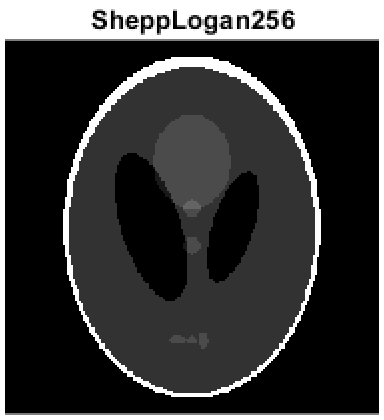
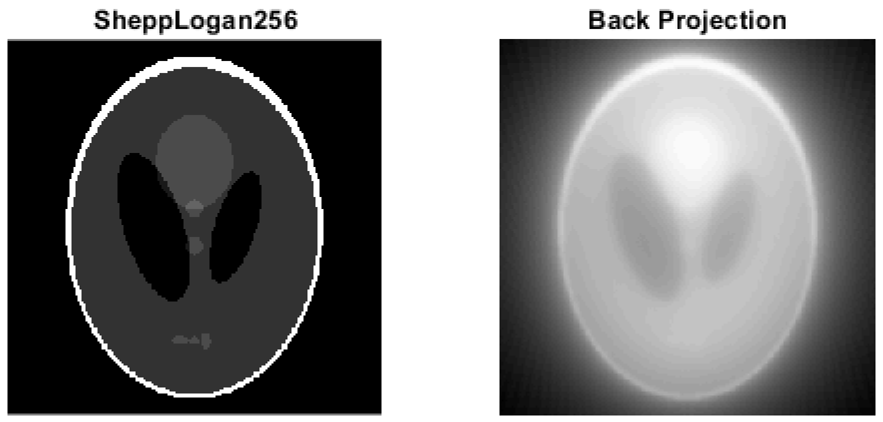
Table of Contents

```
clc;
clear;
tic;

%%%%%%%%%%%%%%% Q1 %%%%%%%%%%%%%%%%%%%%
% reading and displaying the image
f = imread('..\data\SheppLogan256.png');
f = double(f);
figure,subplot(1,2,1), imshow(uint8(f)), daspect([1 1 1]),
 colormap('gray')
title('SheppLogan256')
```



Q1----a-----

```
theta = 0:3:177;
% radon transform of the image
RadonTransform= radon(f,theta);
% inverse radon transform
Back_Projection = iradon(RadonTransform,theta,'linear','none',1,256);
% normalization
mx=max(max(Back_Projection));
mn = min(min(Back_Projection));
bp_min =ones(size(Back_Projection)).*mn;
bp_max =ones(size(Back_Projection)).*mx;
bp = 255*(((Back_Projection)-bp_min)./(bp_max-bp_min));
% displaying the reconstructed image
subplot(1,2,2), imshow(uint8(bp)), daspect([1 1 1]), colormap('gray')
title('Back Projection')
```



Filtered back projection

1 - Ram-Lak

L = w/2

```
FilteredImage = myFilter(RadonTransform, 0.5,1);
```

```
backprojImage_0_5_1 =
 iradon(FilteredImage,theta,'linear','none',1,256);
% normalization
mx=max(max(backprojImage_0_5_1));
mn = min(min(backprojImage_0_5_1));
bp_min =ones(size(backprojImage_0_5_1)).*mn;
bp_max =ones(size(backprojImage_0_5_1)).*mx;
backprojImage_0_5_1 = 255*(((backprojImage_0_5_1)-bp_min)./(bp_max-
bp_min));
% L=W
FilteredImage = myFilter(RadonTransform, 1,1);
backprojImage_1_1 = iradon(FilteredImage,theta,'linear','none',1,256);
% normalization
mx=max(max(backprojImage_1_1));
mn = min(min(backprojImage_1_1));
bp_min =ones(size(backprojImage_1_1)).*mn;
bp_max =ones(size(backprojImage_1_1)).*mx;
backprojImage_1_1 = 255*(((backprojImage_1_1)-bp_min)./(bp_max-
bp_min));
```

