A REPORT

ON

IMPLEMENTATION OF 3D CONE BEAM RECONSTRUCTION ALGORITHM

BY

George Joseph 2013B5A7112G

AT

Indira Gandhi Centre of Atomic Research, Kalpakkam

A Practice School-I Station

BIRLA INSITUTE OF TECHNOLOGY & SCIENCE

PILANI

August, 2015

A REPORT

ON

IMPLEMENTATION OF 3D CONE BEAM RECONSTRUCTION ALGORITHM

BY

George Joseph 2013B5A7112G MSc (Hons) - Physics & BE (Hons) - CS

Prepared in Partial Fulfillment of the Practice School-I Course

AT

Indira Gandhi Centre of Atomic Research, Kalpakkam

A Practice School-I Station

BIRLA INSITUTE OF TECHNOLOGY & SCIENCE

PILANI

August, 2015

ACKNOWLEDGEMENT

I would like to show my sincere gratitude to the Dr. P.R. Vasudeva Rao, Director, Indira Gandhi Center for Atomic Research and Prof. Niranjan Swain, Dean Practice School Unit of Birla Institute of Technology and Science, Pilani for providing me with the chance to work in this esteemed research center of the Indian Government during this Practice School -1 Course of Birla Institute of Technology and Science, Pilani.

I would also like to thank Dr. B.P.C Rao, Head, NDED for giving me the opportunity to work on this project “IMPLEMENTATION OF 3D CONE BEAM RECONSTRUCTION ALGORITHM” and Mr. K. Arunmuthu and Dr. T. Saravanan, the project guides who helped and guided me in various aspects to complete the project.

I would also like to express my thanks to Dr. R. Parameshwaran, PS instructor, Indira Gandhi Centre for Atomic Research, who is handling our complete PS activities, for allocating this project.

BIRLA INSITUTE OF TECHNOLOGY AND SCIENCE
PILANI (RAJASTHAN)
Practice School Division

Station- Indira Gandhi Centre for Atomic Research **Centre-**Kalpakkam

Duration- 55 days **Date of Start**- 22 May, 2015

Date of Submission: 15 July, 2015

Title of the Project: IMPLEMENTATION OF 3D CONE BEAM RECONSTRUCTION ALGORITHM

Name: George Joseph **ID**: 2013B5A7112G

Name and Designation of Expert: Mr K Arunmuthu, Scientist, RTS

Name of the PS Faculty: Dr R Parameshwaran

Key Words: Non-Destructive Testing, Cone Beam Reconstruction

Project Areas: Radiography, Tomography, Reconstruction

Abstract:

Computerized Tomography (CT) is a radiographic method that provides an ideal examination technique whenever the primary goal is to locate planar and volumetric details in three dimensions. CT has been used almost entirely for medical purposes but over the past two decades there has been an interesting increase in using CT as a powerful tool in non-destructive testing of industrial materials. In IGCAR parallel beam and fan beam reconstruction algorithms have been implemented for the in-house CT scanner. In order to further reduce the scanning time, and reconstruction time cone-beam computed tomography (CBCT) based reconstruction algorithm is being attempted. The amount of noise generated by cone-beam image acquisition is substantially higher, reducing image contrast and sharpness. Hence, filtering of projection data plays a vital role in improving the contrast of features reconstructed. Once the projection frames have been acquired, data must be processed to create the 3D object. This process is called reconstruction. The number of individual projection frames may be from 100 to more than 600, each with more than one million pixels, with 16 bits of data assigned to each pixel. The reconstruction of the data is therefore computationally complex. All codes for reconstruction are written in MATLAB.

Signature of Student	Signature of PS Faculty
Date	Date

TABLE OF CONTENTS

1. INTRODUCTION

1.1 ABOUT IGCAR

Indira Gandhi Centre for Atomic Research (IGCAR) is the second largest establishment of Department of Atomic Energy (DAE) next to Bhabha Atomic Research Centre (BARC) in the nuclear industry. Started under the name of Reactor Research Centre (RRC) in 1971, it is one of the leading research institute in the field of nuclear and atomic sciences. It is situated in Kalpakkam, a town 80 Km south of Chennai.

The main mission of this centre, put in the director's words, is "To conduct a broad based multidisciplinary programme of scientific research and advanced engineering development, directed towards the establishment of sodium cooled Fast Breeder Reactor (FBR) and the associated fuel facilities in the country."

This is the second stage of India's Atomic Energy Programme aimed at preparing the country for utilization of extensive Thorium reserves and providing electricity for the nation.

IGCAR had its modest beginning with the construction of sodium cooled Fast Breed Test Reactor (FBTR) with a nominal power of 40MWt. FBTR was based on a French reactor, RAPSODIE and attained its first criticality on 18th Oct 1985 and is operational with its maximum power level of 10.5MWt. It is the first of its kind to use Plutonium and Uranium Carbide as its driver fuel. In 1994, High-Power Physics and Engineering Experiments were undertaken in the FBTR.

Over the years, the centre has built research facilities covering the entire FBTR technology such as Sodium Technology, Reactor Engineering, Reactor Physics, Metallurgy and Materials, Chemistry of fuels and its materials, Fuel Reprocessing, Reactor Safety, Control and Instrumentation, Computer Applications etc., and has developed a strong base in a variety of disciplines related to this advanced technology. As a part of efforts for closing the fuel cycle, a Fast Reactor Fuel Reprocessing Plant is under construction.

With the experience and expertise gained by the successful operation of FBTR, a 500MWe Prototype Fast Breeder Reactor (PFBR) using Mixed Oxide (Uranium oxide + Plutonium oxide) fuel, is under construction. Various R&D activities like structural mechanics, thermal hydraulics, flow induced vibrations, high temperature testing, sodium-water reaction and various types non-destructive testing are pursued towards freezing of the final design for PFBR.

A 30 KWt, U233 fuelled mini reactor [KAMINI] has been made operational for neutron radiography, neuron activation analysis etc. IGCAR utilizes its expertise and resources in enhancing its standing as a leading Centre of research in various branches of basic, applied and engineering sciences that have a bearing on Nuclear Technology like Structural Mechanics, Heat and Mass Transfer, Material Science, Fabrication Processes, Non-Destructive Testing, Chemical sensors, High temperature thermodynamics, Radiation Physics, Computer science etc. In 2009, FBTR was operated at a maximum power level of 18.6 MWt with 55 sub-assemblies for 1732 hours.

Apart from the thrust areas related to nuclear technology, the centre has embarked on a leading research in various frontiers like quasi crystals, oxide superconductors, nano structures, clusters, exo-polymers among others. IGCAR has also extended its expertise and facilities to defence, space and other industries. Kalpakkam is the only place in India where all the three stages of Indian Nuclear program has been achieved .

A modern Library comprising 62,000 volumes of books, 28,400 back volumes, about 785 journals and 1.95 lakhs reports in all disciplines caters to the technical needs of the scientists and engineers.

The Centre is being steered by the dynamic leadership of Dr. Vasudeva Rao , Director,IGCAR.

1.2 About NDED

Non-Destructive Evaluation Division (NDED) is engaged in development of NDE techniques and methodologies for detection and characterization of defects, residual stresses and microstructures. While it gives thrust to science based technology for robustness of the developed techniques / methodologies, it also gives special emphasis to develop field-implemental technologies for pre-service and in-service inspection of components and structures in nuclear installations. The division is also engaged in development of in-house instrumentation, sensors, numerical models, signal & image processing schemes and data management packages. The division also supports strategic and core sectors for finding solutions to challenging problems related to NDE.

Dr. B. PURNA CHANDRA RAO is the current head of NDED.

1.2.1. Activities under NDED

- Pre-service and in-service inspection of components and systems of nuclear Installations.
- High sensitivity defect detection and materials characterization using ultrasonic.
- Remote field eddy current (RFEC) technology demonstration .
- Finite element modelling & design of eddy current (EC) sensors .
- Micro-magnetic and Barkhausen emission (MBE) studies for characterization of microstructures and residual stresses.
- Acoustic emission for crack propagation, phase transformation studies .
- Infrared thermography (IRT) for dynamic deformation, crack propagation and condition monitoring .
- Evaluation of residual stress using MBE, UT, XRD and other techniques.
- Failure analysis, damage assessment and life extension .
- X-ray Tomography .
- NDE of concrete structures.
- Fiber optic based distributed temperature and strain measurement.

1.3 INTRODUCTION TO NDT

Non Destructive Testing is the use of non-invasive techniques to determine the integrity of a material, component or structure or quantitatively measure some characteristic of an object. The field of Non-destructive Testing (NDT) is a very broad, interdisciplinary field that plays a critical role in assuring that structural components and systems perform their function in a reliable and cost effective fashion.

Radiography is one of the oldest NDT techniques, which permits an “inside view” of the material under test. X rays or gamma rays are projected onto a specimen under examination and the intensity of radiation transmitted through the object is recorded using a photosensitive film, widely known as industrial X ray film. For several years, radiography is used detecting defects in the structures and welds.

Of all these various NDT techniques, radiography has the following advantages over the other NDT techniques which make it one of the best techniques to classify defects-

- Can detect flaws in any type of object, internal or external.
- Depth of penetration is variable and large.
- Ability to inspect complex shapes and multi-layered structures without disassembly

Reconstruction is the process of forming the 3D image of the object from the projection data. There are mainly three types of projections

- Parallel Beam
- Fan Beam
- Cone Beam

Parallel Beam is the simplest type of projection, where a point source and detector are used. It provides very accurate projection data, but is very time consuming.

In Fan Beam projection, a point source and a line detector is used to get the projection data. It is faster than the Parallel Beam method, but not as accurate.

Cone Beam projection uses a point source and a two dimensional detector. It is the fastest method of acquiring projection data for a three dimensional object.

The main aim of this report is to explain a cone beam reconstruction algorithm.

2. PARALLEL BEAM PROJECTION DATA

2.1 Beer Equation

The first major part of tomography is understanding what the projection data is. A fundamental equation used in CT is the Beer equation

$$I(x) = I_0 e^{-\mu x}$$

Where I_0 - Initial intensity

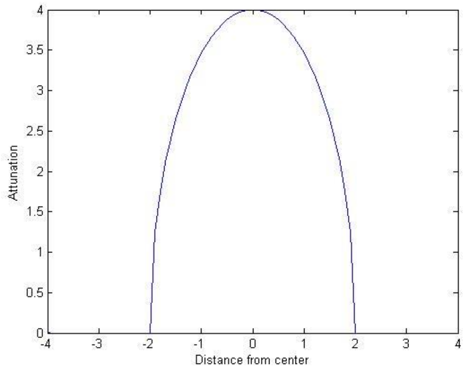
I - Final intensity

μ - Linear attenuation coefficient

x – Thickness of the material

When an X-ray passes through a material, its intensity gets attenuated, which is given by the above equation. Consider a solid circle at the origin, made of a uniform material with attenuation coefficient $u = 1\text{cm}^{-1}$ and radius R=2cm. If we allow a parallel beam of X-rays to pass through this circle and fall on the detector on the other side. Assuming that u_{air} is zero, we get

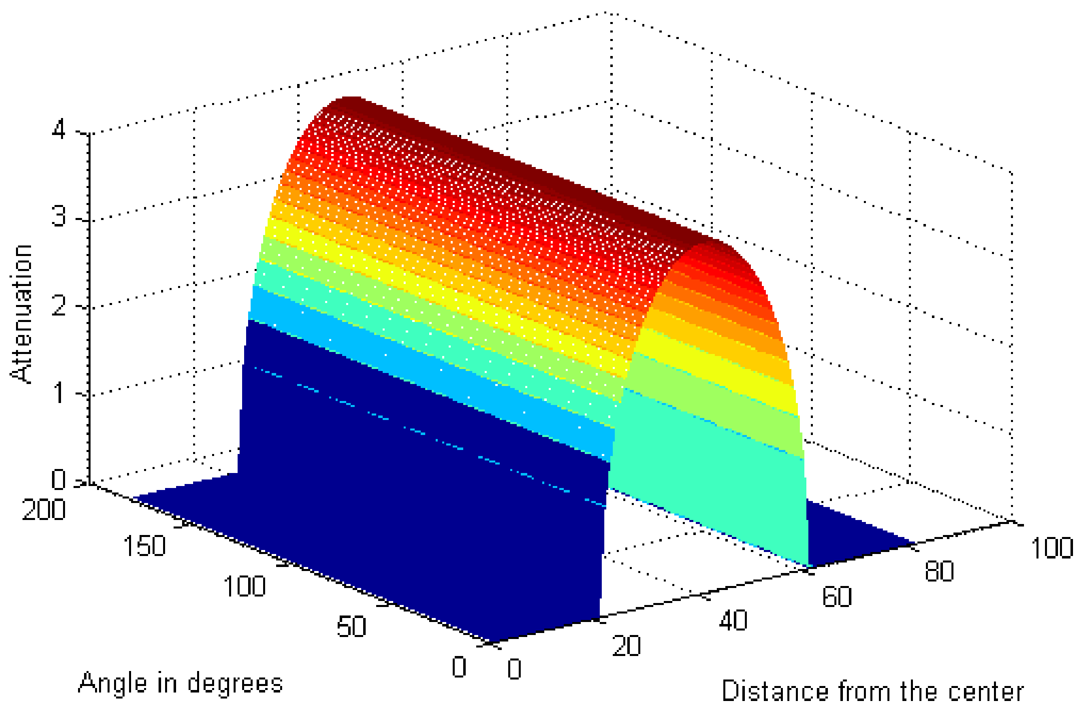
Projection Data of a Circle



(Fig generated in Matlab)

Now if we rotate the detector an the source around the origin, taking projection data at every degree to 180 degrees, we get this,

Sinogram of a Circle



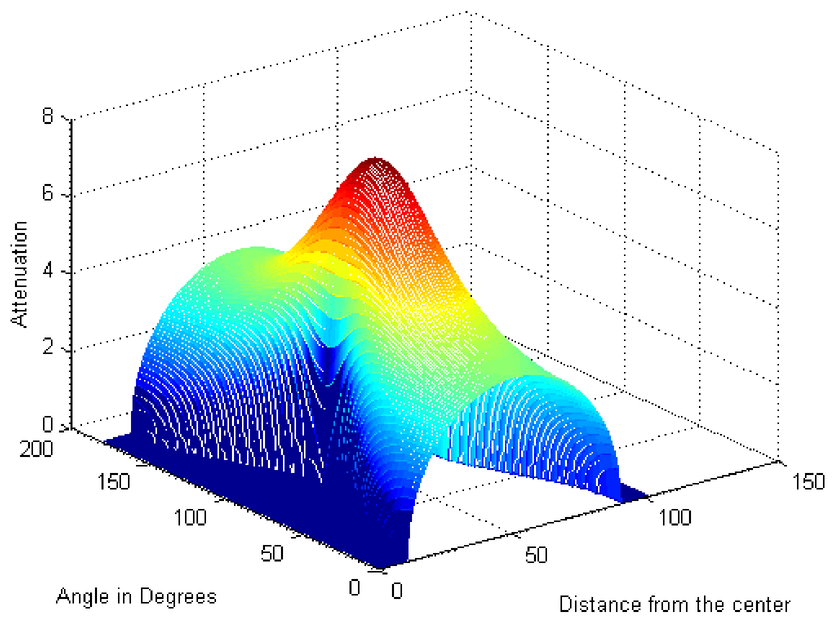
(Fig generated in Matlab)

We can see that the projection data is the same at all angles, because it is a circle at the origin.

2.2 Projection Data of Ellipse

Now we consider a solid ellipse with center at origin made of a uniform material with attenuation coefficient u = 1 cm^{-1}, semi major axis = 4 cm and semi minor axis = 2 cm. The projection data at all angles, is given by

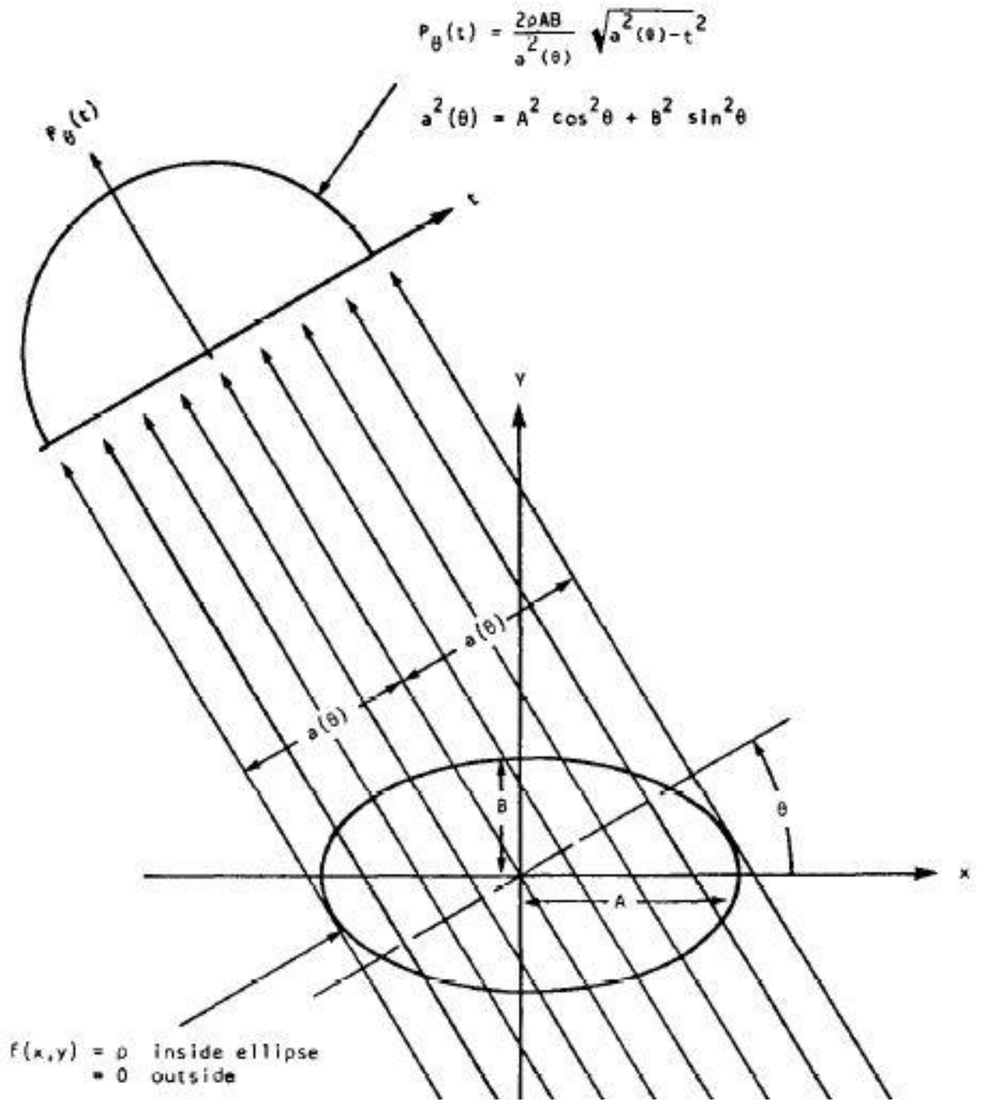
Sinogram of an Ellipse



(Fig generated in Matlab)

For the ellipse the projection data varies with the angle of projection because the total material that the ray has to pass through changes with the angle.

