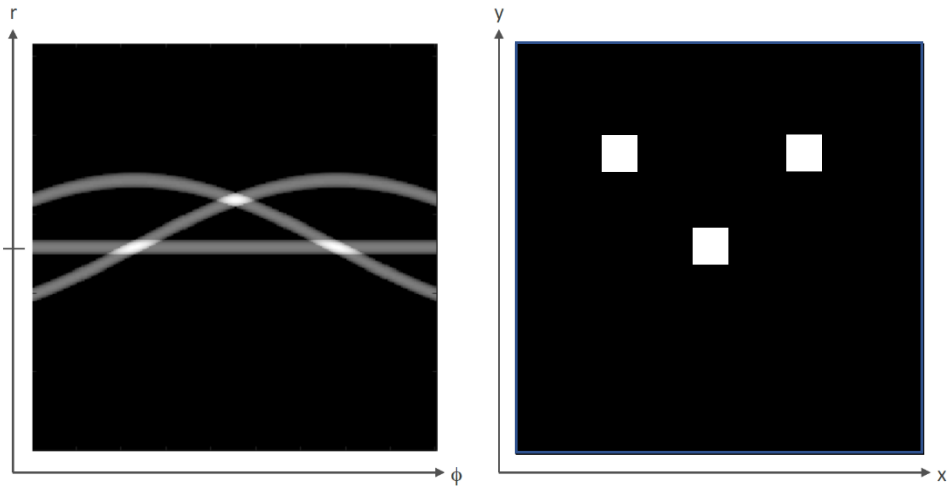
Biomedical Imaging Exercise – Week 7

Name: Alkinoos Sarioglou

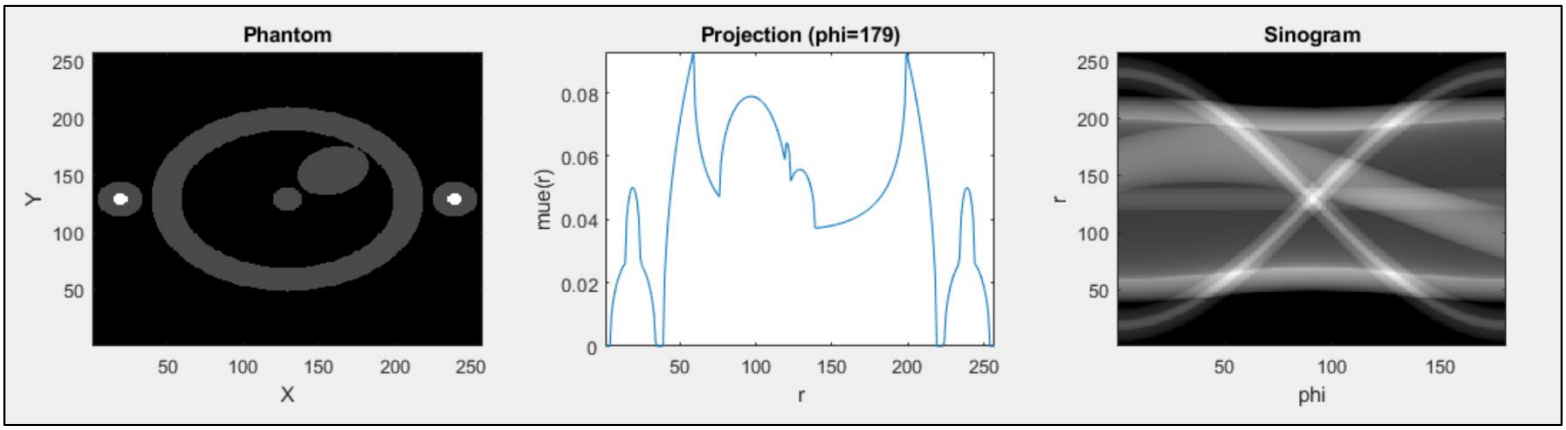
Student ID: 20-947-743

1. Task 2.1

a) The object can be reconstructed by looking at its sinogram as following:



b) In the sinogram, the projection of each structure as a function of the polar coordinates (r,ϕ) in the spatial domain can be seen. It is worth noting that even though the graph is plotted for r from 0 to 256 from bottom-up, here we use the convention that r=0 is the middle of the y-axis. Therefore, for the values of r under the middle, r<0. Also for the values of r above the middle of the y-axis, r>0.



