3D reconstruction in computerized tomography

BY SABIR ILYASS.

December 2022.

Note 1.

The MATLAB code is between ✠✠ ... ✠✠

1 Firt analysis of the data

Q1: We have images of tomographic projections acquired of an object over the range of angles from 0 to 180.

In figure 1, we present 3 images of the object at 3 different angles: 0, 45 and 180.

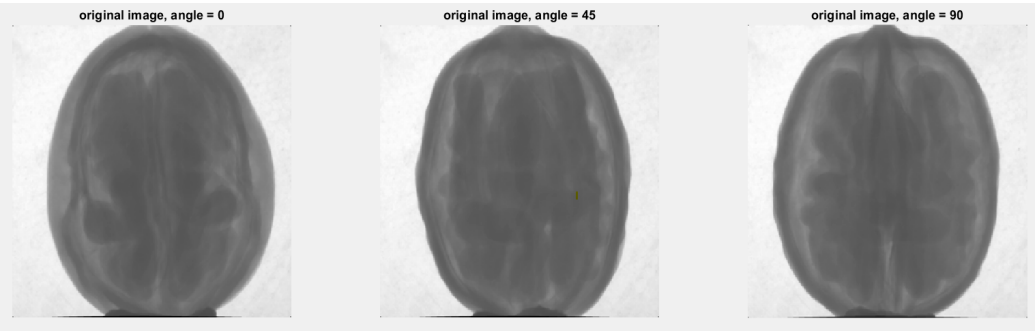

Figure 1. Images of the object at 3 different angles: 0, 45 and 180

Here is the MATLAB code for displaying the images in Figure 1.

✠✠✠✠✠✠✠✠✠✠✠✠✠✠✠✠✠✠✠✠

```
% show some images from different angles:
im0 = double(imread(sprintf("galopa_acqui/frame_%d.0.tif", 0)));
im45 = double(imread(sprintf("galopa_acqui/frame_%d.0.tif", 45)));
im90 = double(imread(sprintf("galopa_acqui/frame_%d.0.tif", 90)));

figure;
subplot(1,3,1); imshow(im0, []); title("original image, angle = 0");
subplot(1,3,2); imshow(im45, []); title("original image, angle = 45");
subplot(1,3,3); imshow(im90, []); title("original image, angle = 90");
```

✠✠✠✠✠✠✠✠✠✠✠✠✠✠✠✠✠✠✠✠

In order to facilitate the reconstructions, we will consider that the images were acquired in parallel geometry, i.e. that the rays originating from the source that cross the object are all parallel (source located at infinity). We will thus be able to treat the 3D reconstruction problem as a succession of 2D problems.

Q2: Here is the sinogram of the object, on line=128.

To speed up the processing, we will subsample the images by at least a factor of 2.

