Direct Fourier Reconstruction Reference Manual[1]

Antony, Celine, Lewis, Samson, and Steven

[1]Generated by Doxygen 1.7.3: Sun May 1 2011 11:02:42

Contents

Chapter 1

CentralSlice

CT Image Reconstruction using Central Slice Theorem. Refer to the project homepage http://code.google.com/p/centralslice/

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

File Documentation

3.1 add_noise.m File Reference

Add noise to the image.

Functions

- ret add_noise (type image, type SNRdB)
  Add gaussian noise to the image.

3.1.1 Detailed Description

Add noise to the image.

Definition in file add_noise.m.

3.1.2 Function Documentation

3.1.2.1 ret add_noise (type *image*, type *SNRdB*)

Add gaussian noise to the image.

Assumes the exposure is long enough that central limit theorem is valid. (Short exposure is currently not supported)

Signal-to-noise ratio is estimated by the ratio ($\frac{\mu}{\sigma}$), where μ is the expected (perfect) measurement result, σ is the variance of the noise. In dB scale it is $20log_{10}(\frac{A_{\text{noise}}}{A_{\text{noise}}})$

Parameters

image	matrix of the image
SNRdB	signal to noise ratio (1∼inf)

Return values

new_image	new image loaded with noise

Definition at line 18 of file add_noise.m.

3.2 add_noise.m

```
1 %%
2 %! @file
3 % Add noise to the image.
4 %
5
6 %! Add gaussian noise to the image. Assumes the exposure is long enough
     that central limit theorem is valid. (Short exposure is currently
    not supported)
7 %
8 % Signal-to-noise ratio is estimated by the ratio ( @f$\frac{\mu}{\
    sigma} @f$ ), where @f$ \mu @f$ is the expected (perfect)
    measurement result, @f$ \sigma @f$ is the variance of the noise. In
     dB scale it is @f$ 20log_{10}(\frac{A_\textrm{noise}}{A_\textrm{
    noise}}) @f$
9 %
10 % @param image matrix of the image
11 % @param SNRdB signal to noise ratio (1¬inf)
12 % @retval new_image new image loaded with noise
13
14 function new_image = add_noise(image,SNRdB)
15
16 [height width] = size(image);
17
18 % estimate the variance of noise in each measurement of each sensor.
19 SNR = 10^(SNRdB/20);
20 signal_amplitude = mean(mean(image));
21 variance = signal_amplitude / SNR;
22
23 noise = randn(height,width) * variance;
24
25 new_image = image + noise;
```

3.3 apply_fft1.m File Reference

Apply Fast Fourier transform to the Radon image.

Functions

- rets apply_fft1 (type Radon, type DEBUG)

  Apply FFT to each columns of the matrix, then shifts the DC to the DFT centre.

3.3.1 Detailed Description

Apply Fast Fourier transform to the Radon image.

Definition in file apply_fft1.m.

3.3.2 Function Documentation

3.3.2.1 rets apply_fft1 (type *Radon*, type *DEBUG*)

Apply FFT to each columns of the matrix, then shifts the DC to the DFT centre.

The value of axis_omega_s in each row is defined by the formula axis_omega_s = x ∗ (2∗pi / dx) where dx=1.

Parameters

Radon	Radon image. Number of rows must be the power of 2 for FFT to work.
DEBUG	Debug mode. Save the Fourier_Radon image in real part and imaginary part.

Return values

Fourier_Radon	Radon image in Fourier Space
axis_omega_s	value of omega_s in each row

Definition at line 16 of file apply_fft1.m.