The following parts of the sinogram can be recognized:

- Central line: The central line represents the aorta, because it is at the center of the Cartesian Coordinate System and it is always at r=0.
- Outermost lines: The lines that have the biggest deviation from r=0 represent the two arm muscles and the two bones. The thicker lines represent the muscles and the thinner lines represent the bones.

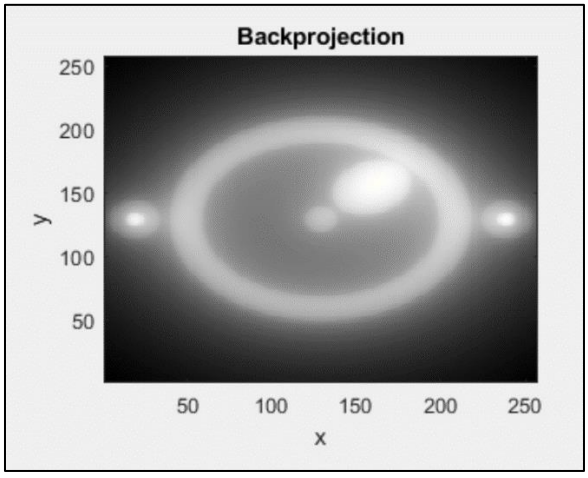
- Straight lines at r=200 and r=50: These straight lines represent the projection of the thorax. The thorax is a circular structure therefore it always has a component at r=0 and projections at r>0 and r<0. However, most of the components are at the same r as parts of the lungs, which reduce the linear attenuation coefficient and therefore they appear as less bright parts of the image. On the other hand, the outermost parts of the walls of the thorax which are not combined with lungs' contribution appear brighter due to higher linear attenuation coefficient and these are seen as the straight lines at positive and negative r extending from $\varphi = 0°$ to $\varphi = 180°$.
- Space between the aorta and the thorax walls: This space represents the lungs in between the aorta and the thorax walls.
- Thick sine-like line: This line represents the heart which starts from positive r at $\varphi = 0°$ and ends up at negative r when $\varphi = 180°$.

2. Task 2.2

a) The following code snippet is written in order to create the backprojection reconstructed image:

```
[x,y]        = meshgrid(-fix(matrix/2):1:+fix(matrix/2)); idx = 1;
phantom.sbp  = zeros(matrix+1,matrix+1);

for phi=projection_angles

    % TASK 2.2 FILL IN HERE

    r = x*cosd(phi)+y*sind(phi);
    rs = round(r)+129;
    ix = find((0<rs) & (rs<=257));

    % TASK 2.2 UNCOMMENT HERE
    projection = phantom.sino(:,idx);
    phantom.sbp(ix) = phantom.sbp(ix)+projection(rs(ix));

    DisplayData(phantom.sbp,[2,3,4]);
    title('Backprojection'); xlabel('x'); ylabel('y'); drawnow;

    idx = idx+1;
end
```

The reconstructed image using simple backprojection looks as following:



The image appears blurry, because many of the high frequency components are not depicted in the image. That means that a lot of information about the surroundings of the object do not appear in the image due to the fact that the added projections give mostly information for the center of the image rather than the surroundings. Therefore, the reconstruction image recreates mostly low-frequency components and most high-frequencies are lost. In other words, the middle of the FT of the image is oversampled and it should be down-weighted in order to get a sharper image.

