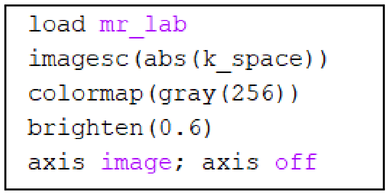
BME 595 - Fall 2023
Nicole Haderer (1005721444), Deniz Uzun (1006035005)

IMAGE RECONSTRUCTION LAB 2

PART 1: MRI IMAGE RECONSTRUCTION

1. ***Display the k-space data of an MR image.***
   The code k-space data of the MR image is displayed below in Figure 1b along with the MATLAB code for reference in 1a.

a) 

```
load mr_lab
imagesc(abs(k_space))
colormap(gray(256))
brighten(0.6)
axis image; axis off
```

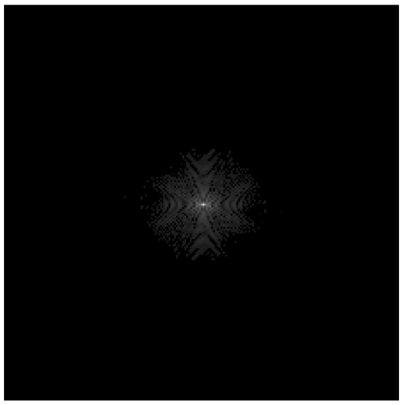
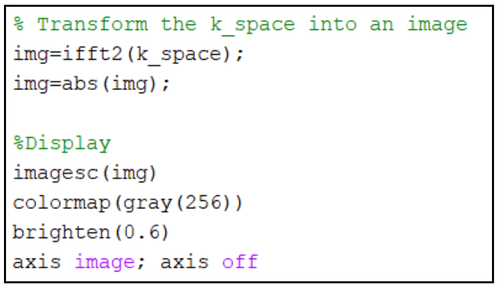
b) 

Figure 1. The MATLAB code (a) used to obtain the image representation of the k-space matrix (b).

2. ***What is the brightest region of the image; what k-space region does this correspond to?***
   The middle is the brightest region of the image and it corresponds to the lowest spatial frequency data. In MRI, low spatial frequencies correspond to features with longer wavelengths and hold information about the overall shape and contrast of the anatomical structures. The central spatial frequency, in other words the DC component, represents the sum of the signal in all voxels in image space.

3. ***Transform this k-space data into an image. Display the image using similar commands used to display the k-space data above.***
   The k-space data shown in Figure 1b was transformed into an image using the ifft2() function as shown in Figure 2a. This takes the 2D inverse Fourier transform of the k-space data, which is equivalent to the 2D image. The abs() function was also applied to this data such that only the magnitude of the signal intensities is applicable in the generation of the image shown in Figure 2b.

a) 

```
% Transform the k_space into an image
img=ifft2(k_space);
img=abs(img);

%Display
imagesc(img)
colormap(gray(256))
brighten(0.6)
axis image; axis off
```

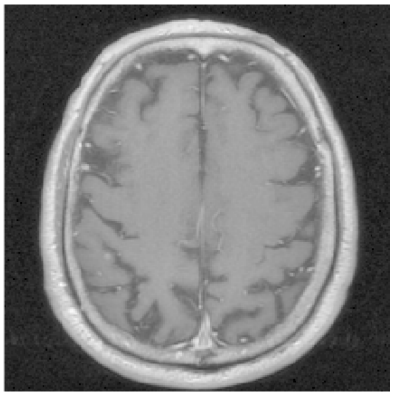
b) 

Figure 2. The MATLAB code (a) used to obtain the image given by the k-space data (b).

4. ***Select a region-of-interest (ROI) in the image. Using roi_meas, determine the image "signal" by calculating the mean intensity in a region of your choice inside the brain. Determine the noise by taking standard deviation of a region in the background of the image (i.e. where there are no anatomical structures). List these quantities, and then calculate the signal-to-noise ratio (SNR).***

An ROI and a section of the background were analyzed using the oimf() function in the "roi_meas.m" file to determine the mean and standard deviation of the signal intensities in these sections. The section of the background was analyzed to determine the noise level. The selected sections are shown in Figure 3a and 3b below.

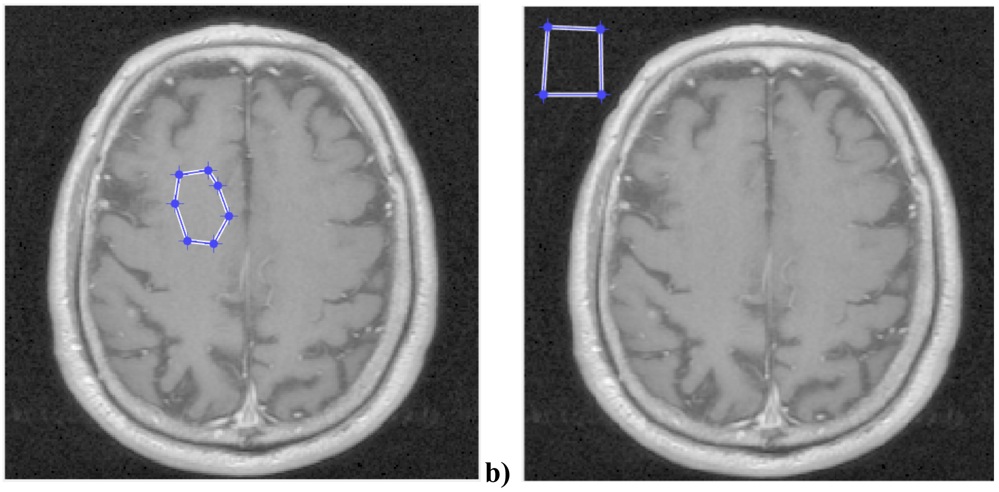


Figure 3. Selected sections for signal analysis of the ROI (a) and the noise (b).