3.4 apply_fft1.m

```
1  %%
2  %! @file
3  % Apply Fast Fourier transform to the Radon image
4  %
5
6  %%
7  %! Apply FFT to each columns of the matrix, then shifts the DC to the
       DFT centre. The value of axis_omega_s in each row is defined by the
        formula axis_omega_s = x * (2*pi / dx) where dx=1.
8  % @param Radon Radon image. Number of rows must be the power of 2 for
       FFT to work.
9  % @param DEBUG Debug mode. Save the Fourier_Radon image in real part
       and imaginary part.
10 % @retval Fourier_Radon Radon image in Fourier Space
11 % @retval axis_omega_s value of omega_s in each row
12 function [Fourier_Radon axis_omega_s] = apply_fft1(Radon,DEBUG)
13
14 % Apply FFT to each column of the radon image
15 Fourier_Radon = fft(ifftshift(Radon,1));
16
17 % Label the axis_omega_s,theta axes;
18 [size_omega_s size_theta] = size(Fourier_Radon);
19 dx=1;
```

```
% d_omega = 2*pi / Period; where dx = Period / N
d_omega = 2*pi/(size_omega_s*dx);
axis_omega_s = [0:(size_omega_s/2-1)  (-size_omega_s/2):-1] * (d_omega / dx);

% Shift the DC to the DFT centre
axis_omega_s = fftshift(axis_omega_s);
Fourier_Radon = fftshift(Fourier_Radon,1);

if(DEBUG)
idx = 1:size_theta; %we do not need to know the angles in this function

save_image(idx,axis_omega_s, real(Fourier_Radon),...
    'Fourier transform of Radon Space, Real Part',...
    'slice index','omega_s');   % Save the radon image (real part)
save_image(idx,axis_omega_s, imag(Fourier_Radon),...
    'Fourier transform of Radon Space, Imaginary Part',...
    'slice index','omega_s');   % Save the radon image (imaginary part)

stem(axis_omega_s, abs(Fourier_Radon(:,1)));
axis tight;
title('Slice at angle theta=0 in Fourier Space')
xlabel('omega_s'),ylabel('Absolute Value');     % save the slice at 0 deg
print -dpng Slice_at_angle_theta_0_in_Fourier_Space.png
end
```

3.5 damage_sensors.m File Reference

Disable some X-ray detectors.

Functions

- ret damage_sensors (type Radon, type damage_ratio)

    Disable X-ray detectors in the CT machine.

3.5.1 Detailed Description

Disable some X-ray detectors.

Definition in file damage_sensors.m.

3.5.2 Function Documentation

3.5.2.1 ret damage_sensors (type *Radon*, type *damage_ratio*)

Disable X-ray detectors in the CT machine.

Sensors are chosen at random and fed with null signal during the CAT scanning.

Parameters

Radon	Radon projection image when all sensors works normally.
damage_-ratio	fraction of sensors damaged. =0 none; =1, all.

Return values

damage_radon	new Radon projection image with damaged sensors.

Definition at line 16 of file damage_sensors.m.

3.6 damage_sensors.m

```
1  %%
2  %! @file
3  % Disable some X-ray detectors
4  %
5
6  %%
7  %! Disable X-ray detectors in the CT machine. Sensors are chosen at random and fed with null signal during the CAT scanning.
8  % @param Radon Radon projection image when all sensors works normally.
9  % @param damage_ratio fraction of sensors damaged. =0 none; =1, all.
10 % @retval damage_radon new Radon projection image with damaged sensors.
11 %
12 function damage_radon = damage_sensors(Radon, damage_ratio)
13 damage_radon = Radon;  %copy the Radon image
14
15 if(damage_ratio ≠ 0) % use this function only when necessary
16
17 [numb_sensor scan_angle] = size(Radon); %find the size of the Radon image
18 total_damage = round((numb_sensor - 1)*damage_ratio) + 1; %total number of sensor damage
19
20 sensor_index = round(1 + (numb_sensor-1).*rand(1, total_damage));
21 damage_radon(sensor_index,:) = 0; %specify which sensors need to be nullified
22
23 end
```

3.7 image_crop.m File Reference

Crop the image to specified range.

Functions

- rets image_crop (type image, type axis_xy, type xy_min, type xy_max, type DEBUG)

3.7.1 Detailed Description

Crop the image to specified range.

Definition in file image_crop.m.

3.7.2 Function Documentation

3.7.2.1 rets image_crop (type *image*, type *axis_xy*, type *xy_min*, type *xy_max*, type *DEBUG*)

Parameters

image	Image to be cropped
axis_xy	original range of axes in the image
xy_min	top left hand corner of the crop box
xy_max	bottom right hand corner of the crop box
DEBUG	Debug mode. If DEBUG=1, save the preview image.