Projection data of an Ellipse



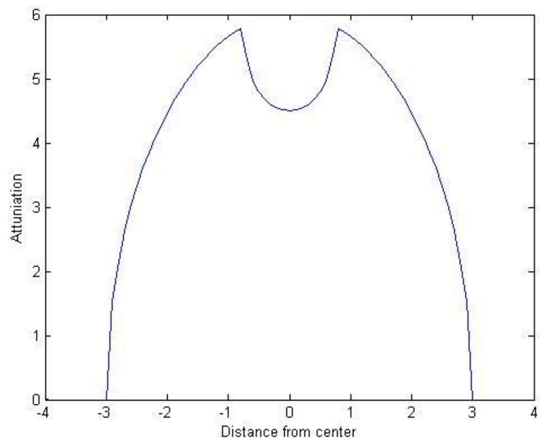
$P_\theta(t) = \frac{2\rho AB}{a^2(\theta)}\sqrt{a^2(\theta)-t^2}$
$a^2(\theta) = A^2\cos^2\theta + B^2\sin^2\theta$
$P_\theta(t)$
t
y
$a(\theta)$
$a(\theta)$
B
θ
x
A
$f(x,y) = \rho$ inside ellipse
$= 0$ outside

Figure 1, Adopted from 'Principles of Computerized Tomography'

2.3 Projection Data of Circle with hole

Now let us consider a solid circle at the origin with attenuation coefficient u = 1 cm^{-1} and radius = 3 cm, with a hole at the center filled with air, its projection data at zero degrees will be

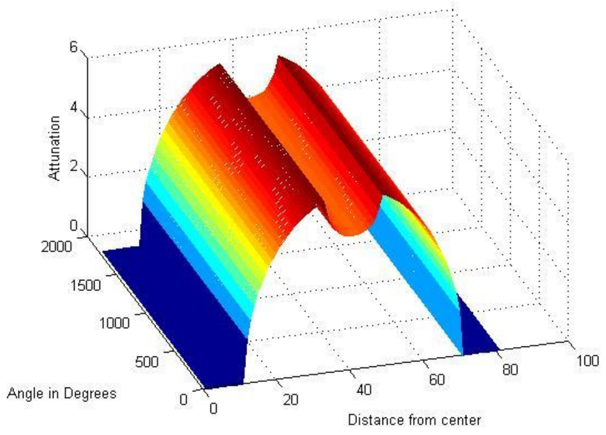
Projection data of a Circle with Hole



(Fig generated in Matlab)

For all angles till 180 degrees the projection data will be

Sinogram of a Circle with Hole

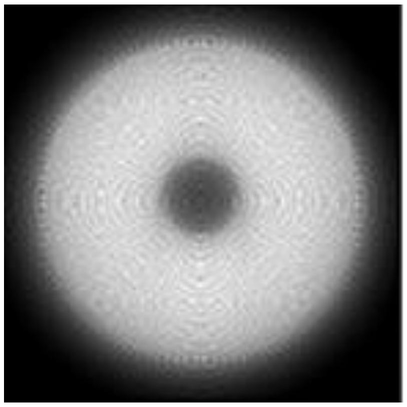


(Fig generated in Matlab)

The main aim of tomography is to reconstruct the actual image from the projection data. In all the cases above, we took only the projection till 180 degrees because the data at the remaining angles are mirror images of given data. So no extra information is obtained.

By creating a reconstruction script in Matlab, the following image was obtained from the above projection data

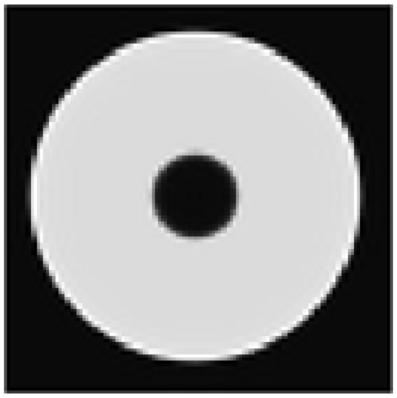
Reconstruction of a Circle with Hole



(Fig generated in Matlab)

Although I mentioned a solid circle with uniform material, the reconstructed image has a lot of noise. This is caused due to reconstruction itself, and can be removed by filtering.

Filtered reconstruction of a Circle with Hole



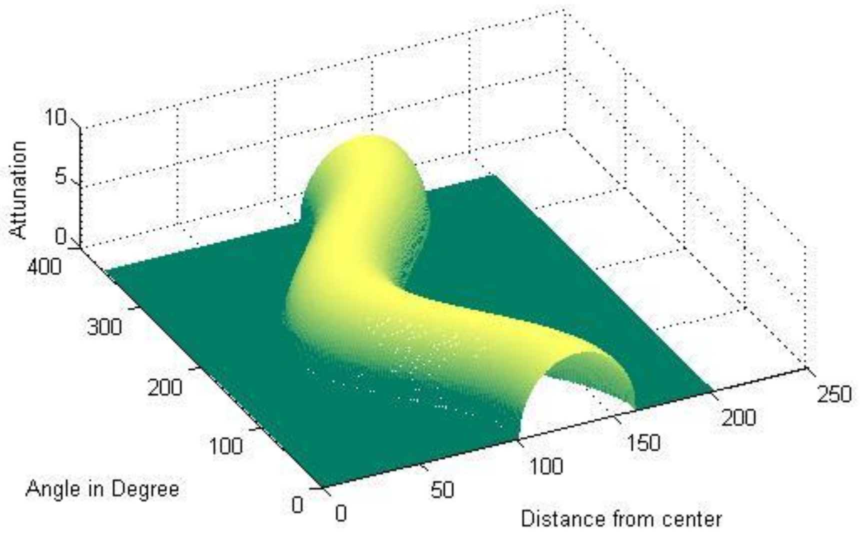
(Fig generated in Matlab)

Each layer of the 3D object is reconstructed and stacked over each other to reconstruct the 3D object.

2.4 Projection Data of Offset Circle

Let us now see the projection data by a solid circle that is offset from the center, for 360 degrees

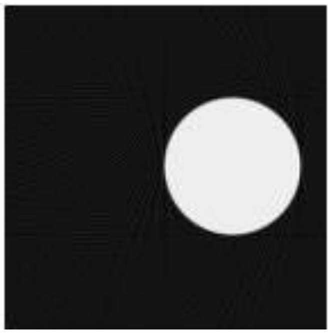
Sinogram of Offset Circle



(Fig generated in Matlab)

Thus as the data represents a sine curve, the projection data is also called a sinogram. The filtered reconstruction of the above sinogram is given by,

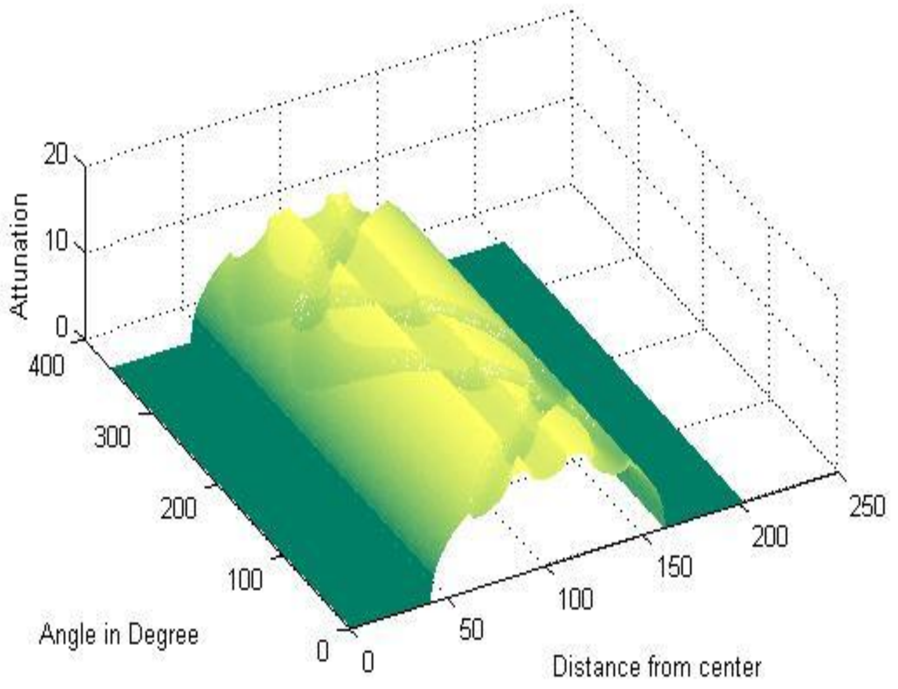
Reconstruction of Offset Circle



(Fig generated in Matlab)

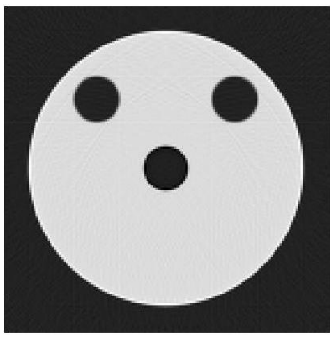
Another sinogram and its filtered reconstruction,

Sinogram of Circle with 3 holes



(Fig generated in Matlab)

Reconstruction of Circle with 3 holes

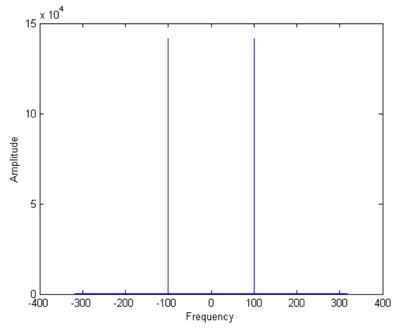


(Fig generated in Matlab)

3. FILTERING

As described before the reconstructed image will have some periodic noise due to the reconstruction itself. But we can remove this noise through appropriate filtering. This section is to explain the filtering process. The main problem of removing noise is how to differentiate between the image and the noise. Since the **noise is periodic and usually with a high frequency**, we can remove it easily in the Fourier domain, which can be obtained through the Fourier transform. The Fourier transform of a function gives the coefficients of the sine and cosine terms at various frequencies. For example for a continuous sine wave from –infinity to infinity, the Fourier transform gives

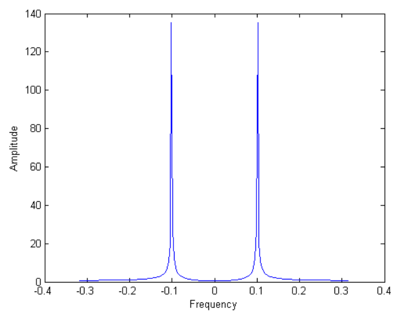
Fourier Transform of continuous sin(x)



(Fig generated in Matlab)

But for a sine wave which is finite in length and zero elsewhere, it can be expressed as a sum of continuous sine wave and many other cosine and sine waves of different frequencies.

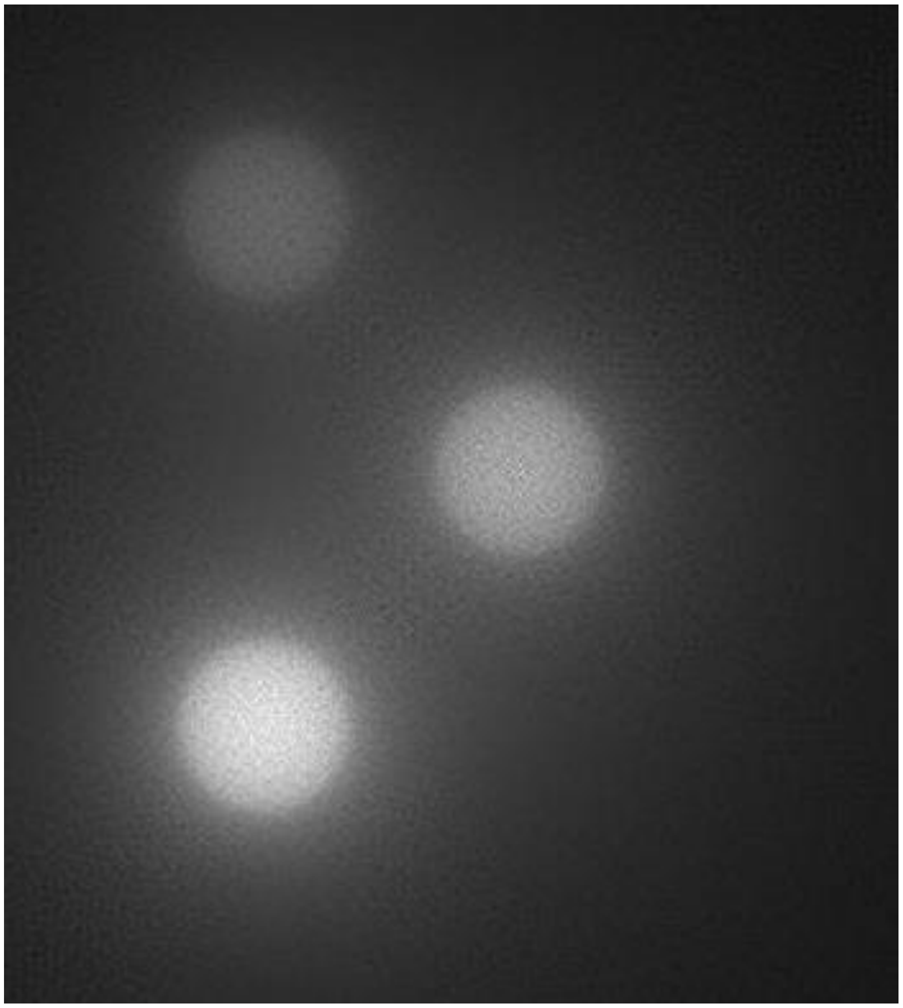
Fourier Transform of discontinuous sin(x)



(Fig generated in Matlab)

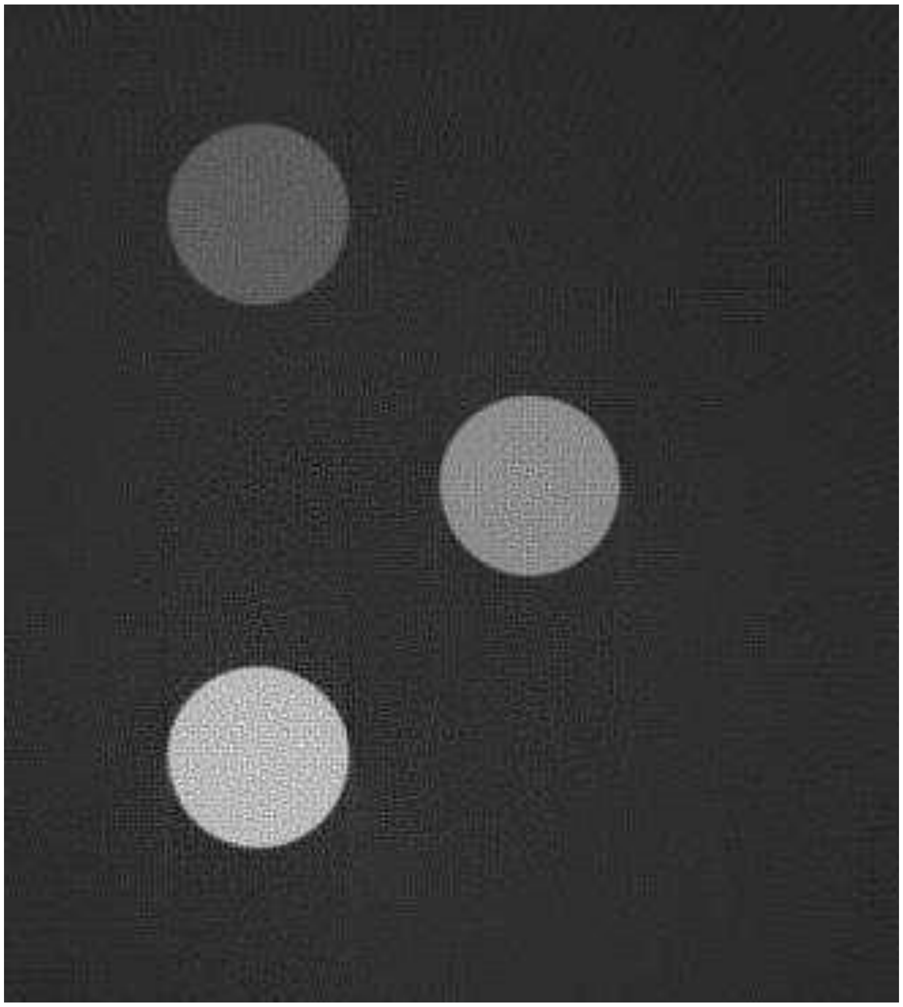
So we can transform the reconstructed image, cut off the high frequency noise and take the inverse to get the filtered image.