The means and standard deviations of the ROI and background (noise) are listed below:

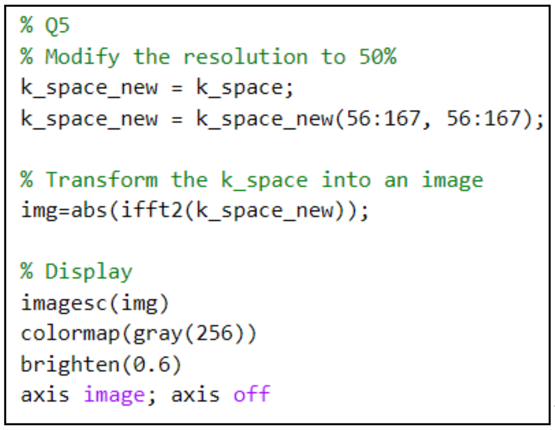
- ROI: Mean = 87, Standard Deviation = 3
- Noise: Mean = 6, Standard Deviation = 1

The signal-to-noise ratio (SNR) can be calculated using the following formula and the mean signal intensity of the ROI and the standard deviation of the signal intensity of the noise.

$$SNR_4 = \frac{mean\ signal\ intensity\ of\ ROI}{standard\ deviation\ of\ noise}$$
$$= 87/1$$
$$= 87$$

∴ The SNR for this ROI is **87.**

5. ***Create a copy of the original k_space matrix and modify the new k-space data matrix so that the resulting image will have a resolution that is coarser by approximately 50% in both the x and y direction. Transform this k-space data into an image. Display the image in a similar manner to step #3.***
To make the resolution of the image coarser in both the x and y direction, the outer 50% of the k-space data must be removed. The outer 50% of data corresponds to the high resolution details of the image; keeping the inner 50% maintains the contrast, but the reduction in matrix size (while maintaining the same FOV) results in a greater pixel size and coarser image. The inner 50% were retrieved to create k_space_new through indexing the rows and columns 56 to 167, creating a 112x112 matrix as shown in Figure 2a (MATLAB indexing is end-inclusive, i.e. the 167th column and row is also saved to k_space_new). The 2D inverse Fourier transform of the manipulated k-space was taken using ifft2() and abs() was applied to this dataset such that only the magnitude of the signal intensities is applicable in the generation of the image shown in Figure 4b.

a)


```
% Q5
% Modify the resolution to 50%
k_space_new = k_space;
k_space_new = k_space_new(56:167, 56:167);

% Transform the k_space into an image
img=abs(ifft2(k_space_new));

% Display
imagesc(img)
colormap(gray(256))
brighten(0.6)
axis image; axis off
```

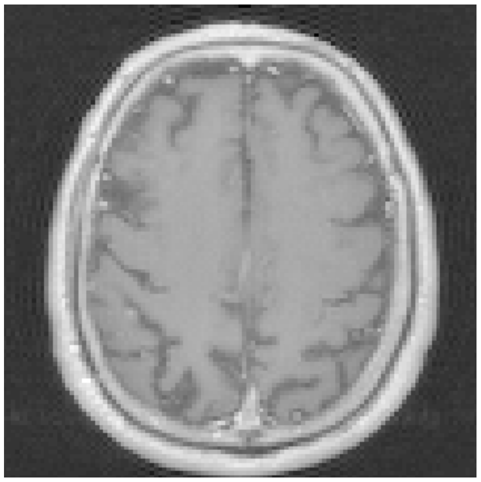
b)


Figure 4. MATLAB code (a) used to obtain the image given by the manipulated k-space data (b).

6. ***Calculate the SNR in this image in a similar manner to step #4 (try and choose signal and noise regions that are similar to the ones used in step #4). How does it compare to the SNR calculated in #4? What do you expect the SNR to be relative to the SNR calculated in #4?***
The process outlined in Step 4 was repeated on the 50% coarser image in Figure 4b. The selected sections for the ROI and noise are shown in Figure 5a and 5b below and were kept as similar as possible to those in Figure 3.

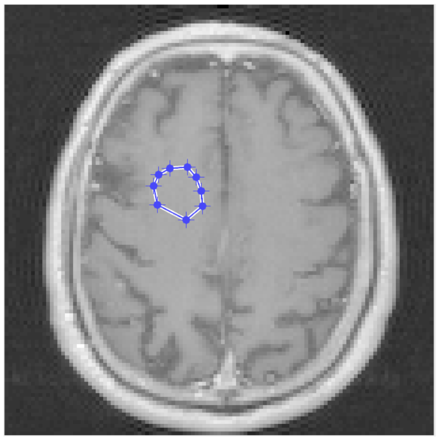
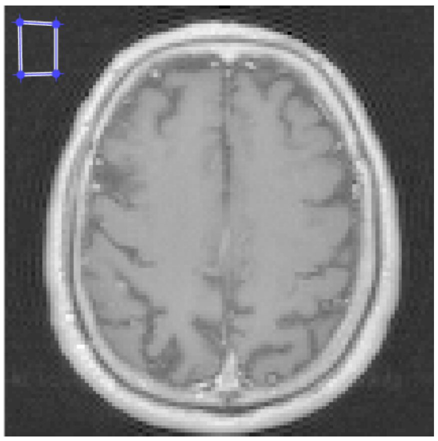
a)  **b)** 

Figure 5. Selected sections for signal analysis of the ROI (a) and the noise (b).

The means and standard deviations of the ROI and background (noise) are listed below:

- ROI: Mean = 342, Standard Deviation = 9
- Noise: Mean = 24, Standard Deviation = 3

The signal-to-noise ratio (SNR) can be calculated using the following formula and the mean signal intensity of the ROI and the standard deviation of the signal intensity of the noise.

$$\mathrm{SNR}_6 = \frac{mean\ signal\ intensity\ of\ ROI}{standard\ deviation\ of\ noise}$$
$$= 342/3$$
$$= 114$$

∴ The SNR for this ROI is **114.**

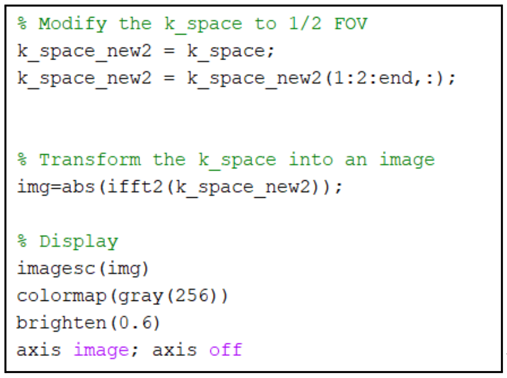
SNR_6 from Figure 4b is ~1.3 times greater than SNR_4 from Figure 2b in Step 4 ($\mathrm{SNR}_4 = 87$).

By reducing the resolution by ~50% in both x and y directions, we reduce the spatial resolution in both dimensions by receiving signals from four voxels in the original image into a single voxel. As a result, the voxel quadruples in size. Total time to acquire data (T) decreases by ¼ as a result of the number of samples in both frequency and phase encoding directions, N_x and N_y, decreasing by half.

Since $SNR \propto V_{vox}\sqrt{T}$, and T is directly proportional to N_x and N_y, the expected relative SNR would be $4\sqrt{0.25} = 2$. The expected SNR_6 is SNR_4 multiplied by this factor (87×2), which produces a value of 174.

