Introduction to Image and Video Processing - Project 1

Pie de Boer, i6272959

8 May 2023

Disclaimer

Most of the theoretical understanding is gained through reading and studying the material in Digital Image Processing 4th edition by Gonzalez. Most practical knowledge, is gained through the documentation of OpenCV, Numpy and various related libraries. The code of the labs, in many cases, served as a starting point. The code contains an extensive "resources.txt" file of articles that were read, but not necessarily used.

Color Spaces

HSV and RGB Color Models

This task investigates the RGB and HSV color models. The pictures chosen clearly help with understanding how these work. RGB images are represented by three components or channels, that represent the primary colors. Combined, fed to a monitor, these generate the composite color for each picture element [3]. Therefore, splitting the channels, and investigating the intensities in each channel, allows use to understand how to the composite color is formed.

When picture elements look more white, they have higher values in that channel. For example, assume there is a lot white in the red channel, we can expect a more red-ish color in the composite image. The RGB Venn Diagram nicely displays this principle.

Figure 1: RGB Venn Diagram that illustrates additive mixing (source: [3]

Let's investigate these principles in practice .

Figure 2: Flamingo image (RGB)

If we correctly applied the splitting of the channels, we will observe, that luckily, the water is not made of blood and flamingo's are still pink/red-ish. The individual channels, reveal, pixel intensities of the Red, Blue and Green channels, used to build the 'additive' composite color.

Figure 3: Flamingo image split into Red, Blue and Green channels

When examining a picture that has less 'popping' colors, we observe that the image of the deer clearly has a orange-ish composite color. This, should correspond to a more white looking Red channel. Using the RGB Venn diagram as an aid, we will find that there will be a bit higher pixel intensities in the Green channel than Blue channel. This is indeed in accordance with our empirical findings.

Figure 4: Deer image (RGB)

Figure 5: Deer image split into Red, Blue and Green channels

Our journey, investigating the RGB color model, has helped us to build composite colors by additive mixing using three primary colors. To investigate, concepts as 'saturation' and 'value' in a more natural way, we will look into the HSV color model.

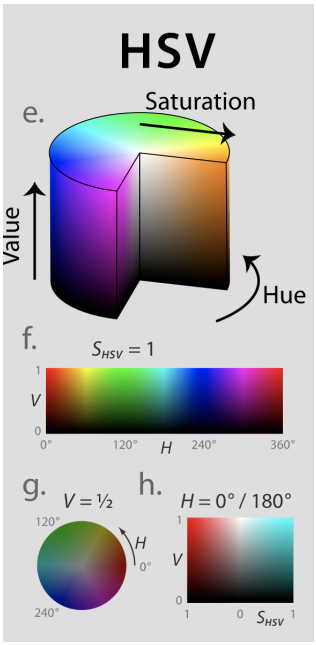

Figure 6: HSV Model (source: [4]

Looking at our pictures, we observe that swimming pool items have a very intense color. The flamingo is light pink, more towards white-ish, which corresponds to less saturation. However, the pink waterbed, is 'intense pink', very saturated, which corresponds with our findings in the separated HSV channels. The values on the hue channel strongly differ for the flamingo and rubber band and indeed they have vastly different colors in the composite image.

Figure 7: Flamingo image split into Hue, Saturation and Value channels

The deer image has a predominant orange tone. As a result, the hue channel should be fairly consistent throughout. The plants in the foreground are more vibrant in color and appear darker because they are less white than the

background. This causes the foreground to have darker intensities in the value channel compared to the lighter background, which has a more white-ish sky.

Figure 8: Deer image split into Hue, Saturation and Intensity channels

Histogram Equalization (HSV)

This exercise investigates histogram equalization, a method that can be used for contrast enhancement [3] Histograms, are plots of intensities, therefore, the distributions reveal, the contrast in images. When plotting a histogram of an image, the color black typically corresponds to the frequency of pixel values that are zero, while white corresponds to the frequency of pixel values that are 255 (in an 8-bit image). The picture from the book, helps to build intuition. Again, the flamingo and deer images are used.