Reconstructed image without Filter



(Fig generated in Matlab)

Reconstructed image with Filter



(Fig generated in Matlab)

The cutoff frequency must be carefully chosen so as to have the optimum sharpness and minimum noise.

4. FAN BEAM PROJECTIONS

Now we can start with fan beam projections. The problem with the parallel beam projection is that for every angle up to 180 degrees, the source and detector must move in parallel across the region of interest. Thus scanning the whole object is very time consuming.

In fan beam projection, the final intensity of all the rays that come from the point source in the plane of the object is measured. Thus the linear movement is no longer necessary, saving a lot of time.

Fan Beam Projection

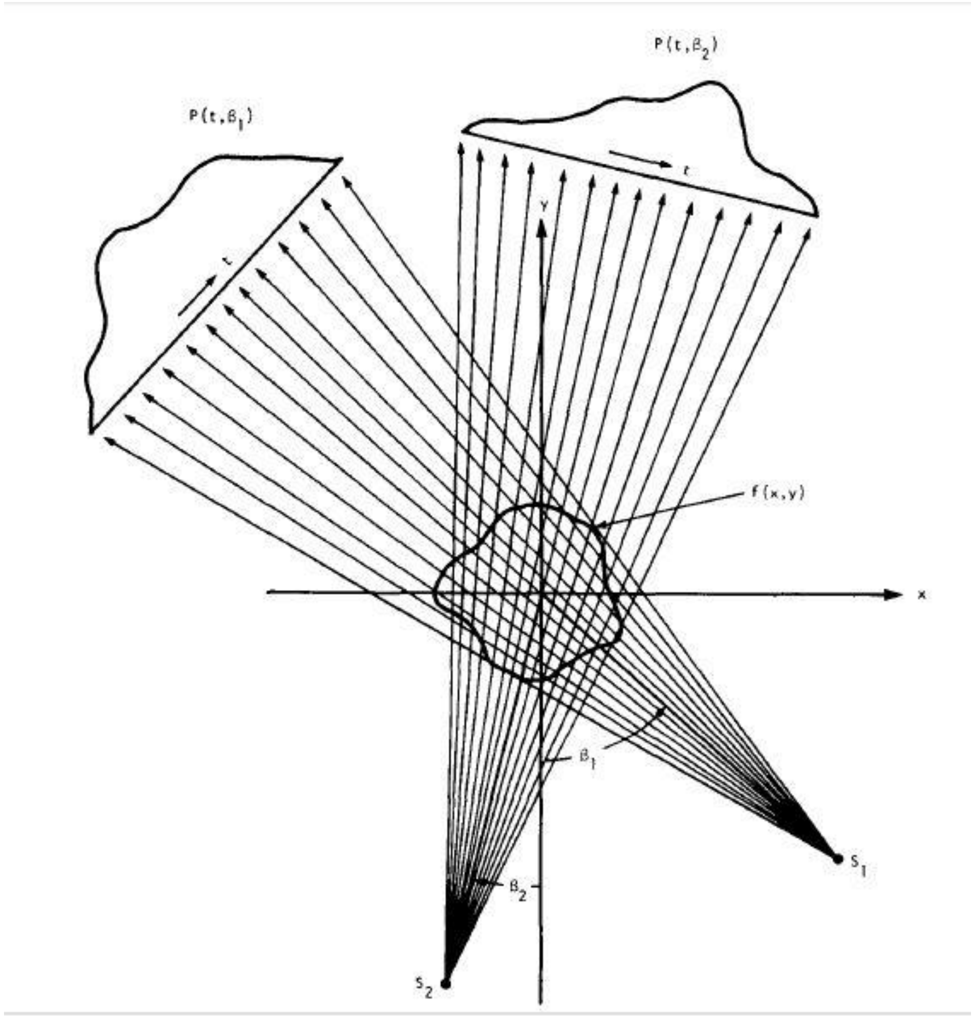
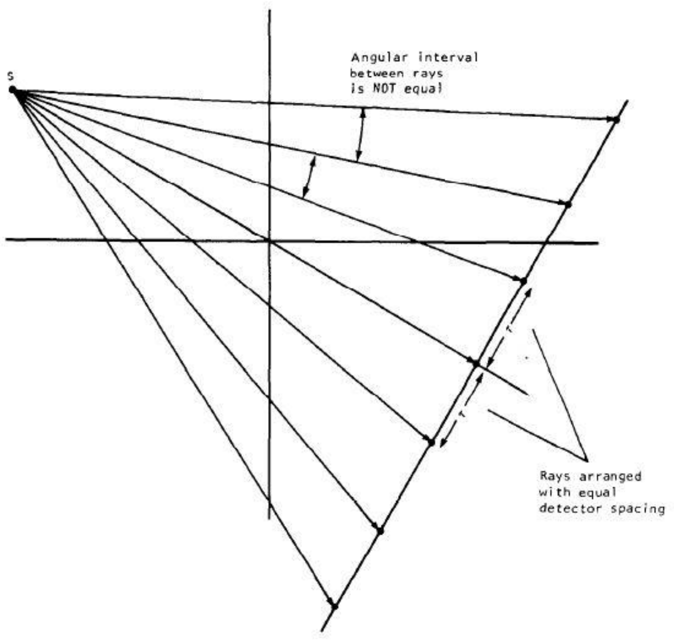


Figure 2, Adopted from 'Principles of Computerized Tomography'

However one disadvantage of fan beam is that due to “cross talk” between the pixels of the detector, the projection data will have more noise. Since the projection data now depends on the location of the source and the detector, they too have to be specified to reconstruct the object.

Fan Beam Projection

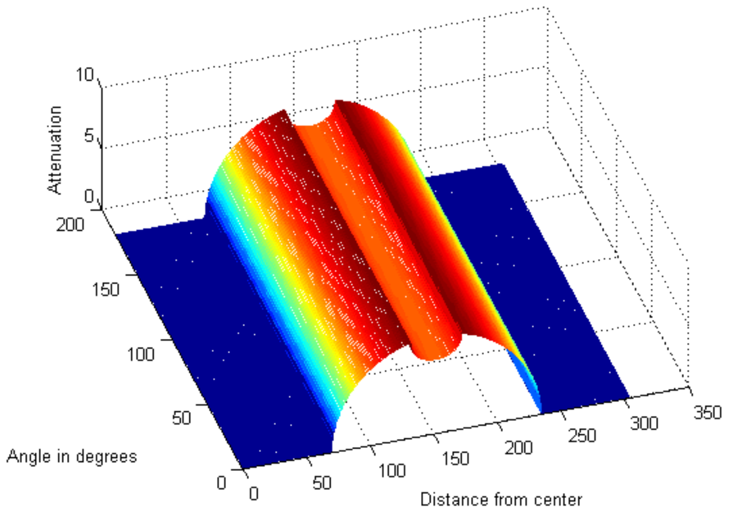


S
Angular interval between rays is NOT equal
Rays arranged with equal detector spacing

Figure 3, Adopted from 'Principles of Computerized Tomography'

Consider a solid circle at the origin with a hole in the center. The radius of the circle is 100cm and the attenuation coefficient u = 1 cm^{-1}. The distance of the source from the origin is 500 cm and the distance between the source and the detector is 1000 cm. The sinogram is given by

Fan Beam Sinogram of circle with hole

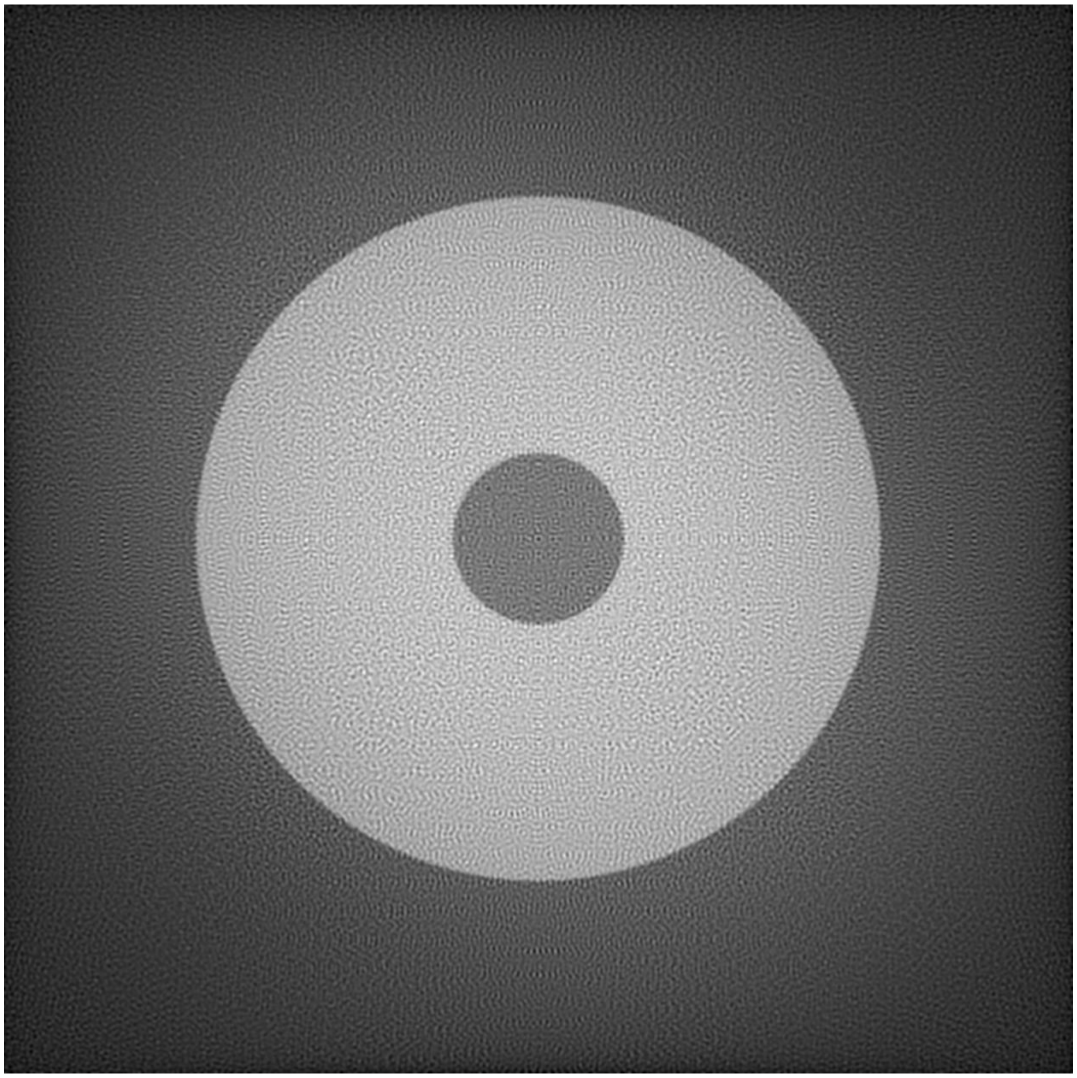


(Fig generated in Matlab)

Notice that it looks the same as the parallel beam data, but if you look carefully you can see that the data has been magnified or spread across the detector. For parallel beam projection, the above sinogram would give a circle of radius greater than 100 cm.

So we can see that a different reconstruction technique is required for fan beam. By using a script written in Matlab (Available in Appendix) the above sinogram was reconstructed to give

Reconstruction of Fan beam projections



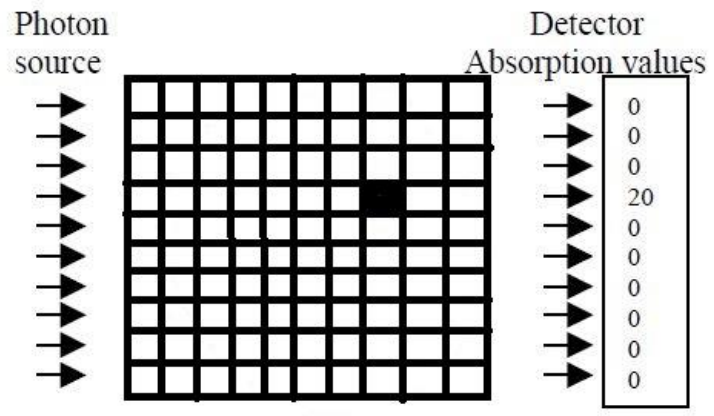
(Fig generated in Matlab)

The reconstruction technique will be explained in the next section.

5.RECONSTRUCTION

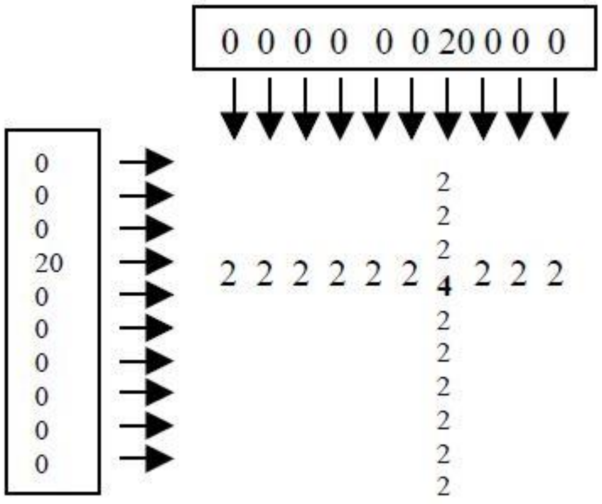
Although there are many techniques of reconstruction, a technique call filtered back projection is used in the Matlab reconstruction script. The technique is best explained through the following example.

Consider a 10x10 matrix with an object with the attenuation coefficient μ = 20 in one element and all other elements are empty. The projection data at zero degree is given by



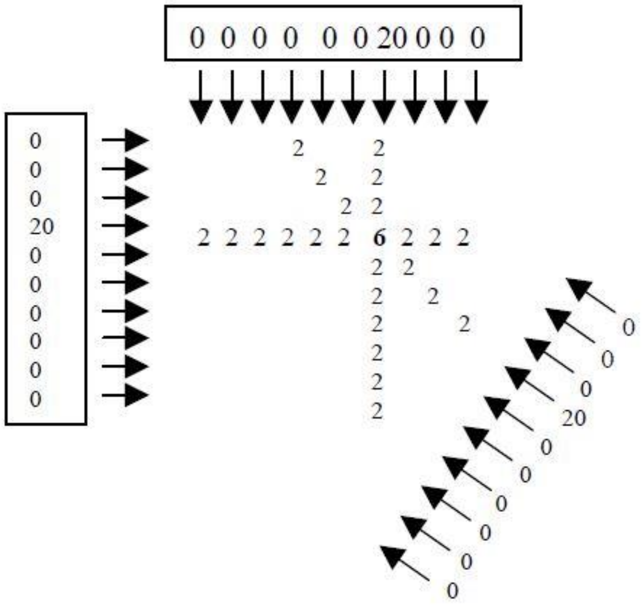
(Fig generated in Paint)

Assuming we have the projection data for 90 degree too, we will attempt to reconstruct the object. We initially take all elements of reconstructed matrix to be zero. Then we 'smear' the projection data for a given angle on the matrix. Then the angle is changed and the new projection data is smeared onto the previous one, adding the values. For the above case the reconstruction for 0 and 90 degree gives



(Fig generated in Paint)

As we increase the number of projections, the reconstruction becomes more accurate.



(Fig generated in Paint)

One can notice that in the reconstructed image the region around the object is not zero, but with a large number of projections creates a pattern around the object. This is the periodic noise created by reconstruction and can be removed by filtering, as demonstrated before.

But the above case was only for the parallel beam case. For the fan beam reconstruction, the data is 'smeared' in the shape of a fan, back to the source.