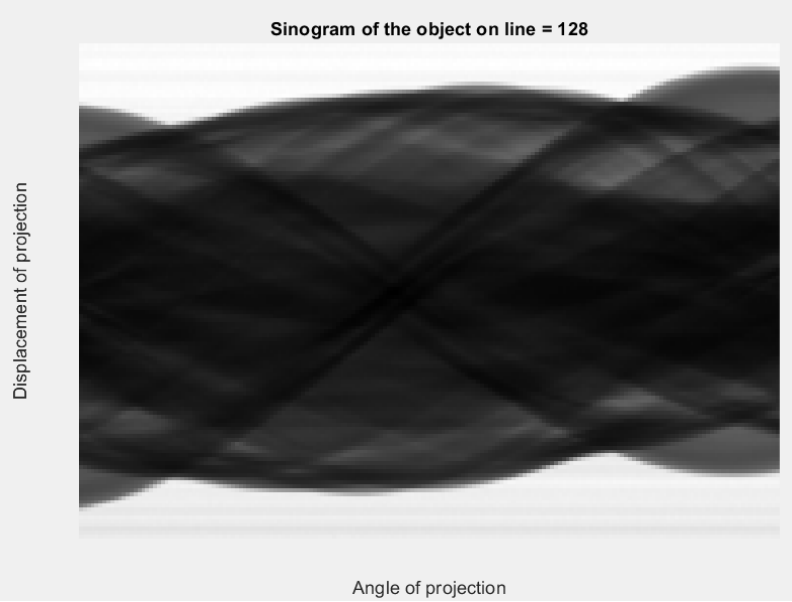

Figure 2. The sinogram of the object

```
function sino = sinogram(line, f)

im0 = imresize(double(imread(sprintf("data2/proj_%d.png", 0))), f);

[~, c] = size(im0);

sino = zeros(c, 180);

for tetha = 0:179
    im = imresize(double(imread(sprintf("data2/proj_%d.png", tetha))), f);
    sino(:, tetha + 1) = im(line, :);
end
```

We apply this function to line=128:

```
f = 0.5;
line = 128;
```

```
sino = sinogram(floor(line * f), f);

figure;
imshow(sino,[]); title('Sinogram of the object on line = 128'); ylabel('Displacement of projection'); xlabel('Angle of projection');
```

2 Modeling the image formation

Q3: We use the Matlab function **radon** to simulate the 1D projections of a black image with size (100, 100), with 3 white disks with center $= (50, 25), (20, 75), (90, 50)$ and radius $=10, 5, 30.$ respectively

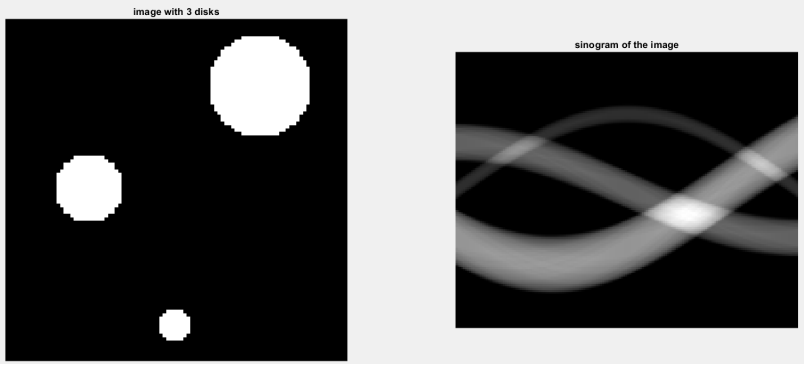

Figure 3. Sinogram of black image with 3 white disks

Here is the code we applied to obtain figure 3.

```
I = zeros(100,100);

radius1 = 10;
center1 = [50, 25];

for i = center1(1) - radius1: center1(1) + radius1
    for j = center1(2) - radius1: center1(2) + radius1
        if ((i - center1(1)) ^2 + (j - center1(2)) ^2 < radius1 ^2)
            I(i, j) = 1;
        end
    end
end

radius2 = 15;
center2 = [20, 75];

for i = center2(1) - radius2: center2(1) + radius2
    for j = center2(2) - radius2: center2(2) + radius2
        if ((i - center2(1)) ^2 + (j - center2(2)) ^2 < radius2 ^2)
```

```
                I(i, j) = 1;
            end
        end
    end

    radius3 = 5;
    center3 = [90, 50];

    for i = center3(1) - radius3: center3(1) + radius3
        for j = center3(2) - radius3: center3(2) + radius3
            if ((i - center3(1)) ^2 + (j - center3(2)) ^2 < radius3 ^2)
                I(i, j) = 1;
            end
        end
    end


    figure;
    subplot(1,2,1); imshow(I,[]); title("image with 3 disks");

    [R,xp] = radon(I);
    subplot(1,2,2); imshow(R,[]); title("sinogram of the image");
```

Q4: We build black images with a single white pixel.