Return values

new_image	cropped image
new_axis_xy	new axes range

Definition at line 18 of file image_crop.m.

3.8 image_crop.m

```
1  %%
2  %! @file
3  % Crop the image to specified range.
4  %
5
6  %%
7  %! @param image Image to be cropped
8  % @param axis_xy original range of axes in the image
9  % @param xy_min top left hand corner of the crop box
10 % @param xy_max bottom right hand corner of the crop box
11 % @param DEBUG Debug mode. If DEBUG=1, save the preview image.
12 % @retval new_image cropped image
13 % @retval new_axis_xy new axes range
14 function [new_image new_axis_xy] = image_crop(image,axis_xy,xy_min,
       xy_max,DEBUG)
15
16 % Make axis_xy a row vector
17 if( size(axis_xy,2) ==1)
18 axis_xy = axis_xy';
19 end
20
21 % define crop box
22 [row_idx col_idx val] = find( axis_xy≥xy_min & axis_xy≤xy_max );
23 idx_begin = col_idx(1);
24 idx_end = col_idx(length(row_idx));
25
```

```
new_image = image(idx_begin:idx_end,idx_begin:idx_end);
new_axis_xy = axis_xy(idx_begin:idx_end);

if(DEBUG)
figure
imagesc(axis_xy,axis_xy,real(image)),colormap(gray),colorbar
xlim([xy_min xy_max]),ylim([xy_min xy_max])
title('Cropbox preview'),xlabel('x'),ylabel('y')
print -dpng 'cropbox_preview.png'
end
```

3.9 inverse_Fourier_2D.m File Reference

Apply inverse Fourier 2D transform to the image.

Functions

- rets inverse_Fourier_2D (type Fourier_2D, type omega_xy, type DEBUG)

3.9.1 Detailed Description

Apply inverse Fourier 2D transform to the image.

Definition in file inverse_Fourier_2D.m.

3.9.2 Function Documentation

3.9.2.1 rets inverse_Fourier_2D (type *Fourier_2D*, type *omega_xy*, type *DEBUG*)

Parameters

Fourier_2D	matrix of the interpolated 2D Fourier space
omega_xy	value of omega_x (or omega_y) in each column (or row) of matrix `Fourier_2D`.
DEBUG	Debug mode. If DEBUG=1, save the image of the reconstructed image in imaginary part.

Return values

Final_image	Inverse Fourier transform of matrix `Fourier_2D`.
axis_xy	value of x (or y) in each column (or row) of `Final_image`

Definition at line 16 of file inverse_Fourier_2D.m.

3.10 inverse_Fourier_2D.m

```
%%
%! @file
% Apply inverse Fourier 2D transform to the image
%

%%
%! @param Fourier_2D matrix of the interpolated 2D Fourier space
% @param omega_xy value of omega_x (or omega_y) in each column (or row) of matrix \c Fourier_2D.
% @param DEBUG Debug mode. If DEBUG=1, save the image of the reconstructed image in imaginary part.
% @retval Final_image Inverse Fourier transform of matrix \c Fourier_2D.
% @retval axis_xy value of x (or y) in each column (or row) of \c Final_image
function [Final_image,axis_xy] = inverse_Fourier_2D(Fourier_2D,omega_xy,DEBUG)

% Shift the DC to the left top corner
shifted_Fourier_2D = ifftshift(Fourier_2D);

% Apply inverse 2D Fourier transform
shifted_Final_image = ifft2(shifted_Fourier_2D);

% Shift the DC back to the centre
Final_image = fftshift(shifted_Final_image);

%Label the axes x and y
size_omega = length(omega_xy);
d_omega = mean(diff(omega_xy));
dx = 2*pi/(d_omega*size_omega);
N_image = 65;
axis_xy = omega_xy * (dx / d_omega);

if(DEBUG)
save_image(axis_xy,axis_xy,imag(Final_image),...
    'Reconstructed Image, imaginary part','x','y');
end
```

3.11 main.m File Reference

Main process of the simulation.

Functions

- void main (type shape, type N_image, type N_theta, type SNRdB, type interp_m, type oversampling_ratio, type damage_ratio, type DEBUG)

    Main process of the simulation.