7. ***Create another copy of the original k_space matrix and modify the new k-space data so that the image is displayed with only half the field-of-view (FOV) in one of the dimensions, and the full FOV in the other . Transform the modified k-space data into an image. Display the image in a similar manner to step #3.***

   The FOV was reduced by one half in the phase-encoding (y-) direction while the FOV of the frequency encoding (x-) direction remained as is; this is shown in Figure 6b. To reduce the phase-encoding FOV by one half, every other row was removed from the original k-space matrix as shown in Figure 6a. This was done since the FOV_y is inversely proportional to the space between data points in k-space in the phase encoding direction, Δk_y. By removing every other row in the k-space matrix, Δk_y doubles allowing the FOV_y to be reduced by half.

a) 

```
% Modify the k_space to 1/2 FOV
k_space_new2 = k_space;
k_space_new2 = k_space_new2(1:2:end,:);


% Transform the k_space into an image
img=abs(ifft2(k_space_new2));

% Display
imagesc(img)
colormap(gray(256))
brighten(0.6)
axis image; axis off
```

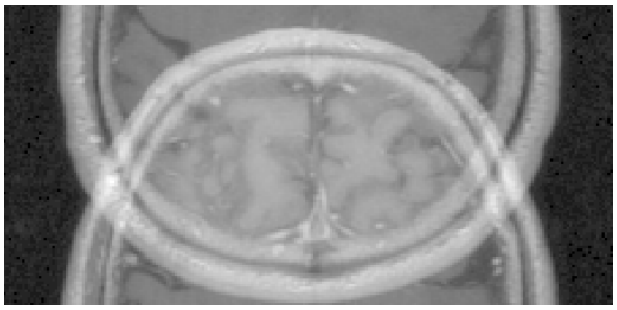
b) 

Figure 6. MATLAB code (a) used to obtain the image given by the manipulated k-space data (b).

8. ***You will notice an artifact in the image that you have just generated. What is the cause of this artifact?***

   The artifact in Figure 6b is called aliasing, also known as wrap-around, and it is a consequence of the Nyquist Sampling Theorem not being met. The Nyquist Theorem states that to accurately represent a continuous signal (or an image) without aliasing in the digital domain, the sampling rate must be at least twice the highest frequency component in the signal. When the MRI image is acquired with the full FOV in one dimension (x-dimension in this case, refer to Figure 6a), this dimension is adequately sampled and the high-frequency components of the image are captured. By taking the half FOV in the y-dimension (sampling every other row) we observe aliasing in the image's y-direction in Figure 6b as a result of not satisfying the Nyquist criterion. This artifact occurs since the FOV is smaller than the body part being imaged, and the part of the image that lies beyond the edge of the FOV is projected onto the other side of the image. In other words, parts of the body part being imaged lie outside the phase-encoding (y-direction) domain. As a result, the phases that lie outside the domain are shifted such that their spatial positions are mismapped to the opposite side causing a wrap-around (the back of the head projected over the front).

9. **Calculate the SNR in this image in a similar manner to step #4 (try and choose signal and noise regions that are similar to the ones used in step #4 if possible. However, it is more important to avoid artifact regions). How does it compare to the SNR calculated in #4? What do you expect the SNR to be relative to the SNR calculated in #4?**
The process outlined in Step 4 was repeated on the image with aliasing in Figure 6b. The selected sections for the ROI and noise are shown in Figure 7a and 7b below and were kept as similar as possible to those in Figure 3.

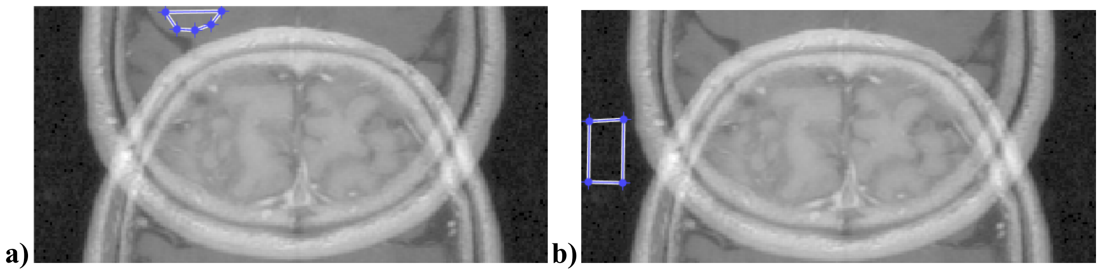


Figure 7. Selected ROIs for the signal avoiding the artifact region (a) and the noise (b).

The means and standard deviations of the ROI and background (noise) are listed below:

- Brain: Mean = 89, Standard Deviation = 3
- Noise: Mean = 12, Standard Deviation = 1

The signal-to-noise ratio (SNR) can be calculated using the following formula and the mean signal intensity of the ROI and the standard deviation of the signal intensity of the noise.

$$SNR_9 = \frac{mean\ signal\ intensity\ of\ ROI}{standard\ deviation\ of\ noise}$$
$$= 89/1$$
$$= 89$$

SNR_9 from Figure 6b is approximately the same as the SNR_4 from Figure 2b calculated in Step 4 ($SNR_4 = 87$).

When the FOV in the y-direction is halved, the y-dimension of the pixel is halved. However, the number of phase-encoding lines, N_y, is also halved which reduces the matrix size in the y-direction by a factor of 2. Therefore, the overall voxel volume remains the same and the pixels are rectangular. The reduction of the phase-encoding lines will reduce the acquisition time, T, in direct proportions. Therefore, the SNR is expected to decrease by $\sqrt{0.5} = 0.7$ and the expected SNR_9 is SNR_4 multiplied by this factor (87×0.7) producing a value of 62.

10. Manipulate the original dataset by retaining the central part of k-space around DC frequency. Retain only the raw data in the central 5 pixel-by-5 pixel square. Display the resulting image. Increase the retained frequency domain to 15 pixel-by-15 pixel, 25 pixel-by-25 pixel, and 50 pixel-by-50 pixel. Display the corresponding images. Explain what is happening. What is this operation called in signal processing terminology?

To adjust k-space's FOV, k_{FOV}, in the manner described above, the inner specified pixels in the frequency domain were retained while the outer ones were given values of 0 as shown in Figures 8a, 9a, 10a, and 11a. The resulting images (2D inverse Fourier transforms) for the 5x5, 15x15, 25x25, and 50x50 inner k-space pixels are shown in Figures 8b, 9b, 10b, and 11b respectively.

