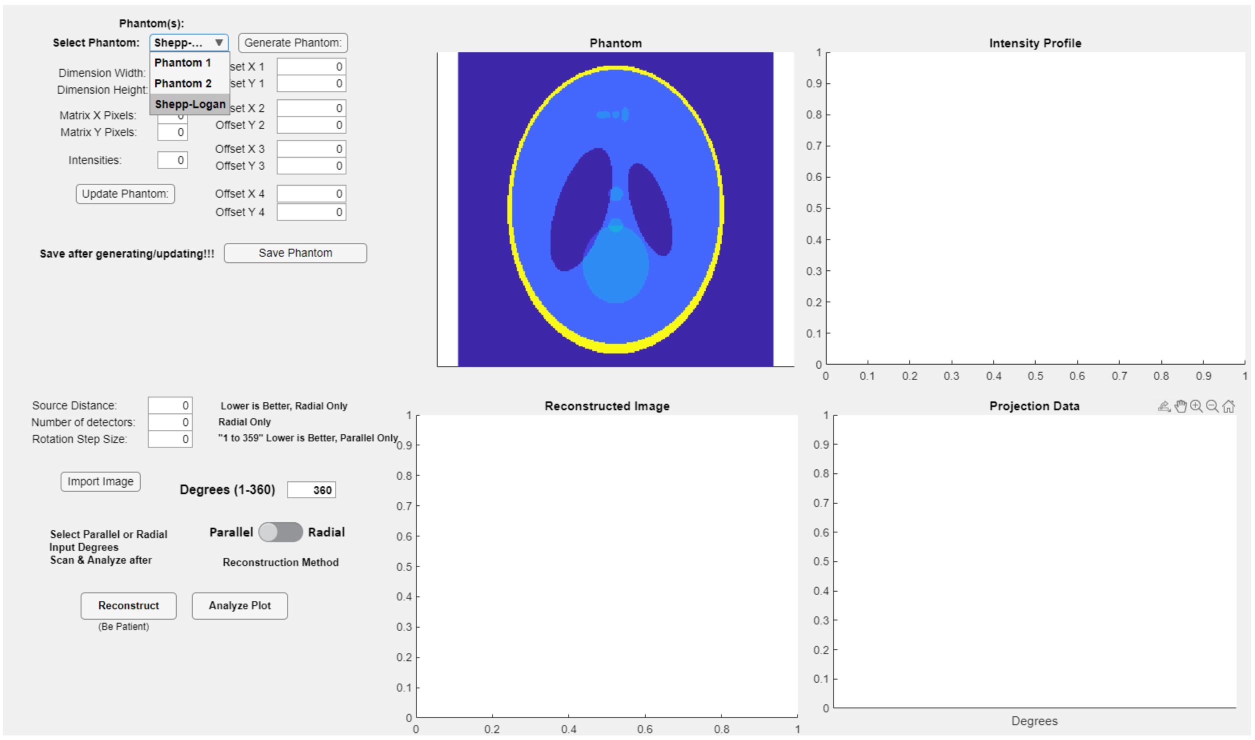
Radial Reconstruction (Radial/Arc):



```
% FAN BEAN / RADIAL RECONSTRUCTION IMAGE
output_size = max(size(P));
Ifan = ifanbeam(F1, D, 'FanSensorSpacing', dsensor, 'OutputSize',output_size);
figure;
imshow(Ifan);
```

Similarly enough, **ifanbeam()**, can be used, in combination with detectors and distance as exclusive variables for Radial Reconstruction.

Graphical User Interface



Dimension Width:
Dimension Height:
Matrix X Pixels: 0
Matrix Y Pixels: 0
Intensities: 0

set X 1: 0
set Y 1: 0
set X 2: 0
Offset Y 2: 0
Offset X 3: 0
Offset Y 3: 0
Offset X 4: 0
Offset Y 4: 0

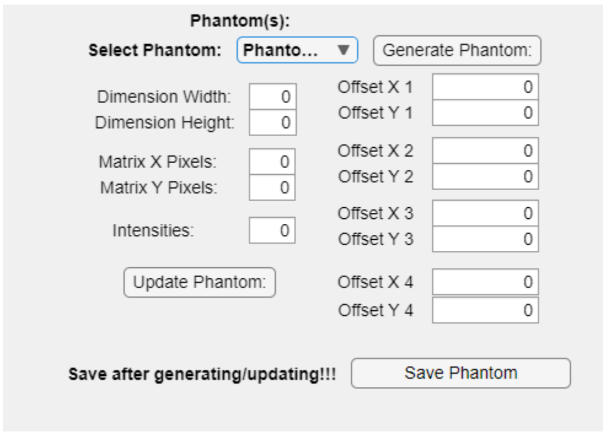
Save after generating/updating!!!