2 - Sheep-Logan

L = W/2

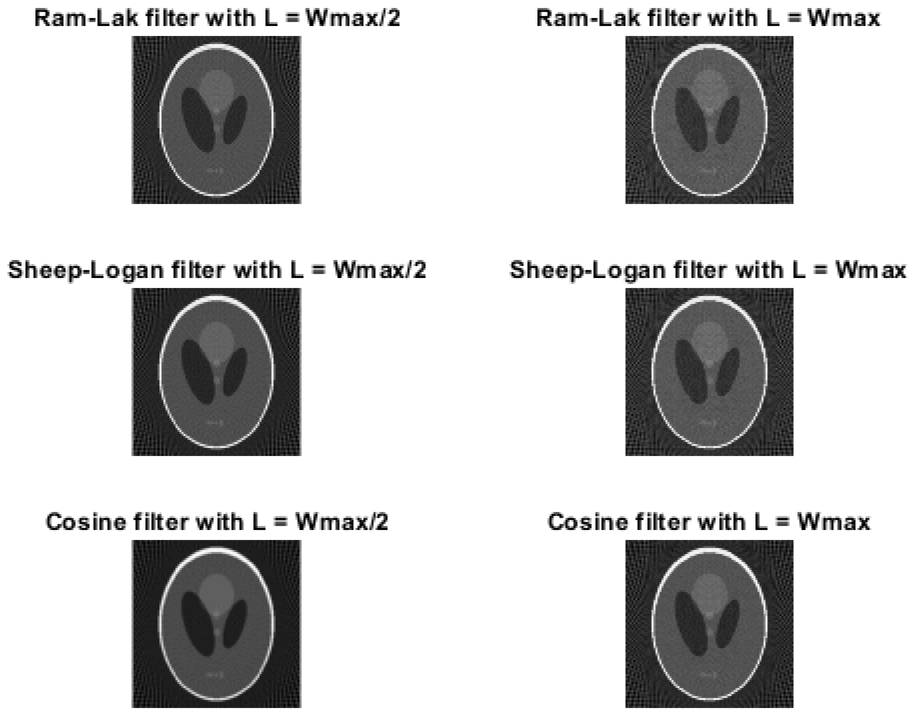
```
FilteredImage = myFilter(RadonTransform, 0.5,2);
backprojImage_0_5_2 =
 iradon(FilteredImage,theta,'linear','none',1,256);
% Normalization
mx=max(max(backprojImage_0_5_2));
mn = min(min(backprojImage_0_5_2));
bp_min =ones(size(backprojImage_0_5_2)).*mn;
bp_max =ones(size(backprojImage_0_5_2)).*mx;
backprojImage_0_5_2 = 255*(((backprojImage_0_5_2)-bp_min)./(bp_max-
bp_min));
% L = W
FilteredImage = myFilter(RadonTransform, 1,2);
backprojImage_1_2 = iradon(FilteredImage,theta,'linear','none',1,256);
% Normalization
mx=max(max(backprojImage_1_2));
mn = min(min(backprojImage_1_2));
bp_min =ones(size(backprojImage_1_2)).*mn;
bp_max =ones(size(backprojImage_1_2)).*mx;
backprojImage_1_2 = 255*(((backprojImage_1_2)-bp_min)./(bp_max-
bp_min));
```

