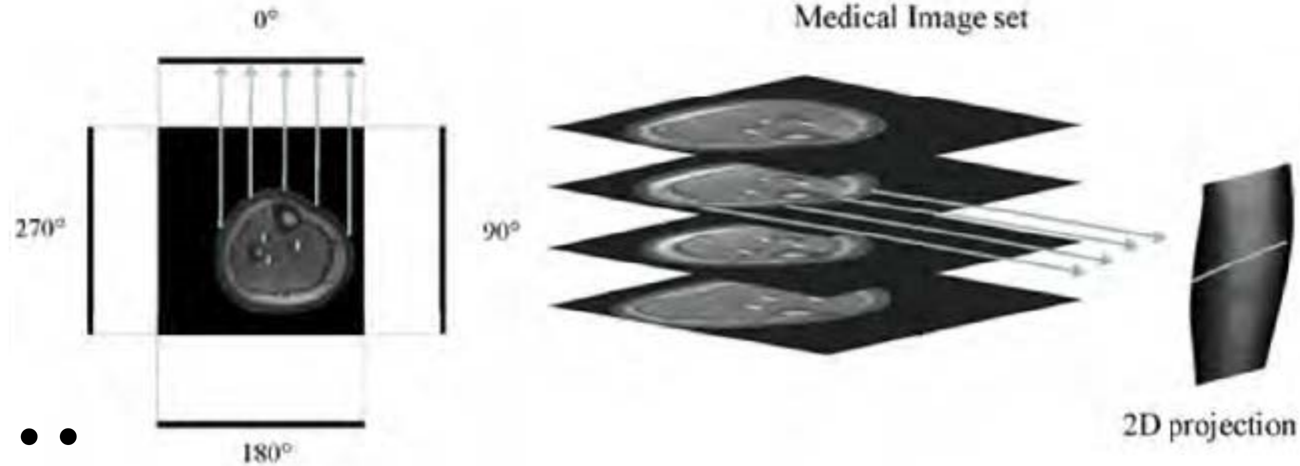
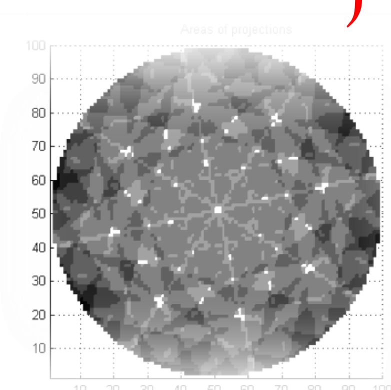
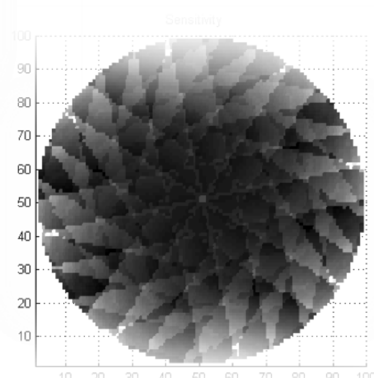


Hands on …

Reconstruction From Projection





Shireen Y. Elhabian
Amal A. Farag
Aly A. Farag
University of Louisville

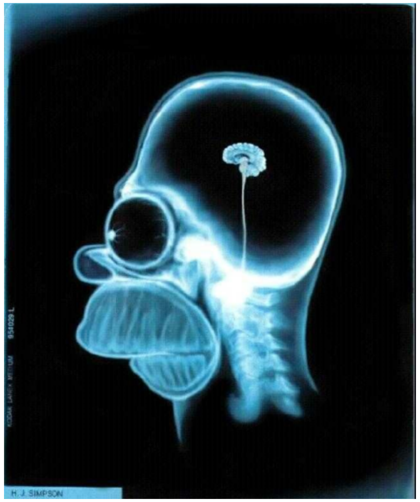
January 2009

An Analogy

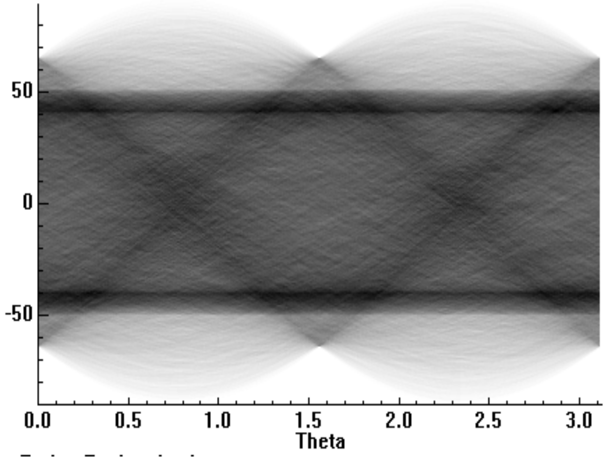
Agenda

- Reconstruction from projections (general)
  - projection geometry and radon transform
- Reconstruction methodology
  - Backprojection, (Fourier slice theorem), Filtered Backprojection.
- Reconstruction examples

Introduction



- Only photography (reflection) and planar x-ray (attenuation) measure spatial properties of the imaged object directly.

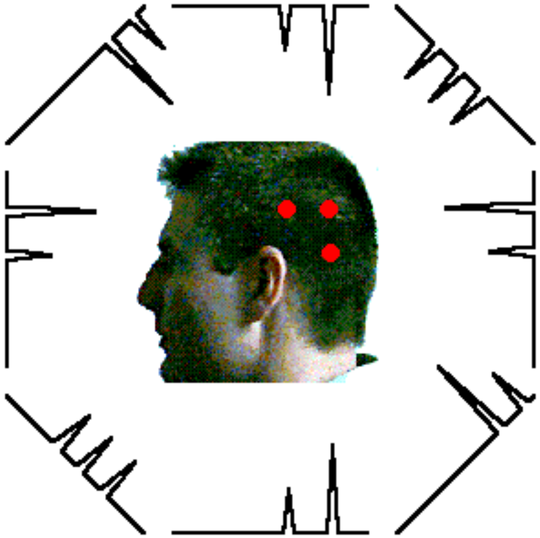


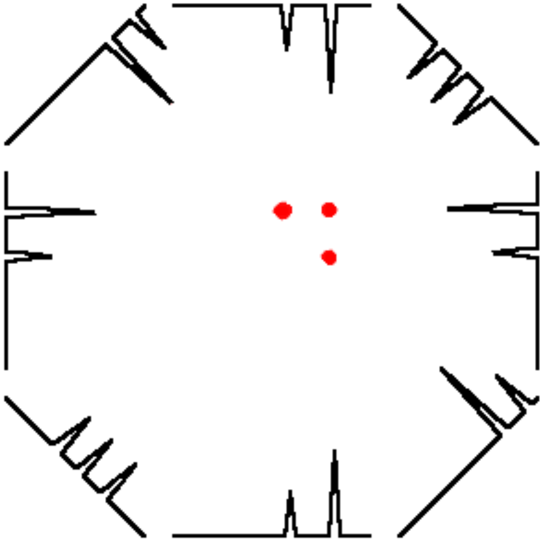
- Otherwise, measured parameters are some how related to spatial properties of imaged object.
  – CT, SPECT and PET (integral projections of parallel rays), MRI (amplitude, frequency and phase) etc...

- **Objective:** We want to construct the object (image) which creates the measured parameters.

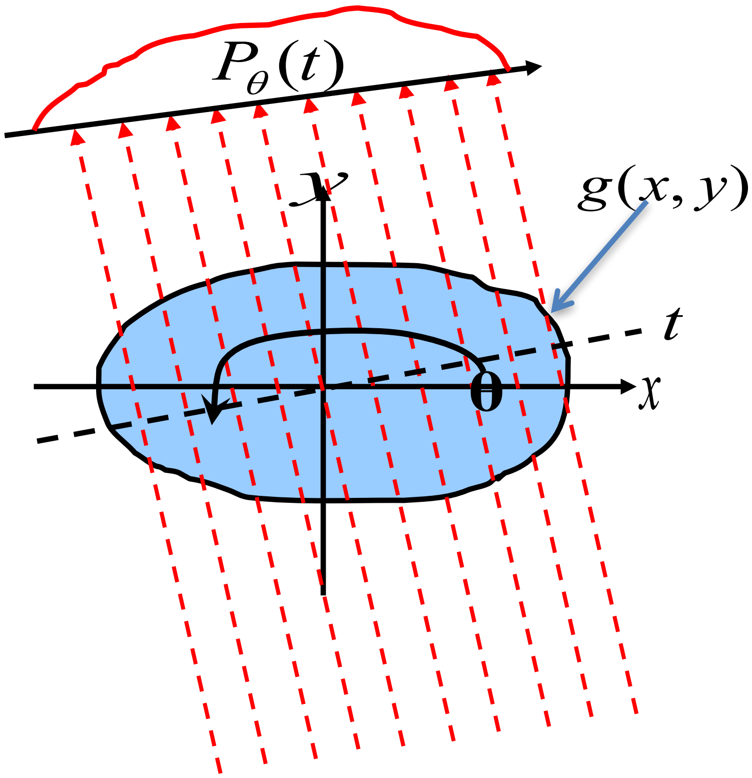
Problem Statement

- Given a set of 1-D projections and the angles at which these projections were taken.
- How do we reconstruct the 2-D image from which these projections were taken?
- Lets look at the nature of those projections … ☹





Parallel Beams Projections

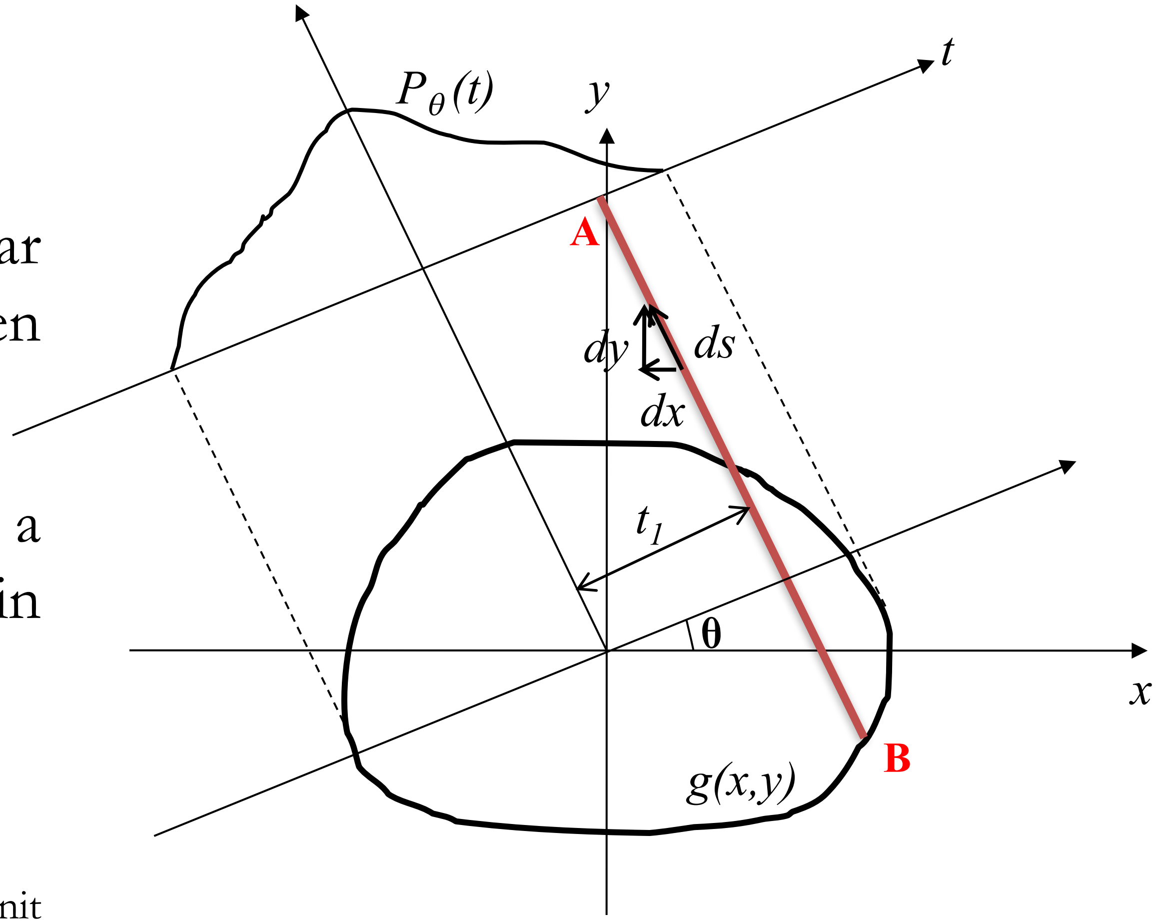


Ray Geometry

- Let x and y be rectilinear coordinates in a given plane.
- A line in this plane at a distance t_1 from the origin is the given by:

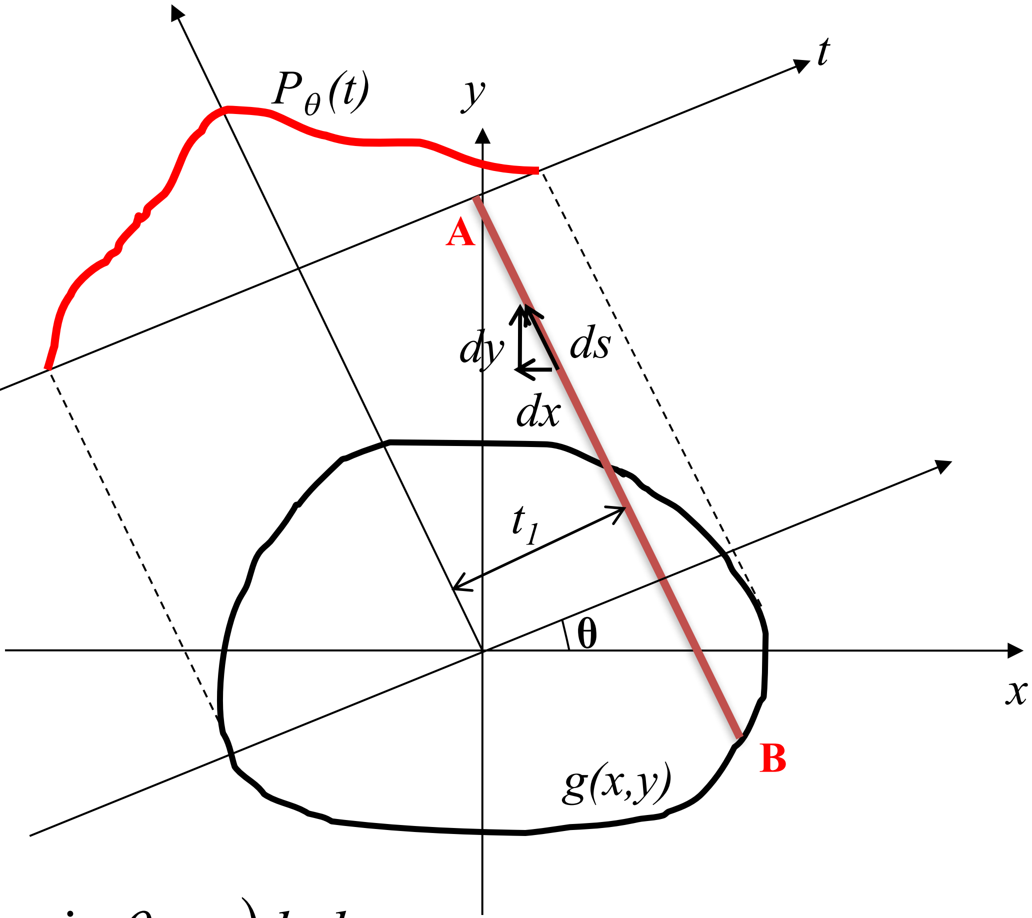
$$t_1 = x\cos\theta + y\sin\theta$$

where θ is the angle between a unit normal to the line and the x-axis.