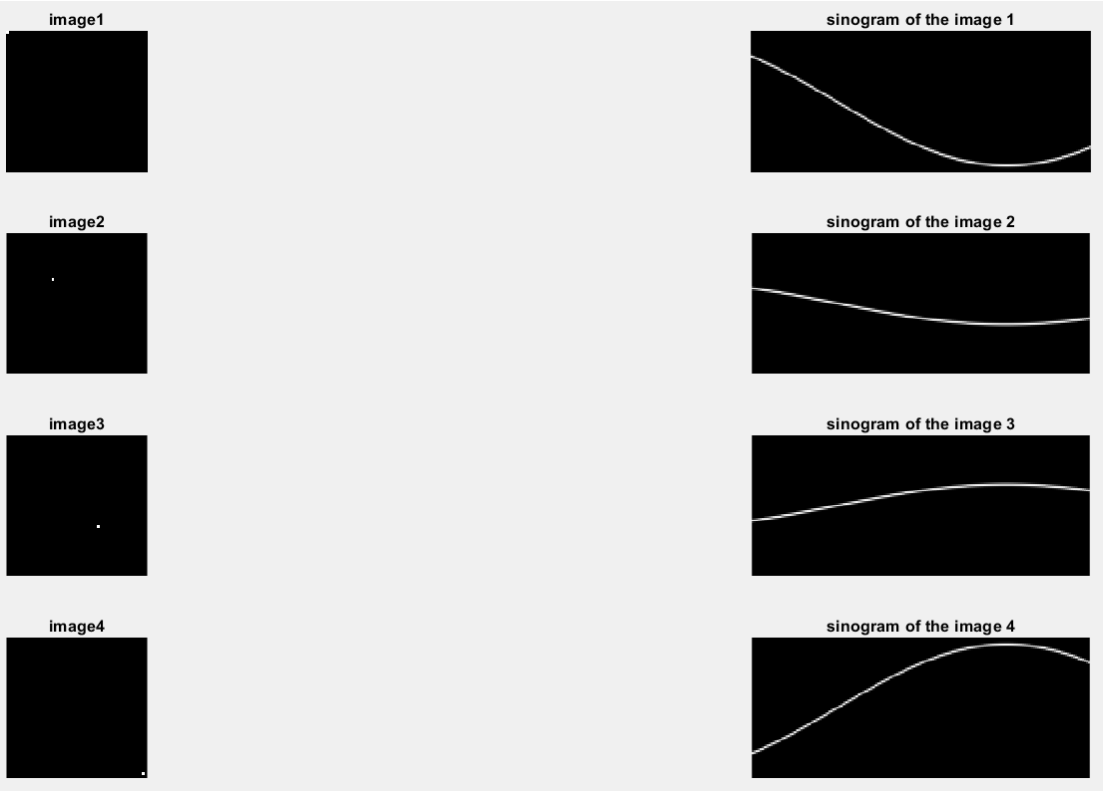

Figure 4. Images with their sinograms.

Here is the code to obtain figure 4

```
figure;
for tetha = 0:3
    x = 1 + 16 * tetha;
    y = 1 + 16 * tetha;
    I = zeros(50,50);
    I(x, y) = 1;
    subplot(4,2,2 * tetha + 1); imshow(I,[]); title(sprintf("image%d", tetha + 1));

    [R,xp] = radon(I);
    subplot(4,2,2 * tetha + 2); imshow(R,[]); title(sprintf("sinogram of the image %d",tetha
+ 1));
end
```

We notice that the sinogram of the image has the same value when the white pixel is located in the middle of the image. The sinogram varies sinusoidally depending on the angle of projection and the amplitude of sinogram is greater the further from the center of the image.

Let's try to prove this result.

For a giving image I with size $(w, h) \in \mathbb{N}^2$.