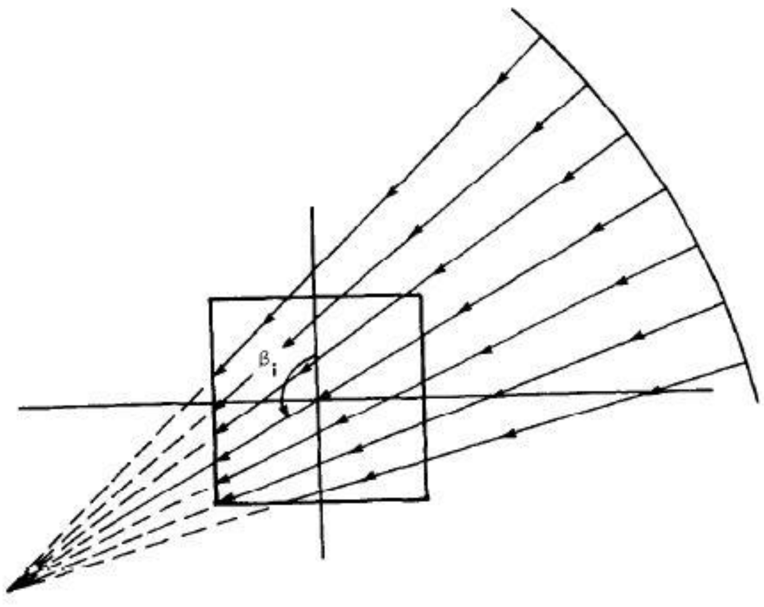
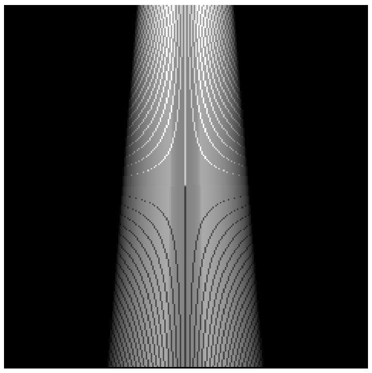


Figure 4, Adopted from 'Principles of Computerized Tomography'

For a solid circle with a hole in the middle, the 'smearing' for 0^0 gives

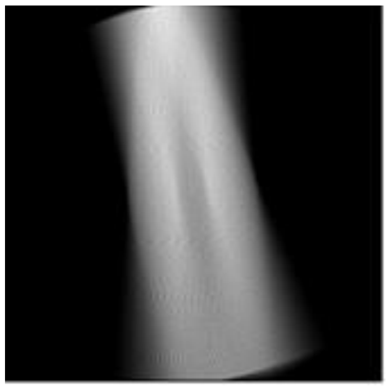
Reconstruction of Fan beam at 0^0



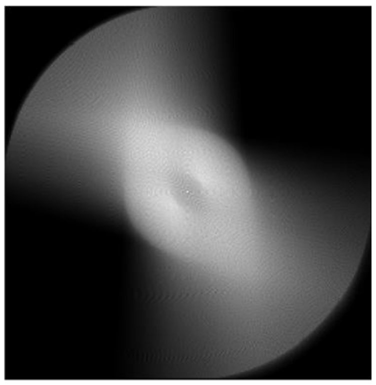
(Fig generated in Matlab)

At 10^0 and 45^0 the reconstructed images,

Reconstruction up to 10^0

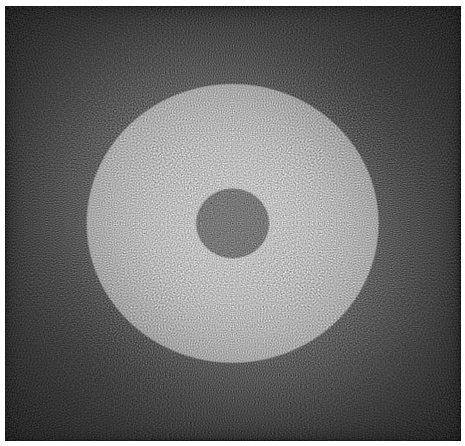


Reconstruction up to 45^0



(Figs generated in Matlab)

Fully Reconstructed Image



(Fig generated in Matlab)

6. CONE BEAM PROJECTION

Cone Beam projection is much faster than fan beam projections. In fan beam, after every 180^0 rotation, the source and the detector have to be moved upwards. In Cone Beam, there is only a need for 180^0 rotation, and all the projection data of the 3D object is captured by the 2D array.

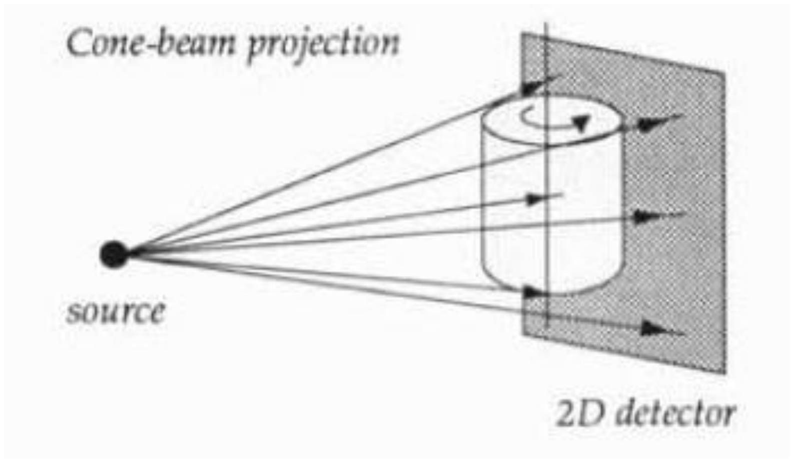
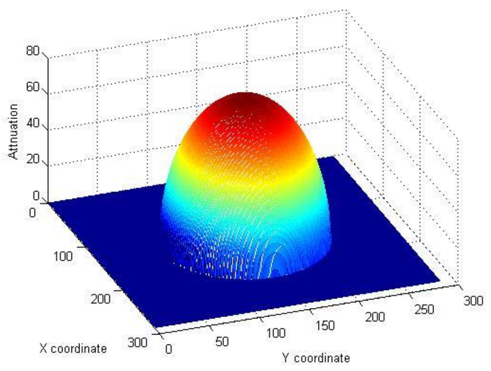


Figure 5, Adopted from 'Principles of Computerized Tomography'

Although cone beam is much faster, it has the disadvantage of more noise in the projection data due to 'cross talk' between pixels in the detector.

The cone beam projection data at 0^0 for a solid sphere is given by

Projection Data for a sphere at 0^0

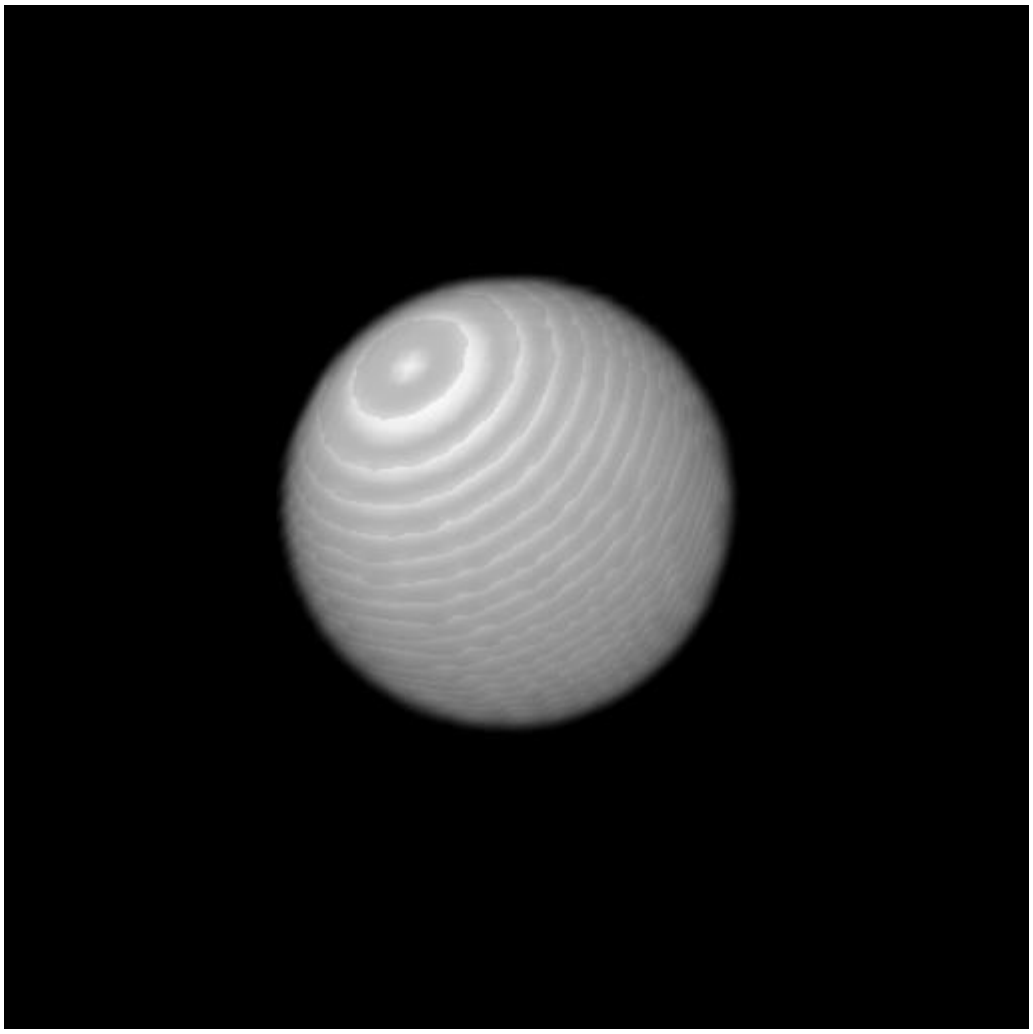


(Fig generated in Matlab)

The reconstruction technique used for cone beam is similar to the one used by fan and parallel beams, smearing the projection data along the ray back to the source.

A reconstruction script has been developed in Matlab (Available in Appendix), however as the whole 3D object is being reconstructed at once, a lot of computational power is required. The above sinogram was processed in the reconstruction script to give the following,

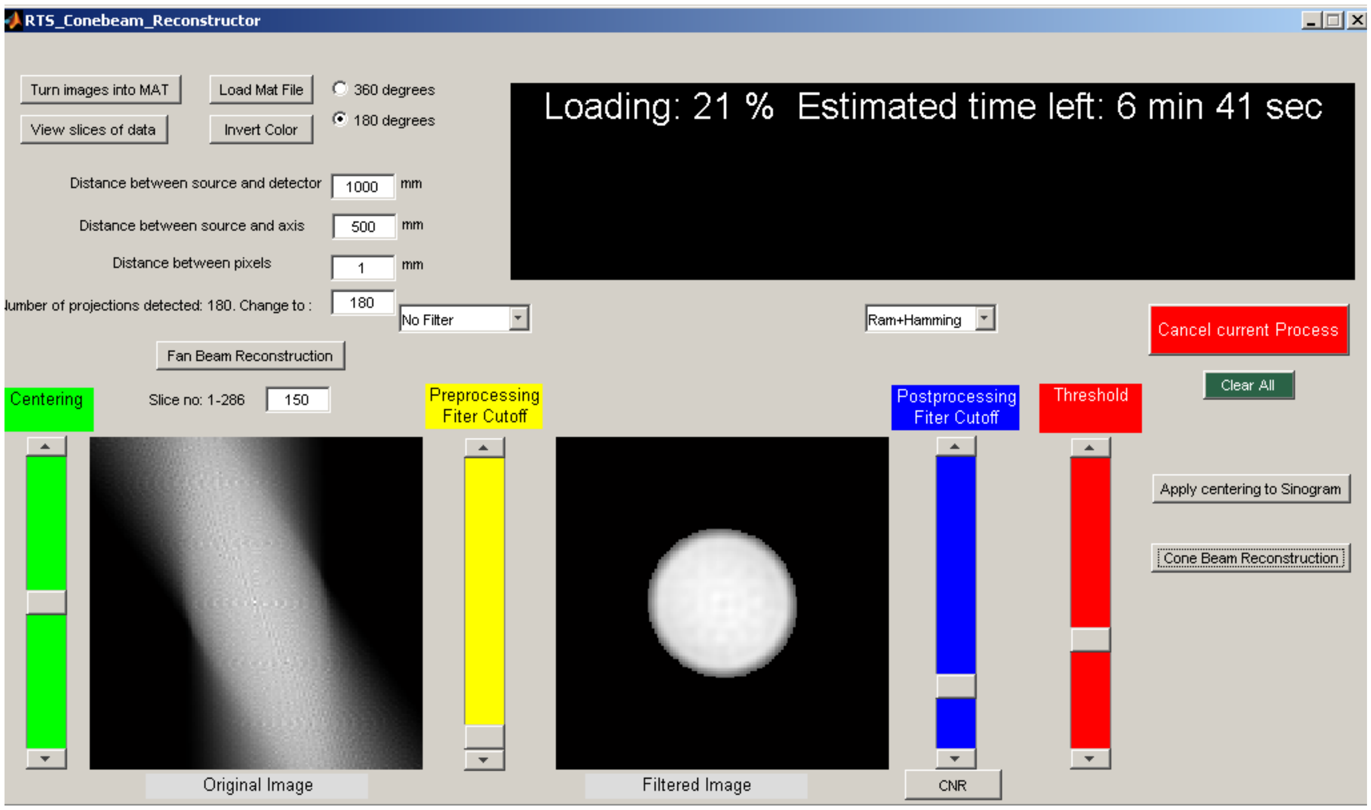
Reconstructed image of sphere



(Fig generated in Imagej)

7. GRAPHICAL USER INTERFACE

To make the implementation of the cone beam reconstruction easier, a graphical user interface was created in Matlab (Code available in Appendix).



360 degrees [✓ 180 degrees]

Distance between source and detector 1000 mm

Distance between source and axis 500 mm

Distance between pixels 1 mm

Number of projections detected: 180. Change to : 180

No Filter

Ram+Hamming

Slice no: 1-286 150

```
Loading: 21 %  Estimated time left: 6 min 41 sec
```

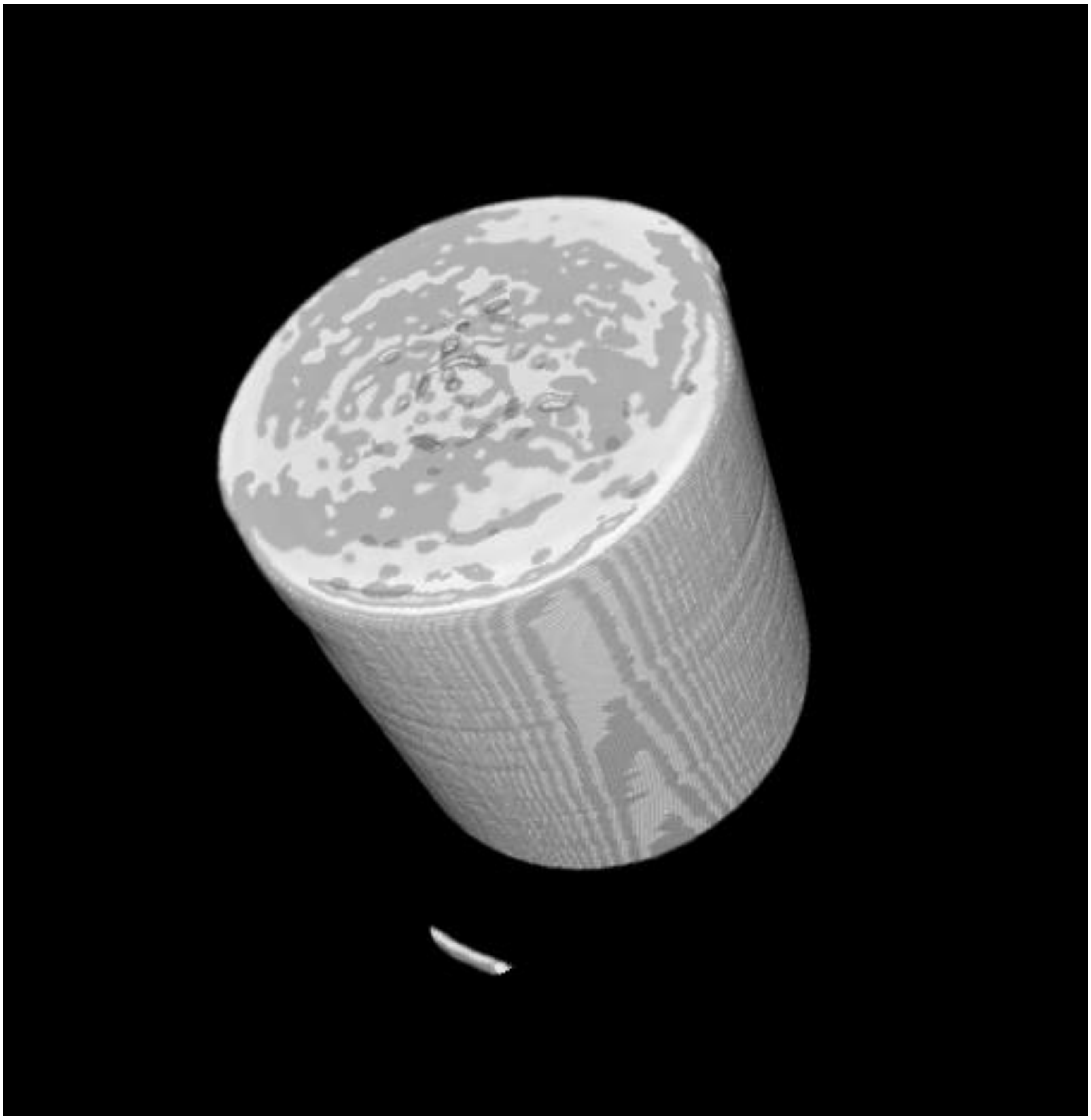
Features:

- Simple user interface, instructions given for every step.
- Can load .mat files or series of images, of various formats (.jpg, .png, .tif, …)
- Can view all projections like a video.
- Automatically calculates the number of slices.
- Automatically centers the object, but centering can still be adjusted.
- Pre-reconstruction filtering.

- Can reconstruct each slice individually.
- Can try filtering and thresholding on a single slice, for the best quality.
- The filtering variables are automatically carried on to the cone beam reconstructor.
- Can calculate the Contrast to noise ratio and Signal to noise ratio, after filtering.
- Cone beam reconstruction uses minimal memory.
- Provides the estimated time left.
- Can view all reconstructed slices like a video.
- Can save the 3D image as a .mat file, or as image slices of various formats.