What is Projection ?!!

- Let $g(x,y)$ be a 2-D function.
- A line running through $g(x,y)$ is called a **ray**.
- The integral of $g(x,y)$ along a ray is called **ray integral**.
- The set of ray integrals forms a **projection** defined as :



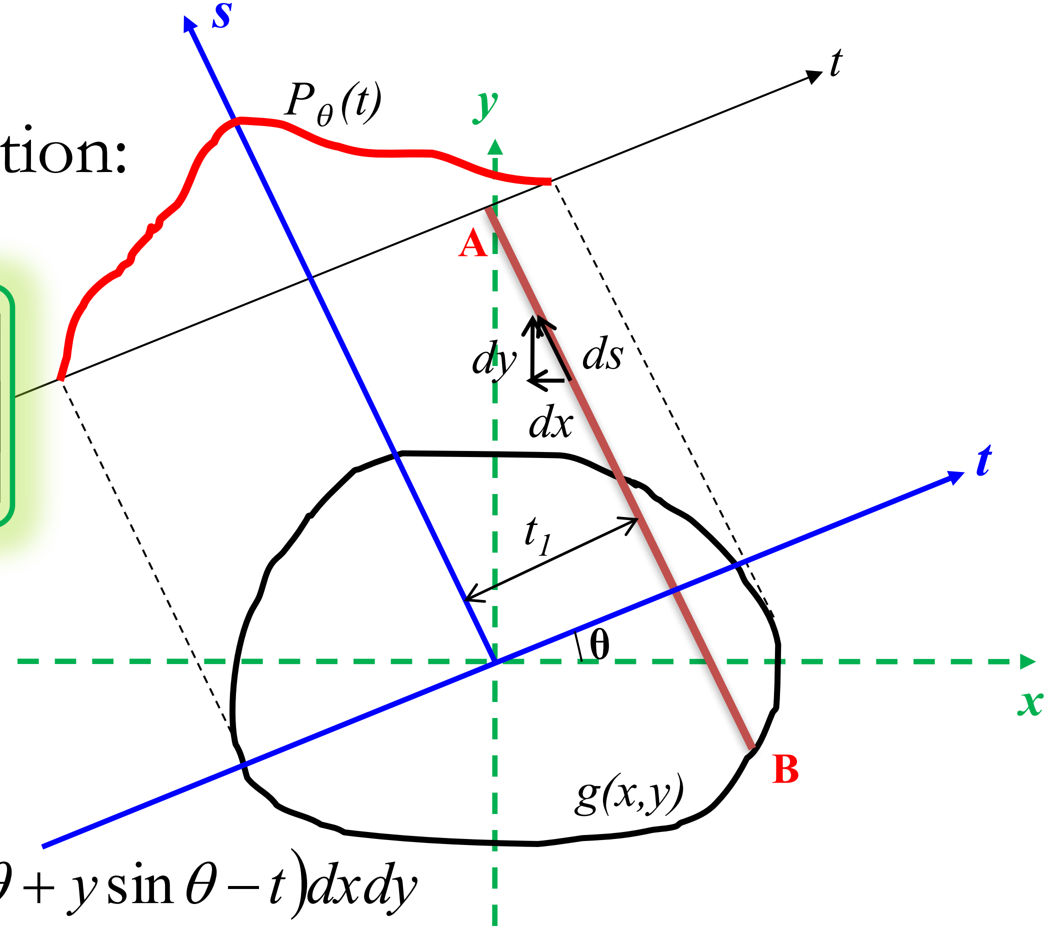
$$P_\theta(t_1) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)\delta(x\cos\theta + y\sin\theta - t_1)dxdy$$

Impulse sheath placed at the points constituting the ray

Radon Transform

- Coordinate transformation:

$$\begin{bmatrix} t \\ s \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



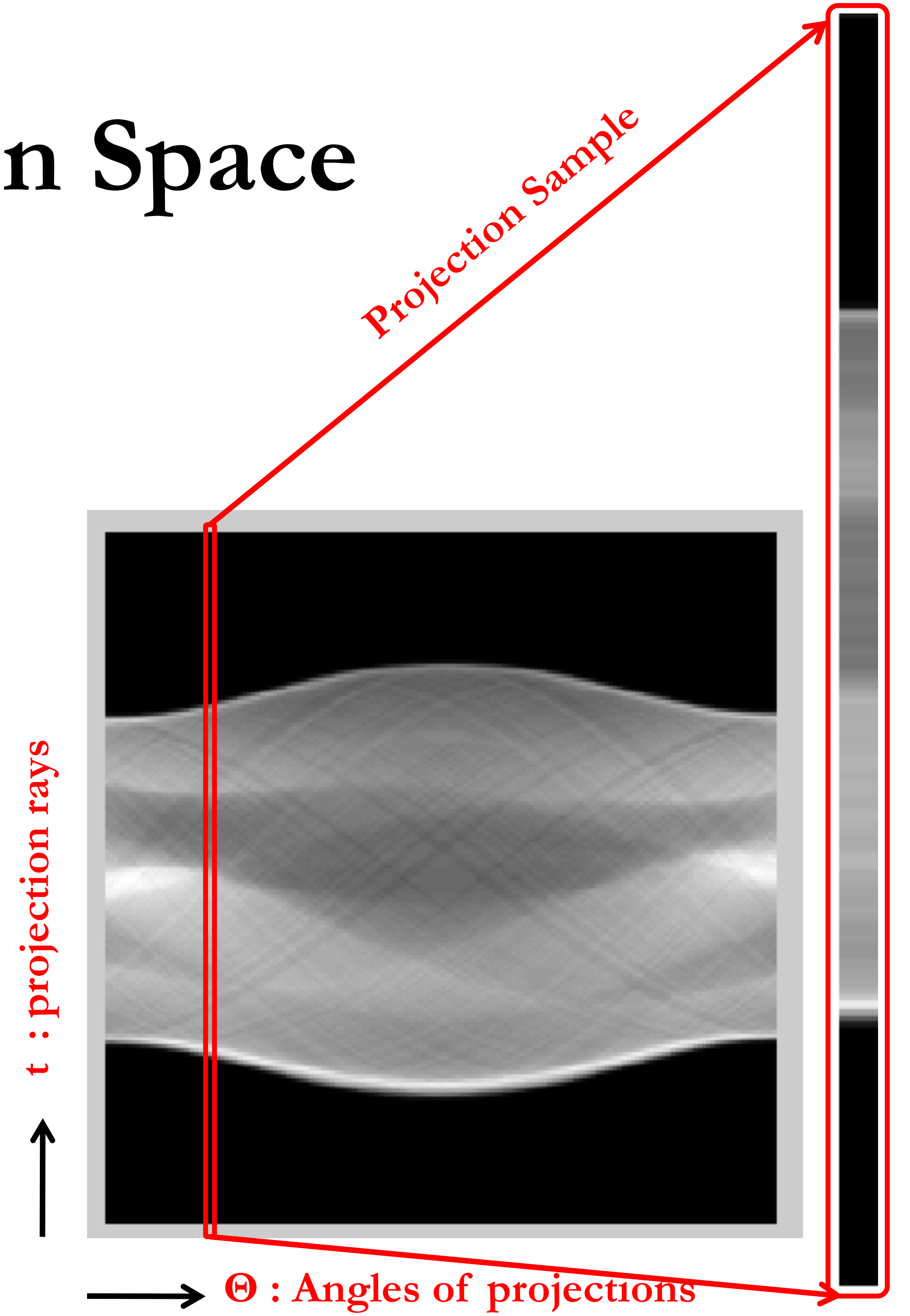
- Radon transform

$$P_\theta(t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(x,y)\,\delta(x\cos\theta + y\sin\theta - t)\,dx\,dy$$

$$= \int_{-\infty}^{\infty} g(t,s)\,ds.$$

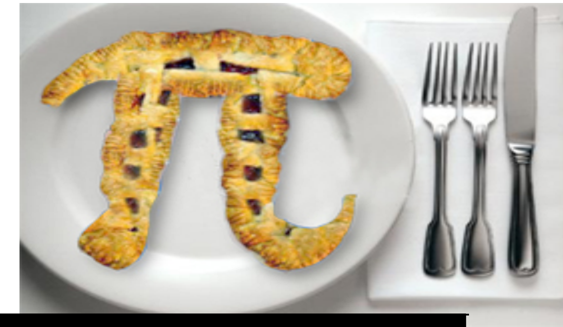
Radon Space

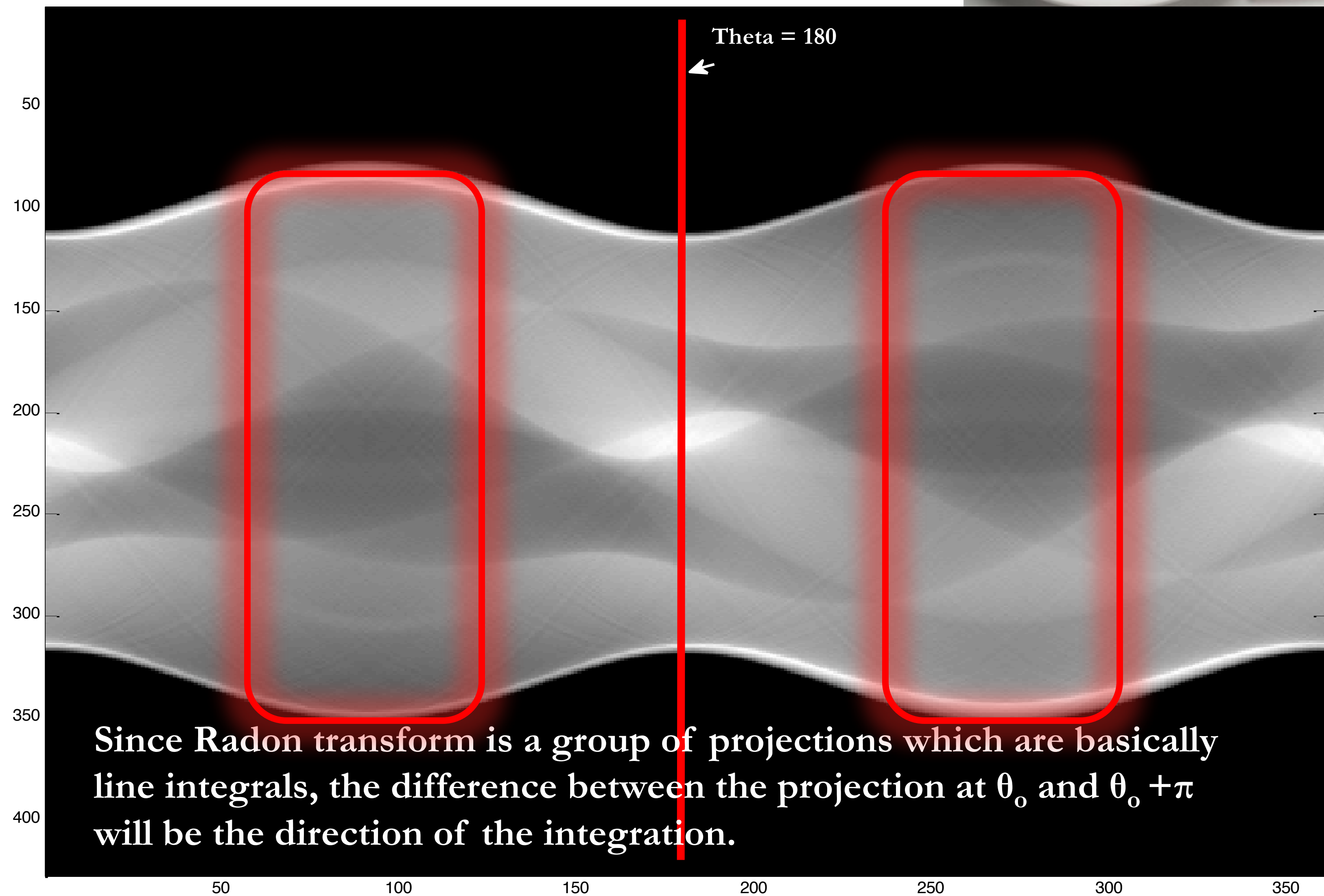
- Projections with different angles are stored in *sinogram (raw data)*.
- Each vertical line in a *sinogram* is a projection with a different angle



$\theta \in [0, \pi)$

The Myth ☺f





Since Radon transform is a group of projections which are basically line integrals, the difference between the projection at θ_o and $\theta_o + \pi$ will be the direction of the integration.

The Fourier Slice Theorem

- This theorem relates the 1D Fourier Transform of a projection and the 2D Fourier transform of the object. It relates the Fourier transform of the object along a radial line.