Source Distance: 0 — Lower is Better, Radial Only
Number of detectors: 0 — Radial Only
Rotation Step Size: 0 — "1 to 359" Lower is Better, Parallel Only

Degrees (1-360): 360

Select Parallel or Radial
Input Degrees
Scan & Analyze after

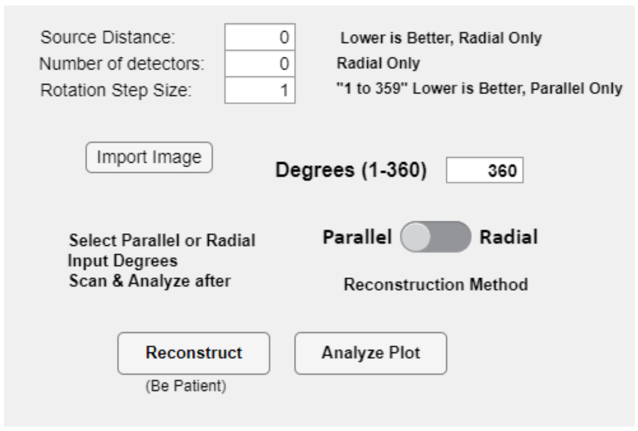
Our GUI consists of 4 image displays, and modular numerical values on the left



Dimension Width: 0
Dimension Height: 0
Matrix X Pixels: 0
Matrix Y Pixels: 0
Intensities: 0

Offset X 1: 0
Offset Y 1: 0
Offset X 2: 0
Offset Y 2: 0
Offset X 3: 0
Offset Y 3: 0
Offset X 4: 0
Offset Y 4: 0

Save after generating/updating!!!



Source Distance: 0 — Lower is Better, Radial Only
Number of detectors: 0 — Radial Only
Rotation Step Size: 1 — "1 to 359" Lower is Better, Parallel Only