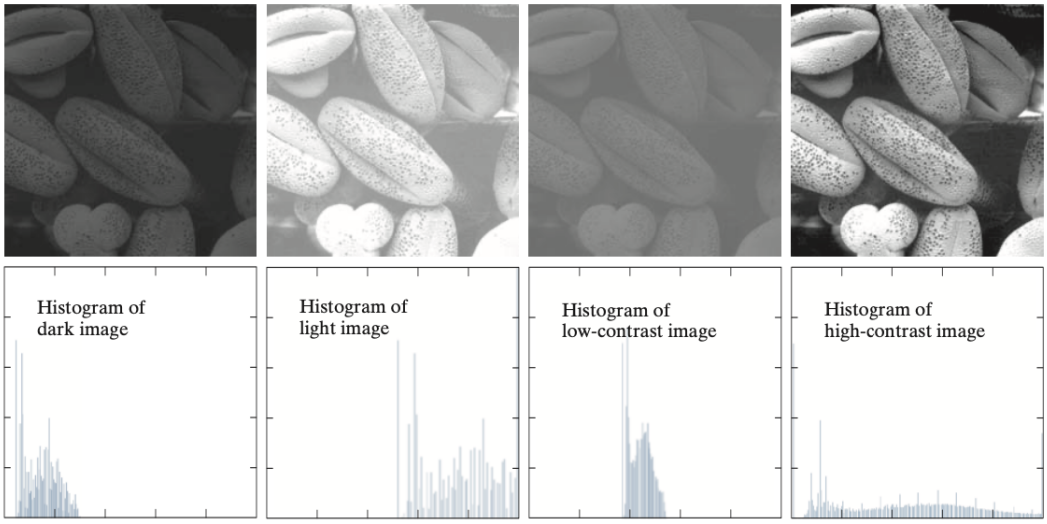

Figure 9: Four images of varying contrast with their corresponding histogram (source: [3])

Separating images into their Hue, Saturation and Intensity channels, gives us the freedom to obtain their histograms. We therefore obtain the distribution of pixel intensities in each of these channels, on which we then can apply equalization, i.e. using build in openCV method 'equalizeHist'.

This process resulted in the following histograms for the flamingo image.

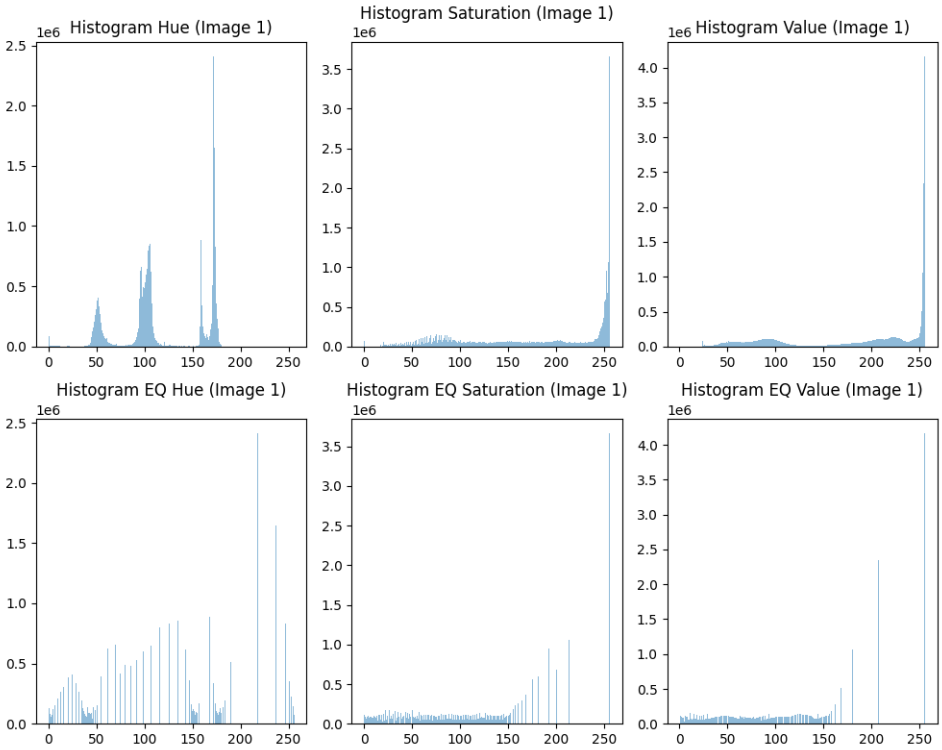

Figure 10: Histograms of HSV channels before and after equalization for the flamingo image

With the corresponding visual representation of the HSV channels before and after histogram equalization.