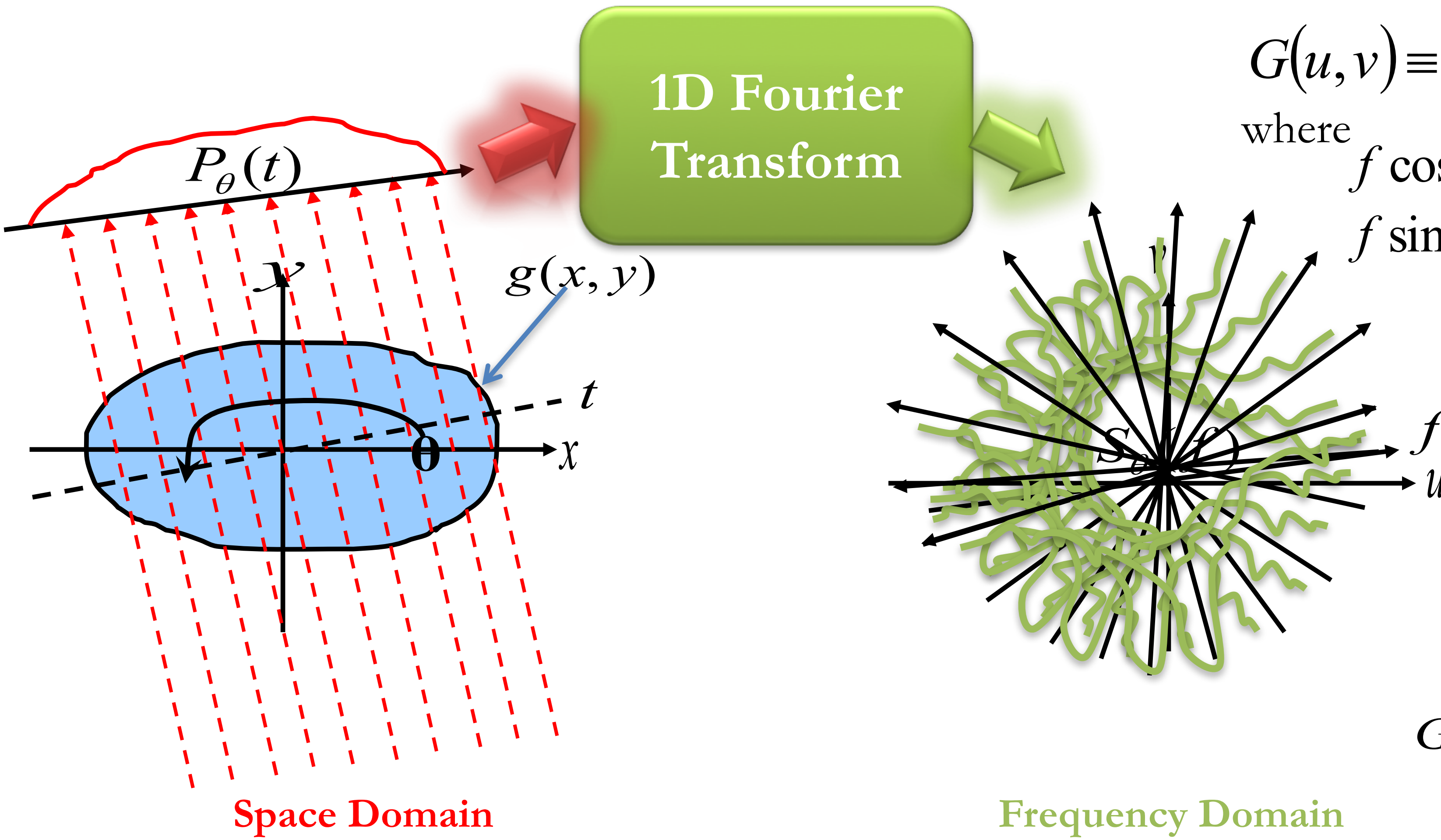
$$S_\theta(f) = \Im_{1D}\{P_\theta(t)\} = \int_{-\infty}^{\infty} P_\theta(t) e^{-j2\pi f t} dt$$

$$G(u,v) \equiv G(f,\theta)$$

where

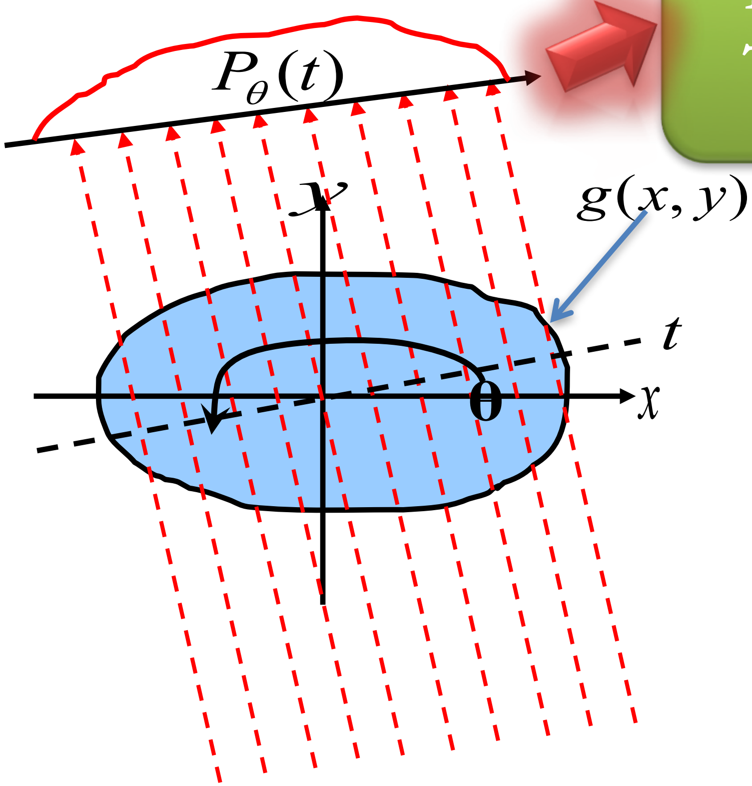
$$f \cos\theta = u$$

$$f \sin\theta = v$$



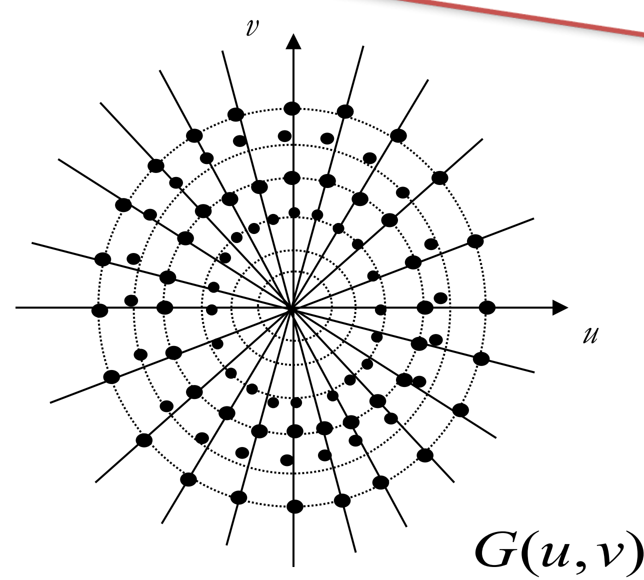
The Fourier Slice Theorem

- This theorem relates the 1D Fourier Transform of a projection and the 2D Fourier transform of the object. It relates the Fourier transform of the object along a radial line.



Space Domain

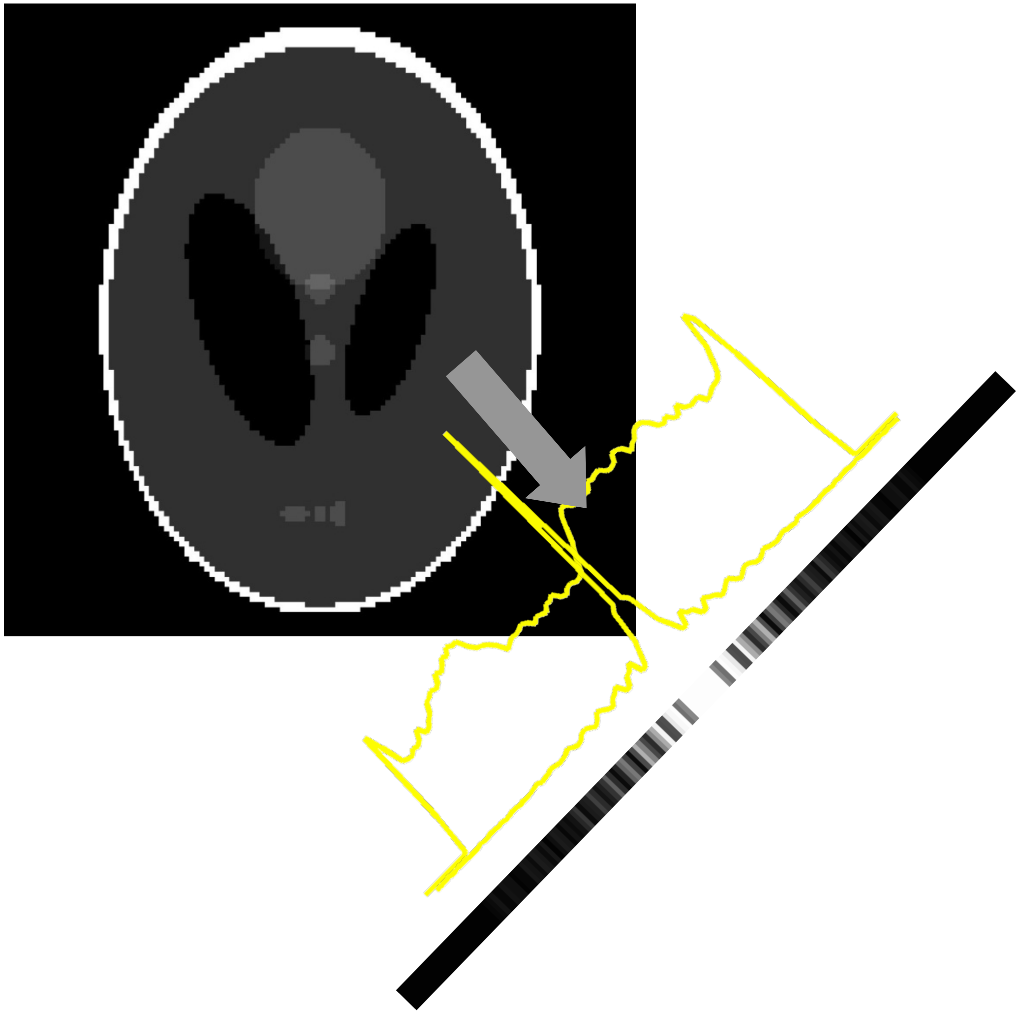
For the reconstruction to be made it is common to determine the values onto a square grid by linear interpolation from the radial points. But for high frequencies the points are further apart resulting in image degradation.



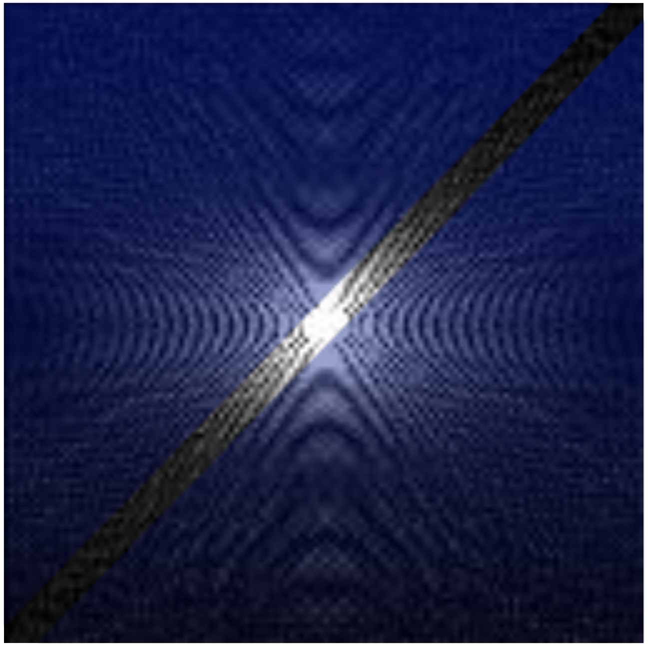
Frequency Domain

The Fourier Slice Theorem

Space Domain

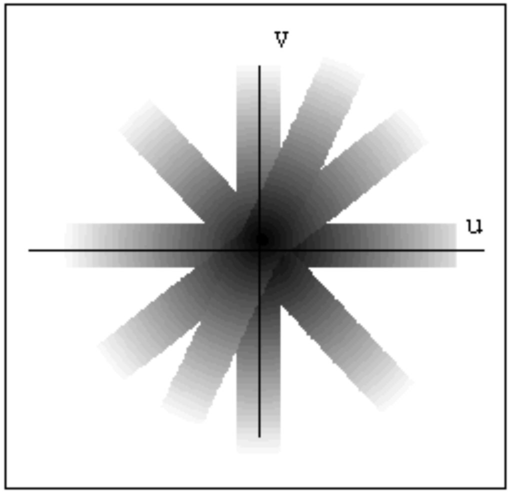


Frequency Domain
S(f,θ)



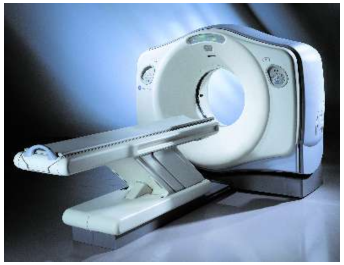
A Problem

- All projections contribute to low freqencies

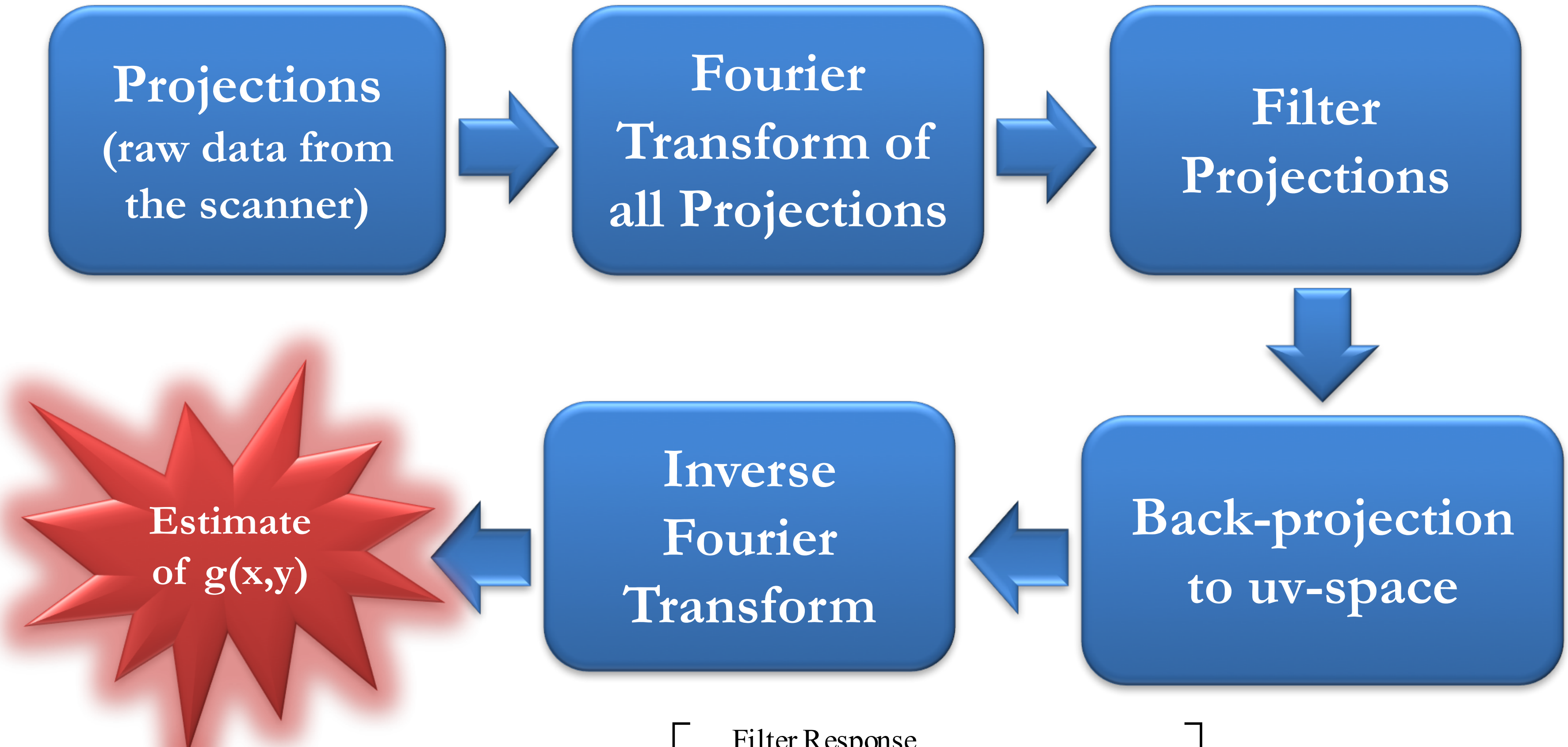


Solution:

use filtered projection



Filtered Back-projection



$$g(x,y)=\int_0^{\pi}\underbrace{\left[\int_{-\infty}^{\infty}\overbrace{|f|}^{\text{Filter Response}}S_\theta(f)e^{j2\pi ft}df\right]}_{Q_\theta(t)}d\theta$$

$$t = x\cos\theta + y\sin\theta$$

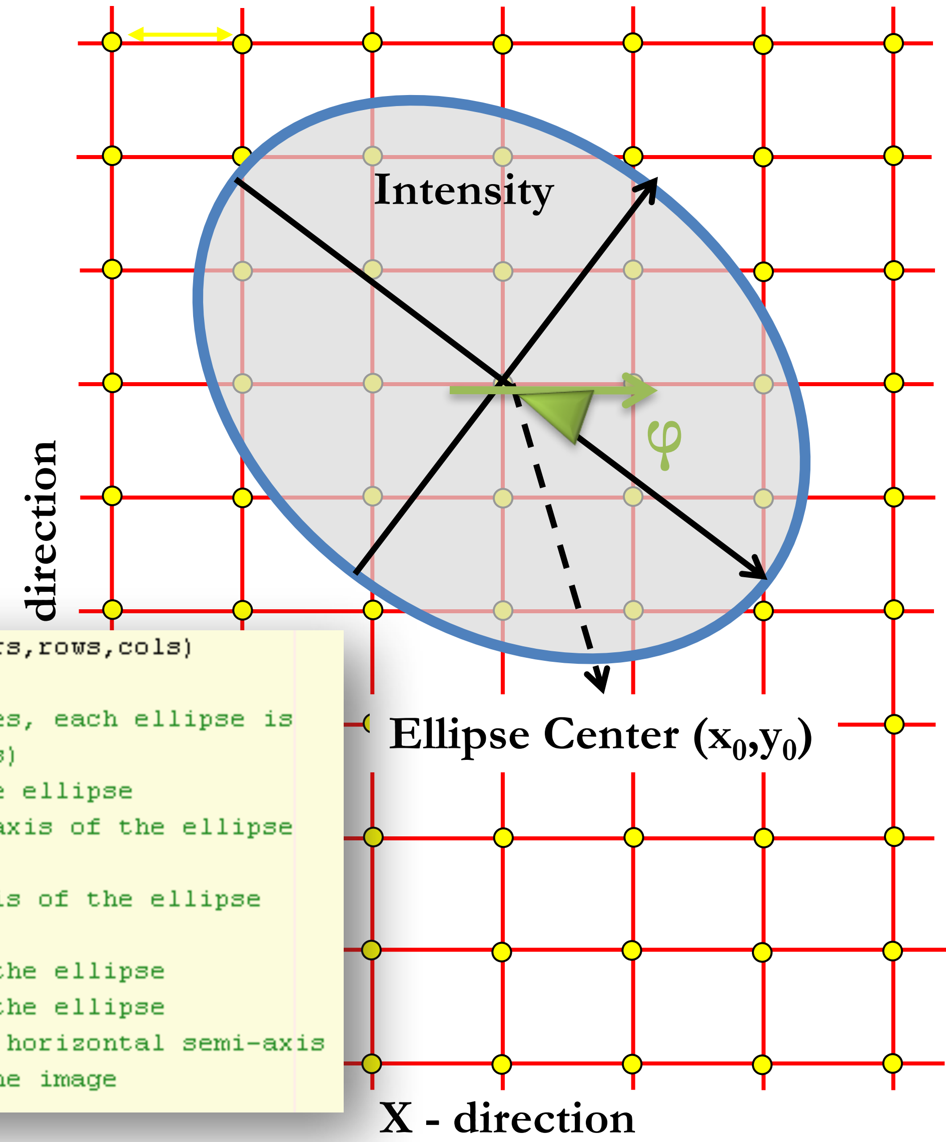
Let's do it …

Tasks :-

- Scanner simulation
  - Phantom Generation
  - Projections computation
- Reconstruction from projections
- Analysis:
  - <u>Experiment 1</u>: the effect of filter type.
  - <u>Experiment 2</u>: the effect of number of projections.
  - <u>Experiment 3</u>: the effect of number of rays

Scanner Simulation – Phantom Generation

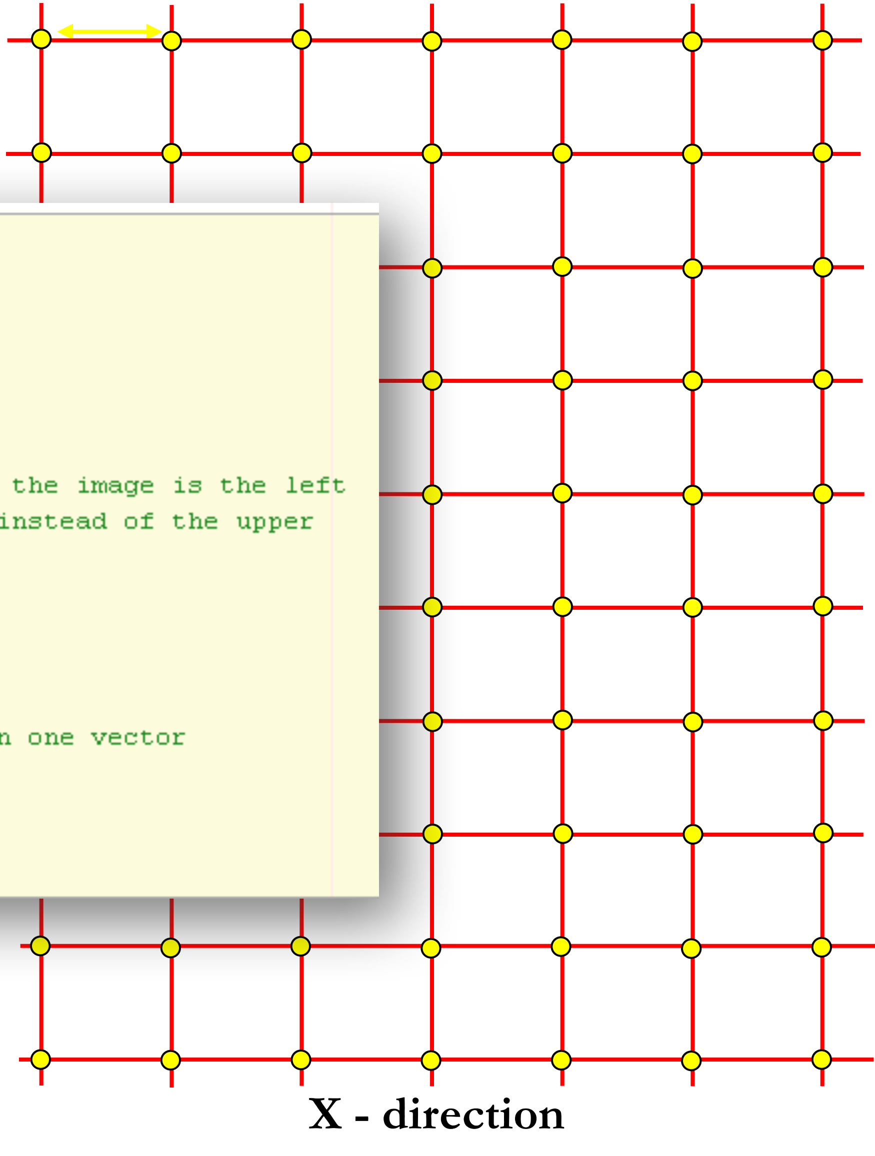
- Given the spatial support of our phantom.
- We assume that our phantom is constructed of a set of ellipses, each has the following parameters:
  - Intensity, ellipse center *(x_0,y_0)*, ellipse major and minor axes length *(a,b)*, and the orientation (φ i.e. rotation angle)




```
function phantom_img = generate_phantom(ellipse_parameters,rows,cols)

% this function generates a 2D synthetic image of ellipses, each ellipse is
% defined by the following parameters (ellipse_parameters)
%     Column 1:  A    the additive intensity value of the ellipse
%     Column 2:  a    the length of the horizontal semi-axis of the ellipse
%                     i.e radius in the x-direction
%     Column 3:  b    the length of the vertical semi-axis of the ellipse
%                     i.e radius in the y-direction
%     Column 4:  x0   the x-coordinate of the center of the ellipse
%     Column 5:  y0   the y-coordinate of the center of the ellipse
%     Column 6:  phi  the angle (in degrees) between the horizontal semi-axis
%                     of the ellipse and the x-axis of the image
```

Scanner Simulation – Phantom Generation

```
%% intialization of our phantom
phantom_img = zeros(rows,cols);

% the spatial support (normalized to -1->1
xmid = (cols-1)/2;
ymid = (rows-1)/2;
x_range = ((0:cols-1) - xmid)./xmid;
y_range = ((rows-1:-1:0) - ymid)./ymid; %the origin of the image is the left
                                        %lower corner instead of the upper
                                        %one

% defining the grid points of our phantom
[x,y] = meshgrid(x_range,y_range);

% getting the xy coordinates and the phantom flatten in one vector
x = x(:);
y = y(:);
phantom_img = phantom_img(:);
```



Scanner Simulation – Phantom Generation

```
%% now lets loop over all ellipses to find the corresponding phantom points
for i = 1 : size(ellipse_parameters,1)
    % the parameters of current ellipse
    A = ellipse_parameters(i,1);
    a = ellipse_parameters(i,2);
    b = ellipse_parameters(i,3);
    x0 = ellipse_parameters(i,4);
    y0 = ellipse_parameters(i,5);
    phi = ellipse_parameters(i,6);

    % lets translate the phantom coordinates to be centered at the ellipe's
    % center
    cur_x = x - x0;
    cur_y = y - y0;

    % lets rotate the translated phantom coordinates to align the x-axis
    % with the ellipse's horizontal semi-axis and the y-axis with the
    % ellipse's vertical semi-axis

    rotation_matrix = [ cosd(phi) sind(phi);
                       -sind(phi) cosd(phi)];

    pts = [cur_x' ; cur_y'];
    pts = rotation_matrix * pts ;

    cur_x = pts(1,:);
    cur_y = pts(2,:);
```

```
    % lets see which points in the phantom that will belong to the current
    % ellipse
    x2 = cur_x.^2;
    y2 = cur_y.^2;
    a2 = a^2;
    b2 = b^2;

    index = (x2./a2) + (y2./b2) <= 1;
    phantom_img(index) = phantom_img(index) + A ;
end
phantom_img = reshape(phantom_img,[rows,cols]);
```

Scanner Simulation – Phantom Generation

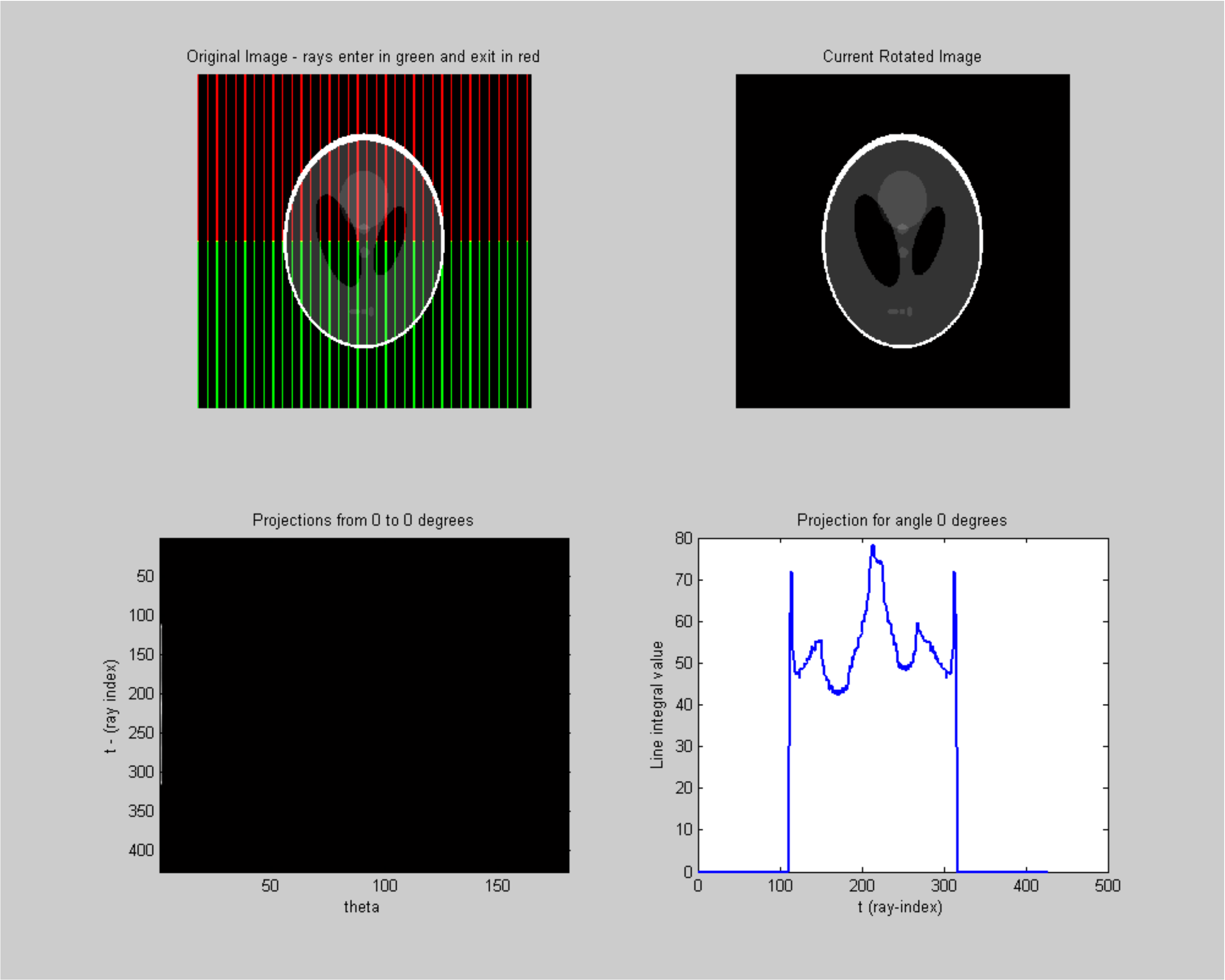
```
ellispes_parameters =

    1.0000    0.6900    0.9200         0         0         0
   -0.8000    0.6624    0.8740         0   -0.0184         0
   -0.2000    0.1100    0.3100    0.2200         0  -18.0000
   -0.2000    0.1600    0.4100   -0.2200         0   18.0000
    0.1000    0.2100    0.2500         0    0.3500         0
    0.1000    0.0460    0.0460         0    0.1000         0
    0.1000    0.0460    0.0460         0   -0.1000         0
    0.1000    0.0460    0.0230   -0.0800   -0.6050         0
    0.1000    0.0230    0.0230         0   -0.6060         0
    0.1000    0.0230    0.0460    0.0600   -0.6050         0
```

