RE4017 Project:

Image Reconstruction from a Sinogram

Date: 20/04/20

Name: Bryan Tiernan

ID: 16169093

Table of Contents

Image Reconstruction Overview:

The aim of this project is to implement image reconstruction from parallel projection sinograms using Python.

A sinogram is a discrete representation of the Radon transform of an image.

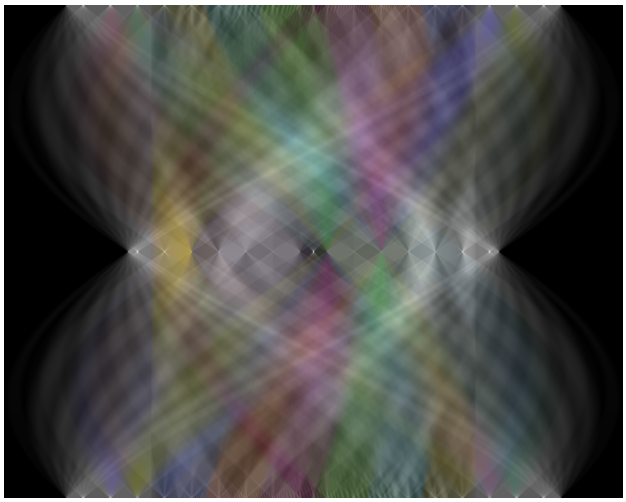

Figure 1: Sinogram used for project

As can be seen by looking at the given sinogram, no information can be extracted from the sinogram by examination. The image needs to be manipulated in order to extract meaningful information from it.

Reconstruction without ramp-filtering:

In order to reconstruct this image, the sinogram needs to be split into 3 single channel sinograms, red, green and blue channels. This is because back projection reconstruction only works on single-channel data, so each channel needs to be back projected individually. Each row of the single channel sinograms represents a Radon transform as a particular angle. After each channel is back projected, the single-channel images need to be rescaled into 8-bit greyscale before they can be recombined into 24-bit colour RBG image.

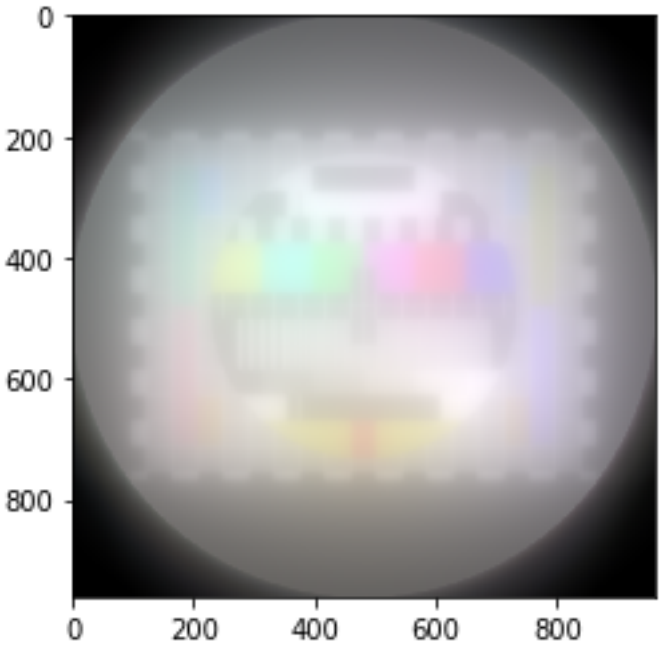

Figure 2: Reconstruction without filtering

Figure 2 shows the reconstructed image without any ramp-filtering. This reconstruction gives us some idea of what the original image looks like, however it has a large amount of blurring. This is a result of the back-projections which are being used to create the image. When the back projections are summed and consequently normalised, this blurring characteristic beings to appear. This blurring can be combatted by implementing the Fourier slice theorem and a ramp filter, as shown in the next section.

Reconstruction with ramp-filtering:

Steps clearly need to be taken in order to reduce the amount of blurring associated with the reconstructed image. This can be tackled using the Fourier slice theorem and a ramp filter, and can be implemented in the following steps:

1. Load the sinogram and split into 3 single channels
2. Translate each row of the sinogram into the frequency domain, which is the 1-d Fourier Transform applied to each row of the sinogram.
3. Ramp-filter the FT of each row.
4. Return the filtered data to the spatial domain using inverse FT.
5. Reconstruct single-channel image using back projections
6. Combine all single channel images into a 3-channel RBG image (rescaling required).