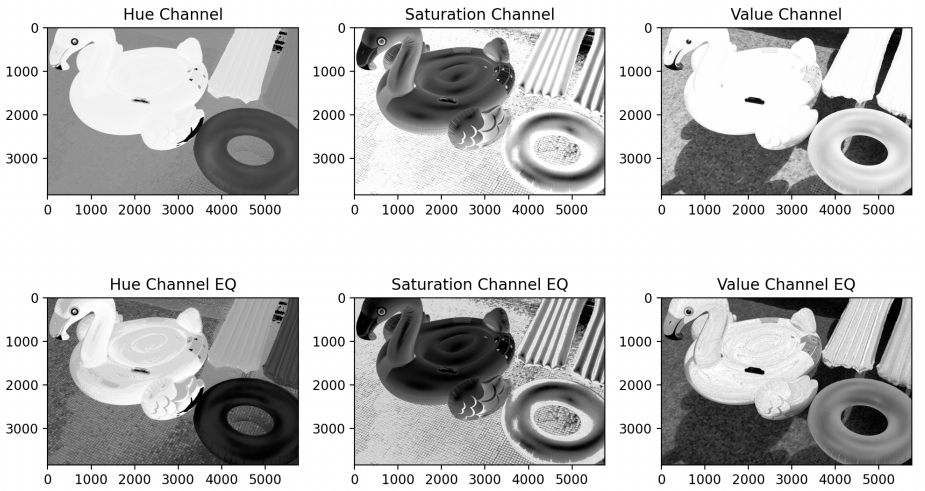

Figure 11: Separate HSV channels of flamingo image before and after histogram equalization

After equalization, we were are able to merge the channels back. This gave us the following image.

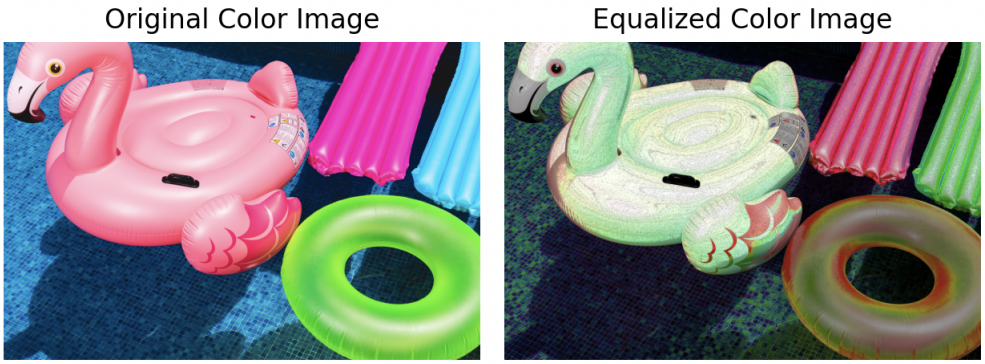

Figure 12: Flamingo image before and after applying equalization all HSV channels

Let's ask ask ourselves the question, is it a wise decision to apply histogram equalization to all the channels of the HSV color model? Unless we are looking for a funny effect, it's unlikely we want the color/hue changed of an image. This happens when the hue channel is being equalized, because the intensities, that

certain pixels have corresponding to certain 'hue' are being shifted.

Looking at the saturation channel, having the waterbed less or more densely/saturated pink, is not of huge interest at this moment. However, later, in the special effects subsection, clipping the saturation channel helped to make it look more cartoony. The colors become more saturated. Which demonstrates there are use cases for this procedure.

Now, looking into the value channel. Assume there is a dense 'area' in a histogram plot for certain pixel intensity values. This means, a lot of pixels share the same intensity, which for the flamingo corresponds to being all pink-white . The flamingo doesn't have a lot of contrast. Therefore, by distributing, the pixels intensity more evenly, the flamingo will have different shades of pink, which corresponds visually to a higher contrast.

To expand our understanding of the HSV color model, we will apply the same procedures listed before to the image of the deer.

We can observe in the hue channel, that there is clearly more intensity in one region around 30, which is in line with an orange-ish color. After equalization, the pixels are more distributed, but shifted to region around 155 and 235. The hue color wheel can aid us in understanding this.

Following histogram equalization, the pixel intensities in the saturation channel are distributed more widely. As a result, all colors become more saturated instead of just a few.