Scanner Simulation – Projections Generation

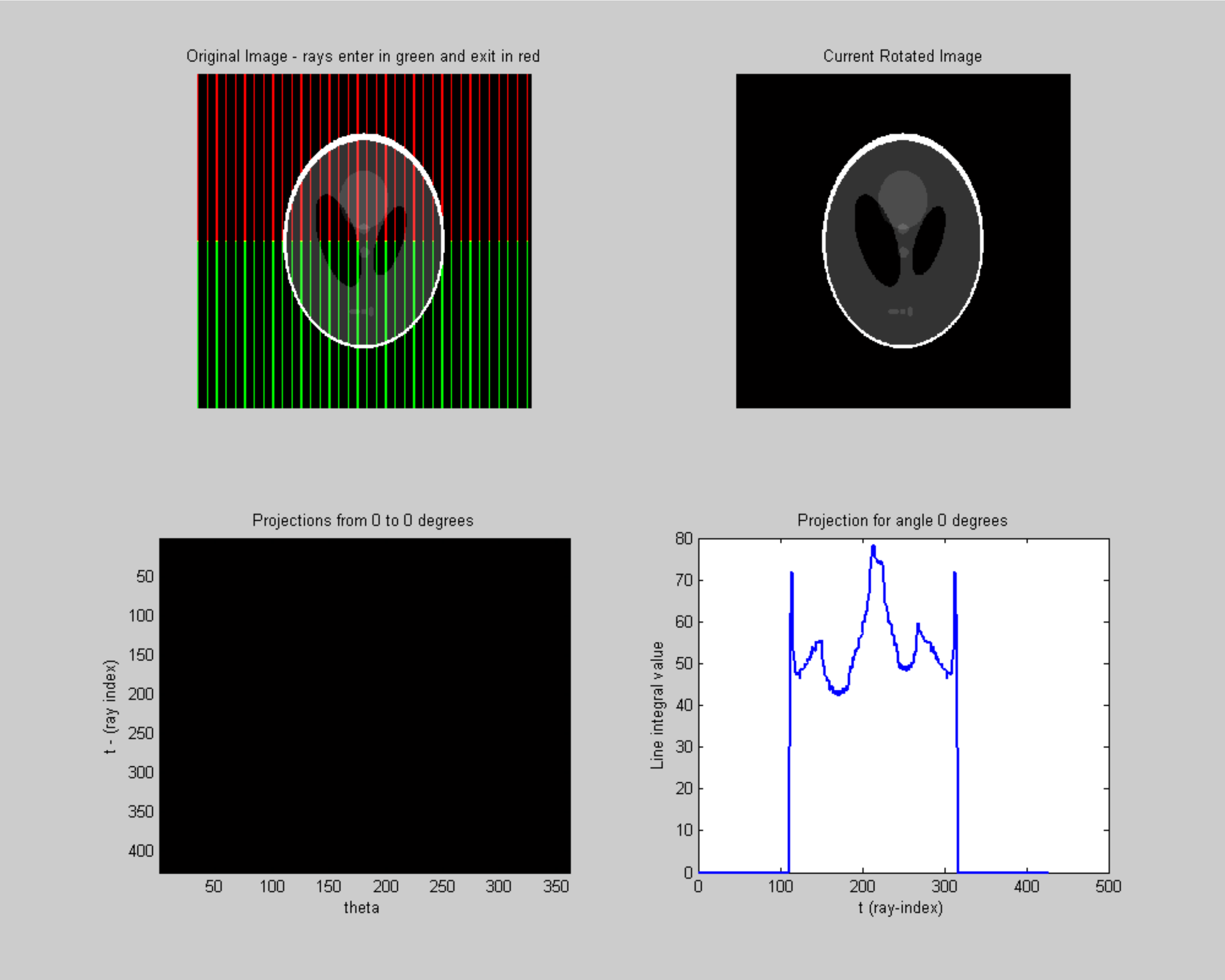
- To be able to study different reconstruction techniques, we first needed to write a program that take projections of a known image.
- Basically, we take the image (which is just a matrix of intensities), rotate it, and sum up the intensities.
- In MATLAB this is easily accomplished with the 'imrotate' and 'sum' commands.
- But first, we zero pad the image so we don't lose anything when we rotate.

```
%% after padding the image, do the following to generate the projections
thetas = 0:180;
no_of_rays = 300;
projections = zeros(length(thetas),no_of_rays);
for i = 1 : length(thetas)
    rotated_phantom = imrotate(padded_phantom_image, theta(i), 'bilinear','crop');
    projections(:,i) = (sum(rotated_phantom))';
end
```

Scanner Simulation – Projections Generation from 0 to π



Scanner Simulation – Projections Generation from 0 to 2π



Reconstruction From Projections

- Given the projections, we first filter them as shown below.

```
% number of rays, which corresponds to number of samples in the discretized
% 1D projection
N = size(projections,1);

% sampling the frequency w = 2*pi*f
w = -pi : (2*pi)/N : pi-(2*pi)/N; % -pi to pi

% shifting the response to 0 to 2*pi
filter_response = fftshift(abs(w));

% number of projections
nProjections = size(projections,2);

filtered_projections = zeros(size(projections));
for i = 1:nProjections
    % filter in the frequency domain
    S_f = fft(projections(:,i));
    filtered_S_f = S_f.*filter_response';
    % return to t-theta domain
    filtered_projections(:,i) = ifft(filtered_S_f);
end

% Remove any remaining imaginary parts
filtered_projections = real(filtered_projections);
```

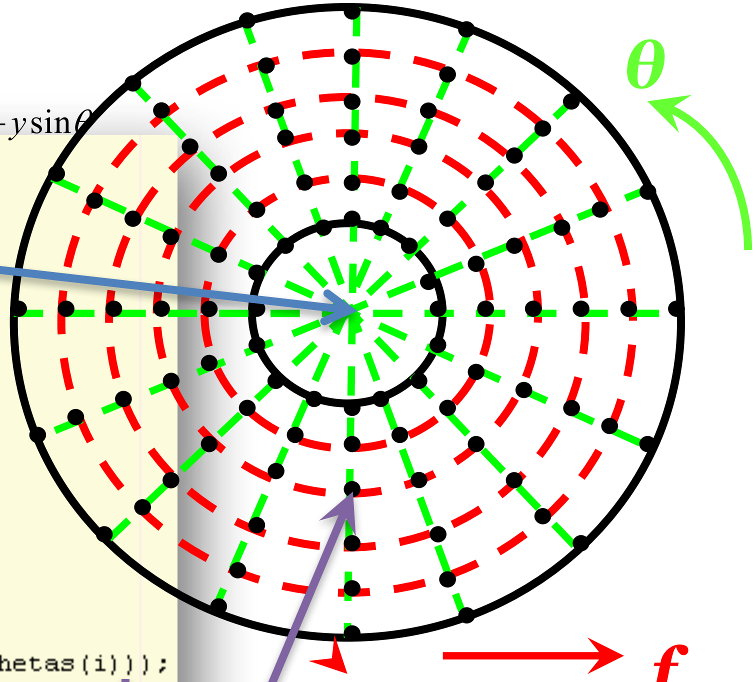
Reconstruction From Projections

- Given the angles where the projections were taken, and the filtered projections, the following will reconstruct an estimate of the original image.

$$g(x,y)=\int_0^{\pi}\underbrace{\left[\int_{-\infty}^{\infty}|f|S_\theta(f)e^{j2\pi ft}\,df\right]}_{Q_\theta(t)}d\theta$$

$t=x\cos\theta+y\sin\theta$

```
% find the middle index of the projections
center = (nProjections+1)/2;

% set up x and y matrices
x = 1:nProjections;
y = 1:nProjections;
[X,Y] = meshgrid(x,y);
% having the origin in the middle of the grid
xproj = X - (nProjections+1)/2;
yproj = Y - (nProjections+1)/2;

reconstructed_image = zeros(nProjections,nProjections);
for i = 1:nProjections
    % figure out which projections to add to which spots
    cur_points = round(center + xproj*cos(thetas(i)) + yproj*sin(thetas(i)));

    % if we are "in bounds" then add the point
    cur_reconstruction = zeros(nProjections,nProjections);
    spot = find((cur_points > 0) & (cur_points <= N));
    new_points = cur_points(spot);
    cur_reconstruction(spot) = filtered_projections(new_points(:),i);
    %keyboard
    reconstructed_image = reconstructed_image + cur_reconstruction;
end
reconstructed_image = reconstructed_image./nProjections;
```

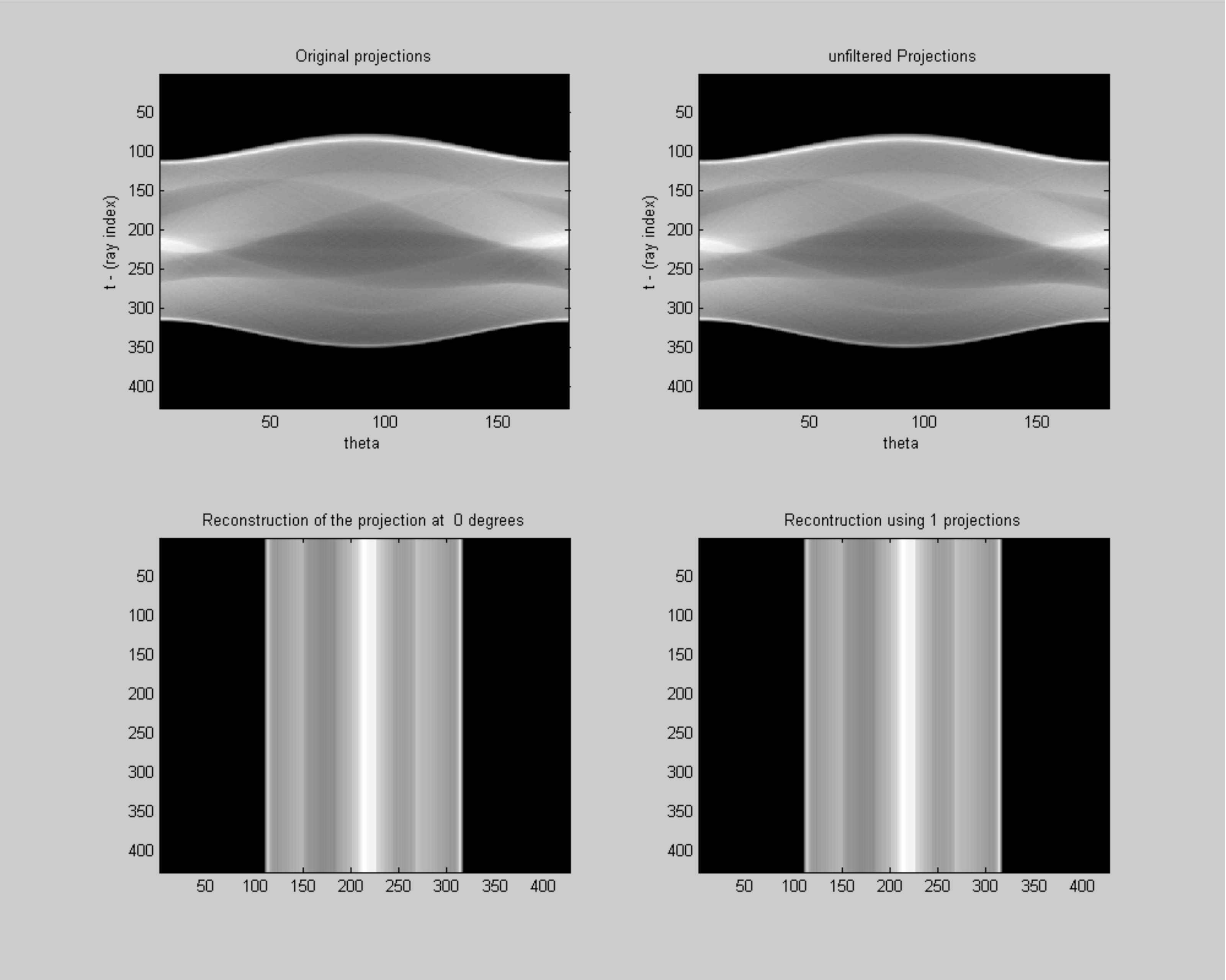


Point on the current radial line which corresponds to the 1D fourier transform of the current projection