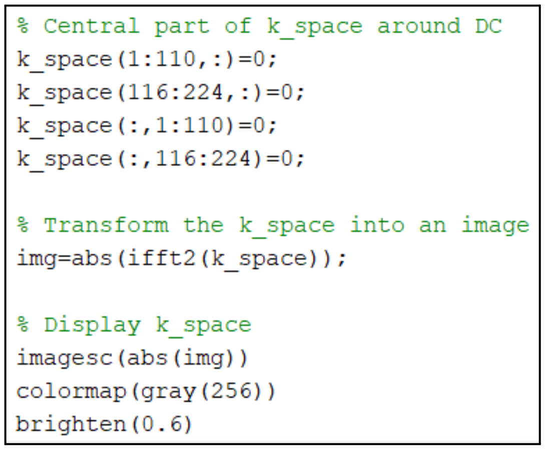
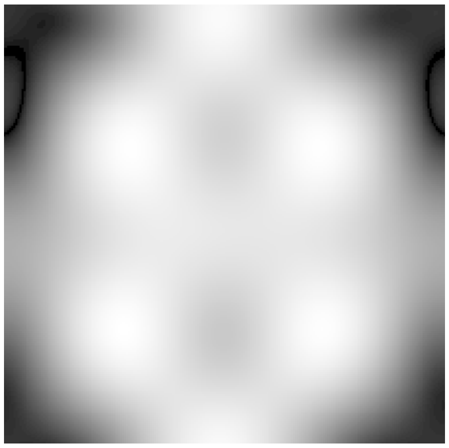

```
% Central part of k_space around DC
k_space(1:110,:)=0;
k_space(116:224,:)=0;
k_space(:,1:110)=0;
k_space(:,116:224)=0;

% Transform the k_space into an image
img=abs(ifft2(k_space));

% Display k_space
imagesc(abs(img))
colormap(gray(256))
brighten(0.6)
```
a) b)


Figure 8. The MATLAB code (a) used to obtain the image given by the central 5x5 k-space pixel square (b).

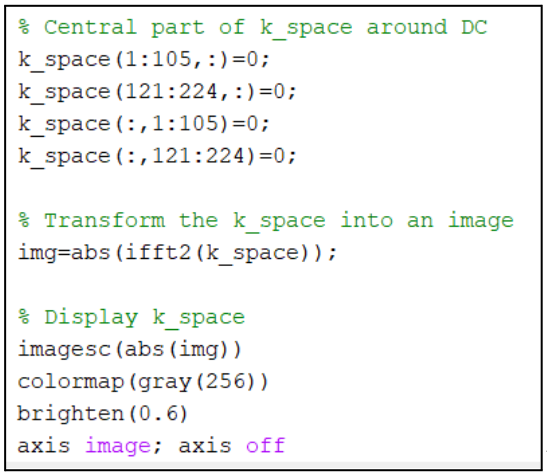
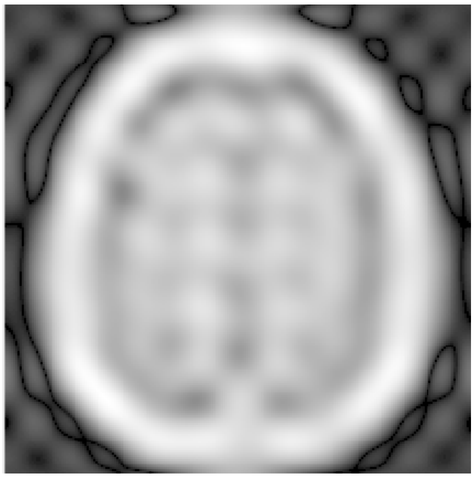
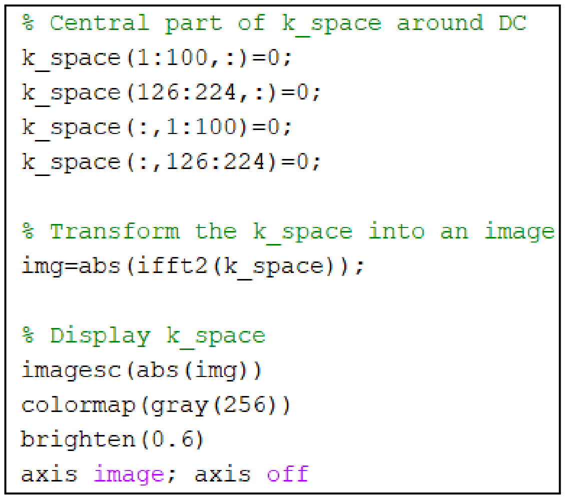

```
% Central part of k_space around DC
k_space(1:105,:)=0;
k_space(121:224,:)=0;
k_space(:,1:105)=0;
k_space(:,121:224)=0;

% Transform the k_space into an image
img=abs(ifft2(k_space));

% Display k_space
imagesc(abs(img))
colormap(gray(256))
brighten(0.6)
axis image; axis off
```
a) b)


Figure 9. The MATLAB code (a) used to obtain the image given by the central 15x15 k-space pixel square (b).

a) 

```
% Central part of k_space around DC
k_space(1:100,:)=0;
k_space(126:224,:)=0;
k_space(:,1:100)=0;
k_space(:,126:224)=0;

% Transform the k_space into an image
img=abs(ifft2(k_space));

% Display k_space
imagesc(abs(img))
colormap(gray(256))
brighten(0.6)
axis image; axis off
```

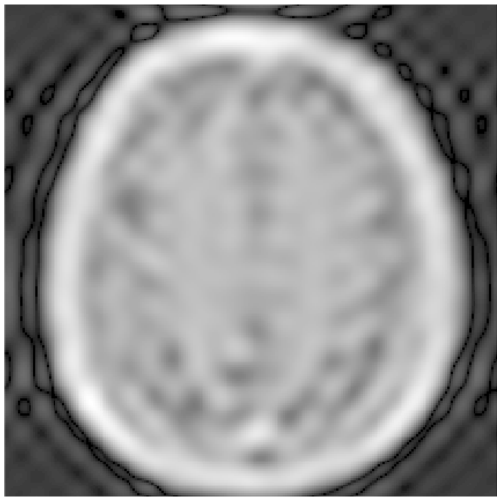
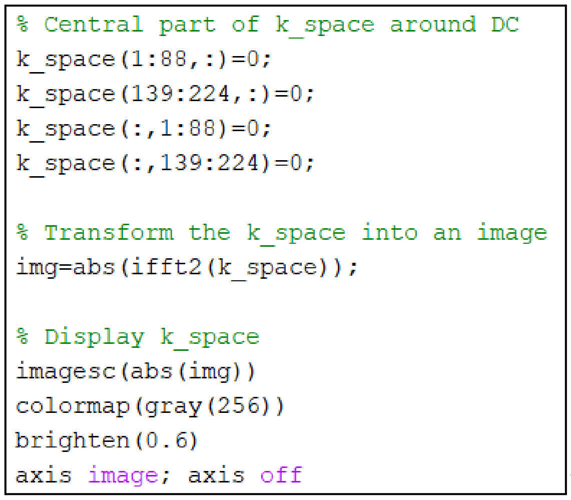
b) 

Figure 10. The MATLAB code (a) used to obtain the image given by the central 25x25 k-space pixel square (b).