For al line ρ, and for all angle θ. The sinogram is given by.

$$g(\rho, \theta) = \sum_{x=1}^{w} \sum_{y=1}^{h} I(x, y)\delta(x.\cos(\theta) + y.\sin(\theta) - \rho)$$

With δ is Kronecker delta defined by: $\delta(x) = \begin{cases} 1 \text{ if } x = 0 \\ 0 \text{ otherwise} \end{cases}$

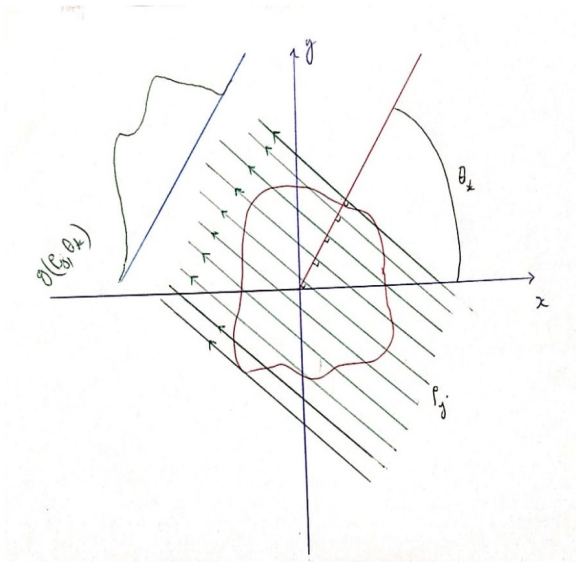

Figure 5. $g(\rho, \theta)$

So for a black image with a single white pixel located in (x_0, y_0), we have

$$g(\rho, \theta) = \delta(x_0.\cos(\theta) + y_0.\sin(\theta) - \rho)$$

So $g(\rho,\theta)=1$ if and only if $x_0.\cos(\theta)+y_0.\sin(\theta)=\rho$.

Q5: For a giving line with parametric equation: $x.\cos(\theta)+y.\sin(\theta)=\rho$
The random transform for a giving image I (In the continuous space).

$$g(\rho,\theta)=\iint_{\mathbb{R}^2} I(x,y)\delta(x.\cos(\theta)+y.\sin(\theta)-\rho)\,dx\,dy$$

With δ is Kronecker delta defined by: $\delta(x)=\begin{cases}1 \text{ if } x=0\\ 0 \text{ otherwise}\end{cases}$

Visualize $g(\rho,\theta)$ as an image is the sinogram of the image I.

We have $(g(\rho,\theta_k))_{0\leqslant k\leqslant 180}$ for a set of angles $\{\theta_k\}_{0\leqslant k\leqslant 180}$ between 0 and π.

In the Fourier space
Consider a fixed angle θ's projection and take the 1-D Fourier trasform with respect to ρ.

$$\rho:''\text{direct space}'' \rightarrow \omega:''\text{Fourier space}''$$

$$g(\rho,\theta)\rightarrow G(\omega,\theta)$$

$$\begin{aligned}
G(\omega,\theta) &= \int_{\mathbb{R}} g(\rho,\theta)e^{-j2\pi\omega\rho}d\rho \\
&= \int_{\mathbb{R}}\iint_{\mathbb{R}^2} I(x,y)\delta(x.\cos(\theta)+y.\sin(\theta)-\rho)e^{-j2\pi\omega\rho}dx\,dy\,d\rho \\
&= \iint_{\mathbb{R}^2} I(x,y)\int_{\mathbb{R}}\delta(x.\cos(\theta)+y.\sin(\theta)-\rho)e^{-j2\pi\omega\rho}d\rho\,dx\,dy \\
&= \iint_{\mathbb{R}^2} I(x,y)e^{-j2\pi\omega(x.\cos(\theta)+y.\sin(\theta))}dx\,dy
\end{aligned}$$

So $G(\omega,\theta)$ is the Fourier transform of the original image along the line $\begin{cases}u=\omega.\cos(\theta)\\ v=\omega.\sin(\theta)\end{cases}$
So

$$I(x,y)=\iint_{\mathbb{R}^2} G(\omega,\theta)e^{j2\pi\omega(x.\cos(\theta)+y.\sin(\theta))}d\omega d\theta$$

So the first idea to build the image I is to calculate the inverse Fourier transform of $(\omega,\theta)\rightarrow G(\omega,\theta)$.