3 - Cosine

L = W/2

```
FilteredImage = myFilter(RadonTransform, 0.5,3);
backprojImage_0_5_3 =
 iradon(FilteredImage,theta,'linear','none',1,256);
% Normalization
```

```
mx=max(max(backprojImage_0_5_3));
mn = min(min(backprojImage_0_5_3));
bp_min =ones(size(backprojImage_0_5_3)).*mn;
bp_max =ones(size(backprojImage_0_5_3)).*mx;
backprojImage_0_5_3 = 255*(((backprojImage_0_5_3)-bp_min)./(bp_max-
bp_min));
% L = W
FilteredImage = myFilter(RadonTransform, 1,3);
backprojImage_1_3 = iradon(FilteredImage,theta,'linear','none',1,256);
% Normalization
mx=max(max(backprojImage_1_3));
mn = min(min(backprojImage_1_3));
bp_min =ones(size(backprojImage_1_3)).*mn;
bp_max =ones(size(backprojImage_1_3)).*mx;
backprojImage_1_3 = 255*(((backprojImage_1_3)-bp_min)./(bp_max-
bp_min));
% Displaying the results
figure
subplot(3,2,1)
imshow(uint8(backprojImage_0_5_1))
title('Ram-Lak filter with L = Wmax/2')
subplot(3,2,2)
imshow(uint8(backprojImage_1_1))
title('Ram-Lak filter with L = Wmax')

subplot(3,2,3)
imshow(uint8(backprojImage_0_5_2))
title('Sheep-Logan filter with L = Wmax/2')
subplot(3,2,4)
imshow(uint8(backprojImage_1_2))
title('Sheep-Logan filter with L = Wmax')

subplot(3,2,5)
imshow(uint8(backprojImage_0_5_3))
title('Cosine filter with L = Wmax/2')
subplot(3,2,6)
imshow(uint8(backprojImage_1_3))
title('Cosine filter with L = Wmax')

%{
The Explanation
? For each filter: the image with L = w_max noisier than the image
 with L = W_max/2
as expected because more high frequencies contribute to producing the
 resulting image when L=w_max.
? The results of cosine filter are smoother than the ones belong to
 other filters, so cosine filter
shows better performance at high frequencies, while Ram-Lak and Shepp-
Logan are better for noise-free
projections (less blur).
%}
```


Ram-Lak filter with L = Wmax/2
Ram-Lak filter with L = Wmax
Sheep-Logan filter with L = Wmax/2
Sheep-Logan filter with L = Wmax
Cosine filter with L = Wmax/2
Cosine filter with L = Wmax

Q1 -------b---------

```
S0 = f;
RadonTransform= radon(S0,theta);
FilteredImage = myFilter(RadonTransform, 1,1);
R0 = iradon(FilteredImage,theta,'linear','none',1,256);
% Normalization
mx=max(max(R0));
mn = min(min(R0));
bp_min =ones(size(R0)).*mn;
bp_max =ones(size(R0)).*mx;
R0 = 255*(((R0)-bp_min)./(bp_max-bp_min));
% convolving the image with a gaussian
mask = fspecial ('gaussian', 11, 1);
S1 = conv2 (f, mask, 'same');
% computing radon transform
RadonTransform= radon(S1,theta);
FilteredImage = myFilter(RadonTransform, 1,1);
% computing inverse radon transform
R1 = iradon(FilteredImage,theta,'linear','none',1,256);
% Normalization
mx=max(max(R1));
mn = min(min(R1));
bp_min =ones(size(R1)).*mn;
bp_max =ones(size(R1)).*mx;
```

```
R1 = 255*(((R1)-bp_min)./(bp_max-bp_min));
% convolving the image with gaussian (sigma = 5)
mask = fspecial ('gaussian', 21, 5);
S5 = conv2 (f, mask, 'same');
% radon transform
RadonTransform= radon(S5,theta);
FilteredImage = myFilter(RadonTransform, 1,1);
% inverse radon transform
R5 = iradon(FilteredImage,theta,'linear','none',1,256);
% normalization
mx=max(max(R5));
mn = min(min(R5));
bp_min =ones(size(R5)).*mn;
bp_max =ones(size(R5)).*mx;
R5 = 255*(((R5)-bp_min)./(bp_max-bp_min));
% computing the errors
rrmse0 = RRMSE(S0,R0);
rrmse1 = RRMSE(S1,R1);
rrmse5 = RRMSE(S5,R5);
% displaying the errors
disp(strcat(['RRMSE of image without blur is ' ,num2str(rrmse0)]))
disp(strcat(['RRMSE with sigma = 1 is ' ,num2str(rrmse1)]))
disp(strcat(['RRMSE with sigma = 5 is ' ,num2str(rrmse5)]))

%{
The Explanation:
? The reconstruction of the image without blur produces the highest
 error
because of the presenting of more high frequencies in the image(noise
 magnification will be large).
? The reconstruction of the blurred image (sigma = 1) produces the
 lowest error,because it has less
 high frequencies than the image without blur, in other words, the
 noise will be magnified less.
 At the same time, it has more high frequencies than the blurred image
 with (sigma = 5), so the
 reconstructed image will be closer to the ground truth.
%}
```