a) 

```
% Central part of k_space around DC
k_space(1:88,:)=0;
k_space(139:224,:)=0;
k_space(:,1:88)=0;
k_space(:,139:224)=0;

% Transform the k_space into an image
img=abs(ifft2(k_space));

% Display k_space
imagesc(abs(img))
colormap(gray(256))
brighten(0.6)
axis image; axis off
```

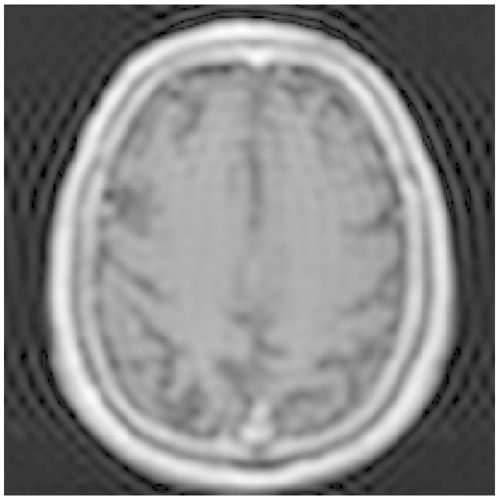
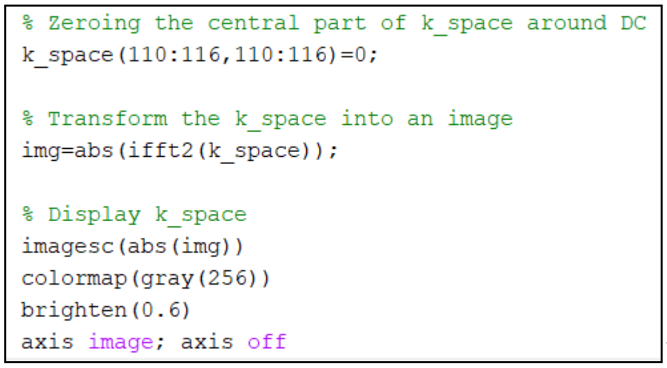
b) 

Figure 11. The MATLAB code (a) used to obtain the image given by the central 50x50 k-space pixel square (b).

The defined FOV of k-space, k_{FOV}, is inversely related to the pixel width (Δw) and determines the number of digitized samples in the k-space required to reconstruct an image with the desired resolution. In Figure 8, by sampling only the center 5-by-5-pixel square, only the lowest spatial frequencies are present in the k-space. As the image is produced by the undersampled data, it appears blurry and lacks the high spatial frequency details. By increasing the retained frequency domain to 15x15, 25x25, and 50x50 pixels, the pixel size is continually decreased. The FOV of the resulting image remains unaffected during this process. As a result of the decreased pixel size, the spatial resolution increases and higher spatial frequency portions of the k-space are sampled. Therefore, higher resolution images are observed as the retained frequency domain (k-space) is increased. This operation is called “k-space subsampling” in MRI or “frequency domain subsampling” in signal processing.

11. ***Manipulate the original dataset by zeroing the central part of k-space around DC frequency. Remove only raw data in the central 5 pixel-by-5 pixel square. Display the resulting image. Increase the zeroed frequency domain to 15 pixel-by-15 pixel, 25 pixel-by-25 pixel, and 50 pixel-by-50 pixel. Display the corresponding images. Explain what is happening. What is this operation called in signal processing terminology?***

To adjust k-space's FOV, k_{FOV}, in the manner described above, the outer pixels in the frequency domain were retained while the inner specified were given values of 0 as shown in Figures 12a, 13a, 14a, and 15a. The resulting images (2D inverse Fourier transforms) for the 5x5, 15x15, 25x25, and 50x50 inner k-space pixels are shown in Figures 12b, 13b, 14b, and 15b respectively.

a)


```
% Zeroing the central part of k_space around DC
k_space(110:116,110:116)=0;

% Transform the k_space into an image
img=abs(ifft2(k_space));

% Display k_space
imagesc(abs(img))
colormap(gray(256))
brighten(0.6)
axis image; axis off
```

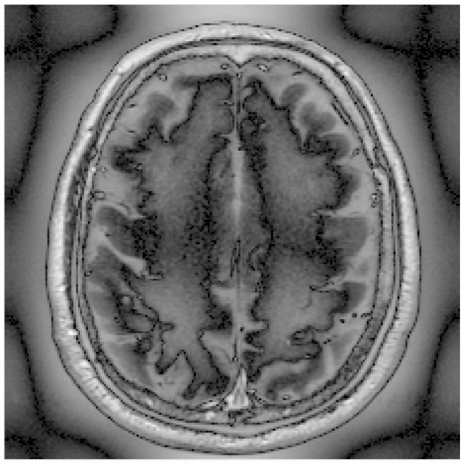
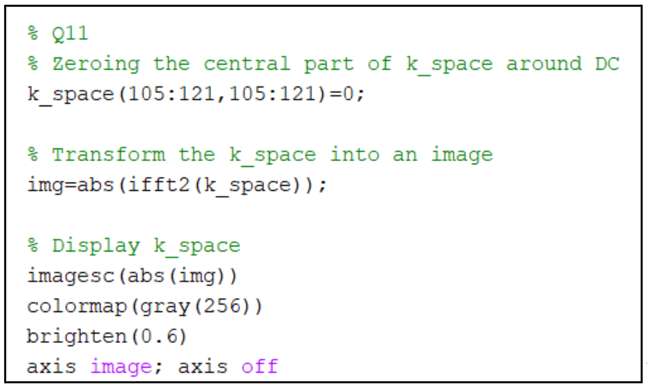
b)


Figure 12. The MATLAB code (a) used to obtain the image given by the k-space outside the 5x5 pixel square (b).

a)


```
% Q11
% Zeroing the central part of k_space around DC
k_space(105:121,105:121)=0;

% Transform the k_space into an image
img=abs(ifft2(k_space));

% Display k_space
imagesc(abs(img))
colormap(gray(256))
brighten(0.6)
axis image; axis off
```

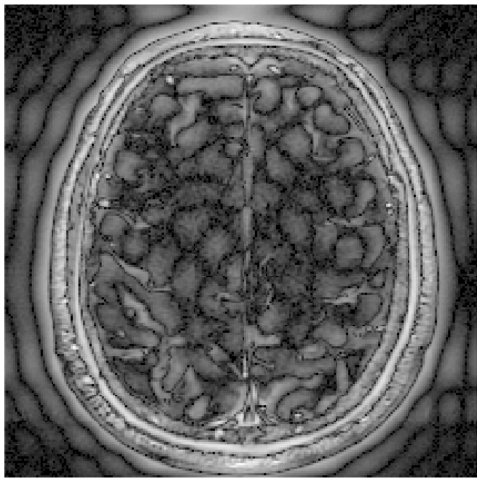
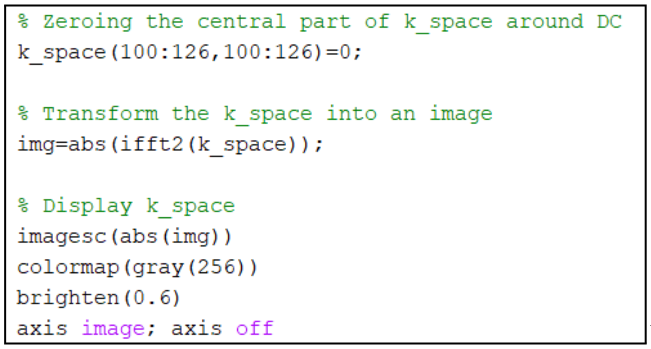
b)