3 First attempt at reconstruction

Q6: we try to reconstruct the line plane = 175, we calculate the sinogram of the object at line 175, then we apply the **iradon** function to calculate the inverse of the sinogram.

```
sinogram_line = sinogram(175, 1, 0);
plan_175 = iradon(sinogram_line, 0:179);
figure;
imshow(plan_175, []);
```

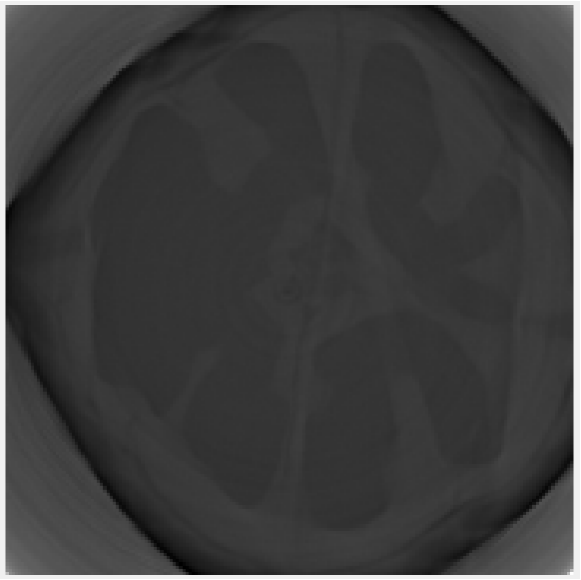

Figure 6. reconstruction of line =175 of the object

We have built the image of the plane 175, except that the size of the image is 180×180, but we should normally have images of size 256×256.

Noises are presented in the image because we did not segment the areas of interest, and we did not subtract the dark field image.

Q7: The size of the image **plan_175** is 180×180. the size of sinogram_radon is 259×180, unlike the size of **sinogram_line** which is 256×180.

This difference comes from the working principle of each method, the sinogram method we developed in question 2 is based on the concatenation of the same line seen from different angles.

On the other hand, Matlab's radon method is based on choosing a grid size for R that will capture the projections of the entire image at all θ angles.

So to cover all the angles, it will be necessary to create an image of length = (length of the initial image) $\times$ (constant to capture all the angles), for a square image this constant is close to sqrt(2), so for the image plan_175 which is of size 1180×180, we obtained a sinogram of size 259×180.

```
sinogram_radon = radon(plan_175);

disp(size(sinogram_radon));
disp(size(sinogram_line));
```

4 System self-calibration

Q8: After a rotation of 180°, the tomographic system should give us (under our appoximation of parallel rays) a projection symmetrical to the rotation axis of the object.

Let I_{180}: projection of the object at angle 180, with size (w,h).
Let's calculate the symmetric image of SymI_{180} defined by:

$$\forall (i,j) \in [\![1,w]\!] \times [\![1,h]\!] \; \mathrm{SymI}_{180}(i,j) = I_{180}(i, h+1-j)$$

We define a pattern $P = I_0\Big(:, \Big\lfloor \frac{h}{2} \Big\rfloor - 30 : \Big\lfloor \frac{h}{2} \Big\rfloor + 30\Big)$ as a sub-image of I_0:projection of the object at angle 180.
We are looking for the translation vector from I_0 to ymI_{180}.
For that, we calculate the correlation between P and $I_0(:, j-30, j+30)$ for all $j \in [\![31, h-30]\!]$.