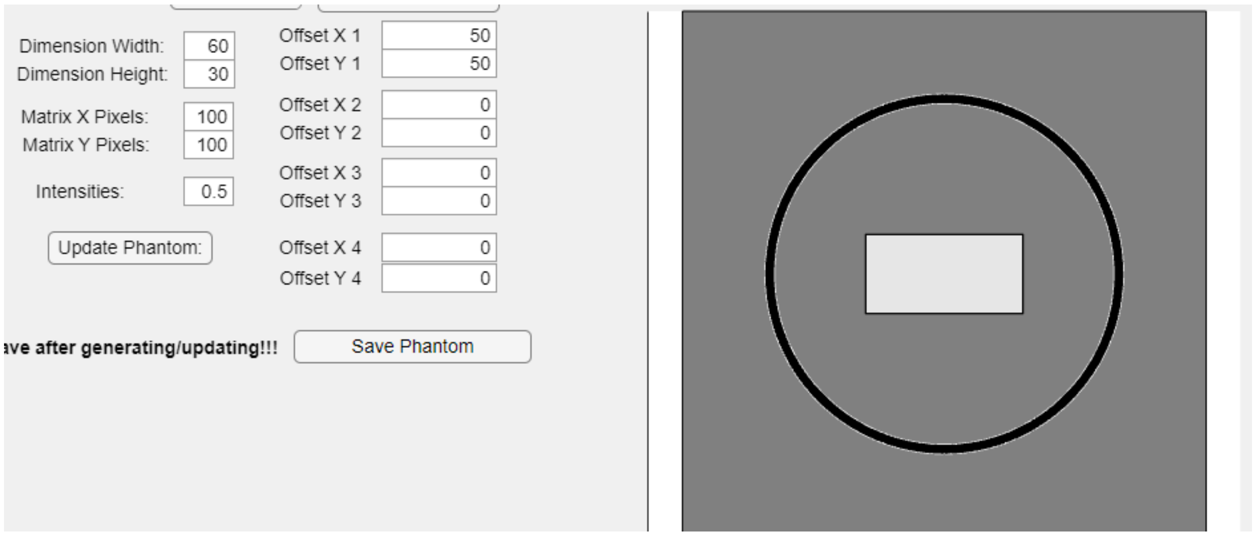
Degrees (1-360): 360

Select Parallel or Radial
Input Degrees
Scan & Analyze after

On the left, we have shown our **Matrix & Phantom Manipulation**, in which on one of the Phantoms, we can **control the positions of the shapes within**, as well as the **dimensions** of the matrix itself, and the **size of said phantom**. Shepp-Logan Phantom is also included as an option in our dropdown menu.
The image to the right shows that we can change **Degrees, Distance, Detectors, Step-Angles**, and whether we want to reconstruct using **Parallel or Radial**, which is **Linear and Arc**, respectively.

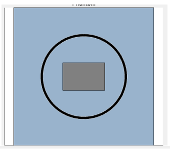
Results & Reconstructions

Now that we have our behind the scenes covered, let's look at some of the results of our findings! Let's start with **Phantom 2**, our phantom designed with a simple rectangle in the center. And to kill two birds with one stone, we will modify the phantom's dimensions to show good contrast and a clear rectangle, showing there are indeed functional changes.

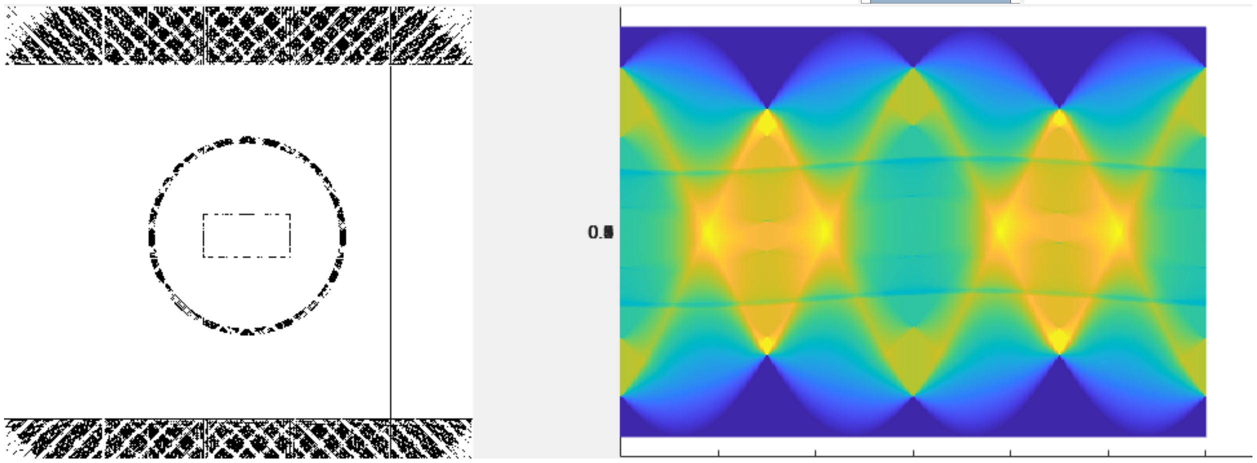


Dimension Width:	60	Offset X 1	50
Dimension Height:	30	Offset Y 1	50
Matrix X Pixels:	100	Offset X 2	0
Matrix Y Pixels:	100	Offset Y 2	0
Intensities:	0.5	Offset X 3	0
		Offset Y 3	0
		Offset X 4	0
		Offset Y 4	0

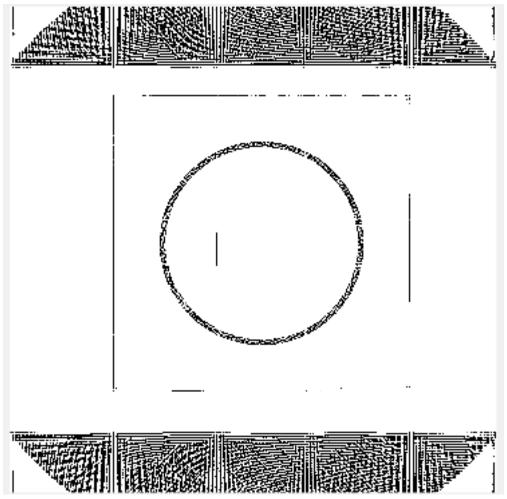
Here is our Phantom with their numerical values, the greyscale image is our new one, while our old one is the small blue background image.