Figure 13. The MATLAB code (a) used to obtain the image given by the k-space outside the 15x15 pixel square (b).

a) 

```
% Zeroing the central part of k_space around DC
k_space(100:126,100:126)=0;

% Transform the k_space into an image
img=abs(ifft2(k_space));

% Display k_space
imagesc(abs(img))
colormap(gray(256))
brighten(0.6)
axis image; axis off
```

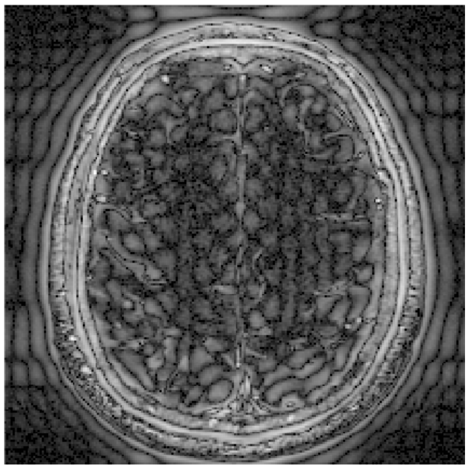
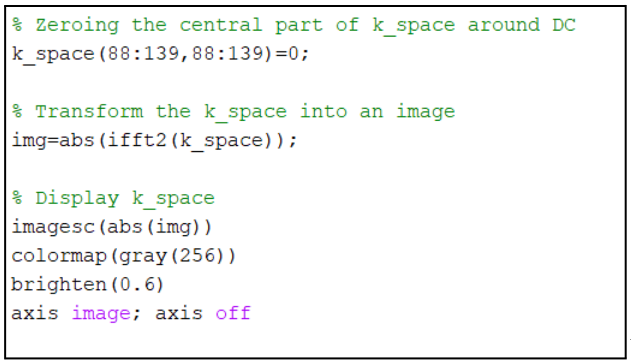
b) 

Figure 14. The MATLAB code (a) used to obtain the image given by the k-space outside the 25x25 pixel square (b).

a) 

```
% Zeroing the central part of k_space around DC
k_space(88:139,88:139)=0;

% Transform the k_space into an image
img=abs(ifft2(k_space));

% Display k_space
imagesc(abs(img))
colormap(gray(256))
brighten(0.6)
axis image; axis off
```

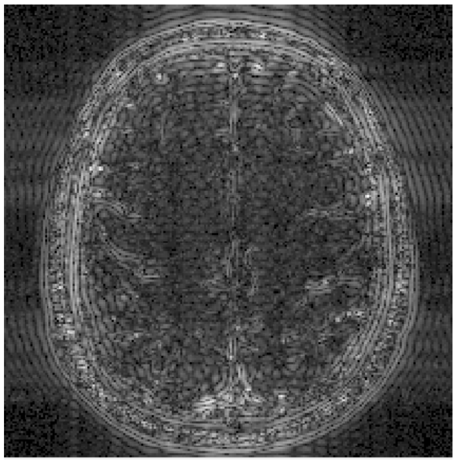
b) 

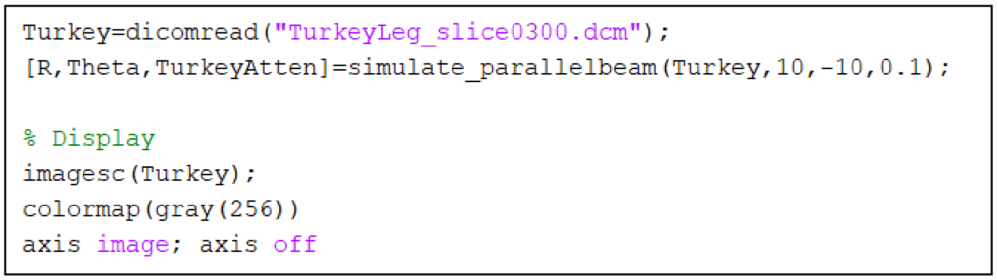
Figure 15. The MATLAB code (a) used to obtain the image given by the k-space outside the 50x50 pixel square (b).

The central region of the k-space contains low spatial frequency information, such as the contrast of the image. By zeroing out the central 5x5 pixel part of k-space around DC frequency, this low spatial frequency information is removed from the image, but the sharp edges and detail from the high spatial frequencies are retained. As seen on Figure 12b, the image appears to lack overall contrast. By increasing the zeroed frequency domain to 15x15, 25x25 and 50x50 pixels, more and more low spatial frequency information is removed. The resulting images in Figures 13b to 15b, have increasingly sharper details. This operation is called “k-space filtering” or “k-space masking” in MRI and “frequency domain filtering” in signal processing.

PART 2: CT IMAGE RECONSTRUCTION

From the Turkey Leg volume set, select a full resolution slice that is "interesting" (i.e. it has lots of bone and soft tissue, but no artifacts). For the purposes of this section, this will be assumed to be a "noisefree" object. Create a sinogram R using the function simulate_parallelbeam().

The "TurkeyLeg_slice0300.dcm" image was used for analysis in the following steps. This image is shown below in Figure 16b.

a)


```
Turkey=dicomread("TurkeyLeg_slice0300.dcm");
[R,Theta,TurkeyAtten]=simulate_parallelbeam(Turkey,10,-10,0.1);

% Display
imagesc(Turkey);
colormap(gray(256))
axis image; axis off
```

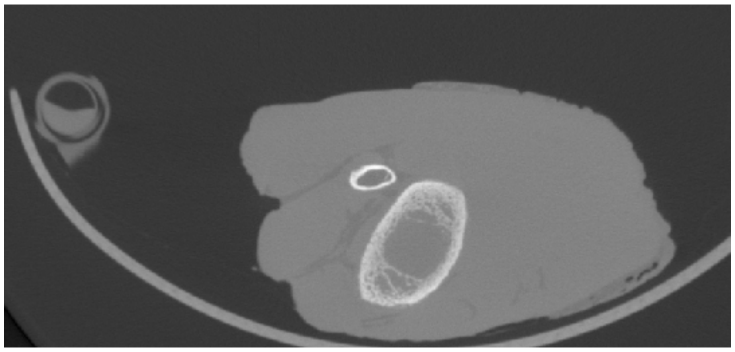
b)


Figure 16. The MATLAB code (a) used to display the selected turkey leg image (slice0300) (b).