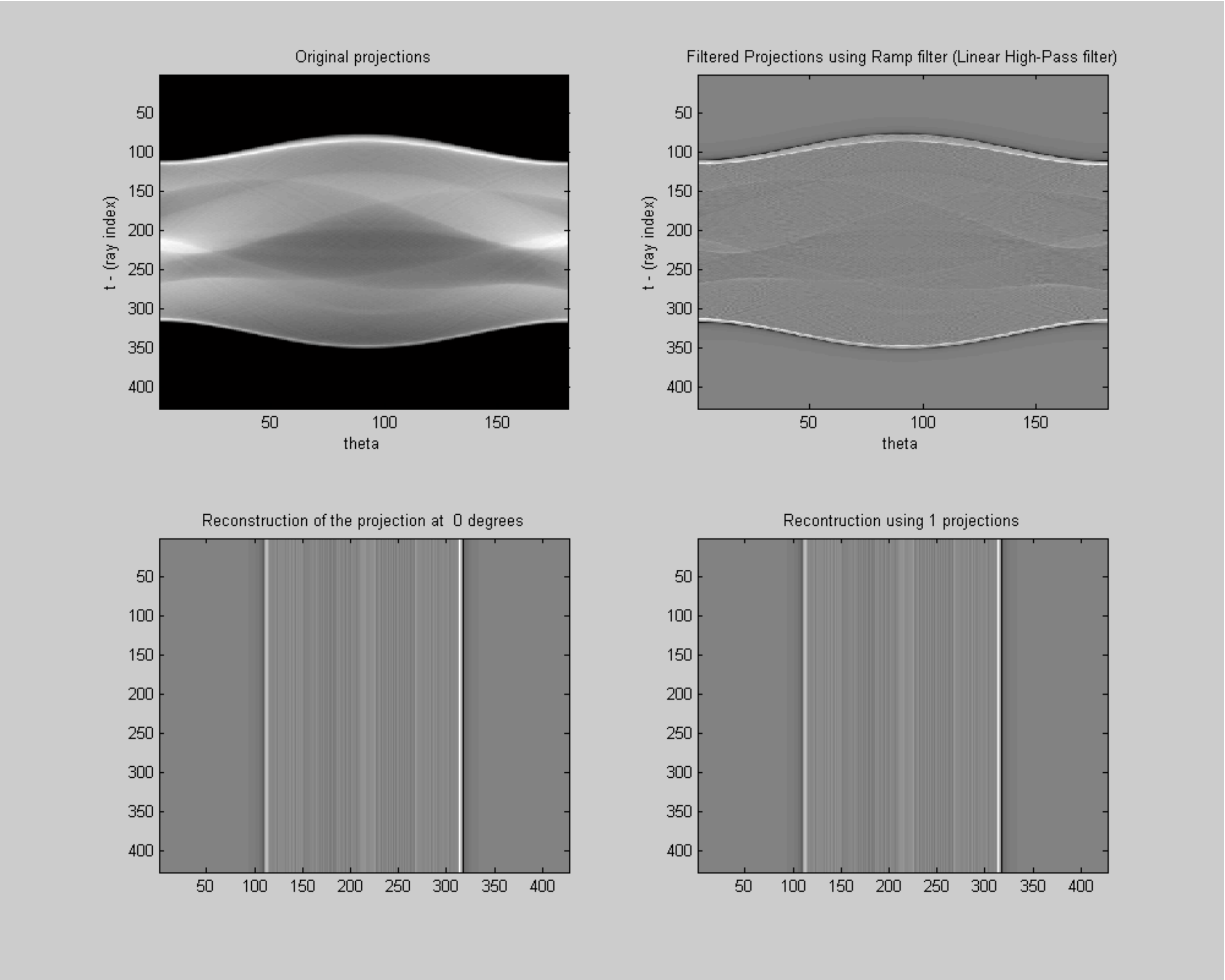
Experiment One

Studying the effect of using different filter types compared to the unfiltered case.

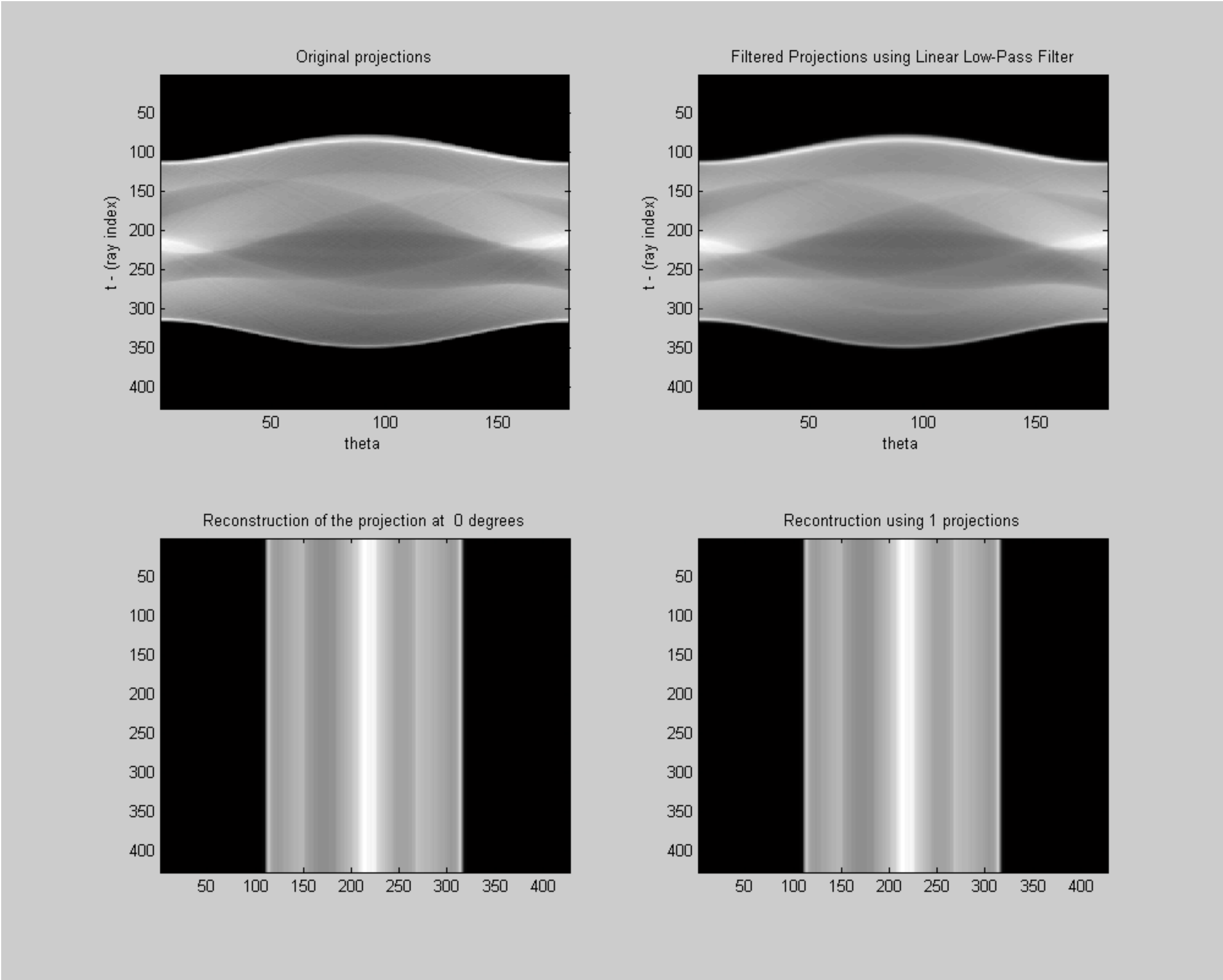
Reconstruction using unfiltered projections