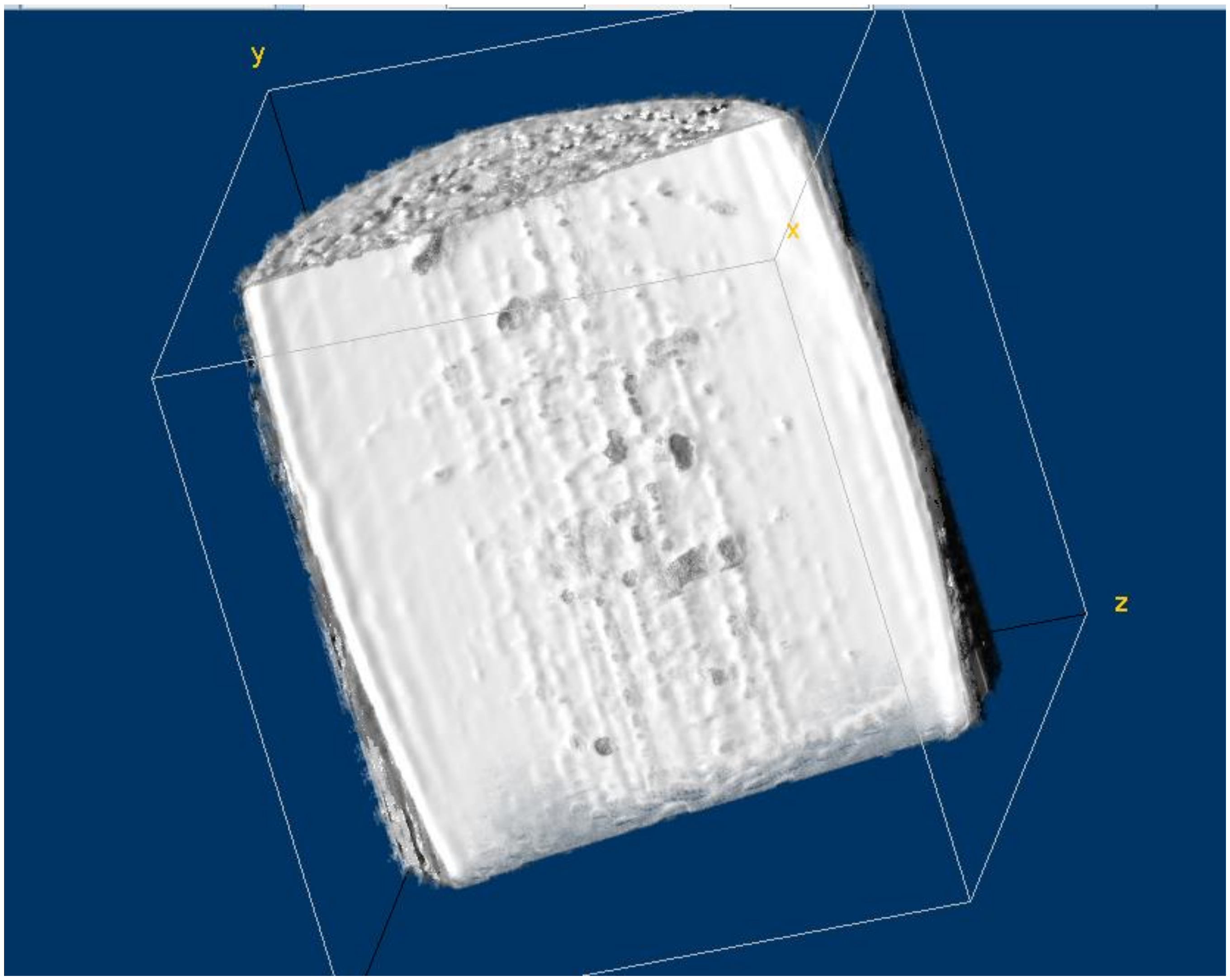
Some more object reconstructed from the GUI:

1.1 3D model of concrete



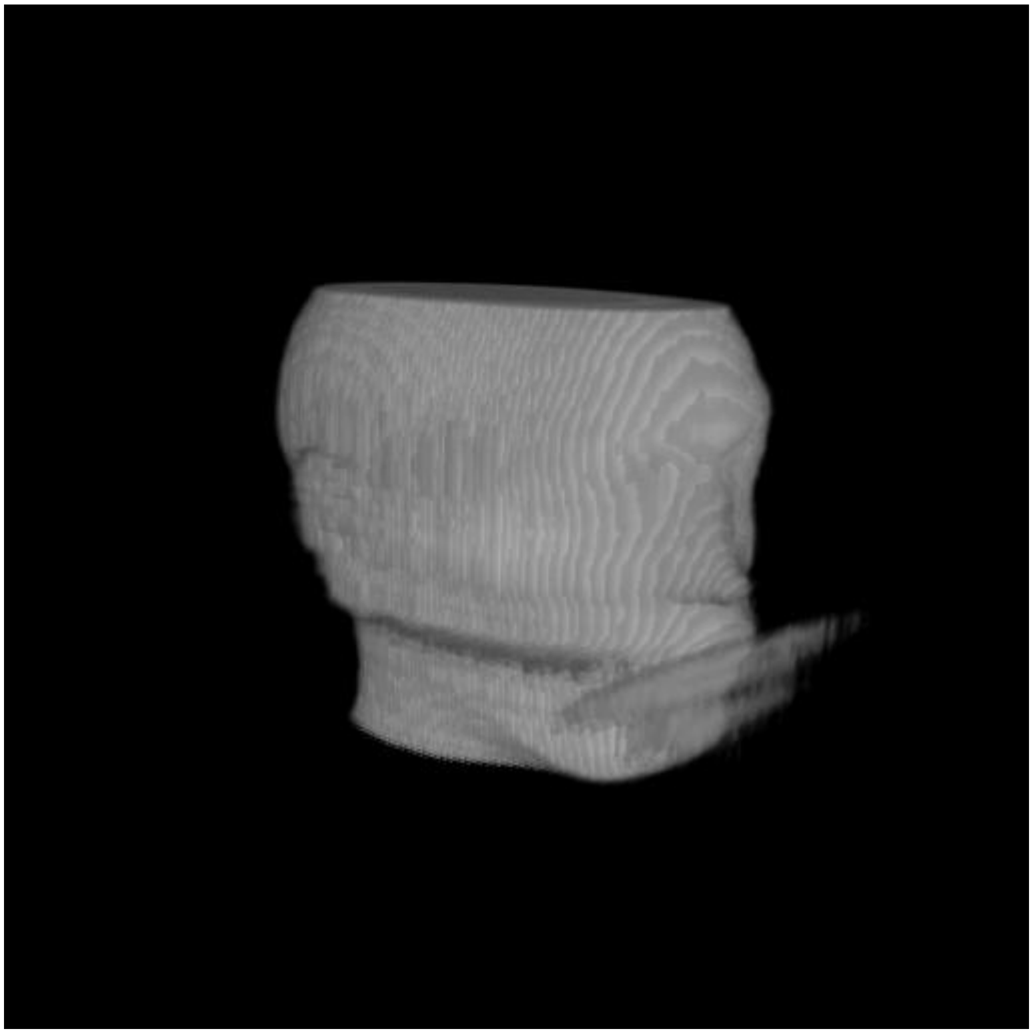
(Fig generated in Imagej)

1.2 Sliced 3D model of concrete



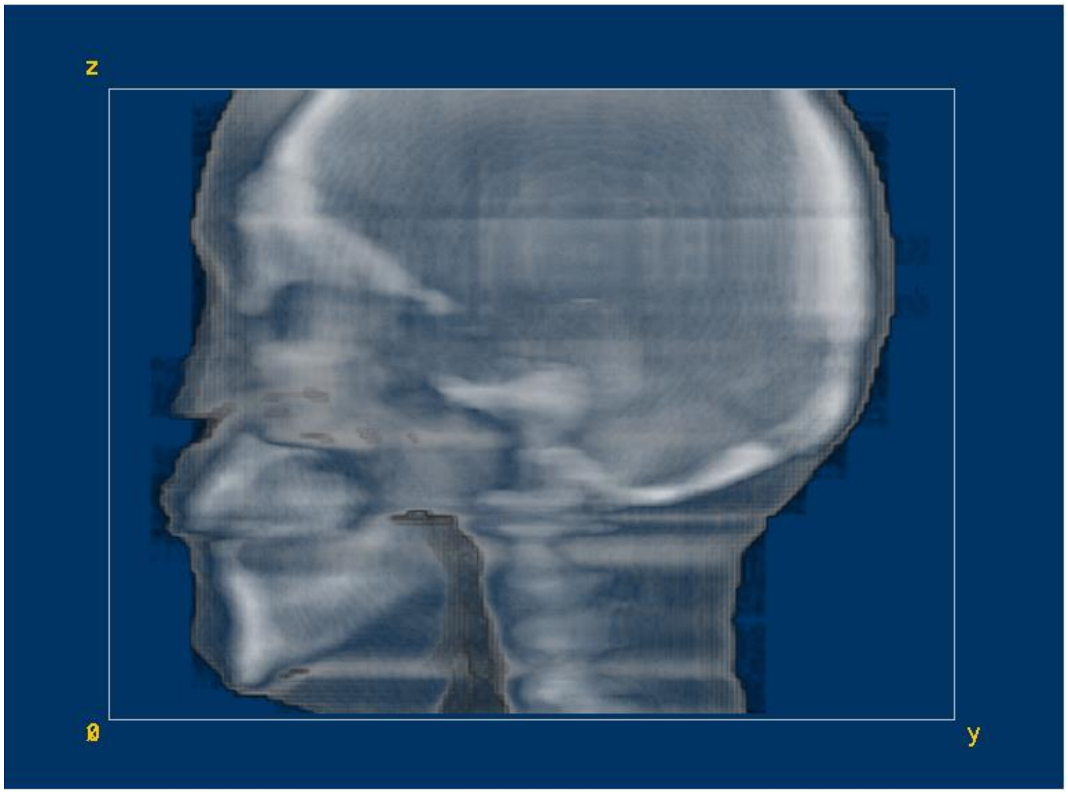
(Fig generated in Imagej)

2.1 3D model of Head



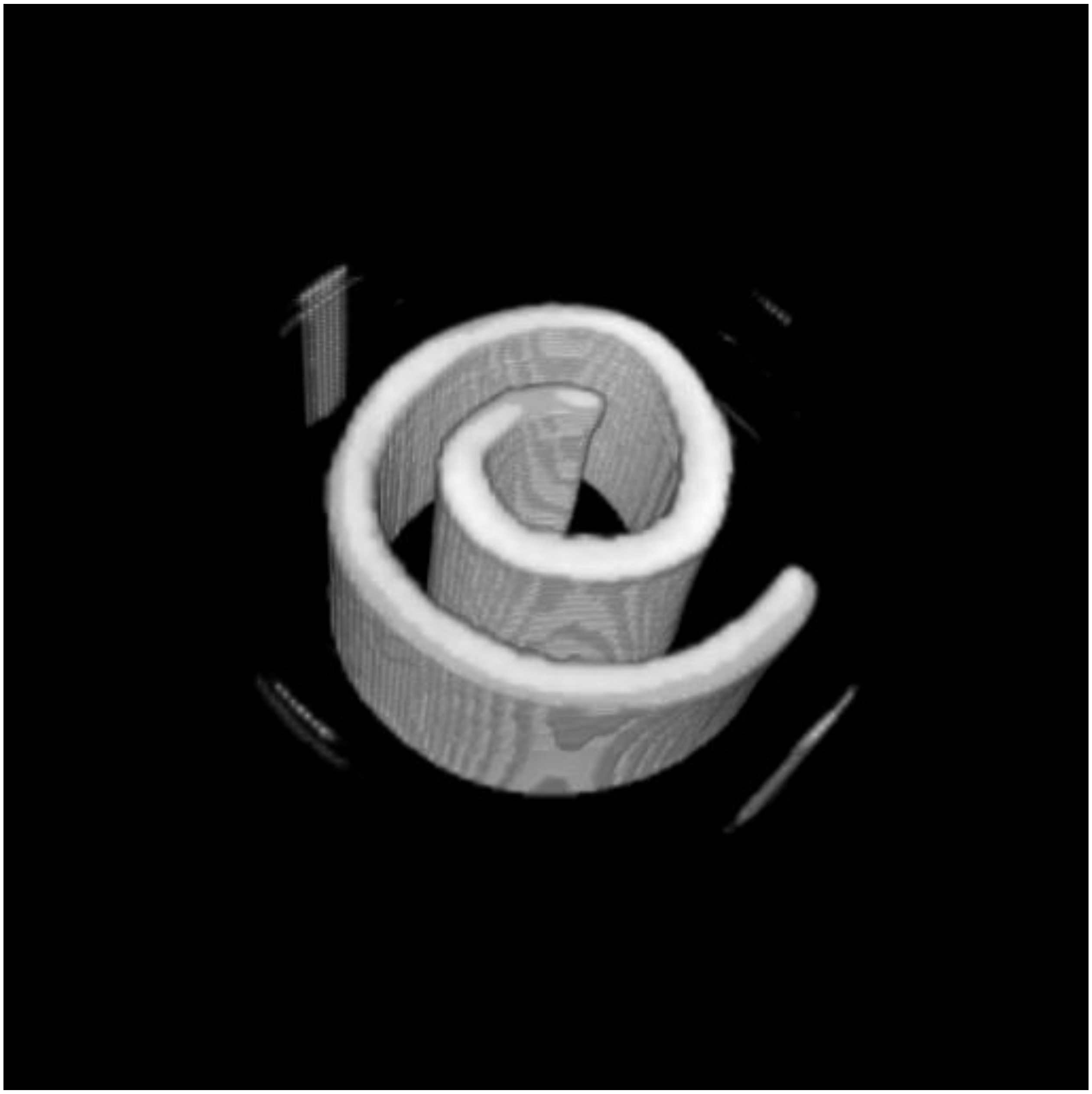
(Fig generated in Imagej)

2.2 Sliced 3D model of Head



(Fig generated in Imagej)

3.1 3D model of Aluminium coil



(Fig generated in Imagej)

8.RESULTS

8.1 Correlation coefficient

To estimate the quality of reconstruction, a simulated image was created along with its projection data. The data was processed in the reconstruction scripts to obtain the final image. Then the correlation coefficient was calculated between the original image and reconstructed image. The equation for correlation coefficient is given by,

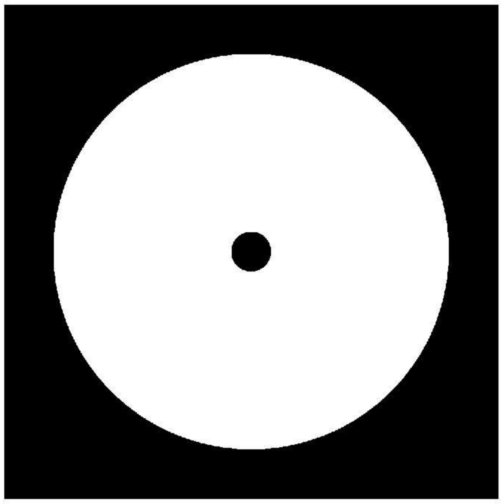
$$\text{Correlation} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\rho(i,j)\overline{\rho}(i,j)}{\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}\rho(i,j)^2\sum_{i=1}^{N}\sum_{j=1}^{N}\overline{\rho}(i,j)^2}}$$

Where $\rho(i, j)$ is the actual image and $\overline{\rho}(i, j)$ is the reconstructed image.

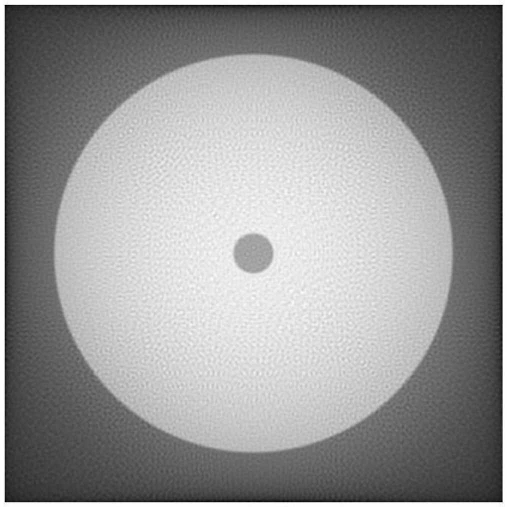
The value of the correlation coefficient varies from 0 to 1. One implies a perfect match to the actual image. The correlation coefficient was calculated for the fan beam and the cone beam reconstruction codes.

Fan Beam:

Original Image



Reconstructed image



(Fig generated in Matlab)

Correlation coefficient before filtering = 0.862

Correlation coefficient after filtering = 0.906

Cone Beam:

Original Image Reconstructed image

(Fig generated in Matlab)

Correlation coefficient before filtering = 0.768

Correlation coefficient after filtering = 0.939

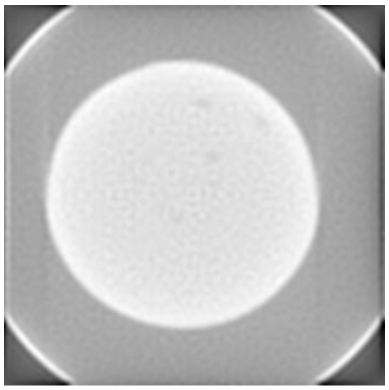
	Fan Beam		**Cone Beam**	
Correlation Coefficient	**Before Filtering**	**After Filtering**	**Before Filtering**	**After Filtering**
	0.862	**0.906**	**0.768**	**0.939**

8.2 Signal to noise ratio

To estimate the quality of reconstruction, the reconstructed image was compared with other images reconstructed through other methods. Then signal to noise ratio was calculated for both the images. The signal to noise ratio is given by,

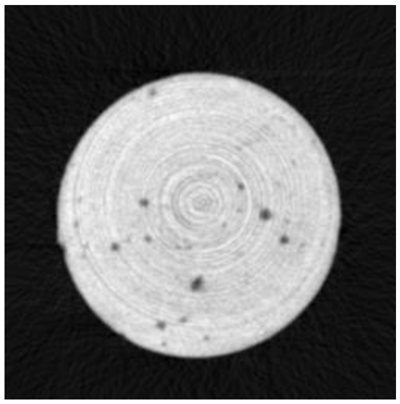
$$\text{SNR} = (\text{Grey value})_{\text{mean}} / (\text{Standard deviation of grey values})$$