Radon Transform

1. ***Plot the radon transform at angle 0°. Be sure to label your axes and include the proper units.***
   Figure 17 below is the plot of the radon transform at the projection angle of 0˚.

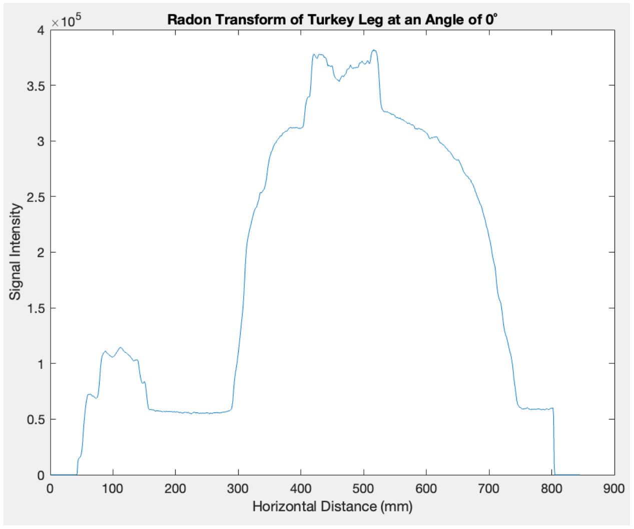
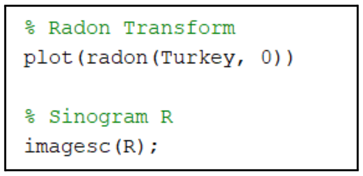


Figure 17. Plot of the radon transform at angle 0˚.

2. ***Show the sinogram, R, as an image with the axes properly labeled.***
   The sinogram of the turkey-leg cross section is plotted below in Figure 18b.

a)


```
% Radon Transform
plot(radon(Turkey, 0))

% Sinogram R
imagesc(R);
```

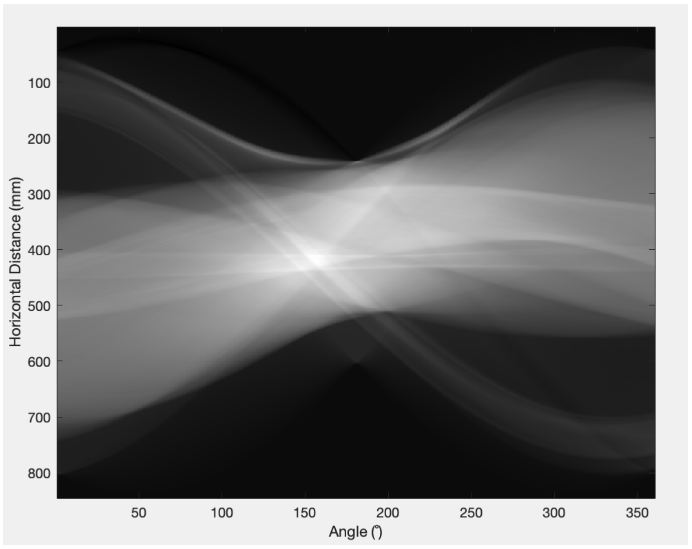
b)


Figure 18. The MATLAB code (a) used to produce the sinogram of the turkey leg (b).

3. ***Compare the figures: original object, projection at zero degrees, and the sinogram. Are they consistent? What is the area under plot #1? How does this area change as a function of angle?***
   Figures 17 and 18 are consistent with the turkey leg cross-section in Figure 16. Figure 17 corresponds to the signal intensities across the projection axis at an angle of 0˚. The angle of 0˚ is equivalent to the horizontal projection through the medium. The four circular objects within the image are of a greater signal and this is shown in the radon plot by the four intensity peaks at 100 mm, 300 mm to 700 mm, and 400 mm and 500 mm (within the 300 mm to 700 mm intensity peak). The location of these peaks are consistent with that shown in the image. The sinogram in Figure 18 also shows these relatively circular objects in the turkey leg. These are indicated by the relatively uniform width in the object projections; circular objects have an even width from all angle views. There are slight variations in the width of each of the object projections since they are not perfectly circular. The spacing of the objects at angle 0˚ on Figure 18 also corresponds to the radon transform in Figure 17 as shown by the consistent peaks in signal intensity.

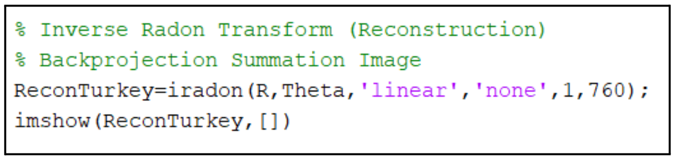
   The area under the plot of Figure 17 was determined to be 1.5136E+8 mm (since signal intensity is unitless, the area has units of mm from the horizontal distance axis) using the trapz() function. The area under the curve should not change as a function of angle since the summation of all signal intensities should be the same as the image is viewed from different angles. However, the shape of the radon transform will be different at each angle due to the differing locations of high and low signal intensity along the axis of projection. This is demonstrated in Figure 18b as the summation of signal intensities remains the same with angle but the location of these intensities differs.

<u>**Inverse Radon Transform (Reconstruction)**</u>

For the lab report: Complete all parts for the Radon and Inverse Radon transform components showing plots or images where appropriate. Comment on the image quality in a, b.i, b.ii, b.iii and any artifacts encountered.

1. *Use the inverse radon transform, iradon(), to perform a CT reconstruction of the sinogram data.*
   a. ***Create a "backprojection summation image" (i.e. simple back-projection), by setting the FilterType='none'. Show the resulting reconstruction image. Describe the overall image quality. Why is this form of reconstruction incorrect?***

   The simple back-projection reconstruction of the turkey leg was created using the code in Figure 19a and is displayed in Figure 19b.

a)


```
% Inverse Radon Transform (Reconstruction)
% Backprojection Summation Image
ReconTurkey=iradon(R,Theta,'linear','none',1,760);
imshow(ReconTurkey,[])
```