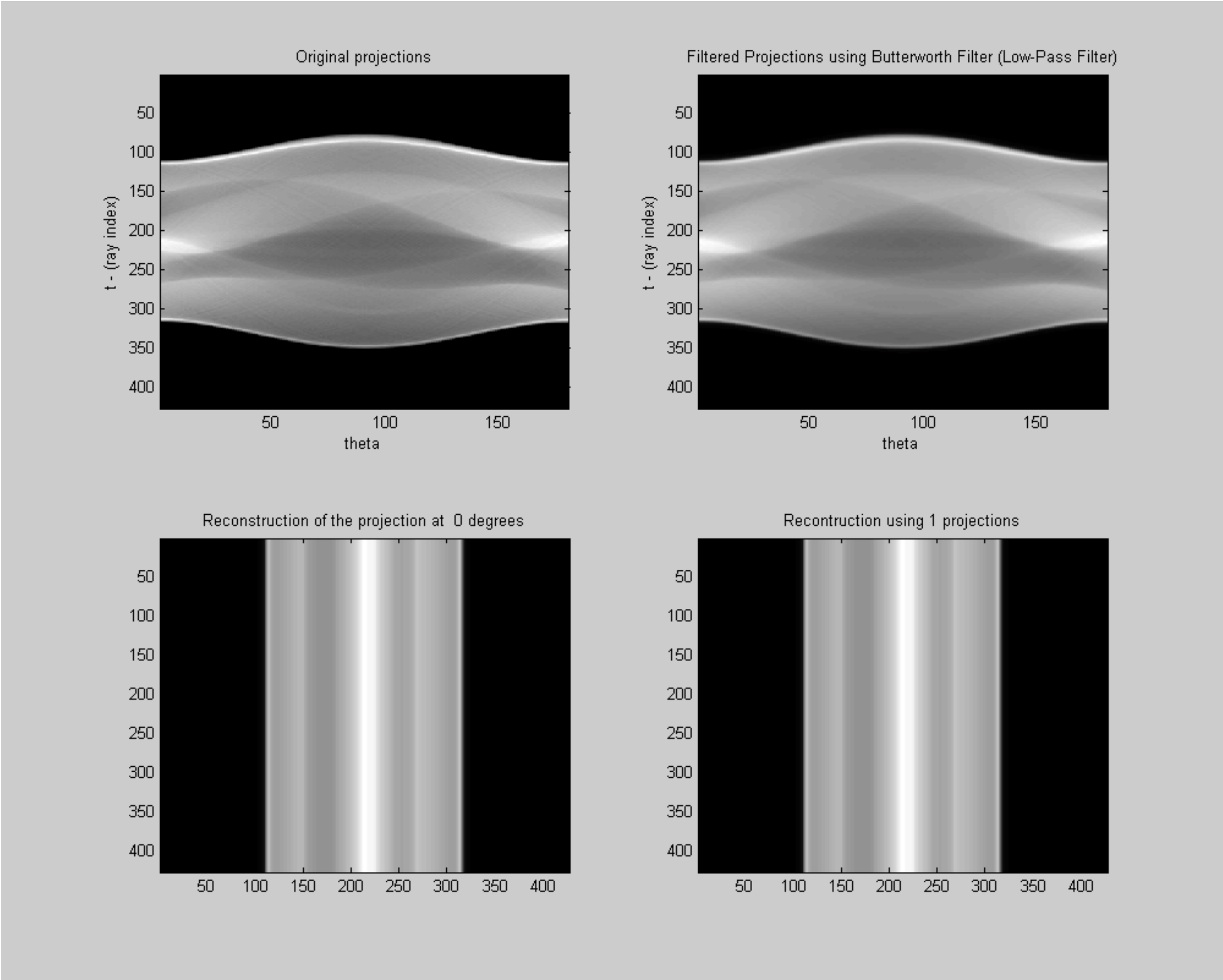
Reconstruction using Ramp filter



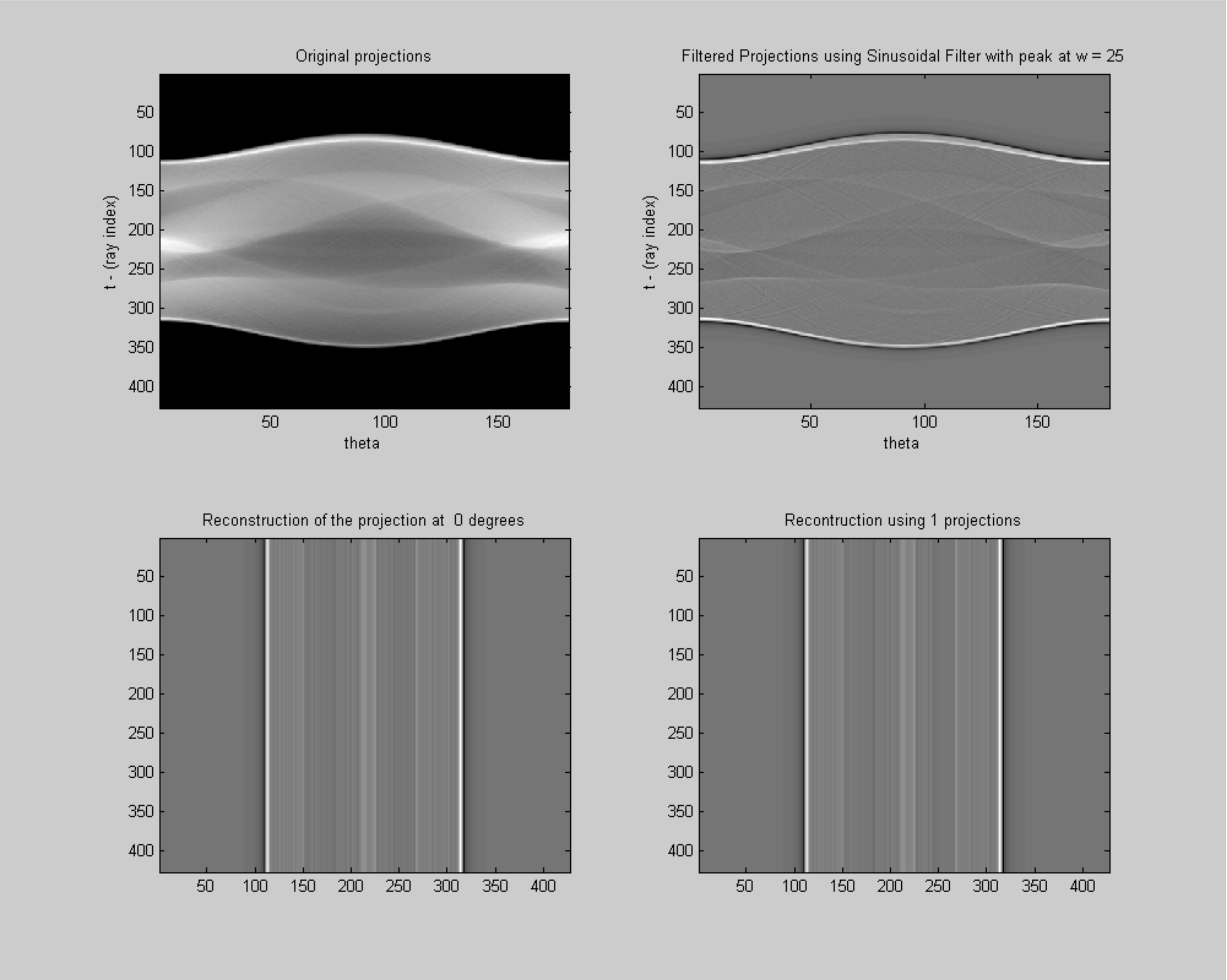
Reconstruction using LPF filter



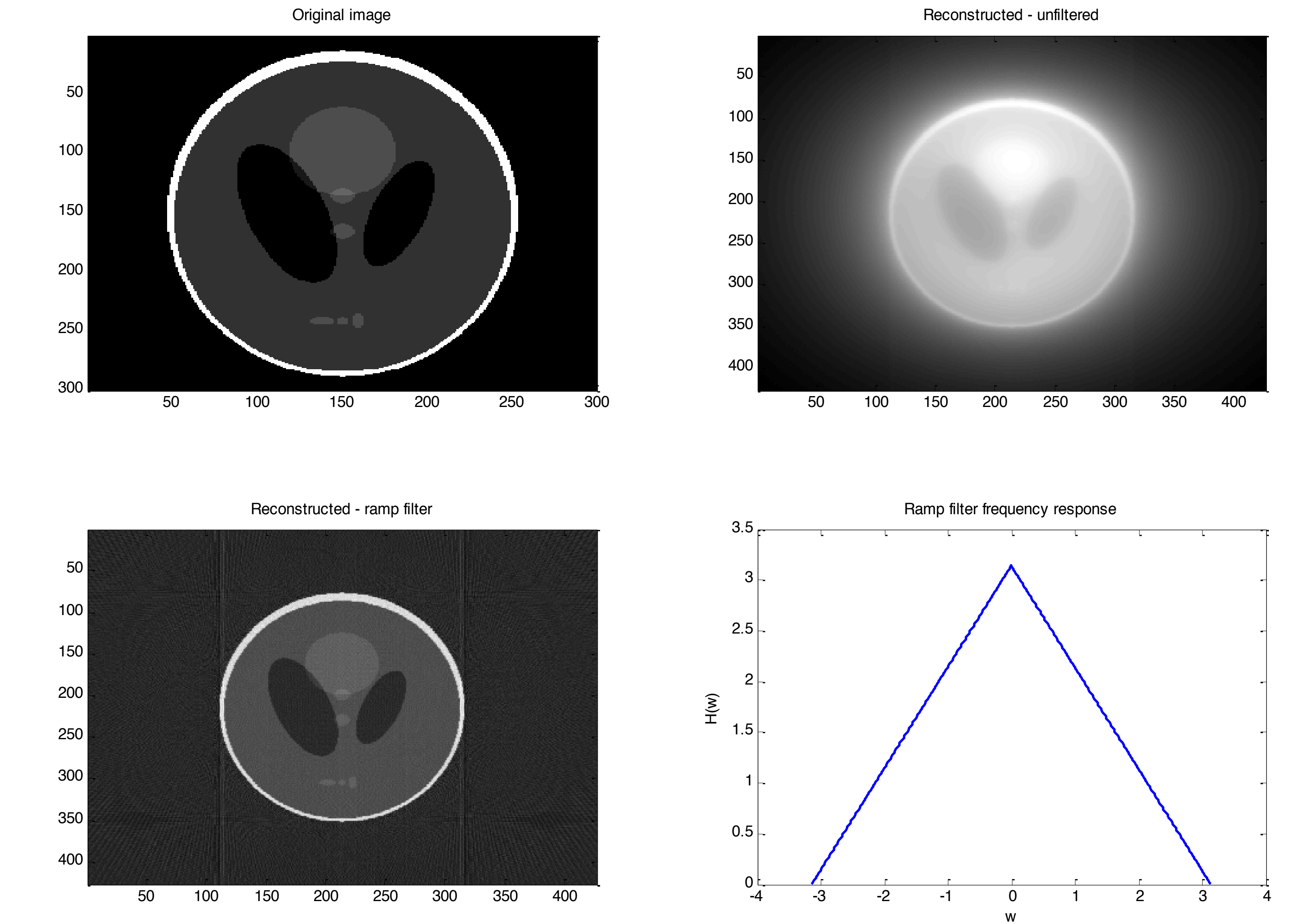
Reconstruction using <u>Butterworth</u> filter



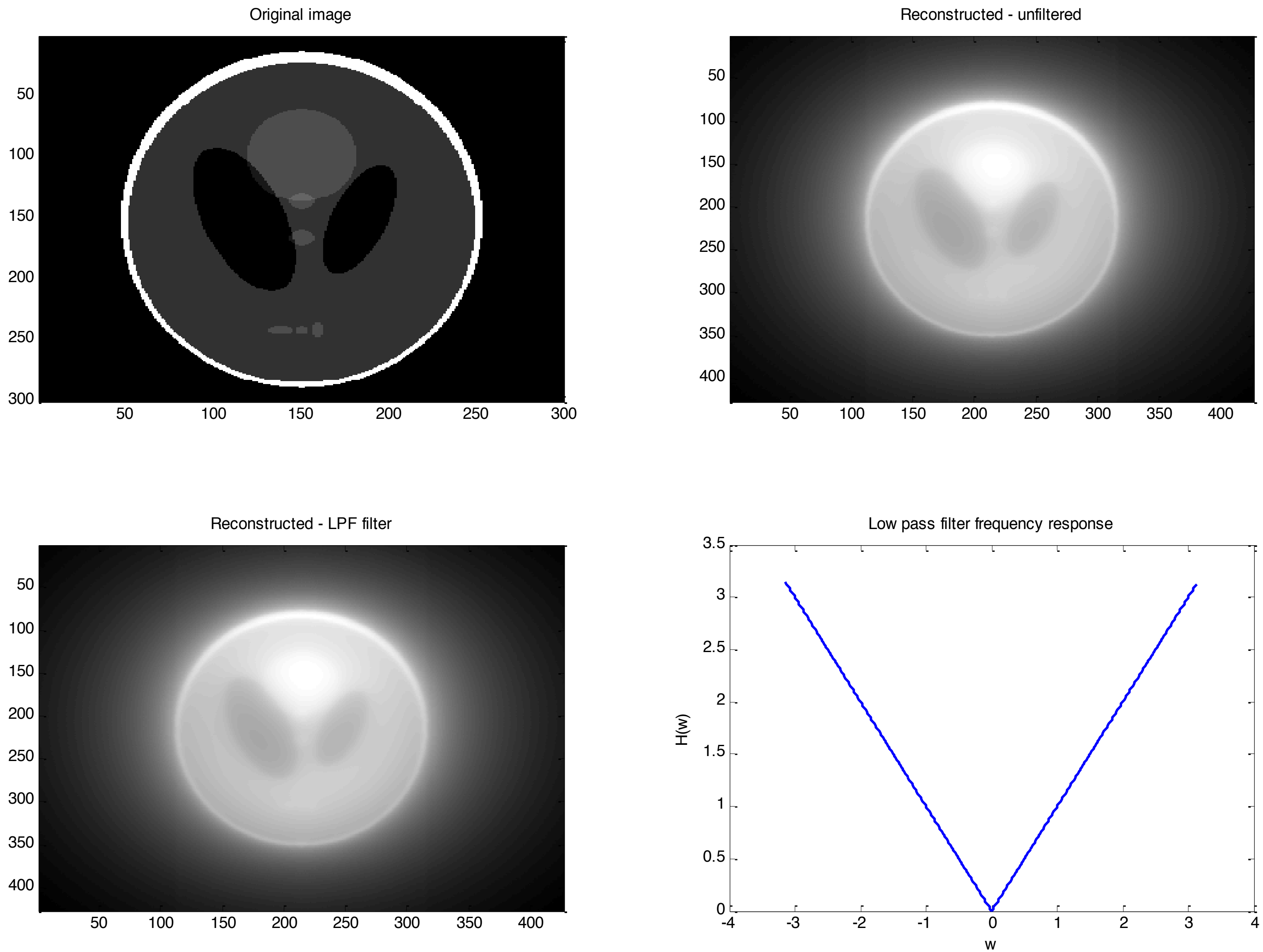
Reconstruction using <u>Sinusoidal</u> filter



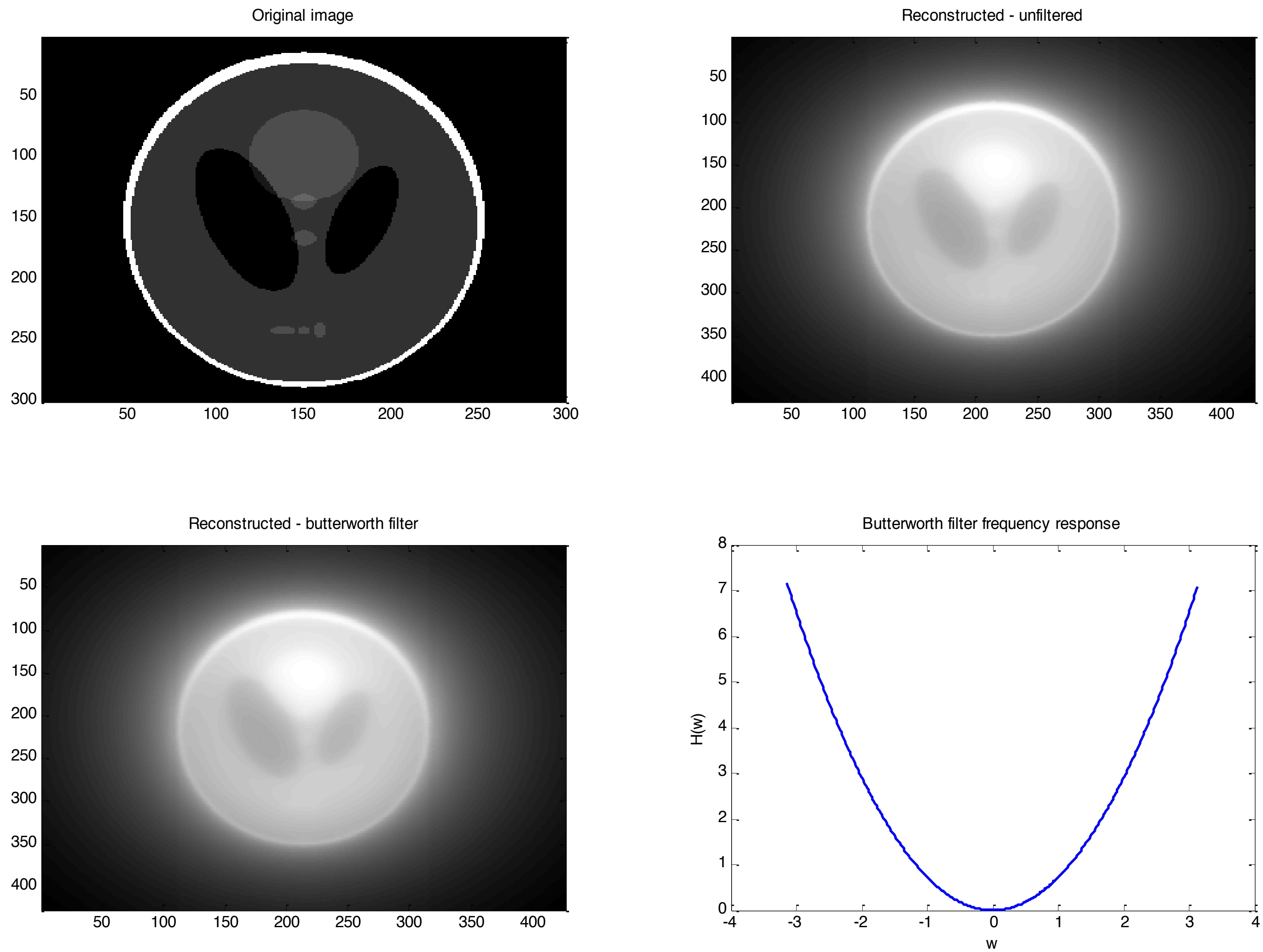
Reconstruction using <u>Ramp</u> filter vs unfiltered case



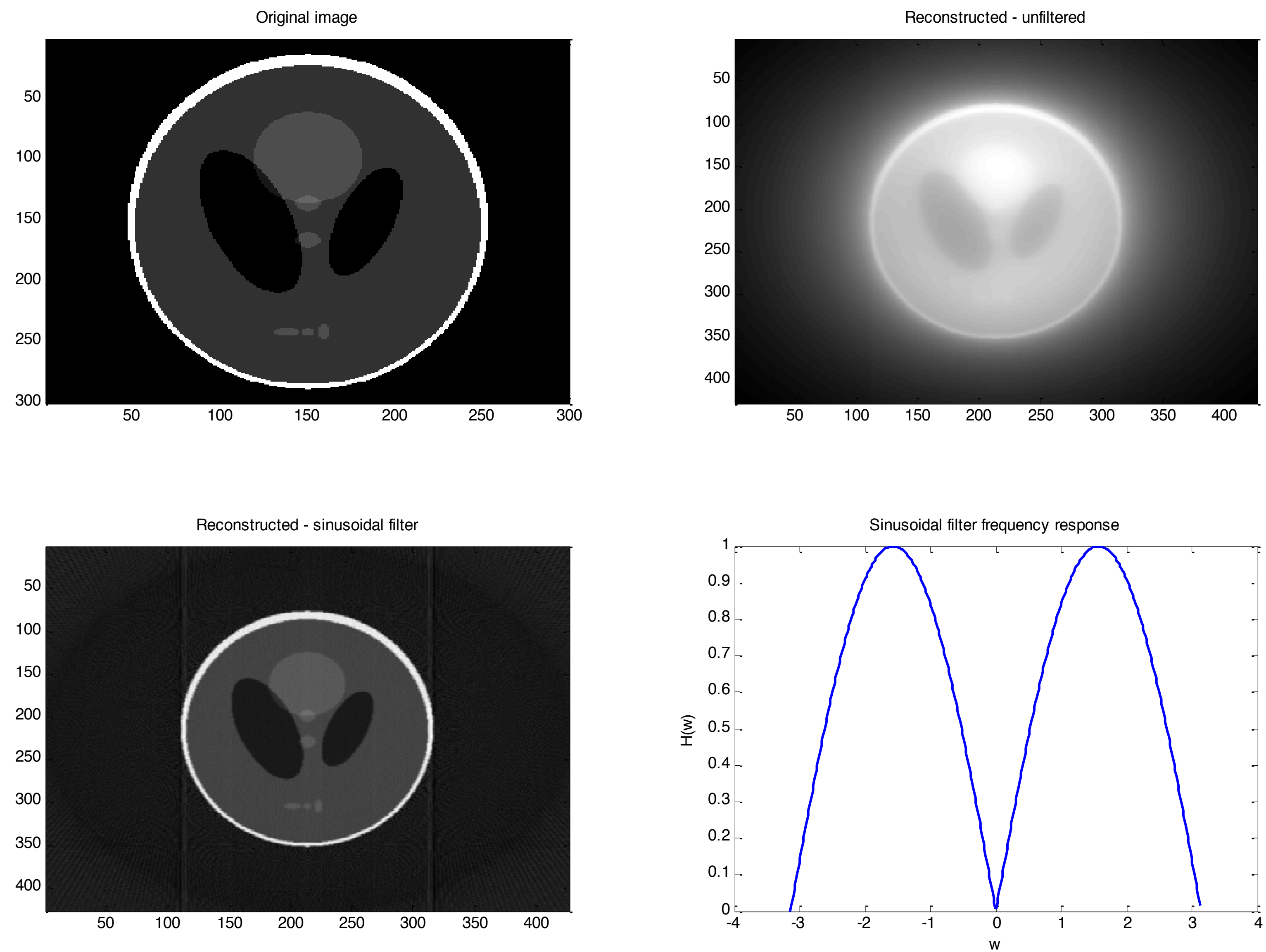
Reconstruction using LPF filter vs unfiltered case