Here is the code to obtain this translation length.

```
% image d'angle = 0 et image d'angle = 180
image0 = imread("data2/proj_0.png");
image180 = imread("data2/proj_180.png");

figure;
subplot(1,3,1); imshow(image0,[]), title("image of the object, angle = 0");
subplot(1,3,2); imshow(image180,[]); title("image of the object, angle = 180");

% determine l'image symétrique de image180
[n, m] = size(image0);
imSymetric = zeros(n, m);
for i = 1:n
    for j = 1:m
        imSymetric(i,j) = image180(i, m + 1 - j);
    end
end
subplot(1,3,3); imshow(imSymetric,[]); title("symetric of the image of the object, angle = 180");

% Pattern
[~, h,~] = size(image0);
x = floor(h / 2);

motif = image0(:, x - 30:30 + x);
maximum = -inf;
index = 0;
for j = 31: h - 30
    im = imSymetric(:, j - 30:j + 30);
    corr = corr2(im, motif);
    if (corr > maximum)
        index = j;
        maximum = corr;
    end
end

decalage = x - index;
```

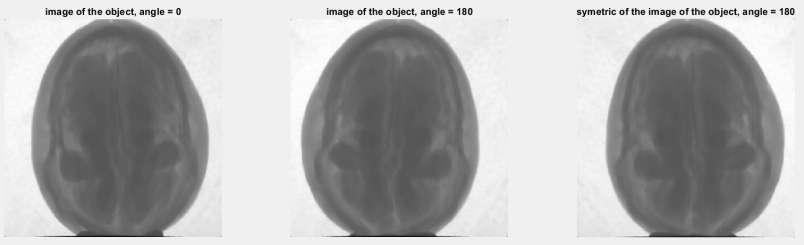

Figure 7. Image of object, angle =0 and angle=180.

To build a sinogram for which the axis of rotation is at the center of the image, we subtract the columns in order to make the images centered with respect to the axis of rotation, for that we modified the code of the sinogram function

```
function sino = sinogram(line, f, decalage)

im0 = imresize(double(imread(sprintf("data2/proj_%d.png", 0))), f);

[h, c] = size(im0);

sino = zeros(c - abs(decalage), 180);
decal = floor(decalage * f);

if decal > 0
    for tetha = 0:179
        im = imresize(double(imread(sprintf("data2/proj_%d.png", tetha))), f);
        im = im(:, decal + 1 : c);
        sino(:, tetha + 1) = im(line, :);
    end
else
    for tetha = 0:179
        im = imresize(double(imread(sprintf("data2/proj_%d.png", tetha))), f);
        im = im(:, 1 : decal + c);
        sino(:, tetha + 1) = im(line, :);
    end
end
```

We applied this function to consrtuire the sinograms for which the axis of rotation is at the center of the image.

```
% Sinogram for which the axis of rotation is at the center of the image.

Sinogram3D = zeros(m - abs(decalage), 180, n);
for line = 1:n
    Sinogram3D(:, :, line) = sinogram(line, 1, decalage);
end

figure;
imshow(Sinogram3D(:, :, 128), []); title('Sinogram of the object for which the axis of rotation is at the center on line = 128');
```

For the line 128, we obtain the following sinogram

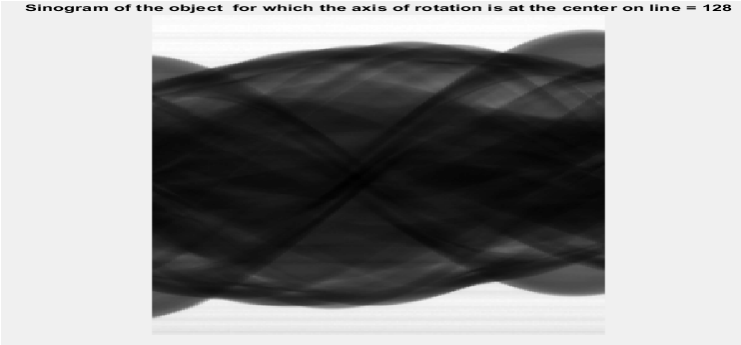

Figure 8. Sinogram of the object for which the axis of rotation is at the center on line = 128.