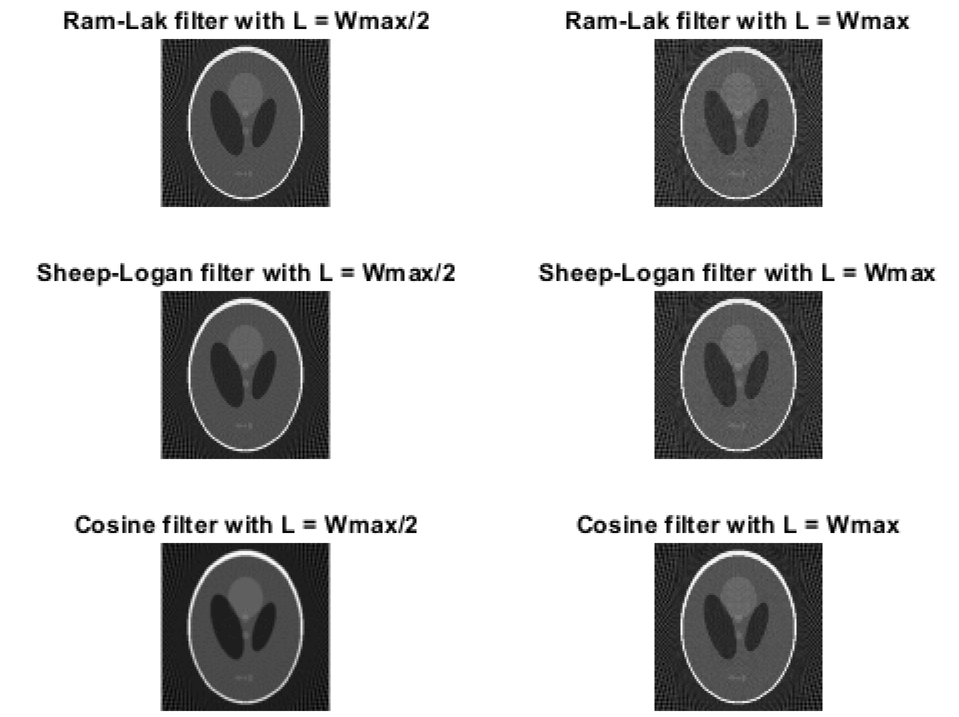
```
RRMSE of image without blur is 0.81561
RRMSE with sigma = 1 is 0.52667
RRMSE with sigma = 5 is 0.59027
```