3.11.1 Detailed Description

Main process of the simulation.

Definition in file main.m.

3.11.2 Function Documentation

3.11.2.1 void main (type *shape*, type *N_image*, type *N_theta*, type *SNRdB*, type *interp_m*, type *oversampling_ratio*, type *damage_ratio*, type *DEBUG*)

Main process of the simulation.

This script generates a radon projection image from a selected phantom. Then 1D Fourier transform is applied to each projection angle. The result is then interpolated onto the cartesian plane according to Central slice theorem. Lastly inverse 2D Fourier transform is applied to reproduce the image.

Parameters

shape	shape of the phantom. Can be 'Shepp-Logan', 'Modified Shepp-Logan', 'dot', 'square', or 'stripe'
N_image	mininium size of the phantom image (in pixels)
N_theta	Number of slices in Radon scan from 0deg to 180deg (excluding 180deg)
SNRdB	Signal to Noise Ratio in log scale.
interp_m	method of interpolation. Can be 'nearest','linear' or 'cubic'
oversampling_ratio	oversampling ratio. Increase the Nyquist frequency to reduce aliasing. =1, none; >1 oversampling.
damage_ratio	fraction of sensors damaged. =0, none; =1, all damaged.
DEBUG	mode. If set to 1, many more figures are printed out for debugging process.

Definition at line 33 of file main.m.

3.12 main.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%! @mainpage CentralSlice
% CT Image Reconstruction using Central Slice Theorem.
%
% Refer to the project homepage http://code.google.com/p/centralslice/

%! @file
% Main process of the simulation.

%! @example start_simulation.m
%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%! Main process of the simulation.
% This script generates a radon projection image from a selected
    phantom.
% Then 1D Fourier transform is applied to each projection angle. The
    result is then interpolated onto the cartesian plane according to
```

```
Central slice theorem. Lastly inverse 2D Fourier transform is applied to reproduce the image.
% @param shape shape of the phantom. Can be 'Shepp-Logan', 'Modified Shepp-Logan', 'dot', 'square', or 'stripe'
% @param N_image mininium size of the phantom image (in pixels)
% @param N_theta Number of slices in Radon scan from 0deg to 180deg (excluding 180deg)
% @param SNRdB Signal to Noise Ratio in log scale.
% @param interp_m method of interpolation. Can be 'nearest','linear' or 'cubic'
% @param oversampling_ratio oversampling ratio. Increase the Nyquist frequency to reduce aliasing. =1, none; >1 oversampling.
% @param damage_ratio fraction of sensors damaged. =0, none; =1, all damaged.
% @param DEBUG mode. If set to 1, many more figures are printed out for debugging process.
function main(shape,N_image,N_theta,SNRdB,interp_m,oversampling_ratio,damage_ratio,DEBUG)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% MAKE A PHANTOM AND APPLY RADON TRANSFROMATION

Phantom = make_phantom(shape,floor(N_image/sqrt(2)));   % Make a phantom.

axis_xy = linspace(-N_image/2,N_image/2,N_image);
save_image(axis_xy,axis_xy,Phantom,...
    'Phantom','x','y');     % Save the phantom image

% Angles for Radon Projection.
% It should be from 0deg to 180deg. The last angular sample normally is smaller than 180deg.
d_theta = 180 / N_theta;
THETA = linspace(0,180-d_theta,N_theta);

% Workaround a bug in Matlab function RADON, which assumes the y-axis points downwards instead of pointing upward
Phantom_flipy = flipud(Phantom);
Radon = radon(Phantom_flipy,THETA);     % Apply Radon transform.

no_of_sensors = size(Radon,1)

Radon = add_noise(Radon,SNRdB);     % Add noise to the image

%% Sensor damage: nullify some sensors
damage_radon = damage_sensors(Radon, damage_ratio);

%% Zeropadding: expand the matrix to power of 2 before doing FFT
[Radon2 axis_s] = zeropad(damage_radon);

save_image(THETA,axis_s,Radon2,...
    'Radon Projection','theta','s');    % Save the radon image