3. Task 2.3

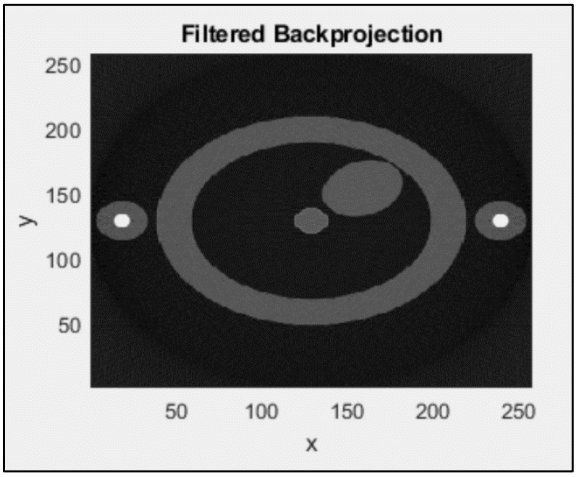
a) The code written to implement filtered backprojection is:

```
for phi=projection_angles

    % TASK 2.3 FILL IN HERE
    projection  = phantom.sino(:,idx);
    projection_fourier  = R2U(projection');

    r = x*cosd(phi)+y*sind(phi);
    rs = round(r)+129;
    ix = find((0<rs) & (rs<=257));

    conv_freq_domain(rs(ix),idx) = projection_fourier(1, rs(ix)).*filter(1, rs(ix));
    filteredprojection = U2R(conv_freq_domain(:,idx));

    % TASK 2.3 UNCOMMENT HERE
    phantom.fbp(ix) = phantom.fbp(ix)+filteredprojection(rs(ix));

    DisplayData(phantom.fbp,[2,3,5]);
    title('Filtered Backprojection'); xlabel('x'); ylabel('y'); drawnow;

    idx = idx+1;
end
```

b) The filter used here is a high-pass filter designed as following:

```
function [filter] = CalcFilter(matrix)

    % ------------------------------------------------------------------
    % Define high-pass filter according to |u|
    % ------------------------------------------------------------------
    hpf = abs(-fix(matrix/2):+fix(matrix/2));

    % TASK 2.3 FILL IN HERE
    hpf = hpf/128; % High-Pass Filter Frequency Domain

    lpf = zeros(1,matrix); % Low-Pass Filter Frequency Domain
    for n=0:1:256
        lpf(1,n+1) = 0.54 - (1-0.54)*cos((2*pi*n)/256); % Hamming Window Function
    end

    filter = hpf; % Combine HPF and LPF to create a BPF
```

The resulting reconstructed image is formed:

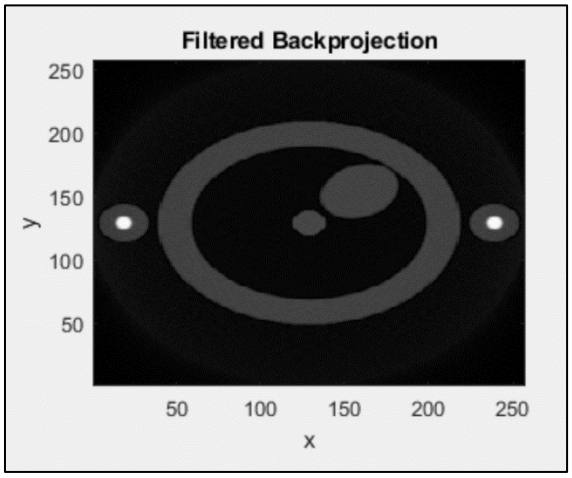


c) After applying the high-pass filter, it can be seen that there are some streaking artefacts, which are caused due to the beam “hardening”. This means that most low-energy photons are attenuated due to the high-pass filter and only the high-energy photons contribute to the imaging, which experience less scattering. The “streaking” artefacts are the result of the