Now what does our Parallel Reconstruction look like?

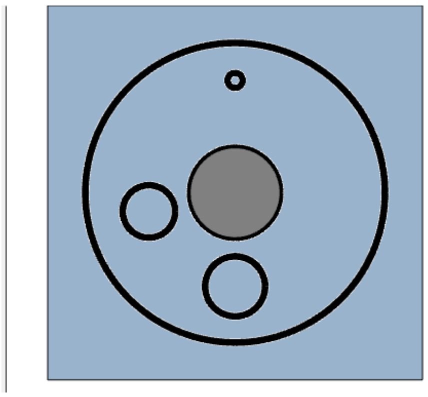
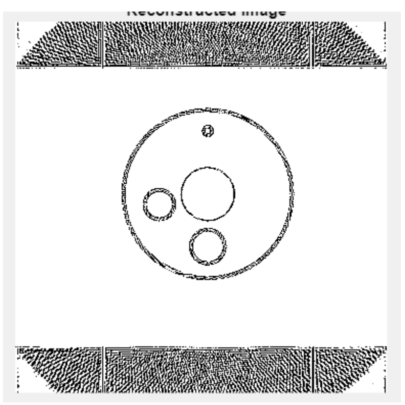
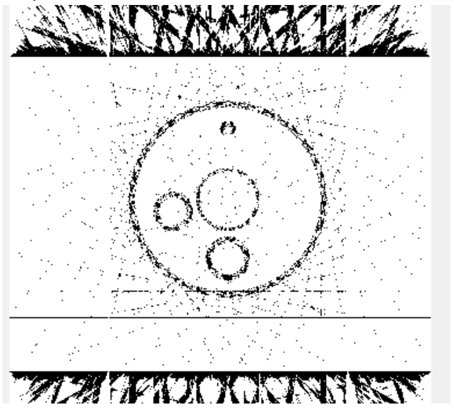


It looks rather clean! Accompanied by its projection data heat map. We used a full 360 degree with 1 angel step rotation for full coverage.



Although we are having a little bit of trouble with our next figure of Radial Reconstruction.
So let’s try **decreasing Distance = .25 to .1** to try to get a better image.

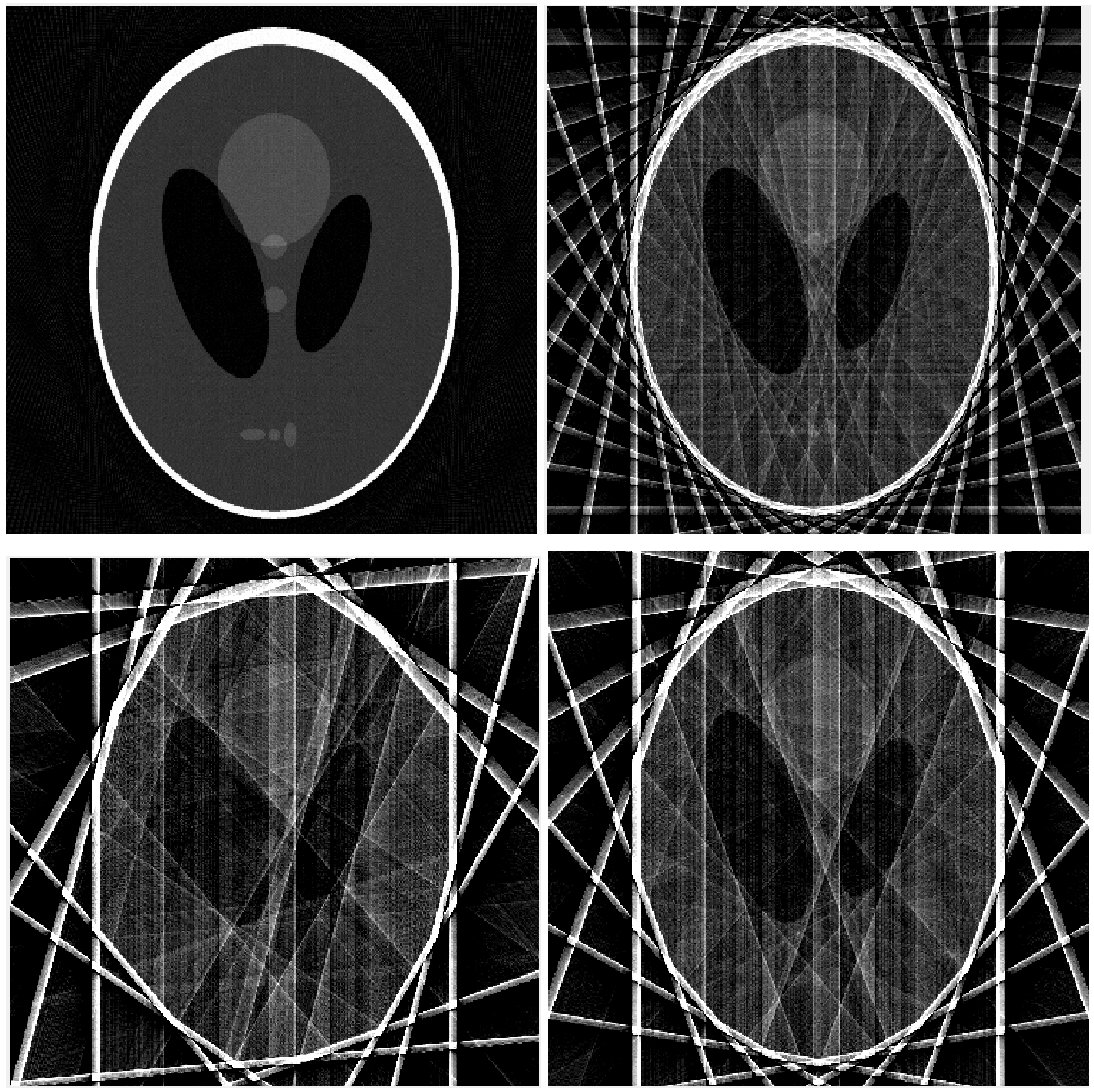
It looks better, although it seems Radial Recon does not work well with rectangular phantoms.
With the same settings with **Distance = .25**, on the circular **Phantom 1**, will it be better?