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% 1D FOURIER TRANSFORM
[Fourier_Radon omega_s] = apply_fft1(Radon2,DEBUG);

save_image(THETA, omega_s, abs(Fourier_Radon),...
    'Fourier transform of Radon Space, Absolute Value',...
    'theta','omega_s');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% INTERPOLATION: Map slices from polar coordinates to rectangular
    coordinates
[Fourier_2D omega_xy] = polar_to_rect(THETA,omega_s,Fourier_Radon,
    N_image*oversampling_ratio,interp_m,DEBUG);

save_image(omega_xy,omega_xy,log(abs(Fourier_2D)),'Interpolated Fourier
     Space (log scale)','omega_x','omega_y')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% INVERSE 2D FOURIER TRANSFORM
[Reconstructed_image axis_xy_2] = inverse_Fourier_2D(Fourier_2D,
    omega_xy,DEBUG);

% Crop image
xy_min = axis_xy(1);
xy_max = axis_xy(length(axis_xy));
[Crop_image new_axis_xy] = image_crop(Reconstructed_image,axis_xy_2,
    xy_min,xy_max,DEBUG);

save_image(new_axis_xy,new_axis_xy,real(Crop_image),...
    'Reconstructed Image','x','y');
```

3.13 make_phantom.m File Reference

Make a phantom.

Functions

- ret make_phantom (type shape, type N)

  Construct a matrix of a selected phantom.

3.13.1 Detailed Description

Make a phantom.

Definition in file make_phantom.m.

3.13.2 Function Documentation

3.13.2.1 ret make_phantom (type *shape*, type *N*)

Construct a matrix of a selected phantom.

Parameters

shape	Type of the phantom. Can be 'Shepp-Logan', 'Modified Shepp-Logan', 'dot', 'square', 'stripe' or 'offcentre dot'
N	Size of the matrix

Return values

P	Matrix of the phantom image

Definition at line 16 of file make_phantom.m.

3.14 make_phantom.m

```
1  %%
2  %! @file
3  % Make a phantom.
4  %
5
6  %%
7  %! Construct a matrix of a selected phantom.
8  % @param shape Type of the phantom. Can be 'Shepp-Logan', 'Modified Shepp-Logan', 'dot', 'square', 'stripe' or 'offcentre dot'
9  % @param N Size of the matrix
10 % @retval P Matrix of the phantom image
11 %
12 function P = make_phantom(shape,N)
13
14     % T width of the square pulse or the stripe. T must be an even number.
15     T=round(N/4)*2;
16     R=T/2;
17
18 switch shape
19   case {'Shepp-Logan','Modified Shepp-Logan'}
20     % Modified Shepp-Logan' gives better visual perception than 'Shepp-Logan'
21     P = phantom(shape,N);
22     P = flipud(P);
23   case {'dot'}
24     R=4;
25     x=linspace(-N/2,N/2,N); y=x; [X, Y]=meshgrid(x,y); P=(X.^2 +Y.^2 ≤ R^2);
26   case {'square'}
27     P=[zeros(N,(N-T)/2) ones(N,T) zeros(N,(N-T)/2)];
28     P=P'*P;
29   case {'stripe'}
30     P=[zeros(N,(N-T)/2) ones(N,T) zeros(N,(N-T)/2)];
31   case {'circle'}
32     x=linspace(-N/2,N/2,N); y=x; [X, Y]=meshgrid(x,y); P=(X.^2 +Y.^2 ≤ R^2);
33   case {'offcentre dot'}
34 % make a off-centre dot
35     P=zeros(N);
36     idx = round(N/4);
37     P(idx:idx+1,idx:idx+1)=1;
38 end
```

3.15 polar_to_rect.m File Reference

Map polar coordinates to rectangular coordinates.

Functions

- rets polar_to_rect (type theta, type omega_s, type Fourier_Radon, type N_image, type interp_m, type DEBUG)

3.15.1 Detailed Description

Map polar coordinates to rectangular coordinates.

Definition in file polar_to_rect.m.

3.15.2 Function Documentation

3.15.2.1 rets polar_to_rect (type *theta*, type *omega_s*, type *Fourier_Radon*, type *N_image*, type *interp_m*, type *DEBUG*)

Parameters

theta	angles of Radon transform. Values of theta in each columns of Fourier_-Radon
omega_s	values of omega_s in each rows of Fourier_Radon
Fourier_-Radon	Matrix of Fourier transformed Radon image
N_image	minimium size of the image
interp_m	method of interpolation, can be 'nearest','linear' or 'cubic'
DEBUG	Debug mode. If DEBUG=1, surface of Fourier_Radon in polar coordinates and in rectangular coordinates will be saved.

Return values

Fourier_2D	Matrix of the mapped Fourier space. By central slice theorem, this is equivalent to the 2D Fourier transform of the original image.
axis_omega_xy	values of omega_x (or omega_y) in the columns (or rows) of Fourier_2D.

Definition at line 20 of file polar_to_rect.m.

3.16 polar_to_rect.m