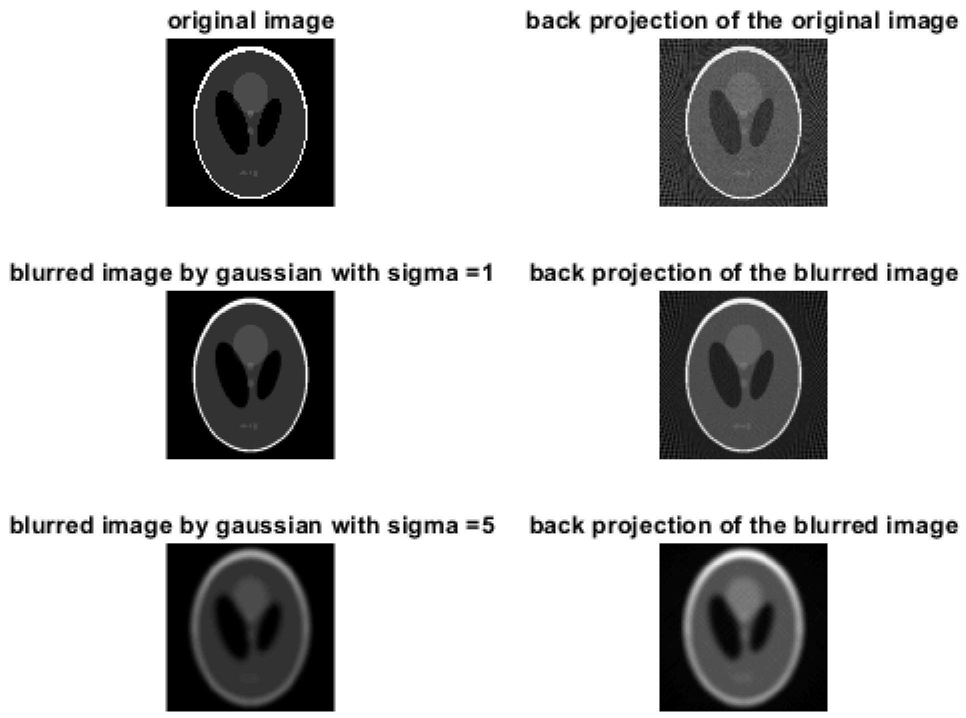
Q1 -------c---------

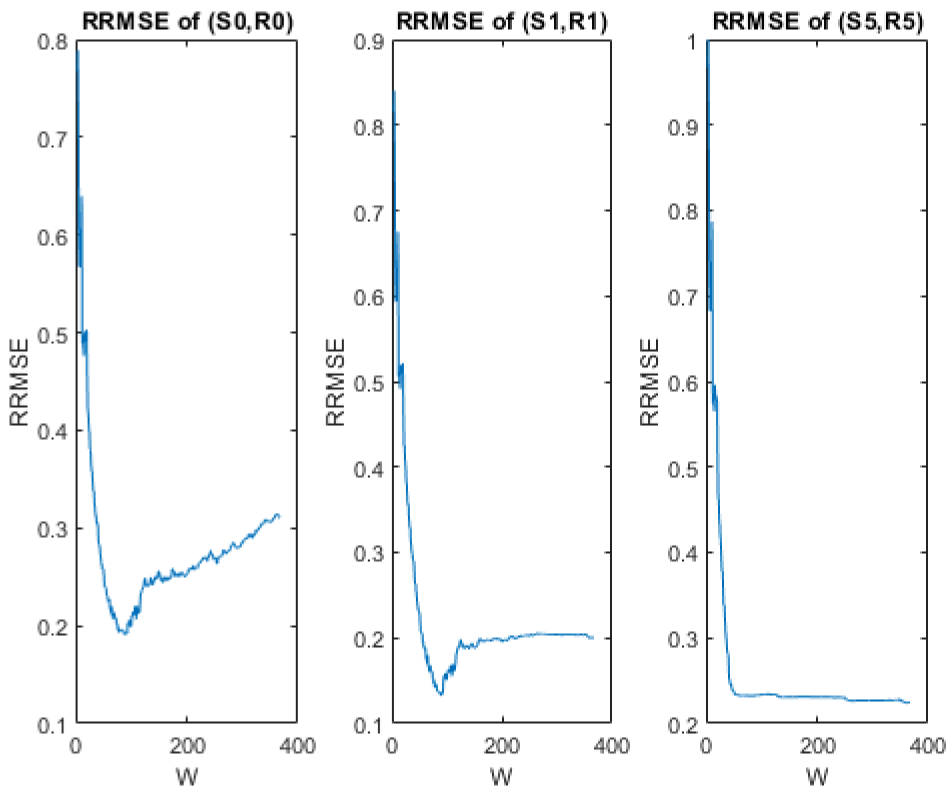
```
M= size(RadonTransform,1);
% the error matrix
Error = zeros(3,M);
for i = 1:M
S0 = f;
% radon transform
RadonTransform= radon(S0,theta);
% L = W*(i/M)
```

```
FilteredImage = myFilter(RadonTransform, i/M,1);
% inverse radon transform
R0 = iradon(FilteredImage,theta,'linear','none',1,256);
% normalization
mx=max(max(R0));
mn = min(min(R0));
bp_min =ones(size(R0)).*mn;
bp_max =ones(size(R0)).*mx;
R0 = 255*(((R0)-bp_min)./(bp_max-bp_min));
% convolving with gaussian (sigma = 1)
mask = fspecial ('gaussian', 11, 1);
S1 = conv2 (f, mask, 'same');
% radon transform
RadonTransform= radon(S1,theta);
FilteredImage = myFilter(RadonTransform, i/M,1);
% inverse radon transform
R1 = iradon(FilteredImage,theta,'linear','none',1,256);
% normalization
mx=max(max(R1));
mn = min(min(R1));
bp_min =ones(size(R1)).*mn;
bp_max =ones(size(R1)).*mx;
R1 = 255*(((R1)-bp_min)./(bp_max-bp_min));
% convolving with gaussian (sigma = 5)
mask = fspecial ('gaussian', 21, 5);
S5 = conv2 (f, mask, 'same');
% radon transform
RadonTransform= radon(S5,theta);
FilteredImage = myFilter(RadonTransform, i/M,1);
% inverse radon transform
R5 = iradon(FilteredImage,theta,'linear','none',1,256);
mx=max(max(R5));
mn = min(min(R5));
bp_min =ones(size(R5)).*mn;
bp_max =ones(size(R5)).*mx;
R5 = 255*(((R5)-bp_min)./(bp_max-bp_min));
% computing the errors
rrmse0 = RRMSE(S0,R0);
rrmse1 = RRMSE(S1,R1);
rrmse5 = RRMSE(S5,R5);
Error(1,i)= rrmse0;
Error(2,i)= rrmse1;
Error(3,i)= rrmse5;
end
% displaying the reconstructed images
figure,
subplot(3,2,1)
imshow(uint8(S0)),daspect([1,1,1]),colormap('gray')
title('original image')
subplot(3,2,2)
imshow(uint8(R0)),daspect([1,1,1]),colormap('gray')