RTS reconstructor Image



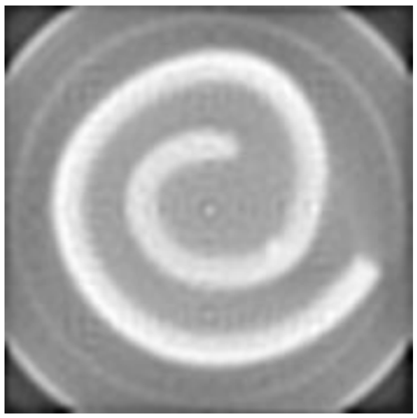
SNR = 15.93

Other reconstructor image



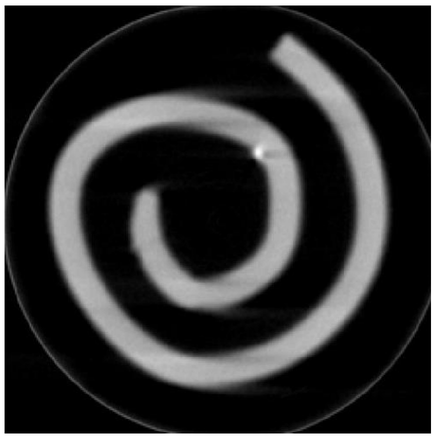
SNR = 8.47

RTS reconstructor Image



SNR = 15.52

Other reconstructor image



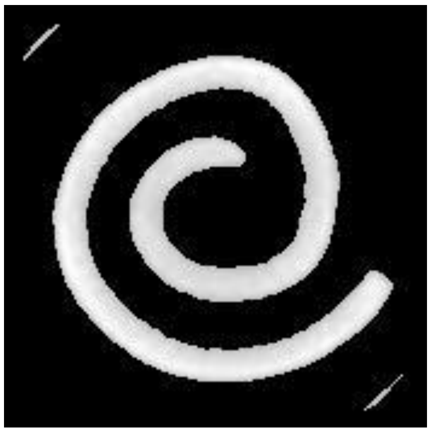
SNR = 11.77

8.3 Contrast to noise ratio

To estimate the quality of reconstruction, the reconstructed image was compared with other images reconstructed through other methods. Then contrast to noise ratio was calculated for both the images. The contrast to noise ratio is given by,

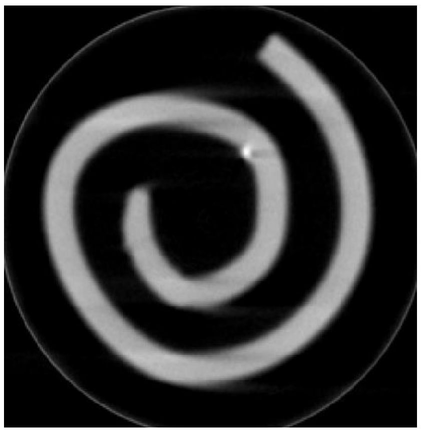
$$CNR_{ROI} = (\text{Grey value}_{ROI} - \text{Grey value}_{BKG}) / (\text{Standard deviation of background})$$

RTS reconstructor Image



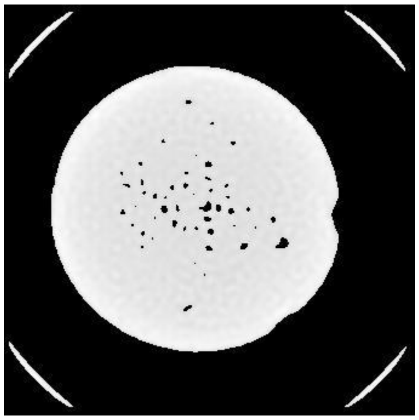
CNR = 21.82

Other reconstructor image



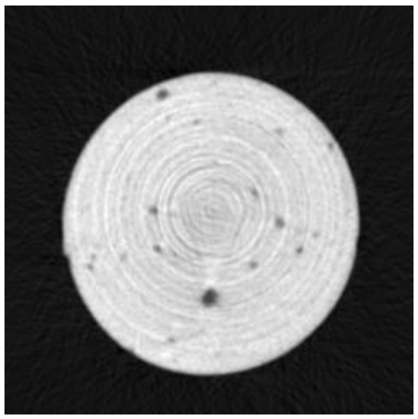
CNR = 17.41

RTS reconstructor Image



CNR = 58.68

Other reconstructor image



CNR = 52.79

Table comparing the values of SNR and CNR for different objects

	Concrete		Aluminium Spiral	
	RTS reconstructor	Other reconstructor	RTS reconstructor	Other reconstructor
SNR	15.93	8.47	15.52	11.77
CNR	58.68	52.79	21.82	17.41

9. CONCLUSION

The GUI created in MATLAB was compiled into a stand-alone application. It can now be used even in computers that don’t have MATLAB installed. All they have to do is install the Matlab complier, which is a small and free software.

The GUI can be used for fan beam and cone beam reconstruction. The user can easily center, filter and threshold the images for a slice. The filter can be carried forward to the whole 3D object. The cone beam reconstruction requires minimal memory. Most of the reconstructions can be done in a personal computer.

If the appropriate filter and threshold are chosen, a high correlation coefficient can be obtained. With the cone beam reconstructor, the total time required to make the 3D model of an object is significantly reduced, as the scanning time is minimal.

10.FUTURE WORK

The Cone Beam Reconstructor can be further improved by including various other methods of reconstruction. Also the filtering and the thresholding can be automated using adaptive techniques.

11.APPENDIX

The Matlab codes

I] Fan Beam projection data for circle with hole

```
clear all
clc

L = 1000;                  %length of tatal area% ;
R = 400;                  %radius of circle% ;

d=5000;                   % distance of source from origin
D=10000;                  % distance of detector from source

Q=zeros(2*round(D*L/(2*d-L)),360);


for q=1:1:360


    P1 = zeros(1,2*round(D*L/(2*d-L)));
    for x=-round(D*L/(2*d-L)):1:round(D*L/(2*d-L));

        g1=atan(x/D);
        tt=round(x+D*L/(2*d-L)+1);

                if -R < d*sin(g1)  && R > d*sin(g1)
                    if -R/10 < d*sin(g1)  && R/10 > d*sin(g1)

           P1(tt)=2*(R*R-(d*sin(g1))^2)^(0.5)-2*(R*R/100-(d*sin(g1))^2)^(0.5);

                    else
                        P1(tt)=2*(R*R-(d*sin(g1))^2)^(0.5);
                    end

                else

                    P1(tt)=0;

                end
                Q(tt,q)=P1(tt);

    end


end
mesh(Q);
```

II] Parallel Beam reconstruction

```
f=size(Q);                          %projection data stored in Q
vm=round(f(2)/2)*2;
F= zeros(vm,vm);

for t=1:1:180
    for x=-vm/2:(vm/2)-2
       for y=-vm/2:vm/2

            u = round(x*sin(pi*t/180) + y*cos(pi*t/180)) + vm/2;

            v = round(y*sin(pi*t/180) - x*cos(pi*t/180)) + vm/2;

            if u>0 && u<vm
                if v>0 && v<vm

                    F(u,v) = F(u,v) + Q(t,x+vm/2+1);

                end
            end
        end
    end
end

figure

imshow(F,[min(min(F)) max(max(F))])
```

III] Fan Beam reconstruction

```
D=10000;                                        %Distance of source from detector
d=5000;                                         %Distance of source from origin
f=size(Q);                                      % Projection data stored in Q
th=320;                                         % number of projections

xl=round(f(1)/2)*2;
xL=2*f(1)*d/(2*D+f(1));
F= zeros(xL,xL);

tim=inf;
for t= 1:360/th:360

    si=sin(pi*(t-1)/180);
    co =cos(pi*(t-1)/180);


    clc
    per=t/360*100;
    fprintf('%s %f %s %f %s \n', 'Loading:', per,'%   Estimated time left: ',tim,'
min');
    tic;

    for i=1:t/4
        fprintf('%s','=');
    end

    fprintf('%s','>>');
       for x=-xl/2:xl/2-2
           for y=-xl/2:xl/2
               x1=(x/D)*(d+y);

                u = round(x1*si + y*co+ xL/2);
                v = round(y*si - x1*co+ xL/2);

                if u>0 && u<xL
                    if v>0 && v<xL

                            F(u,v) = F(u,v)+Q(x+xl/2+1,round(t*th/360));

                    end
               end
           end

      end

    tim=toc;
    tim=tim*(360-t)/60;

end
```

IV] Cone Beam reconstruction

```
D=155;                       %Distance of source from detector
d=100;                       %Distance of source from origin
f=size(Q);                   %Projection data stored in Q
th=320;                      % number of projections
xl=round(f(2)/2)*2;
zl=round(f(1)/2)*2;

xL=2*f(2)*d/(2*D+f(1));
zL=2*f(1)*d/(2*D+f(1));

F= zeros(xL,xL,zL);

tim=inf;
for t= 1:360/th:360

    si=sin(pi*(t-1)/180);
    co =cos(pi*(t-1)/180);


    clc
    per=t/360*100;
fprintf('%s %f %s %f %s \n', 'Loading:', per,'%   Estimated time left: ',tim,'
min');

    tic;

    for i=1:t/4
        fprintf('%s','=');
    end

    fprintf('%s','>>');

    for z=-zl/2:(zl/2)-2

       for x=-xl/2:xl/2-2

           for y=-xl/2:xl/2

               x1=(x/D)*(d+y);

               z1=(z/D)*(d+y);


                u = round(x1*si + y*co+ xL/2);

                v = round(y*si - x1*co+ xL/2);

                w = round(z1 + zL/2);
                if u>0 && u<xL
                    if v>0 && v<xL
                        if w>0 && w<zL
                        F(u,v,w) = F(u,v,w)+Q(z+zl/2+1,x+xl/2+1,round(t*th/360));

                        end
                    end
               end
           end
        end
    end


    tim=toc;
    tim=tim*(360-t)/60;

end
```

V] GUI for converting images to .mat

```
function varargout = img2mat(varargin)
% IMG2MAT M-file for img2mat.fig
%      IMG2MAT, by itself, creates a new IMG2MAT or raises the existing
%      singleton*.
%
%      H = IMG2MAT returns the handle to a new IMG2MAT or the handle to
%      the existing singleton*.
%
%      IMG2MAT('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in IMG2MAT.M with the given input arguments.
%
%      IMG2MAT('Property','Value',...) creates a new IMG2MAT or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before img2mat_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to img2mat_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help img2mat

% Last Modified by GUIDE v2.5 04-Jul-2015 13:53:43

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @img2mat_OpeningFcn, ...
                   'gui_OutputFcn',  @img2mat_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before img2mat is made visible.
function img2mat_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to img2mat (see VARARGIN)

% Choose default command line output for img2mat
handles.output = hObject;
global str1;
global str2;
global num1;
global num2;
global dum3;
dum3=1;

set(handles.eta2,'String','Choose if you want Mat to images, or images to Mat');

set(handles.pushbutton1, 'Visible', 'Off');
set(handles.pushbutton2, 'Visible', 'Off');
set(handles.pushbutton3, 'Visible', 'Off');
set(handles.pushbutton4, 'Visible', 'Off');
set(handles.pushbutton5, 'Visible', 'Off');

set(handles.edit1, 'Visible', 'Off');
set(handles.edit2, 'Visible', 'Off');
set(handles.edit3, 'Visible', 'Off');
set(handles.edit4, 'Visible', 'Off');
set(handles.text1, 'Visible', 'Off');
set(handles.text2, 'Visible', 'Off');
set(handles.text3, 'Visible', 'Off');
set(handles.text4, 'Visible', 'Off');
set(handles.text5, 'Visible', 'Off');


str1='Img';
str2='.jpg';
num1=1;
num2=364;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes img2mat wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = img2mat_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)

global dim;
global xL;
global yL;
global folder;
```

```
[baseFileName, folder] = uigetfile('*.*', 'Specify an image file');
if baseFileName == 0
    % User clicked the Cancel button.
    return;
end
fullFileName = fullfile(folder, baseFileName)

I = imread(fullFileName);

axes(handles.axes1);
imshow(I,[min(min(I)) max(max(I))]);

set(handles.eta2,'String','Now drag and select the region to crop. Right click and
click on "crop image" to confirm ');

[Q,dim]=imcrop(I);
f=size(Q);
xL=f(1);
yL=f(2);


set(handles.edit1, 'Visible', 'On');
set(handles.edit2, 'Visible', 'On');
set(handles.edit3, 'Visible', 'On');
set(handles.edit4, 'Visible', 'On');

set(handles.text1, 'Visible', 'On');
set(handles.text2, 'Visible', 'On');
set(handles.text3, 'Visible', 'On');
set(handles.text4, 'Visible', 'On');
set(handles.text5, 'Visible', 'On');
set(handles.eta2,'String','Now enter the name of the images and press "Convert to
Mat" . Make sure the file format is correct');

set(handles.pushbutton2, 'Visible', 'On');
axes(handles.axes1);
imshow(Q,[min(min(Q)) max(max(Q))]);

% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global str1;
% Hints: get(hObject,'String') returns contents of edit1 as text
str1=get(hObject,'String');


% --- Executes during object creation, after setting all properties.
```

```matlab
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global num1;
% Hints: get(hObject,'String') returns contents of edit2 as text
num1 = str2double(get(hObject,'String'));


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global num2;
% Hints: get(hObject,'String') returns contents of edit2 as text
num2 = str2double(get(hObject,'String'));


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global str2;
str2=get(hObject,'String');
%        str2double(get(hObject,'String')) returns contents of edit4 as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
global str1;
global str2;
global num1;
global num2;
global dim;
global xL;
global yL;
global Q;
global folder;

display(num1);
Q=zeros(xL,yL,num2-num1+1);

for z=num1:num2;


    g=strcat(str1,int2str(z),str2);
    fullFileName = fullfile(folder,g)
    I=imread(fullFileName);
    S=imcrop(I,dim);

    for x=1:xL
        for y=1:yL
            Q(x,y,z)=S(x,y);
        end
    end

 set(handles.eta2,'String',horzcat('Image No: ',int2str(z)));
 axes(handles.axes1);
imshow(S,[min(min(S)) max(max(S))]);
 pause(.1);


end
set(handles.pushbutton3,'Visible','On');
set(handles.eta2,'String','Loading complete, Now click on "Save as Mat"');









% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)

global Q;


uisave('Q');
set(handles.eta2,'String','Saved Sucessfully. Now you can close this window');
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function eta2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to eta (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
global dum3;

set(handles.pushbutton1, 'Visible', 'Off');
set(handles.pushbutton2, 'Visible', 'Off');
set(handles.pushbutton3, 'Visible', 'Off');
set(handles.pushbutton4, 'Visible', 'Off');
set(handles.pushbutton5, 'Visible', 'Off');

set(handles.edit1, 'Visible', 'Off');
set(handles.edit2, 'Visible', 'Off');
set(handles.edit3, 'Visible', 'Off');
set(handles.edit4, 'Visible', 'Off');
set(handles.text1, 'Visible', 'Off');
set(handles.text2, 'Visible', 'Off');
set(handles.text3, 'Visible', 'Off');
set(handles.text4, 'Visible', 'Off');
set(handles.text5, 'Visible', 'Off');

sa=get(hObject,'Value');
if sa==1
dum3=1;
set(handles.radiobutton2, 'Value', 0);
set(handles.eta2,'String','Images to Mat selected');
pause(.5);
set(handles.pushbutton1, 'Visible', 'On');
set(handles.pushbutton4, 'Visible', 'Off');


set(handles.eta2,'String','Now load an image to crop');

end
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton1


% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton2 (see GCBO)

global dum3;

set(handles.pushbutton1, 'Visible', 'Off');
set(handles.pushbutton2, 'Visible', 'Off');
set(handles.pushbutton3, 'Visible', 'Off');
set(handles.pushbutton4, 'Visible', 'Off');
set(handles.pushbutton5, 'Visible', 'Off');

set(handles.edit1, 'Visible', 'Off');
set(handles.edit2, 'Visible', 'Off');
set(handles.edit3, 'Visible', 'Off');
set(handles.edit4, 'Visible', 'Off');
set(handles.text1, 'Visible', 'Off');
set(handles.text2, 'Visible', 'Off');
set(handles.text3, 'Visible', 'Off');
set(handles.text4, 'Visible', 'Off');
set(handles.text5, 'Visible', 'Off');


sa=get(hObject,'Value');
if sa==1
dum3=2;
set(handles.radiobutton1, 'Value', 0);
set(handles.eta2,'String','Mat to images selected');
pause(.5);
set(handles.pushbutton4, 'Visible', 'On');
set(handles.pushbutton1, 'Visible', 'Off');


set(handles.eta2,'String','Now load the mat file');
end
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton2


% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)


% hObject    handle to pushbutton4 (see GCBO)
startingFolder = pwd;
global num2;
% Get the name of the mat file that the user wants to use.
defaultFileName = fullfile(startingFolder, '*.mat');
[baseFileName, folder] = uigetfile(defaultFileName, 'Select a mat file');
if baseFileName == 0
    % User clicked the Cancel button.
    return;
end
fullFileName = fullfile(folder, baseFileName)

I = load(fullFileName);
set(handles.eta2,'String',horzcat('Loaded ',baseFileName));
pause(.1);
global F;
try
F=I.Fi;
catch
F=I.Q;
end
f=size(F);
S=zeros(f(1),f(2));
num2=f(3);
set(handles.edit3,'String',int2str(num2));
t=round(f(3)/2);
```

```
for x=1:f(1)
    for y=1:f(2)
        S(x,y)=F(x,y,t);
    end
end
axes(handles.axes1);
imshow(S,[min(min(S)) max(max(S))]);

set(handles.edit1, 'Visible', 'On');
set(handles.edit2, 'Visible', 'On');
set(handles.edit3, 'Visible', 'On');
set(handles.edit4, 'Visible', 'On');

set(handles.text1, 'Visible', 'On');
set(handles.text2, 'Visible', 'On');
set(handles.text3, 'Visible', 'On');
set(handles.text4, 'Visible', 'On');
set(handles.text5, 'Visible', 'On');

set(handles.pushbutton5, 'Visible', 'On');

set(handles.eta2,'String',horzcat(int2str(f(3)),' Slices Found! ','Now select the
name of the images to be saved as'));

% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)

global str1;
global str2;
global num1;
global num2;


global F;
F=F./(max(max(max(F))));
startingFolder = pwd;

% Get the name of the mat file that the user wants to use.
defaultFileName = fullfile(startingFolder,str2);
folder = uigetdir(defaultFileName, 'Select the folder to save images to');

f=size(F);
S=zeros(f(1),f(2));
for x=num1:num2
    for y=1:f(1)
        for z=1:f(2)
            S(y,z)=F(y,z,x);
        end
    end
```

```
baseFileName=strcat(str1,int2str(x),str2);


fullFileName = fullfile(folder, baseFileName)

imwrite(S,fullFileName);

set(handles.eta2,'String',horzcat(baseFileName,' Saved Sucessully!'));
pause(.01);

end

set(handles.eta2,'String','All files Saved Sucessully! You can now close this
window');
```