```
1 %%
2 %! @file
3 % Map polar coordinates to rectangular coordinates
4 %
5
6 %%
7 %!
8 % @param theta angles of Radon transform. Values of theta in each
    columns of Fourier_Radon
9 % @param omega_s values of omega_s in each rows of Fourier_Radon
```

```
% @param Fourier_Radon Matrix of Fourier transformed Radon image
% @param N_image minimium size of the image
% @param interp_m method of interpolation, can be 'nearest','linear' or 'cubic'
% @param DEBUG Debug mode. If DEBUG=1, surface of Fourier_Radon in polar coordinates and in rectangular coordinates will be saved.
% @retval Fourier_2D Matrix of the mapped Fourier space. By central slice theorem, this is equivalent to the 2D Fourier transform of the original image.
% @retval axis_omega_xy values of omega_x (or omega_y) in the columns ( or rows) of Fourier_2D.
function [Fourier_2D axis_omega_xy] = polar_to_rect(theta,omega_s, Fourier_Radon,N_image,interp_m,DEBUG)
%% Check correctness of input data
[size_omega_s size_theta] = size(Fourier_Radon);
length_theta = length(theta);
length_omega_s = length(omega_s);

if(length_theta ≠ size_theta)
 error('size of theta does not match with the size of Fourier_Radon!')
elseif(length_omega_s ≠ size_omega_s)
 error('size of omega_s does not match with the size of Fourier_Radon!' )
end

%% Preparations
% entend the range of Fourier Radon space so that value at theta=0 and theta =180 can be interpolated
% Disabled so that the effect of scan range could be investigated
%Extended_Fourier_Radon = horzcat( Fourier_Radon, Fourier_Radon(:, size_theta) );
%theta = [theta 180];

% Label each elements in the matrix Fourier_Radon with the corresponding theta and omega_s:
[THETA OMEGA_S] = meshgrid(theta,omega_s);

%Define the desired scale of the rectangular coordinates
x = linspace(-N_image/2,N_image/2,N_image);
y = x;
dx=1;
d_omega = 2*pi/N_image;
omega_x = x * (d_omega / dx);
omega_y=omega_x;

% Label each (omega_x, omega_y) to (omega_s, theta)
[OMEGA_X OMEGA_Y] = meshgrid(omega_x, omega_y);
[THETA_I OMEGA_SI] = cart2pol(OMEGA_X,OMEGA_Y);

% map from {theta,omega_s : [-pi,pi],[0,inf]} to {theta,omega_s : [0, 180],[-inf,inf]}
OMEGA_SI = OMEGA_SI .* sign(THETA_I);
THETA_I = mod( THETA_I * (180/pi), 180);

%% Apply interpolation
% disable extrapolation: everything outside the defined space is set to zero.
Fourier_2D = interp2(THETA,OMEGA_S,Fourier_Radon,...
    THETA_I,OMEGA_SI,interp_m,0);
axis_omega_xy = omega_x;