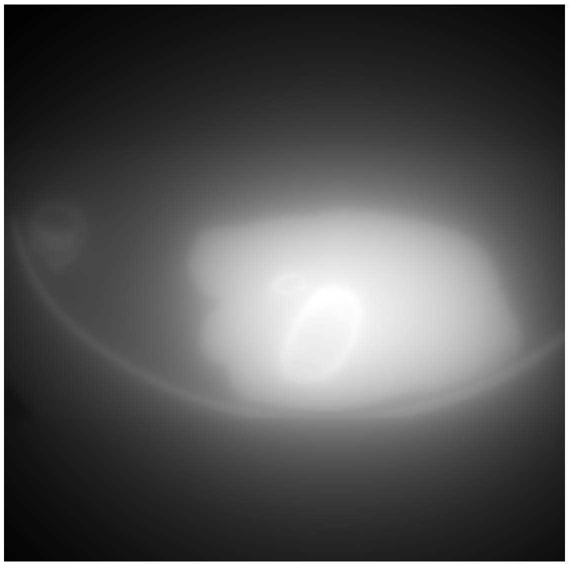
b)


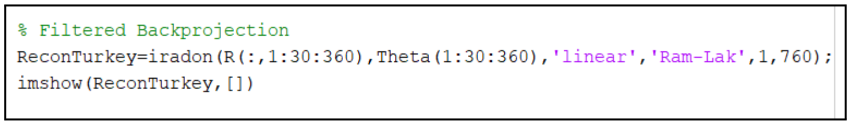
Figure 19. The MATLAB code (a) used to reconstruct the image through simple back-projection (b).

Simple back-projection is the back-projection of the signal intensity data along the sampled axes of projection. This can be visualized as a smearing of the signal along these axes, which results in a blurry (poor quality) reconstruction as shown in Figure 19b. This form of reconstruction results in the removal of high spatial frequency content, which is why smaller details in the image are not resolved, but contrast is maintained. This can be mitigated by applying a high-pass filter (the ramp filter is the ideal inverse blurring function) to recover high spatial frequency data in a process called filtered back-projection, which is detailed in the following sections and images.

b. Create a "filtered backprojection" reconstruction image by setting. FilterType='Ram-Lak' which implements the ramp filter. Show the resulting reconstructed slices for:

i. A reconstruction from every 30th projection.

The turkey leg image was reconstructed using every 30th projection from the data set of 360 projections using the code in Figure 20a and is displayed in Figure 20b.

a) 

```
% Filtered Backprojection
ReconTurkey=iradon(R(:,1:30:360),Theta(1:30:360),'linear','Ram-Lak',1,760);
imshow(ReconTurkey,[])
```

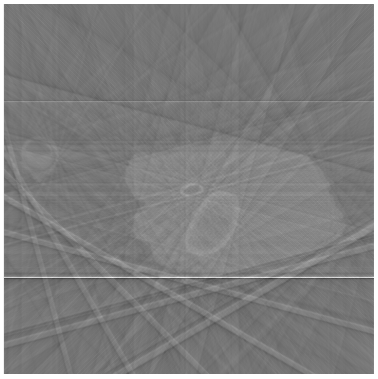
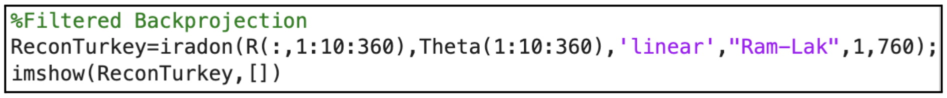
b) 

Figure 20. The MATLAB code (a) used to reconstruct the image using filtered back-projection from every 30th projection (b).

In comparison with the simple-backprojection (Figure 19b), the filtered back-projection in Figure 20b allows for better resolution of smaller details in the image and therefore, improved quality. However, there is very low contrast and the projection lines are clearly shown. This is due to the low number of projection lines and therefore larger intervals in angle sampling used to reconstruct the image.

ii. A reconstruction from every 10th projection.

The turkey leg image was reconstructed using every 10th projection from the data set of 360 projections using the code in Figure 21a and is displayed in Figure 21b.

a) 

```
%Filtered Backprojection
ReconTurkey=iradon(R(:,1:10:360),Theta(1:10:360),'linear',"Ram-Lak",1,760);
imshow(ReconTurkey,[])
```

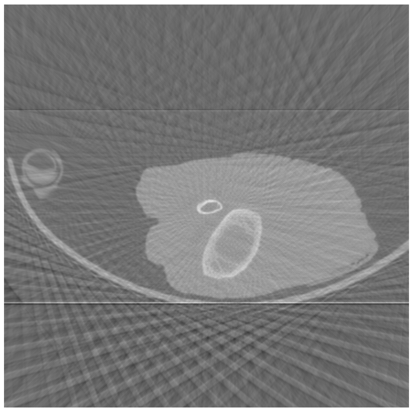
b) 

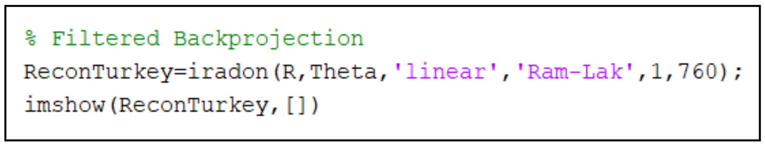
Figure 21. The MATLAB code (a) used to reconstruct the image using filtered back-projection from every 10th projection (b).

In comparison with the simple-backprojection (Figure 19b) and filtered-backprojection in Figure 20b, Figure 21b further increases the resolution of smaller details in the image and therefore, is of better quality. The increase in projection lines allows for better contrast since there are more radial axes in which the signal is "smeared" back. The projection lines are still clearly shown though due to large intervals in angle sampling.

iii. A reconstruction using all the projections.

The turkey leg image was reconstructed using every projection from the data set of 360 projections using the code in Figure 21a and is displayed in Figure 21b.

a)



```
% Filtered Backprojection
ReconTurkey=iradon(R,Theta,'linear','Ram-Lak',1,760);
imshow(ReconTurkey,[])
```

b)

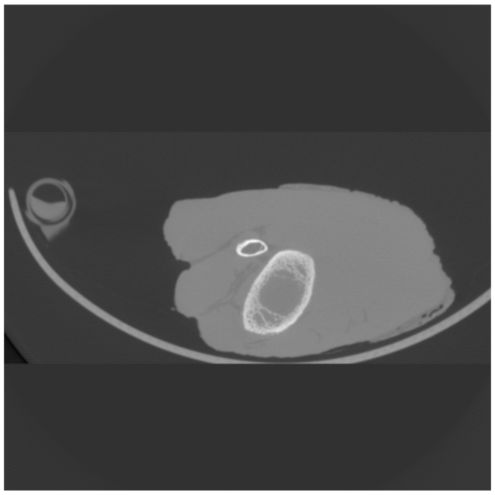


Figure 22. The MATLAB code (a) used to reconstruct the image using filtered back-projection from every projection (b).

The utilization of all 360 filtered back-projections creates a very high quality image comparable to the original shown in Figure 16b. The high spatial frequency data recovered from the filter contributes to improved resolution while the increase in the number of back-projections increases contrast. As a result of the number of back-projections being increased, projection lines can not be easily distinguished on the image due to the smaller intervals in angle sampling.