Yes! Much better than in fact, we can conclude **Radial Recon is not great for right angular**, and better for **abstract spherical shapes. (Middle Figure)**

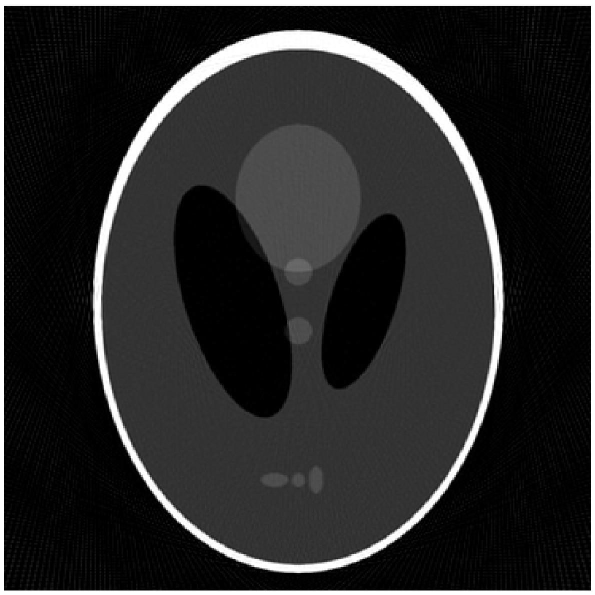
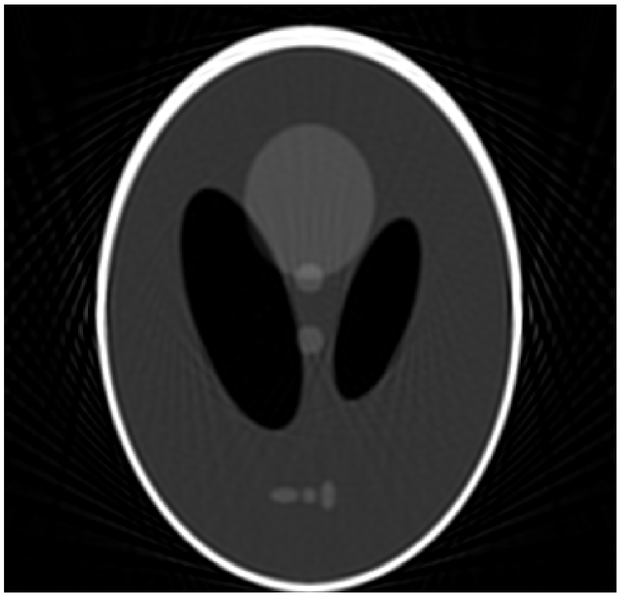
Conversely, **Parallel Recon is weaker at circular objects (Rightmost Figure)**, for dramatic effect and clear visual, we set the **Angle Step to 10**, up to 360. Although not the best image, it shows a clear display of the step angle and how it can be difficult to reconstruct such an image.