%% DEBUG: Print surface of Fourier_Radon before and after interpolation
```

```
if(DEBUG)
[WX WY] = pol2cart(THETA,OMEGA_S);
figure
surf(WX,WY,abs(Fourier_Radon),...
    'edgecolor','none')
colormap(jet),colorbar
title('Surface of Fourier transformed Radon space, before interpolation
    ')
xlabel('omega_x'),ylabel('omega_y')
print -dpng '
    Surface_of_Fourier_transformed_Radon_space_before_interpolation.png
    '

figure
surf(omega_x,omega_y,abs(Fourier_2D),...
        'edgecolor','none')
colormap(jet),colorbar
title('Surface of Fourier transformed Radon space, after interpolation'
    )
xlabel('omega_x'),ylabel('omega_y')
print -dpng '
    Surface_of_Fourier_transformed_Radon_space_after_interpolation.png'
end
```

3.17 save_image.m File Reference

Save the gray-scale representation of the image.

Functions

- void save_image (type x, type y, type Z, type Title, type Xlabel, type Ylabel)
  Save the gray-scale representation of the image.

3.17.1 Detailed Description

Save the gray-scale representation of the image.

Definition in file save_image.m.

3.17.2 Function Documentation

3.17.2.1 void save_image (type *x*, type *y*, type *Z*, type *Title*, type *Xlabel*, type *Ylabel*)

Save the gray-scale representation of the image.

Parameters

x	value of x in each column
x	value of y in each row

Z	matrix of the image
Title	title of the graph
Xlabel	label of the x-axis
Ylabel	label of the y-axis

Definition at line 17 of file save_image.m.

3.18 save_image.m

```
%%
%! @file
% Save the gray-scale representation of the image.

%! Save the gray-scale representation of the image.
% @param x value of x in each column
% @param x value of y in each row
% @param Z matrix of the image
% @param Title title of the graph
% @param Xlabel label of the x-axis
% @param Ylabel label of the y-axis

function save_image(x,y,Z,Title,Xlabel,Ylabel)
figure
imagesc(x,y,Z)
% flip the y-axis to pointing upward
set(gca,'YDir','normal')
colormap(gray),colorbar
title(Title)
if(nargin > 2)
  xlabel(Xlabel),ylabel(Ylabel)
end
print('-dpng',strcat(strrep(Title,' ','_'),'.png'))
```

3.19 start_simulation.m File Reference

Program Initiation.

Variables

- DEBUG = 1

  Debug the program.

- shape

  Shape of the phantom.

3.19.1 Detailed Description

Program Initiation. Defines several important variables before starting the main process.

Definition in file start_simulation.m.

3.19.2 Variable Documentation

3.19.2.1 DEBUG = 1

Debug the program.

When DEBUG=1, extra figures will be saved in current directory which can be used for debugging process.

Definition at line 15 of file start_simulation.m.

3.19.2.2 shape

Initial value:

```
'Modified Shepp-Logan'
```

Shape of the phantom.

Can be 'Shepp-Logan', 'Modified Shepp-Logan', 'dot', 'square', 'stripe' or 'offcentre dot'.

Definition at line 23 of file start_simulation.m.

3.20 start_simulation.m

```
1  %%
2  %! @file
3  % Program Initiation.
4  % Defines several important variables before starting the main process.
5
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7  %% DEBUG MODE
8
9  %! Debug the program.
10 % When DEBUG=1, extra figures will be saved in current directory which
       can be used for debugging process.
11 DEBUG = 1;
12
13 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
14 %% Parameters
15
16 %! Shape of the phantom. Can be 'Shepp-Logan', 'Modified Shepp-Logan',
       'dot', 'square', 'stripe' or 'offcentre dot'.
```

```
shape='Modified Shepp-Logan'

%! Size of the phantom.
% This specifies the number of rows and columns in the matrix of
    Phantom.
% N_image is suggested to be an odd number.
N_image = 371

%! Number of slices in Radon scan from 0deg to 180deg (excluding 180deg
    )
N_theta = 180

%number of sensors damaged, damage_ratio varies from 0 to 1
damage_ratio = 0

%! Signal to noise ratio, in deciBell (dB)
SNRdB = inf

%! Interpolation method
interp_m = 'linear'

%! Oversampling_ratio oversampling ratio. Increase the Nyquist
    frequency to reduce aliasing. =1, none; >1 oversampling.
oversampling_ratio=1