The reason that the sinogram needs to translated from the spatial domain into the frequency domain is because the ramp-filter is not band-limited and therefore has no meaningful spatial domain representation.

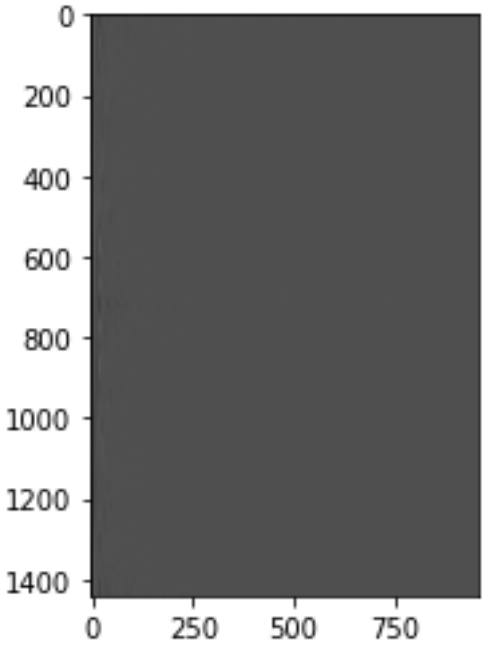

Figure 3: Frequency domain translation of red channel sinogram

Figure 3 is the red channel sinogram represented in the frequency domain. It is quite hard to make out the frequency components from this image, but there are a number of them on the left hand side of the image. In order to generate this image, the sinogram was passed through scipy.fftpack.rfft(sinogram,axis=1) as a numpy array, which transforms each row sinogram into a 1-d real FFT separately. This one call computes each FFT, and outputs as shown in figure 3. Each of these projections is one row in the array outputted.

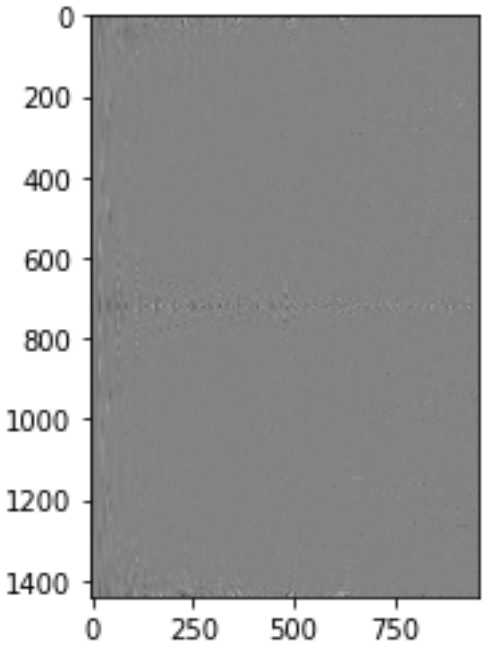

Figure 4: Frequency domain translation of red channel sinogram with ramp filter

Figure 4 again shows the frequency domain translation of the red channel sinogram, except this time with the ramp filter implemented. This is implemented by multiplying each of the projections or FFT rows by the ramp function. This ramp function accounts for mixed real and imaginary components or elements in the FFTs of these projections. This manipulation gives us the image shown in figure 4. Similar to the other frequency domain representation, it does not visually represent anything significant to us now, but this transform is meaningful as we will see when we reconstruct the image.