Now, let’s test step-angles on Shepp-Logan.

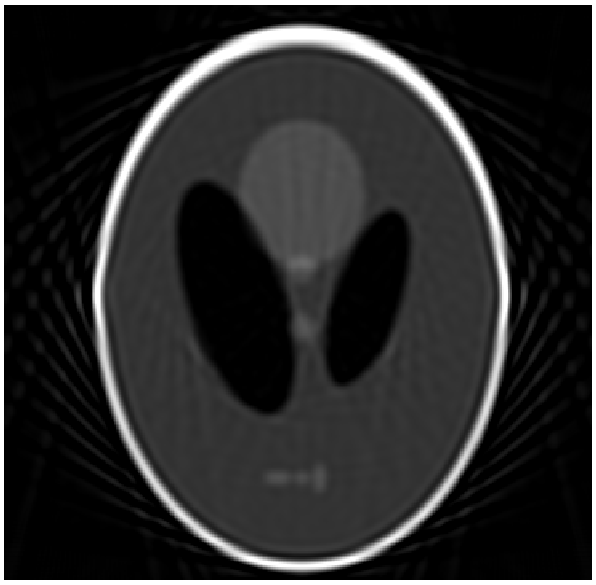
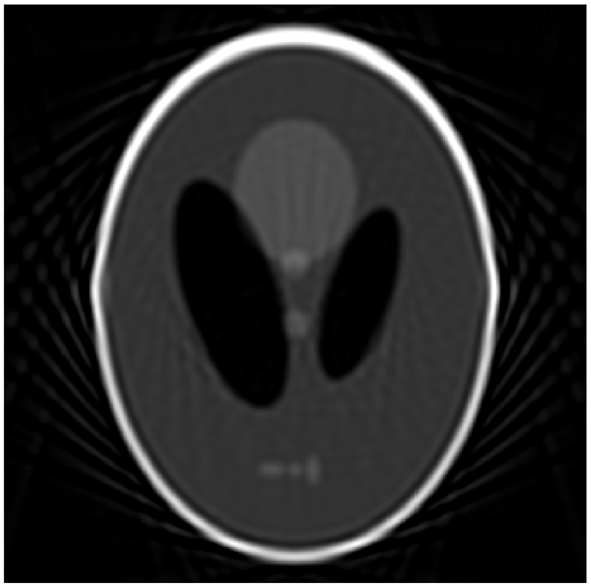


Indeed, you can see a difference when **Step-Angle** is from **1,10, 50, 20**.
(Top Left to Right, Bottom Left to Right)
The image shows **artifacts of the parallel lines** intersecting at such given degrees up to 360. In Parallel Recon, they essentially draw each line, making a complete image.

And to conclude every functional detail, we must see Radial Recon, starting with a .25 distance and 800 D value.

Notice a difference? The distance is .6, it is a little blurrier. (Top Right)
It becomes very apparent at a distance of 1. (Bottom Left)

Similarly, if our D value is too high, it may end up causing similar deformities. Using our base .25 distance, but having a D value of 3000 instead of normal 800. (Bottom Right)

Addendum

That concludes our findings and implementations. For a better understanding of the operation,, please read the “readme” made by Justin Lam which is included.
This Report is made by Darion Le and in partnership with Justin Lam. Due to broken mic and other issues, unfortunately, the presentation was still able to be solely voiced by Justin Lam. We hope the additional details in this report will fill any shortcomings of the showcase and represent more intricacies and details of our project.

~Darion Le, responsible for SI Plots, Image Reconstruction functions and numerical modularity
~Justin Lam, responsible for the base GUI,, bug-fixing, image optimization and matrix modularity

~Ammar… spent minimal time on this, with insignificant results (Graders, this is up to your discretion)