5 Reconstruction by the filtered back-projections algorithm

Q9: 3D reconstruction of the object using **iradon**

```
object3D = zeros(178, 178, n);

for line = 1:n
    plan = iradon(Sinogram3D(:, :, line), 0:179);
    object3D(:, :, line) = plan(:, :);
end

figure;
isosurface(object3D);
```

Here is a section of the 3D object we obtained, the following figure shows a part of the object from line=0 to line=100.

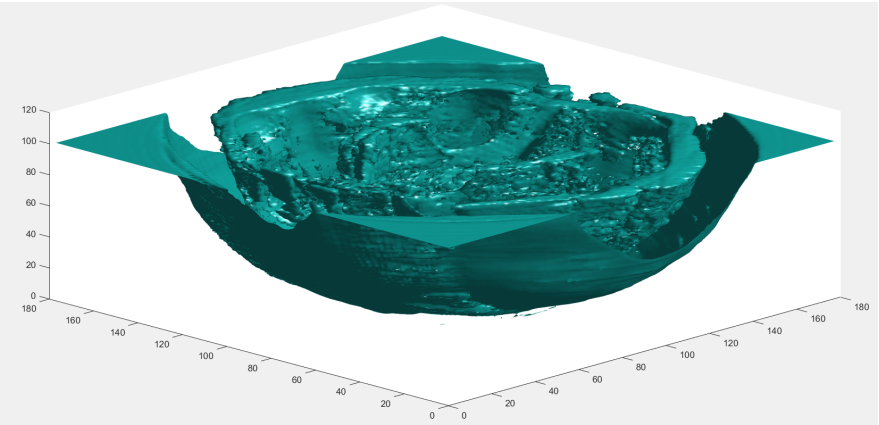

Figure 9. A section of the 3D object.

Here is the entire 3D object

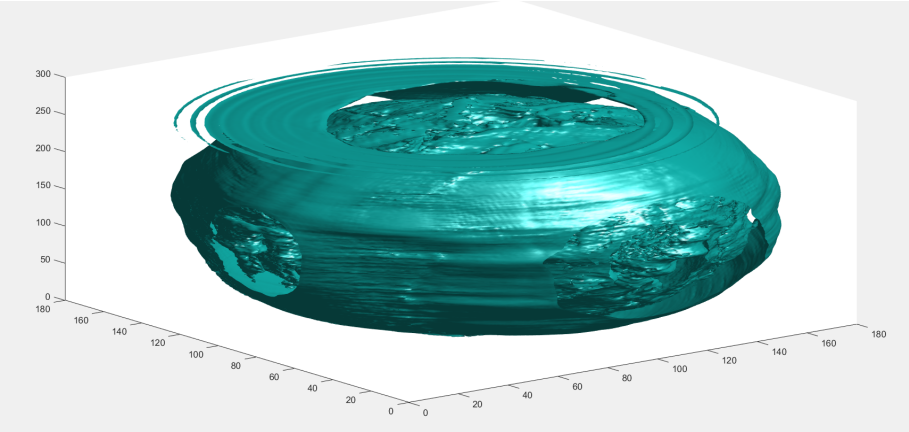

Figure 10. The 3D object.

We notice that there is a little noise on the reconstruction, since we did not do any preprocessing (segmentation of the area of interest and extraction of the dark field image...), before computing the sinogram.