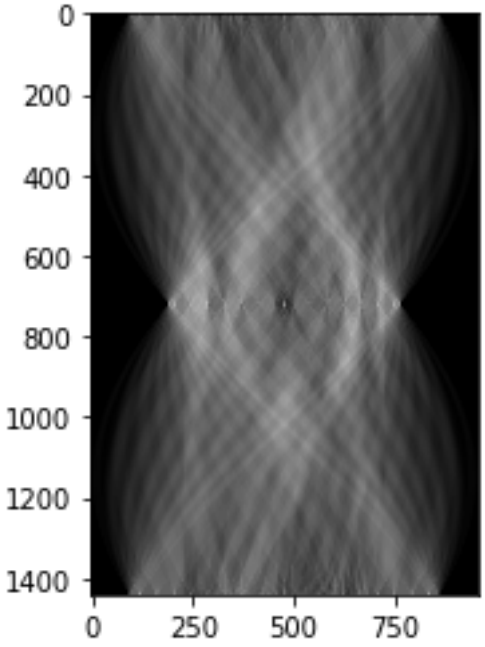

Figure 5: Sinogram of red channel originally

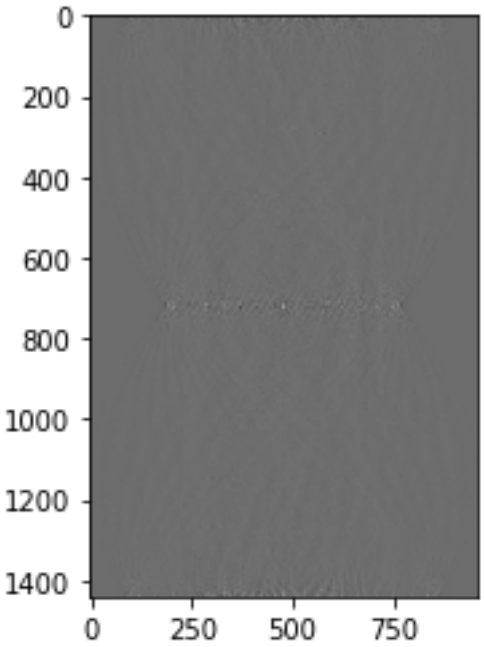

Figure 6: Spatial domain translation of red channel sinogram with ramp filter

Figure 6 shows the red channel sinogram translated back to the spatial domain. The sinogram is translated back to the spatial domain by using the inverse Fourier transform. The shape and general outline of the sinogram can be seen in figure 6, in contrast with figure 5 which is the channel originally with no manipulations at all. When looking at the 2 images side by side now the original version seems much sharper and clearer. However, the later representation of the sinogram will be shown to give us a much clearer version of the original image when we recombine the sinograms. This is because much of the unneeded frequency components have been filtered out.

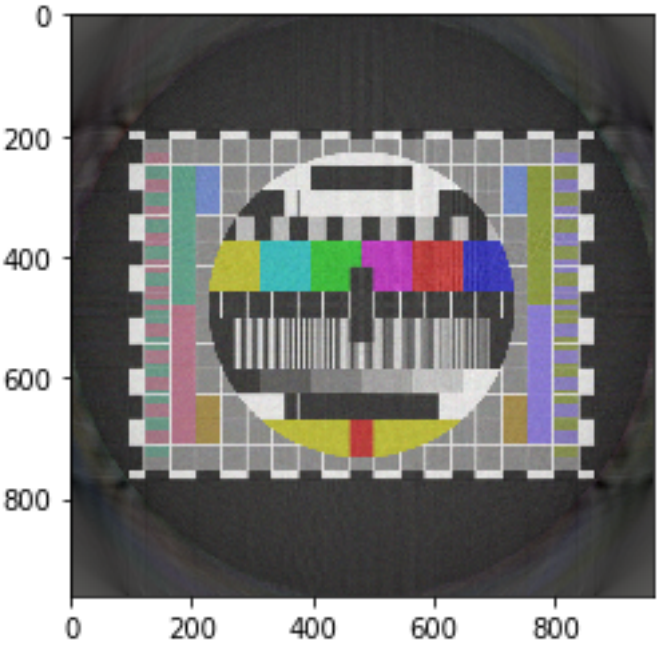

Figure 7: Reconstructed Image

Figure 7 is the reconstructed image. Comparing this with figure 2, the reconstructed image without filtering, we can clearly see that performing the filtration steps have given us a much clearer representation of the original image.

However, this reconstruction process is not without flaws. There is still a large amount of blurring upon further inspection of the image, on the borders or lines in the image. An interference pattern can be seen in the background of the image, and there are artefacts around the edges of the image. This is a result of the fact that the ramp filter allows high frequency components into the output image, causing these artefacts. This shows that improvements can still be made in the filtration or reconstruction process.

Reconstruction using a Hamming-windowed ramp filter:

As discussed, the artefacts seen in figure 7 are a result of high frequency components being emphasised by the ramp filter. We can reduce artefacts by implementing a Hamming window on the FFT, which will dampen the high frequency terms produced by the FFT.

Unfortunately, I was unable to implement a Hamming window into this project. However, based on the Hamming window applied, the ringing artefacts should be reduced, as well as the inference pattern.

Conclusion:

Aside from the implementation of the Hamming window, I feel that the project has been successfully implemented. It is clear that in order to successfully reconstruct an image from a sinogram, a filter needs to be carefully implemented in order to get a clear representation of the original image. Furthermore, a window function should be implemented to further improve the quality of the image and reduce artefacts.

While I only used the red channel sinogram for the purposes of explaining the process, all images created throughout this process have been included with the project, alongside the code. (The final image appears in black and white in the images, but if you run the code it comes out in colour, I couldn’t figure this out? Type error?)