V] Final GUI for Cone beam reconstruction

```
function varargout = Coneinv(varargin)
% CONEINV M-file for Coneinv.fig
%      CONEINV, by itself, creates a new CONEINV or raises the existing
%      singleton*.
%
%      H = CONEINV returns the handle to a new CONEINV or the handle to
%      the existing singleton*.
%
%      CONEINV('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CONEINV.M with the given input arguments.
%
%      CONEINV('Property','Value',...) creates a new CONEINV or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Coneinv_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Coneinv_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Coneinv

% Last Modified by GUIDE v2.5 07-Jul-2015 14:47:17

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Coneinv_OpeningFcn, ...
                   'gui_OutputFcn',  @Coneinv_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Coneinv is made visible.
function Coneinv_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Coneinv (see VARARGIN)

% Choose default command line output for Coneinv
handles.output = hObject;
%clc
global player;

%try

%[y,Fs] = wavread('Flight.wav');
%player=audioplayer(y,Fs);

%end

try
```

```
play(player);
    display('song');
end


global flag;
global cen;
global dum3;
global orde;
global orde2;
orde2=1;
orde=1;
dum3=1;
cen=0;
flag=0;


% Update handles structure
guidata(hObject, handles);
set(handles.eta,'String','                    Start by loading a .mat file
Or converting images to .mat');

%%%%%%%%%%%%%%%%%%%%%hide%%%%%%%%%%%%%%%%%



set(handles.pushbutton2 , 'Visible', 'Off');
set(handles.pushbutton3 , 'Visible', 'Off');
set(handles.pushbutton4 , 'Visible', 'Off');
set(handles.pushbutton5 , 'Visible', 'Off');
set(handles.pushbutton7 , 'Visible', 'Off');
set(handles.pushbutton10 , 'Visible', 'Off');
set(handles.pushbutton12 , 'Visible', 'Off');
set(handles.pushbutton18 , 'Visible', 'Off');
set(handles.pushbutton19 , 'Visible', 'Off');

set(handles.radiobutton1, 'Visible', 'Off');
set(handles.radiobutton2, 'Visible', 'Off');

set(handles.text1, 'Visible', 'Off');
set(handles.text2, 'Visible', 'Off');
set(handles.text3, 'Visible', 'Off');
set(handles.text5, 'Visible', 'Off');
set(handles.text6, 'Visible', 'Off');
set(handles.text9, 'Visible', 'Off');
set(handles.text10, 'Visible', 'Off');
set(handles.text11, 'Visible', 'Off');
set(handles.text12, 'Visible', 'Off');
set(handles.text13, 'Visible', 'Off');
set(handles.text14, 'Visible', 'Off');
set(handles.text17, 'Visible', 'Off');
set(handles.text18, 'Visible', 'Off');
set(handles.text19, 'Visible', 'Off');

set(handles.ax1, 'Visible', 'Off');
set(handles.ax2, 'Visible', 'Off');

set(handles.edit1, 'Visible', 'Off');
set(handles.edit2, 'Visible', 'Off');
set(handles.edit3, 'Visible', 'Off');
set(handles.edit4, 'Visible', 'Off');
set(handles.edit6, 'Visible', 'Off');
set(handles.edit7, 'Visible', 'Off');
set(handles.edit8, 'Visible', 'Off');

set(handles.axes1, 'Visible', 'Off');
set(handles.axes2, 'Visible', 'Off');

set(handles.slider1, 'Visible', 'Off');
set(handles.slider2, 'Visible', 'Off');
set(handles.slider3, 'Visible', 'Off');
set(handles.slider4, 'Visible', 'Off')

set(handles.popupmenu1, 'Visible', 'Off');
set(handles.popupmenu2, 'Visible', 'Off');

global ang;
ang=360;

% UIWAIT makes Coneinv wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = Coneinv_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
startingFolder = pwd;


% Get the name of the mat file that the user wants to use.
defaultFileName = fullfile(startingFolder, '*.mat');
[baseFileName, folder] = uigetfile(defaultFileName, 'Select a mat file');
if baseFileName == 0
    % User clicked the Cancel button.
    return;
end
fullFileName = fullfile(folder, baseFileName)

I = load(fullFileName);
```

```
set(handles.eta,'String',horzcat('Loaded ',baseFileName));
pause(.1);
set(handles.eta,'String','Select if the data is for 180 or 360');
global Q1;

try
Q1=I.Q;
catch
    Q1=I.Fi;
end

f=size(Q1);
S=zeros(f(1),f(2));
for x=1:f(1)
    for y=1:f(2)
        S(x,y)=Q1(x,y,1);
    end
end
axes(handles.axes1);
imshow(S,[min(min(S)) max(max(S))]);
for x=1:f(1)
    for y=1:f(2)
        S(x,y)=Q1(x,y,round(f(3)/4));
    end
end
axes(handles.axes2);
imshow(S,[min(min(S)) max(max(S))]);
set(handles.radiobutton1, 'Visible', 'On');
set(handles.radiobutton2, 'Visible', 'On');
set(handles.pushbutton18 , 'Visible', 'On');
set(handles.ax1, 'Visible', 'On');
set(handles.ax2, 'Visible', 'On');
set(handles.ax1,'String','Projection at zero degree');
set(handles.ax2,'String','Projection at 90 degree');
set(handles.text5,'String',horzcat('Slice no: 1-',int2str(f(1))));

% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)


sa=get(hObject,'Value');
if sa==1
ang=360;
set(handles.radiobutton2, 'Value', 0);
set(handles.eta,'String','360 selected');
pause(1);
set(handles.pushbutton12 , 'Visible', 'On');
set(handles.edit1 , 'Visible', 'On');
set(handles.text1 , 'Visible', 'On');
set(handles.text6 , 'Visible', 'On');
set(handles.eta,'String','             Now Invert color if the object if black.
If the object is white, Enter the distance variables.');
end
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton1


% --- Executes on button press in radiobutton2.
function radiobutton2_Callback(hObject, eventdata, handles)
global ang;
sa=get(hObject,'Value');
if sa==1
ang=180;
set(handles.radiobutton1, 'Value', 0);
set(handles.eta,'String','180 selected');
pause(1);
set(handles.pushbutton12 , 'Visible', 'On');
set(handles.edit1 , 'Visible', 'On');
set(handles.text1 , 'Visible', 'On');
set(handles.text6 , 'Visible', 'On');
set(handles.eta,'String','             Now Invert color if the object if black.
If the object is white, Enter the distance variables.');
end
% hObject    handle to radiobutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton2


% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)

global player;
try

play(player);

end

global Qsli;
global cen;
global flag;
global Qsli3;
cen=get(hObject,'Value');
try

play(player);

end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%5
Qsli3=Qsli;

f=size(Qsli3);
S=zeros(f(1),f(2));

for x=1:f(1)

    if flag==1
        flag=0;
        return;

    end

    for y=1:f(2)
        u=round(x-cen*100);
        if u>0 && u<f(1)
        S(u,y)=Qsli3(x,y);
        end
    end
end
Qsli3=S;

 axes(handles.axes2);
 imshow(S,[min(min(S)) max(max(S))]);
 set(handles.eta,'String',horzcat('Done!!',' moved by ',int2str(cen*100),'
pixels'));


% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
set(handles.eta,'String','Please wait');
```

```
pause(.1);
global F;
global J;
global filt;
global dum;
global cu;
global flag;
global orde;

dum=1;
if strcmp(filt,'Ram-Lak')
    dum=2;
elseif strcmp(filt,'Butterworth')
    dum=3;
elseif strcmp(filt,'Ram+Butter')
    dum=4;
elseif strcmp(filt,'Ram+Hamming')
    dum=5;
end


f=size(F);
plo=zeros(f(1),1);

J=zeros(f(1),f(2));
s=get(hObject,'Value');
cu=f(1)*s;

for x=1:cu
    if dum==1
        plo(x)=1;
    elseif dum==2

        plo(x)=x;

    elseif dum==3

        plo(x)=1/(1+(x/cu)^(2*orde))^.5;

    elseif dum==4

        plo(x)=(x/(1+(x/cu)^(2*orde))^.5);
    elseif dum==5

        plo(x)=(x*cos(pi*x/(2*cu)));

    end
end
 axes(handles.axes1);
 plot(plo);
  set(handles.ax1,'String','Filter Plot');

J=F;
```

```
 J=fftshift(fft2(J));

 if flag==1
     flag=0;
     return;

 end

 for x=1:f(1)
     for y=1:f(2)
         tx=x-f(1)/2;
         ty=y-f(2)/2;
         tt=(tx*tx+ty*ty)^(.5);

         if dum~=1

         if -cu <tt && cu >tt

             if dum==2

                 J(x,y)=J(x,y)*tt;

             elseif dum==3

                  J(x,y)=J(x,y)/(1+(tt/cu)^(2*orde))^.5;

             elseif dum==4

                  J(x,y)=J(x,y)*tt/(1+(tt/cu)^(2*orde))^.5;

                  (x*cos(pi*x/(2*cu)));
             elseif dum==5

                  J(x,y)=J(x,y)*(tt*cos(pi*tt/(2*cu)));
             end

         else
             J(x,y)=0;
         end
         end

     end

 end

 J=real(ifft2(ifftshift(J)));

J=J-min(min(J));
J=J./max(max(J));

ima=max(J(:));
  imi=min(J(:));
```

```
     ims=std(J(:));
     snr=20*log10((ima-imi)./ims);

 axes(handles.axes2);
 imshow(J);
 pause(1);
 axes(handles.axes1);
 imshow(F,[min(min(F)) max(max(F))]);


 set(handles.ax1,'String','Original Image');
 set(handles.ax2,'String','Filtered Image');

 set(handles.eta,'String',horzcat('                           Done!',' Cutoff
',int2str(100*s),' %  SNR: ',num2str(snr),'   Now set the threshold until the noise
has been removed'));
set(handles.slider3, 'Visible', 'On');
set(handles.text12, 'Visible', 'On');
 % hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider






% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
global Qsli3;
global player;

global ang;
global D1;  %Distance of source from detector
global d1;   %Distance of source from origin
global flag;
D=1000*D1/pi;
d=1000*d1/pi;
f=size(Qsli3);
th=f(2);  % number of projections
global F;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global dum3;
global cu2;
global flag;
global orde2;
```

```
  axes(handles.axes1);
 imshow(Qsli3,[min(min(Qsli3)) max(max(Qsli3))]);


f=size(Qsli3);

J=zeros(f(1),f(2));


J=Qsli3;

    if dum3~=1;

    J=fftshift(fft2(J));

    if flag==1
        flag=0;
        return;

    end

    for x=1:f(1)
        for y=1:f(2)
            tx=x-f(1)/2;
            ty=y-f(2)/2;
            tt=(tx*tx+ty*ty)^(.5);

            if dum3~=1

            if -cu2 <tt && cu2 >tt

                if dum3==2

                    J(x,y)=J(x,y)*tt;

                elseif dum3==3

                     J(x,y)=J(x,y)/(1+(tt/cu2)^(2*orde2))^.5;

                elseif dum3==4

                     J(x,y)=J(x,y)*tt/(1+(tt/cu2)^(2*orde2))^.5;

                elseif dum3==5

                     J(x,y)=J(x,y)*(tt*cos(pi*tt/(2*cu2)));

                end

            else
```

```
                J(x,y)=0;

            end
            end


        end

    end

    J=real(ifft2(ifftshift(J)));

   J=J-min(min(J));
   J=J./max(max(J));


   end

Qsli3=J;


xl=round(f(1)/2)*2;
xL=2*f(1)*d/(2*D+f(1));
F= zeros(xL,xL);

tim=inf;
for t= 1:ang/th:ang
  try

play(player);

end

     if flag==1
         flag=0;
        return;

     end


    si=sin(pi*(t-1)/180);
    co =cos(pi*(t-1)/180);


    clc
    per=t/ang*100;
   % fprintf('%s %f %s %f %s \n', 'Loading:', per,'%   Estimated time left: ',tim,'
min');

    tic;
```

```
        for x=-xl/2:xl/2-2

            for y=-xl/2:xl/2

                x1=(x/D)*(d+y);

                 u = round(x1*si + y*co+ xL/2);

                 v = round(y*si - x1*co+ xL/2);

                 if u>0  && u<xL
                     if v>0 && v<xL


                             F(u,v) = F(u,v)+Qsli3(x+xl/2+1,round(t*th/ang));


                     end
                end
            end

       end


     axes(handles.axes1);
     imshow(F,[min(min(F)) max(max(F))]);

       tim=toc;
     tim=tim*(ang-t)/60;

     set(handles.eta,'String',horzcat('Loading: ',int2str(per),' % ',' Estimated
time left: ',int2str(floor(tim)),' min ',int2str(tim*60-60*floor(tim)),' sec'));

end

 set(handles.eta,'String','Done');
 set(handles.ax1,'String','Reconstructed Slice');
set(handles.popupmenu1, 'Visible', 'On');
set(handles.slider2, 'Visible', 'On');
set(handles.popupmenu1, 'Visible', 'On');
set(handles.text11, 'Visible', 'On');



axes(handles.axes1);
 imshow(F,[min(min(F)) max(max(F))]);
 pause(1);
 set(handles.eta,'String','Now choose a Filter');
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
global Fi;


uisave('Fi');

set(handles.eta,'String','Saved');
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



function edit1_Callback(hObject, eventdata, handles)

global D1;
set(handles.edit2 , 'Visible', 'On');
set(handles.text2 , 'Visible', 'On');
set(handles.text9 , 'Visible', 'On');
D1=str2double(get(hObject,'String'));
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)

global d1;
set(handles.edit3 , 'Visible', 'On');
set(handles.text3 , 'Visible', 'On');
set(handles.text10 , 'Visible', 'On');
d1=str2double(get(hObject,'String'));
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function edit3_Callback(hObject, eventdata, handles)

global pi;
global Q1;
f=size(Q1);
pi=get(hObject,'String');



set(handles.edit8 , 'Visible', 'On');
set(handles.text19 , 'Visible', 'On');
set(handles.text19,'String',horzcat('Number of projections detected:
',int2str(f(3)),'. Change to : '));
set(handles.edit8,'String',int2str(f(3)));
set(handles.eta,'String','Now select the number of projections' );
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end




% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
global Q1;
global flag;


f=size(Q1);
G=zeros(f(1),f(2));
for z=1:f(3)
    for x=1:f(1)
        for y=1:f(2)

            G(x,y)=Q1(x,y,z);

        end
    end

axes(handles.axes2);
    imshow(G,[min(min(G)) max(max(G))]);
    pause(.1);

    if flag==1
         flag=0;
        return;

    end

end

% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)




function edit4_Callback(hObject, eventdata, handles)
global slino;
global Q1;
global cen;
global flag;
global Qsli2;
global numang;
slino=str2double(get(hObject,'String'));
f=size(Q1);
global Qsli
Qsli=zeros(f(2),numang);

for x=1:f(2)
```

```
        for y=1:numang
            Qsli(x,y)=Q1(slino,x,y);
        end
    end
    set(handles.eta,'String','Auto centering please wait');
    pause(.1);
    fde=0;
    ma=max(max(Qsli));
    if ma~=0
    Qsli=Qsli./ma;
    end
    h=size(Qsli);
    xc2=h(1)-1;
    xc1=0;
    for x=1:round(h(1)/2)
        for y=1:h(2)
            if Qsli(x,y)>.3

                xc1=x;
                fde=1;
                break;
            end
        end

        if fde==1
            break
        end
    end
    fde=0;
    for x=round(h(1)):-1:round(h(1)/2)
        for y=1:h(2)
            if Qsli(x,y)>.3


                xc2=x;
                fde=1;
                break;
            end
        end

        if fde==1
            break
        end
    end

    cen=(round((xc1+xc2-h(1))/2)+1)/100;

     axes(handles.axes1);
     imshow(Qsli,[min(min(Qsli)) max(max(Qsli))]);


    Qsli2=Qsli;

    f=size(Qsli2);
    S=zeros(f(1),f(2));
```

```
for x=1:f(1)

    if flag==1
        flag=0;
        return;

    end

    for y=1:f(2)
        u=round(x-cen*100);
        if u>0 && u<f(1)
        S(u,y)=Qsli2(x,y);
        end
    end
end
Qsli2=S;

 axes(handles.axes2);
 imshow(S,[min(min(S)) max(max(S))]);
 set(handles.slider1, 'Visible', 'On');
 set(handles.text13, 'Visible', 'On');
 set(handles.pushbutton2, 'Visible', 'On');
 set(handles.slider4, 'Visible', 'On');
 set(handles.text18, 'Visible', 'On');
 set(handles.popupmenu2, 'Visible', 'On');
 set(handles.ax1,'String','Before centering');
 set(handles.ax2,'String','After centering');
set(handles.eta,'String',horzcat('Auto centering Complete. Press apply if the
sinogram is the center, or use the scroll bar and bring it to the center.
(',int2str(cen*100),' pixels)'));

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global filt
 contents = cellstr(get(hObject,'String'));
 filt=contents{get(hObject,'Value')} ;
 if strcmp(filt,'Butterworth')
     set(handles.text14, 'Visible', 'On');
     set(handles.edit6, 'Visible', 'On');
     set(handles.eta, 'String', 'Now choose the order. Then set the cutoff in the
blue scrollbar until the image is sharp, with no noise.');
 elseif strcmp(filt,'Ram+Butter')
     set(handles.text14, 'Visible', 'On');
     set(handles.edit6, 'Visible', 'On');
     set(handles.eta, 'String', 'Now choose the order. Then set the cutoff in the
blue scrollbar until the image is sharp, with no noise.');
 else
     set(handles.text14, 'Visible', 'Off');
     set(handles.edit6, 'Visible', 'Off');
     set(handles.eta, 'String', 'Now  set the cutoff in the blue scrollbar until
the image is sharp, with no noise.')
 end




% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
function slider3_Callback(hObject, eventdata, handles)

set(handles.eta,'String','Please wait');
pause(.1);
global thresh;