6 Understanding of the principle of the filtered back projection algorithm

Q10: the back-projection of the sinograms of the absorption point studied in part 2.

```
figure;
for tetha = 0:3
    x = 1 + 16 * tetha;
    y = 1 + 16 * tetha;
    I = zeros(50,50);
    I(x, y) = 1;
    subplot(4,2,2 * tetha + 1); imshow(I,[]); title(sprintf("image%d", tetha + 1));

    [R,xp] = radon(I);
    iradon_I = iradon(R, 0:179, “linear", “Ram-Lak");

    subplot(4,2,2 * tetha + 2); imshow(iradon_I,[]); title(sprintf("iradon of the sinogram of the image %d",tetha + 1));
end
```

Figure 11. the back-projection of the sinograms of the absorption point studied in part 2.

We notice that the reconstruction of the absorbing point depends on its position in the volume. We take the equation obtained in question 5, we have

$$I(x,y) = \iint_{\mathbb{R}^2} G(\omega,\theta) e^{j2\pi\omega(x.\cos(\theta)+y.\sin(\theta))} d\omega d\theta$$

Let $\begin{cases} u=\omega\cos\theta \\ v=\omega\sin\theta \end{cases}$ and $H(u,v)=G(\omega,\theta)$, we have

$$I(x,y) = \iint_{\mathbb{R}^2} H(u,v) e^{j2\pi\omega(x.\cos(\theta)+y.\sin(\theta))} du dv$$

We change the coordinates of Fourier transform to the polar coordinates:

$$\begin{aligned} I(x,y) &= \int_0^{2\pi}\int_{\mathbb{R}} H(\omega\cos\theta,\omega\sin\theta) e^{j2\pi\omega(x.\cos(\theta)+y.\sin(\theta))} \omega d\omega d\theta \\ &= \int_0^{2\pi}\int_{\mathbb{R}} G(\omega,\theta) e^{j2\pi\omega(x.\cos(\theta)+y.\sin(\theta))} \omega d\omega d\theta \end{aligned}$$

Because of $G(\omega,\theta+\pi)=G(-\omega,\theta)$. so

$$I(x,y) = \int_0^{\pi}\int_{\mathbb{R}} |\omega| G(\omega,\theta) e^{j2\pi\omega(x.\cos(\theta)+y.\sin(\theta))} d\omega d\theta$$

So $I(x,y)$ is the inverse of Fourier transform of $G(\omega,\theta)$, but multiplied by a filtred function $|\omega|$.

The matlab method to implement this technique is "Ram-Lak", there are other methods that are more accurate.

We summarize the above in the following reconstruction algorithm:
for each pixel (x,y)

1- Compute 1D FT of each projection.
2- Multiply each FT ($G(\omega, \theta)$) by $|\omega|$.
3- Take the 1-D inverse FT.
4- Integrate over all angles to get the original image $I(x, y)$.

Q11: We implement the code developed in the previous question to build plan 175.

```
I = sinogram(175, 1, decalage);
[n, m] = size(I);
line = floor(n / 2);

iw = 2 * floor(n / (2 * sqrt(2)));
iw_n = iw / 2;

% Filtered Back Projection, Compute the Ramlak filter
g = [0:line, line - 1:-1:1];

H = 2 * g / n;

% Compute Fourier transformation of sinogram
gf = fft(I, [], 1);

% Multiply the filter in frequency domain
gff = bsxfun(@times, gf, H');

% Inverse Fourier transformation
gffi = real(ifft(gff, [], 1));

% Initialize the reconstructed image
img = zeros(iw);

% Compute some arguments for back projection

[posX, posY] = meshgrid((1:iw) - iw_n);

for t = 1:180

    % Calculate the position in sinogram
    pos = posX * cosd(t) + posY * sind(t) + line;

    % Accumulate projection of each degree sinogram
    img = img + interp1(1:n, gffi(:, t), pos);

end
N = 180;
% Multiply the factor
img = img * (pi / (2 * N));

figure;
imshow(img, []); title("reconstruct the plan 175");
```

We obtain the following figure:

Figure 12. reconstruction of the line 175.

We can follow the same algorithm to build all the planes of the object, which allows us to reconstruct the object in 3D.