Reconstruction using <u>Butterworth</u> filter vs unfiltered case



Reconstruction using <u>Sinusoidal</u> filter vs unfiltered case

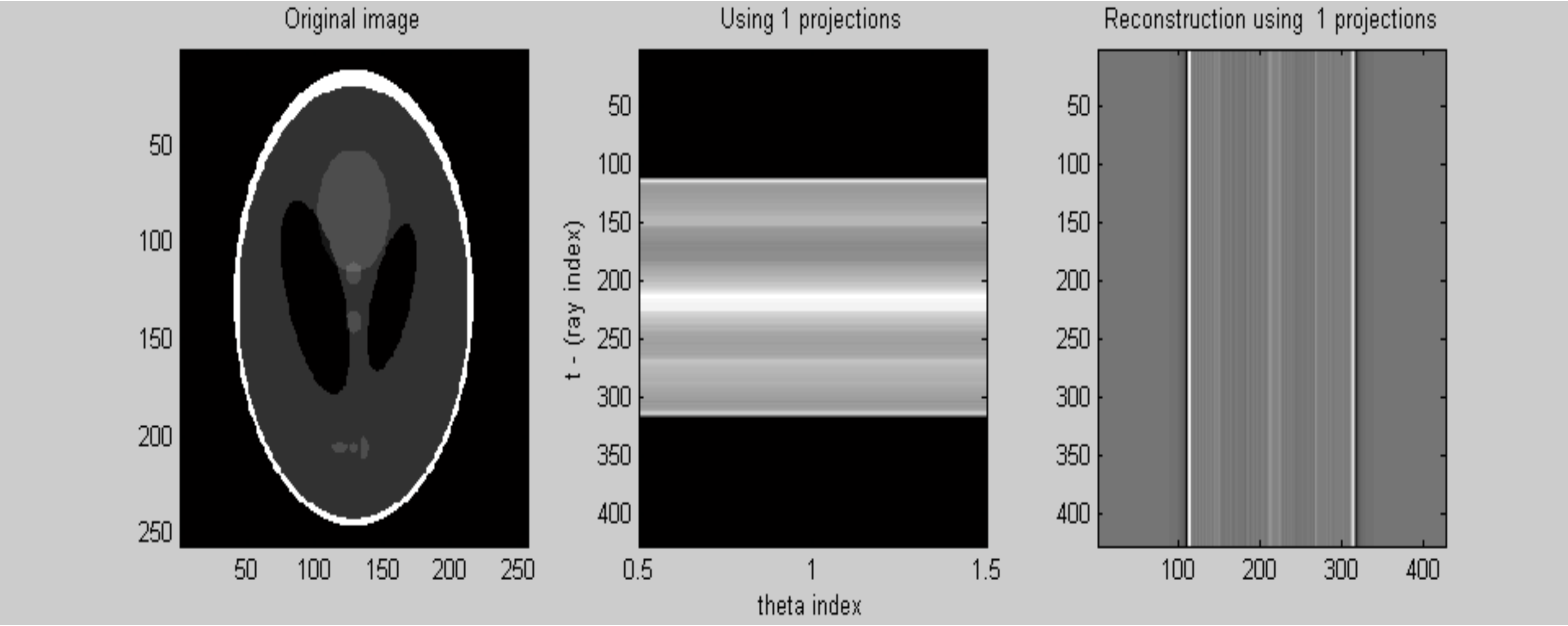


Experiment Two

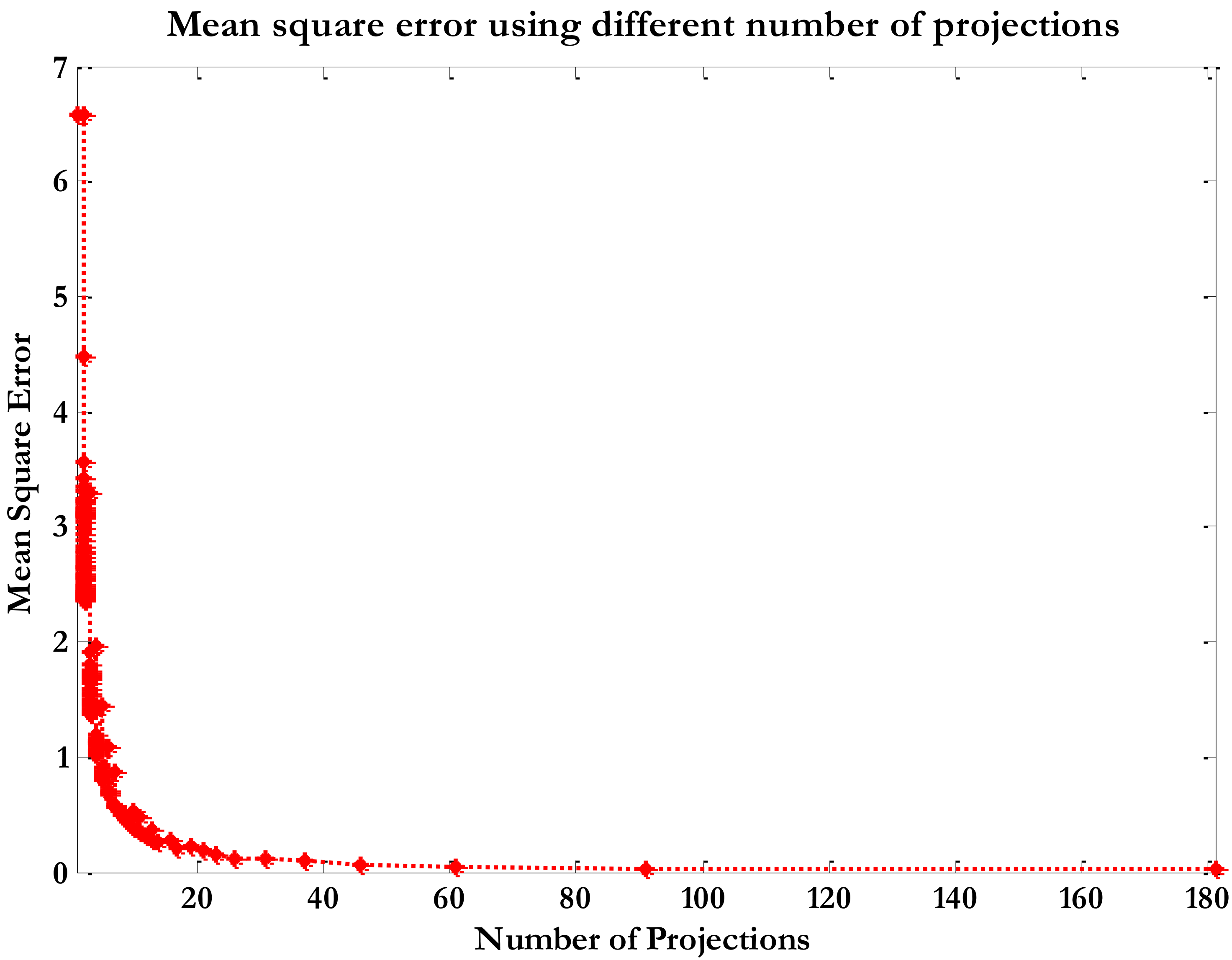
Studying the effect of reconstruction using different number of projections

Reconstruction using different number of projections

Using sinusoidal filter and number of rays equal to image number columns



Quantifying the reconstruction error

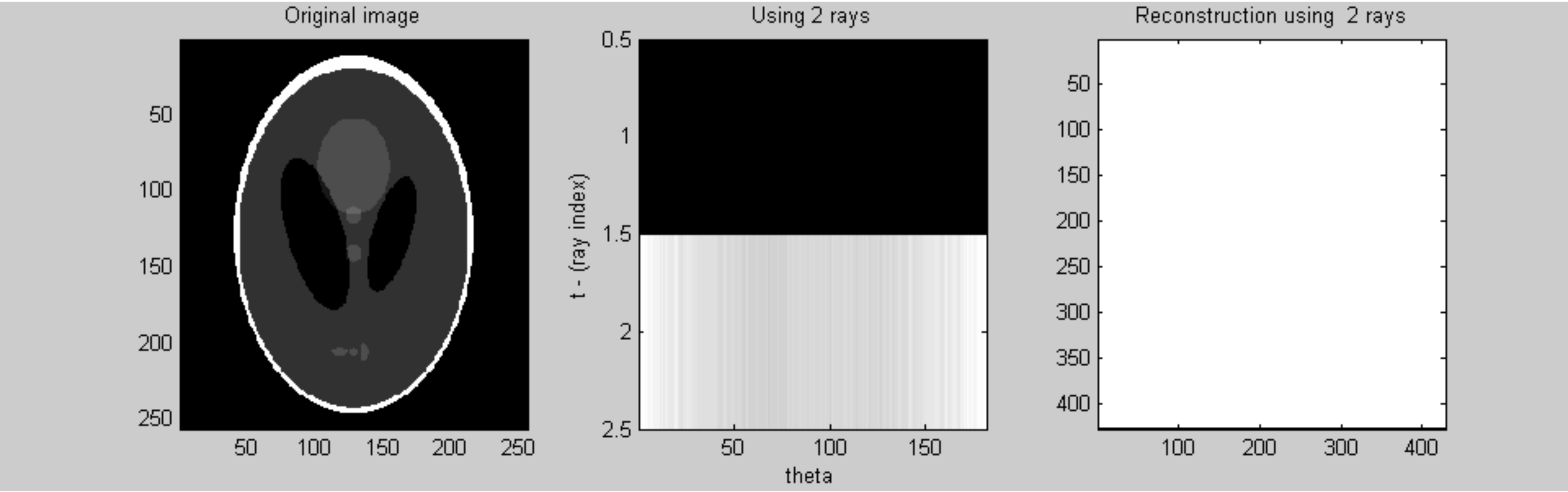


Experiment Three

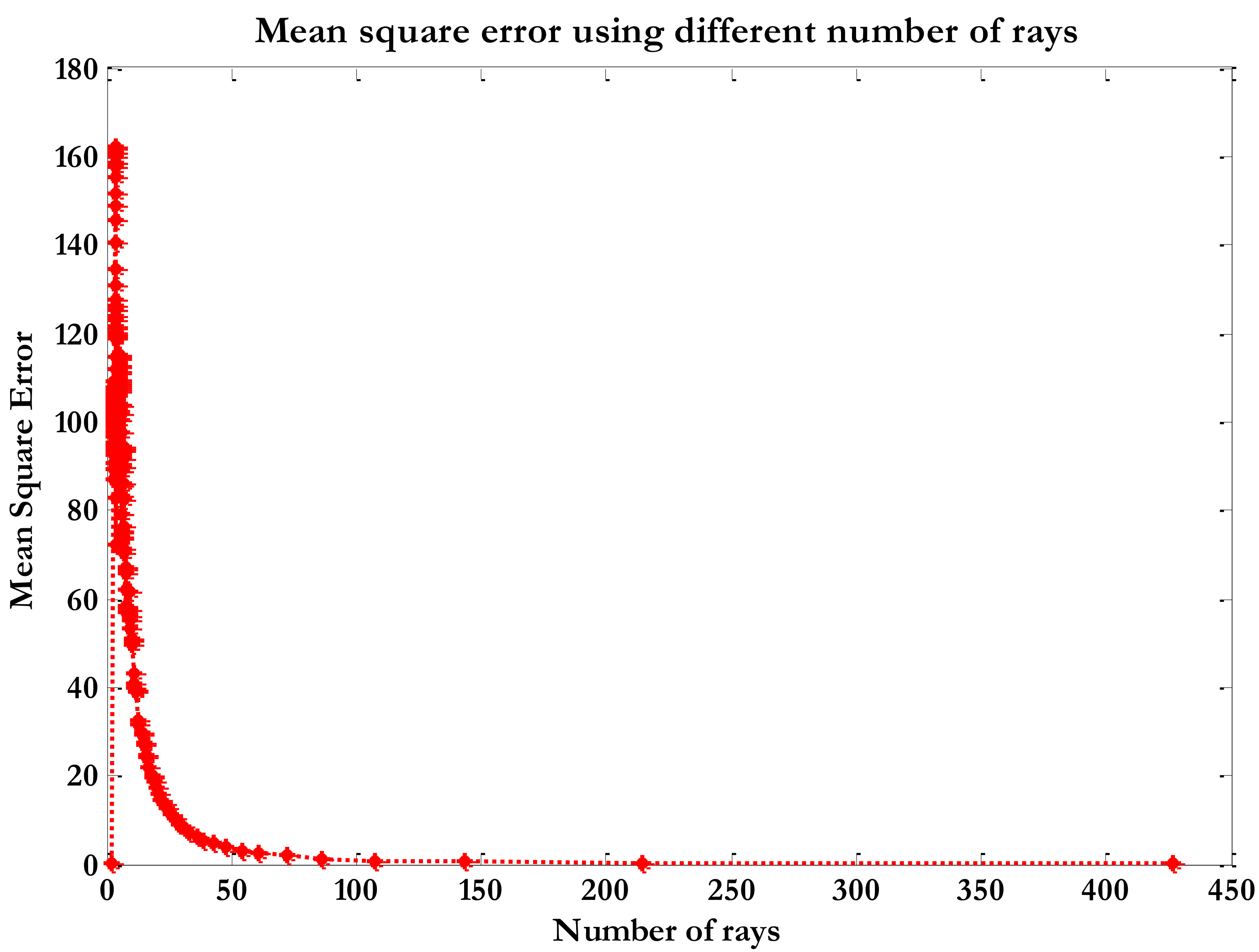
Studying the effect of reconstruction using different number of rays

Reconstruction using different number of rays

Using sinusoidal filter and fixed number of projections



Quantifying the reconstruction error



Thank You