thresh=get(hObject,'Value');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global J;
global F1;
f=size(J);
F1=zeros(f(1),f(2));

F1=J;
```

```
   for x=1:f(1)
       for y=1:f(2)
           if F1(x,y)<thresh
               F1(x,y)=0;
           end
       end
   end
%F1=histeq(F1);
   axes(handles.axes2);
 imshow(F1,[min(min(F1)) max(max(F1))]);

 set(handles.pushbutton10 , 'Visible', 'On');
 set(handles.eta,'String',horzcat('Done!!',' threshold ',int2str(100*thresh),' %Now
if the final image is sharp and represents the actual object, click on Apply
centering to Sinogram. Otherwise, choose a slice again and repeat the process!'));


 % hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
set(handles.eta,'String','ConeBeam Reconstruction has been initiated. This will
take time depending on the size of the images. Please be Patient!');
pause(.2);
global Fi;
global player;

global Q1;
global ang;
global D1;  %Distance of source from detector
global d1;   %Distance of source from origin
global flag;
```

```
D=1000*D1/pi;
d=1000*d1/pi;
f=size(Q1);
th=f(3);  % number of projections



xl=round(f(2)/2)*2;
zl=round(f(1)/2)*2;

xL=2*f(2)*d/(2*D+f(1));
zL=2*f(1)*d/(2*D+f(1));

Fi= zeros(xL,xL,zL);
S=zeros(xL);
tim=inf;
for t= 1:ang/th:ang

  try

play(player);

end



    if flag==1
        flag=0;
        return;

    end

    si=sin(pi*(t-1)/180);
    co =cos(pi*(t-1)/180);



    per=t/ang*100;
    %fprintf('%s %f %s %f %s \n', 'Loading:', per,'%   Estimated time left: ',tim,'
min');

    tic;



    for z=-zl/2:(zl/2)-2

       for x=-xl/2:xl/2-2

           for y=-xl/2:xl/2
```

```
                x1=(x/D)*(d+y);

                z1=(z/D)*(d+y);


                 u = round(x1*si + y*co+ xL/2);

                 v = round(y*si - x1*co+ xL/2);

                 w = round(z1 + zL/2);



                 if u>0  && u<xL
                     if v>0  && v<xL
                         if w>0  && w<zL

                                Fi(u,v,w)  =
Fi(u,v,w)+Q1(z+zl/2+1,x+xl/2+1,round(t*th/ang));

                         end
                     end
                end
            end
         end
     end

     for x7=1:xL
         for y7=1:xL
             S(x7,y7)=Fi(x7,y7,round(zL/2));
         end
     end

      axes(handles.axes1);
     imshow(S,[min(min(S)) max(max(S))]);

       tim=toc;
     tim=tim*(ang-t)/60;

     set(handles.eta,'String',horzcat('Loading: ',int2str(per),' % ',' Estimated
time left: ',int2str(floor(tim)),' min ',int2str(tim*60-60*floor(tim)),' sec'));

end


for x7=1:xL
        for y7=1:xL
            S(x7,y7)=Fi(x7,y7,round(zL/2));
        end
end
axes(handles.axes1);
 imshow(S,[min(min(S)) max(max(S))]);
  set(handles.pushbutton3 , 'Visible', 'On');
```

```
   set(handles.pushbutton7 , 'Visible', 'On');
   set(handles.pushbutton19 , 'Visible', 'On');
  set(handles.eta,'String','Reconstruction Done');
  pause(1);
  set(handles.eta,'String','Now you can directly save the 3D image as a Matfile or
apply the filter and save it . ');

 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
global flag;
set(handles.eta,'String','Cancelled!!');
flag=1;

% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 set(handles.eta,'String','Applying Filters');
 pause(.1);
global Fi;
   %Distance of source from origin
global thresh;
global dum;
global cu;
global orde
```

```
f=size(Fi);

xL=f(1);
zL=f(3);

S=zeros(xL);

for z7=1:zL

     per=z7/zL*100;
     tic;

    for x7=1:xL
        for y7=1:xL
            S(x7,y7)=Fi(x7,y7,z7);
        end
    end




    S=fftshift(fft2(S));



    for x=1:f(1)
        for y=1:f(2)
            tx=x-f(1)/2;
            ty=y-f(2)/2;
            tt=(tx*tx+ty*ty)^(.5);


            if -cu <tt && cu >tt

                if dum==2

                    S(x,y)=S(x,y)*tt;

                elseif dum==3

                     S(x,y)=S(x,y)/(1+(tt/cu)^(2*orde))^.5;

                elseif dum==4

                     S(x,y)=S(x,y)*tt/(1+(tt/cu)^(2*orde))^.5;

                elseif dum==5

                     S(x,y)=S(x,y)*tt*cos(pi*tt/(2*cu));

                end
            else
               S(x,y)=0;
            end


        end

    end

    S=real(ifft2(ifftshift(S)));

   S=S-min(min(S));
   S=S./max(max(S));

   for x9=1:xL
       for y9=1:xL
           if S(x9,y9)<thresh
               S(x9,y9)=0;
           end
       end
   end

%   S=histeq(S);


    for x7=1:xL
        for y7=1:xL
            Fi(x7,y7,z7)=S(x7,y7);
        end
    end

    tim=toc;
    tim=tim*(zL-z7)/60;

    set(handles.eta,'String',horzcat('Applying Filter: ',int2str(per),' % ','
Estimated time left: ',int2str(floor(tim)),' min ',int2str(tim*60-60*floor(tim)),'
sec'));
 end


 for x7=1:xL
        for y7=1:xL
            S(x7,y7)=Fi(x7,y7,round(zL/2));
        end
    end
axes(handles.axes1);
imshow(S,[min(min(S)) max(max(S))]);
 set(handles.eta,'String','Done');


 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)

clear all;
clc;





% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
global numang;
global cen;
global Q1;
global flag;
set(handles.eta,'String','Please wait');
pause(.1);
f=size(Q1);
S=zeros(f(1),f(2),numang);





for z=1:f(1)
    for x=1:f(2)

        if flag==1
            flag=0;
            return;

        end

        for y=1:f(3)
            u=round(x-cen*100);

            if u>0 && u<f(2)
            S(z,u,y)=Q1(z,x,y);
            end


        end
    end
end
Q1=S;
clear S;
```

```
set(handles.eta,'String','Applying Filters');
 pause(.1);

   %Distance of source from origin

global dum3;
global cu2;
global orde2;

f=size(Q1);

xL=f(1);
yL=f(2)
zL=f(3);

if dum3~=1

S=zeros(yL,zL);

for z7=1:xL

     per=z7/zL*100;


    for x7=1:yL
        for y7=1:zL
            S(x7,y7)=Q1(z7,x7,y7);
        end
    end




    S=fftshift(fft2(S));



    for x=1:f(2)
        for y=1:f(3)
            tx=x-f(2)/2;
            ty=y-f(3)/2;
            tt=(tx*tx+ty*ty)^(.5);


            if -cu2 <tt && cu2 >tt

                if dum3==2

                    S(x,y)=S(x,y)*tt;

                elseif dum3==3
```

```
                          S(x,y)=S(x,y)/(1+(tt/cu2)^(2*orde2))^.5;

                     elseif dum3==4

                          S(x,y)=S(x,y)*tt/(1+(tt/cu2)^(2*orde2))^.5;

                     elseif dum3==5

                          S(x,y)=S(x,y)*tt*cos(pi*tt/(cu2*2));

                     end

                 else
                    S(x,y)=0;
                 end


             end

         end

         S=real(ifft2(ifftshift(S)));

        S=S-min(min(S));
        S=S./max(max(S));


         for x7=1:yL
             for y7=1:zL

                 Q1(z7,x7,y7)=S(x7,y7);
             end
         end



      end

     end

      for x7=1:f(1)
             for y7=1:f(2)
                 S(x7,y7)=Q1(x7,y7,round(f(3)/2));
             end
      end

     axes(handles.axes1);
     imshow(S,[min(min(S)) max(max(S))]);

     set(handles.pushbutton5 , 'Visible', 'On');
     set(handles.eta,'String','                           Centering completed!
     Now click on Cone Beam Reconstruction');
     % hObject    handle to pushbutton10 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)

img2mat;


% hObject    handle to pushbutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton12.
function pushbutton12_Callback(hObject, eventdata, handles)
global Q1;
global flag;
set(handles.eta,'String','Please Wait');
pause(.1);
f=size(Q1);
r=max(max(max(Q1)));
Q1=Q1./r;
S=zeros(f(1),f(2));

for z=1:f(3);
    for x=1:f(1)
        for y=1:f(2)
            S(x,y)=Q1(x,y,z);
         end
    end
            S=imcomplement(S);
     for x=1:f(1)
        for y=1:f(2)
            Q1(x,y,z)= S(x,y);
         end
     end

   if flag==1
        flag=0;
        return;
   end
end

S=zeros(f(1),f(2));
for x=1:f(1)
    for y=1:f(2)
        S(x,y)=Q1(x,y,1);
    end
end
axes(handles.axes1);
imshow(S,[min(min(S)) max(max(S))]);
for x=1:f(1)
    for y=1:f(2)
```

```
        S(x,y)=Q1(x,y,round(f(3)/4));
    end
end
axes(handles.axes2);
imshow(S,[min(min(S)) max(max(S))]);


set(handles.eta,'String','Inversion Complete! Now continue adding the data');

% hObject    handle to pushbutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.




function edit6_Callback(hObject, eventdata, handles)
global orde;
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
orde= str2double(get(hObject,'String'));


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes when figure1 is resized.
function figure1_ResizeFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function pushbutton2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% --- Executes on button press in pushbutton13.
function pushbutton13_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton15.
function pushbutton15_Callback(hObject, eventdata, handles)
img2mat;
% hObject    handle to pushbutton15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on slider movement.
function slider4_Callback(hObject, eventdata, handles)


set(handles.eta,'String','Please wait');
pause(.1);

global Q1;
global filt2;
global dum3;
global cu2;
global flag;
global orde2;


g=size(Q1);
Qsli2=zeros(g(1),g(2));
for x=1:g(1)
    for y=1:g(2)
        Qsli2(x,y)=Q1(x,y,1);
    end
end

  axes(handles.axes1);
 imshow(Qsli2,[min(min(Qsli2)) max(max(Qsli2))]);


dum3=1;
if strcmp(filt2,'Ram-Lak')
    dum3=2;
elseif strcmp(filt2,'Butterworth')
    dum3=3;
elseif strcmp(filt2,'Ram+Butter')
    dum3=4;
elseif strcmp(filt2,'Ram+Hamming')
    dum3=5;
end
```

```
f=size(Qsli2);
plo=zeros(f(1),1);

J=zeros(f(1),f(2));
s=get(hObject,'Value');
cu2=f(1)*s;

for x=1:cu2
    if dum3==1
        plo(x)=1;
    elseif dum3==2

        plo(x)=x;

    elseif dum3==3

        plo(x)=1/(1+(x/cu2)^(2*orde2))^.5;

    elseif dum3==4

        plo(x)=(x/(1+(x/cu2)^(2*orde2))^.5);

    elseif dum3==5

        plo(x)=(x*cos(pi*x/(2*cu2)));

    end
end
 axes(handles.axes1);
 plot(plo);
  set(handles.ax1,'String','Filter Plot');

J=Qsli2;


    J=fftshift(fft2(J));

    if flag==1
        flag=0;
        return;

    end

    for x=1:f(1)
        for y=1:f(2)
            tx=x-f(1)/2;
            ty=y-f(2)/2;
            tt=(tx*tx+ty*ty)^(.5);

            if dum3~=1
            if -cu2 <tt && cu2 >tt

                if dum3==2

                    J(x,y)=J(x,y)*tt;

                elseif dum3==3

                     J(x,y)=J(x,y)/(1+(tt/cu2)^(2*orde2))^.5;

                elseif dum3==4

                     J(x,y)=J(x,y)*tt/(1+(tt/cu2)^(2*orde2))^.5;

                elseif dum3==5

                     J(x,y)=J(x,y)*(tt*cos(pi*tt/(2*cu2)));

                end

            else
                J(x,y)=0;

            end
            end


        end

    end

    J=real(ifft2(ifftshift(J)));

   J=J-min(min(J));
   J=J./max(max(J));
 axes(handles.axes2);
 imshow(J);
 pause(1);
 axes(handles.axes1);
 imshow(Qsli2,[min(min(Qsli2)) max(max(Qsli2))]);


 set(handles.ax1,'String','Original Image');
 set(handles.ax2,'String','Filtered Image');

 set(handles.eta,'String',horzcat('                           Done!',' Cutoff
',int2str(100*s),' %                        Now set the threshold until the noise
has been removed'));

% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function slider4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end



% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
global filt2
 contents = cellstr(get(hObject,'String'));
 filt2=contents{get(hObject,'Value')} ;





 if strcmp(filt2,'Butterworth')
     set(handles.text17, 'Visible', 'On');
     set(handles.edit7, 'Visible', 'On');
     set(handles.eta, 'String', 'Now choose the order. Then set the cutoff in the
yellow scrollbar until the image is sharp, with no noise.');
 elseif strcmp(filt2,'Ram+Butter')
     set(handles.text17, 'Visible', 'On');
     set(handles.edit7, 'Visible', 'On');
     set(handles.eta, 'String', 'Now choose the order. Then set the cutoff in the
yellow scrollbar until the image is sharp, with no noise.');
 else
     set(handles.text17, 'Visible', 'Off');
     set(handles.edit7, 'Visible', 'Off');
     set(handles.eta, 'String', 'Now  set the cutoff in the yellow scrollbar until
the image is sharp, with no noise.');
 end
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu2 contents as
cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu2


% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
global orde2;
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
orde2= str2double(get(hObject,'String'));
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton17.
function pushbutton17_Callback(hObject, eventdata, handles)
global Q1;
global flag;


f=size(Q1);
G=zeros(f(1),f(2));
for z=1:f(3)
    for x=1:f(1)
        for y=1:f(2)
```

```
            G(x,y)=Q1(x,y,z);

        end
    end

axes(handles.axes2);
    imshow(G,[min(min(G)) max(max(G))]);
    pause(.1);

    if flag==1
         flag=0;
        return;

    end

end
% hObject    handle to pushbutton17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton18.
function pushbutton18_Callback(hObject, eventdata, handles)

global Q1;
global flag;

set(handles.eta,'String','Click "Cancel current process" to stop');
f=size(Q1);
G=zeros(f(1),f(2));
for z=1:f(3)
    for x=1:f(1)
        for y=1:f(2)

            G(x,y)=Q1(x,y,z);

        end
    end

axes(handles.axes2);
    imshow(G,[min(min(G)) max(max(G))]);
    pause(.1);

    if flag==1
         flag=0;
        return;

    end

end
set(handles.eta,'String','Done!');
% hObject    handle to pushbutton18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton19.
function pushbutton19_Callback(hObject, eventdata, handles)
global Fi;
global flag;

set(handles.eta,'String','Click "Cancel current process" to stop');
f=size(Fi);
G=zeros(f(1),f(2));
for z=1:f(3)
    for x=1:f(1)
        for y=1:f(2)

            G(x,y)=Fi(x,y,z);

        end
    end
display(min(min(G)))
display(max(max(G)))
axes(handles.axes2);
    imshow(G,[min(min(G)) max(max(G))]);
    pause(.1);

    if flag==1
         flag=0;
        return;

    end

end
set(handles.eta,'String','Done!');
% hObject    handle to pushbutton19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes during object creation, after setting all properties.
function text12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to text12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global numang;
numang=str2double(get(hObject,'String'));
global Q1;

f=size(Q1);
```

```
set(handles.eta,'String','Please wait');

S=zeros(f(2),f(3));

for x=1:f(2)
    for y=1:f(3)
        S(x,y)=Q1(round(f(1)/2),x,y);
    end
end

axes(handles.axes1);
 imshow(S,[min(min(S)) max(max(S))]);
S=zeros(f(2),numang);
for x=1:f(2)
    for y=1:numang
        S(x,y)=Q1(round(f(1)/2),x,y);
    end
end

axes(handles.axes2);
 imshow(S,[min(min(S)) max(max(S))]);




set(handles.edit4 , 'Visible', 'On');
set(handles.text5 , 'Visible', 'On');


set(handles.eta,'String','Now select a slice, and the program will auto center the
sinogram.');
% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton21.
function pushbutton21_Callback(hObject, eventdata, handles)

global J;
global F;
```

```
axes(handles.axes2);
set(handles.eta,'String','Select the ROI');
masr=roipoly(J);
set(handles.eta,'String','Select the background');
masb=roipoly(J);


ima=max(J(:));
     imi=min(J(:));
     ims=std(J(:));
     snr=20*log10((ima-imi)./ims);
 ob=mean(J(masr));
 ba=mean(J(masb));
 cnr=(ob-ba)/ims;


 ima2=max(F(:));
     imi2=min(F(:));
     ims2=std(F(:));
     snr2=20*log10((ima2-imi2)./ims2);
 ob2=mean(F(masr));
 ba2=mean(F(masb));
 cnr2=(ob2-ba2)/ims2;




 set(handles.eta,'String',horzcat('Filtered image === CNR: ',num2str(cnr),' SNR:
',num2str(snr),'       Original image === CNR: ',num2str(cnr2),' SNR:
',num2str(snr2)));

% hObject    handle to pushbutton21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

12.REFERENCES

1. Indira Gandhi Centre for Atomic Research, *NDED Home Page*, http://www.igcar.ernet.in/mmg/nded/nded.html
2. MATLAB Help, *MATLAB Functions*
3. A. C. Kak and Malcolm Slaney, *Principles of Computerized Tomographic Imaging*, IEEE Press, 1988.