title('back projection of the original image')

subplot(3,2,3)
```

```
imshow(uint8(S1)),daspect([1,1,1]),colormap('gray')
title('blurred image by gaussian with sigma =1')
subplot(3,2,4)
imshow(uint8(R1)),daspect([1,1,1]),colormap('gray')
title('back projection of the blurred image')

subplot(3,2,5)
imshow(uint8(S5)),daspect([1,1,1]),colormap('gray')
title('blurred image by gaussian with sigma =5')
subplot(3,2,6)
imshow(uint8(R5)),daspect([1,1,1]),colormap('gray')
title('back projection of the blurred image')

Max = max(max(Error));
Error_Normalized = Error./Max;

% plotting the errors for different values of L(in other words
 sifferent values of cut-off frequency)
figure,
subplot(1,3,1), plot(Error_Normalized(1,:))
title('RRMSE of (S0,R0)')
xlabel('W')
ylabel('RRMSE')

subplot(1,3,2), plot(Error_Normalized(2,:))
title('RRMSE of (S1,R1)')
xlabel('W')
ylabel('RRMSE')

subplot(1,3,3), plot(Error_Normalized(3,:))
title('RRMSE of (S5,R5)')
xlabel('W')
ylabel('RRMSE')
%{
The Explanation:
? Similar to the results of part(b). R0 has the highest error, and R1
 has the lowest error.
? In the three figures we can see that the error will be minimum at a
 cut-off frequency less than W/2
 then the error will increase for larger frequencies because the noise
 will be magnified more.
%}
```


Ram-Lak filter with L = Wmax/2
Ram-Lak filter with L = Wmax
Sheep-Logan filter with L = Wmax/2
Sheep-Logan filter with L = Wmax
Cosine filter with L = Wmax/2
Cosine filter with L = Wmax


original image
back projection of the original image
blurred image by gaussian with sigma =1
back projection of the blurred image
blurred image by gaussian with sigma =5
back projection of the blurred image



```
toc;
```

```
Elapsed time is 19.140769 seconds.
```

Published with MATLAB® R2019b