%! Zeropadding ratio. Avoid overlapping of artefacts to the phantom
    after applying inverse Fourier transform. 1 ¬ mininal; >1 ¬
    zeropadding.
%zeropadding_ratio=1;

main(shape,N_image,N_theta,SNRdB,interp_m,oversampling_ratio,
    damage_ratio,DEBUG);
```

3.21 zeropad.m File Reference

Expand the matrix to power of 2 before doing FFT.

Functions

- rets zeropad (type Radon)

    Zeropad each column to preprare for FFT.

3.21.1 Detailed Description

Expand the matrix to power of 2 before doing FFT.

Definition in file zeropad.m.

3.21.2 Function Documentation

3.21.2.1 rets zeropad (type *Radon*)

Zeropad each column to preprare for FFT.

Expand the length of each column to the power of 2. The value of s in each row is also computed.

Parameters

Radon	matrix of Radon image

Return values

Radon2	expaned matrix of Radon image
axis_s	value of s in each row

Definition at line 15 of file zeropad.m.

3.22 zeropad.m

```
1 %%
2 %! @file
3 % Expand the matrix to power of 2 before doing FFT.
4 %
5
6 %%
7 %! Zeropad each column to preprare for FFT. Expand the length of each column to the power of 2. The value of s in each row is also computed.
8 % @param Radon matrix of Radon image
9 % @retval Radon2 expaned matrix of Radon image
10 % @retval axis_s value of s in each row
11 function [Radon2 axis_s] = zeropad(Radon)
12
13 [size_s size_theta] = size(Radon);
14 next_power_of_2 = pow2(nextpow2(size_s));
15
16 % Shift the DC to the left
17 shifted_Radon = ifftshift(Radon,1);
18
19 % Estimate the size of zeropad required
20 size_zeropad = next_power_of_2 - size_s;
21 zeropad = zeros(size_zeropad,size_theta);
22
23 if(size_s & 2) % if length is an odd number, the 'middle' is between (size_s + 1)/2 and (size_s + 1)/2+1
24     mid_position = (size_s + 1)/2;
25 else    % if length is an even number, the 'middle' is between size_s/2 and size_s/2+1
26     mid_position = size_s / 2;
27 end
28
29 % Add zeros to the middle of the shifted signal
```

```
shifted_Radon2 = vertcat(shifted_Radon(1:mid_position,:),zeropad,
    shifted_Radon((mid_position+1):size_s,:));

% Shift the DC back to the centre
Radon2 = fftshift(shifted_Radon2,1);

% label the new s axis
axis_s = linspace(-next_power_of_2/2,next_power_of_2/2,next_power_of_2)
    ;
```

Chapter 4

Example Documentation

4.1 start_simulation.m

Refer to Section 3.19 on page 20 for detailed explanation.

```
%%
%! @file
% Program Initiation.
% Defines several important variables before starting the main process.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DEBUG MODE

%! Debug the program.
% When DEBUG=1, extra figures will be saved in current directory which can be used for debugging process.
DEBUG = 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Parameters

%! Shape of the phantom. Can be 'Shepp-Logan', 'Modified Shepp-Logan', 'dot', 'square', 'stripe' or 'offcentre dot'.
shape='Modified Shepp-Logan'

%! Size of the phantom.
% This specifies the number of rows and columns in the matrix of Phantom.
% N_image is suggested to be an odd number.
N_image = 371

%! Number of slices in Radon scan from 0deg to 180deg (excluding 180deg)
N_theta = 180

%number of sensors damaged, damage_ratio varies from 0 to 1
damage_ratio = 0

%! Signal to noise ratio, in deciBell (dB)
SNRdB = inf
```

```

%! Interpolation method
interp_m = 'linear'

%! Oversampling_ratio oversampling ratio. Increase the Nyquist frequency to reduce aliasing. =1, none; >1 oversampling.
oversampling_ratio=1

%! Zeropadding ratio. Avoid overlapping of artefacts to the phantom after applying inverse Fourier transform. 1 ¬ mininal; >1 ¬ zeropadding.
%zeropadding_ratio=1;

main(shape,N_image,N_theta,SNRdB,interp_m,oversampling_ratio,damage_ratio,DEBUG);
```

Index