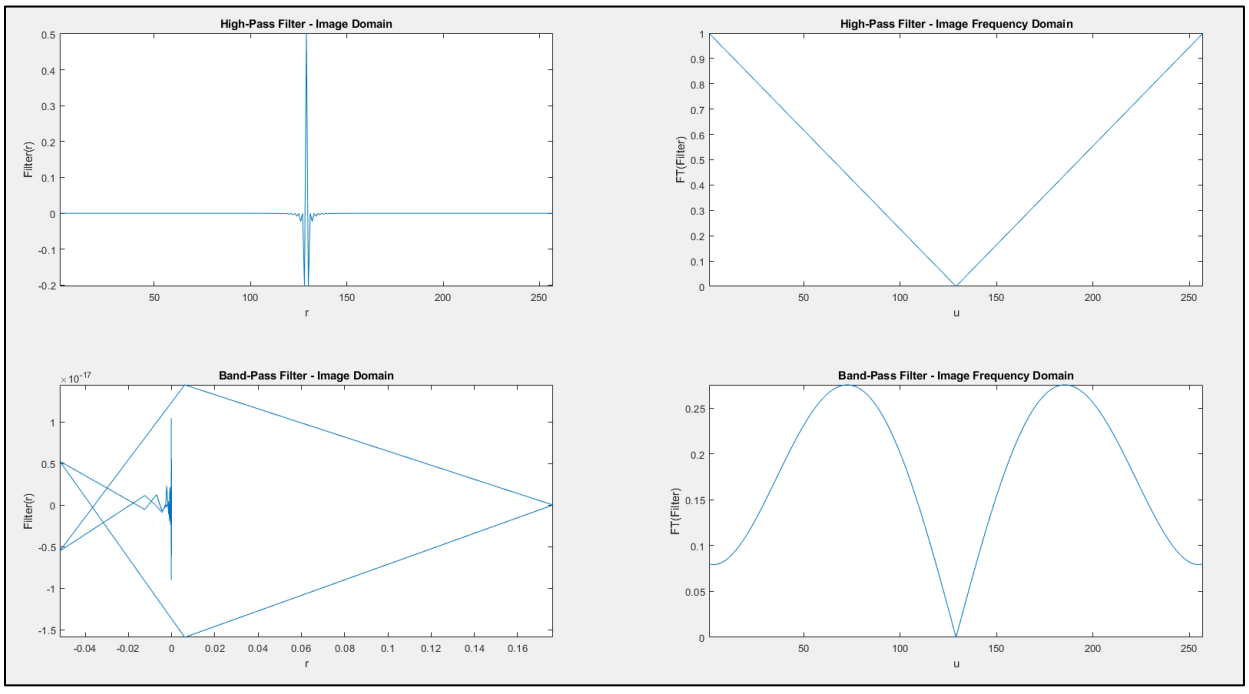
X-Ray beam being filtered at different rates from the high-pass filter, according to the angle of the source and the detector. Therefore, a low-pass filter can be added as well in order for the combined filter to be a band-pass filter. Here a Hanning window function is used as a low pass filter. This is coded as following:

```
function [filter] = CalcFilter(matrix)

    % ------------------------------------------------------------------
    % Define high-pass filter according to |u|
    % ------------------------------------------------------------------
    hpf = abs(-fix(matrix/2):+fix(matrix/2));

    % TASK 2.3 FILL IN HERE
    hpf = hpf/128; % High-Pass Filter Frequency Domain

    lpf = zeros(1,matrix); % Low-Pass Filter Frequency Domain
    for n=0:1:256
        lpf(1,n+1) = 0.54 - (1-0.54)*cos((2*pi*n)/256); % Hamming Window Function
    end

    filter = hpf.*lpf; % Combine HPF and LPF to create a BPF
```

The resulting reconstructed image is the following, in which it can be seen that the “streaking” artefacts have been eliminated:



d) The Fourier transform pairs of the ideal and the modified filters are seen in the following figure:



When selecting the frequency range of the band-pass filter, there is a trade-off between the advantages of keeping the highest-frequencies and the advantages of keeping only a portion of the highest frequencies by using a low-pass filter as well.

When selecting higher frequencies by minimizing the impact of the low-pass filter at these frequencies, the image is sharper because the beam includes mostly high-energy photons which experience less scattering. On the other hand, as seen before, keeping only high frequencies results in streaking artefacts which distort the quality of the image and additionally the SNR is reduced.

Therefore, a good balance between these advantages is required by selecting the band-pass frequencies appropriately in order to achieve the best trade-off possible.

4. Task 2.4

a) In the Fourier-Slice Theorem, many projections are taken along the direction s of the beam in separate angles ϕ, which are then Fourier-transformed to give information in the Spatial Frequency Domain. Once all the projections have been extracted, the Inverse Fourier Transform is taken in order to reconstruct the image.

A projection under angle ϕ and as a function of position r is given by: $P_\varphi(r) = \int \mu(r,s)ds$

Taking the FT of $P_\varphi(r)$:

$$F\{P_\varphi(r)\} = \int P_\varphi(r) \cdot e^{-iur} dr = \iint \mu(r,s) \cdot e^{-iur} drds = \iint \mu(x,y) \cdot e^{-ixucos(\varphi)} \cdot e^{-iyusin(\varphi)} dxdy$$

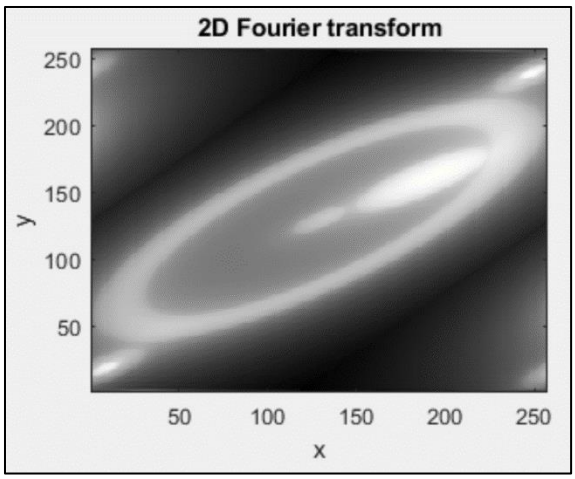
But $ucos(\varphi) = p$ and $usin(\varphi) = q$, therefore:

$$F\{P_\varphi(p,q)\} = \iint \mu(x,y) \cdot e^{-ixp} \cdot e^{-iyq} dxdy$$

b) The code used to implement image reconstruction from projections using the 2D Fourier Transform is shown below:

```
for phi=projection_angles

    % TASK 2.4 FILL IN HERE

    r = x*cosd(phi)+y*sind(phi);
    rs = round(r)+129;
    ix = find((0<rs) & (rs<=257));

    projection = phantom.sino(:,idx);
    phantom.sbp(ix) = phantom.sbp(ix)+projection(rs(ix));

    phantom.fft(ix) = R2U(phantom.sbp(ix));

    idx = idx+1;

end

phantom.fft = U2R(phantom.fft);

DisplayData(phantom.fft*length(projection_angles),[2,3,6]);
title('2D Fourier transform'); xlabel('x'); ylabel('y'); drawnow;
```

The resulting reconstructed image is the following:



c) The reconstructed image using FFT looks tilted and not close to the original image. The image reconstruction with simple backprojection (Task 2.2) results in a much better reconstructed image.

d) The artefacts in the reconstructed image using FFT are caused because the resolution varies as the source and the detector rotate around the structures of interest. Therefore, at some angles the high-frequency components are suppressed and as a result when converting back to the image domain these components are reconstructed distorted. For that reason, artefacts in the high-frequency components of the image appear, which correspond to the distorted information in the surroundings of the image rather than the center. The further away from the center, the more distortion in frequencies occurs.