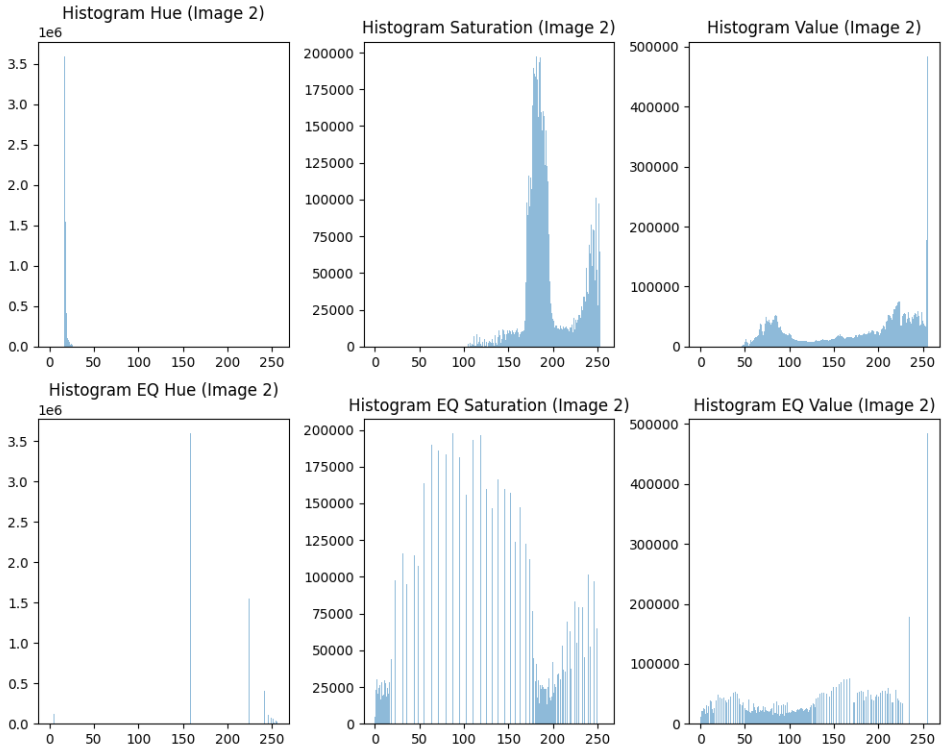

Figure 13: Histograms of HSV channels of deer image before and after equalization

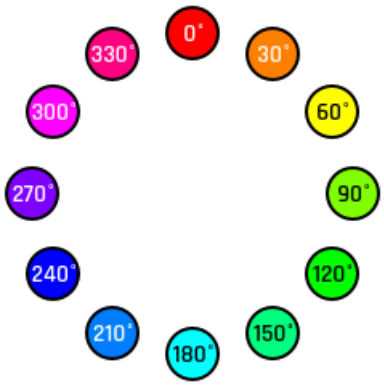

Figure 14: Hue color wheel [1]

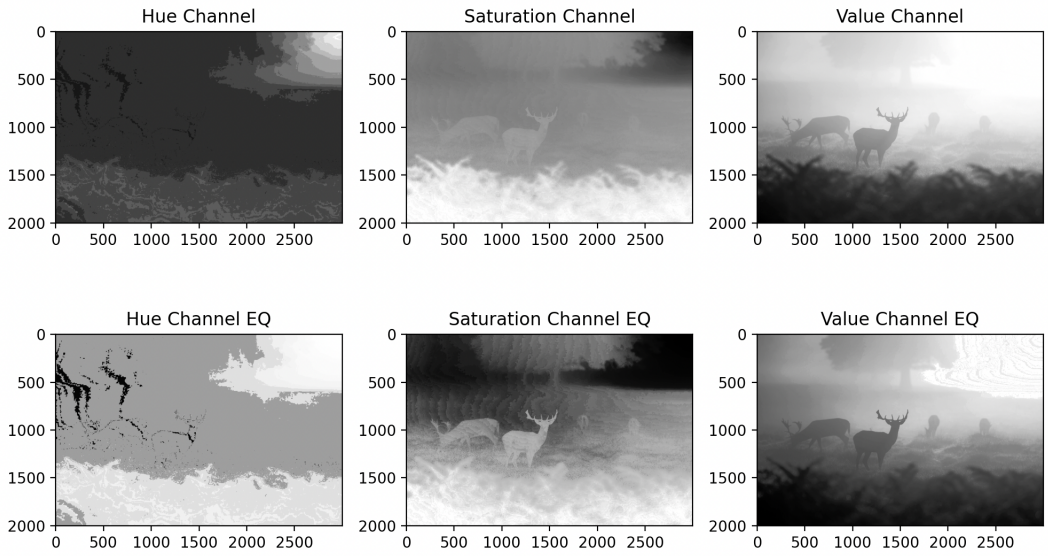

Figure 15: Separate HSV channels of deer image before and after histogram equalization

Once we applied histogram equalization to all HSV channels, the resulting merged image exhibited a purple hue. A majority of the regions appeared more saturated, and the overall contrast of the image was enhanced. This outcome aligns with our observations from the histograms and visually perceived equalized channels.

Figure 16: Deer image before and after applying equalization all HSV channels

Histogram Equalization (V-Channel)

Given that our flamingo has turned green, it could be a wise decision to only equalize the value channel. As explained before, we do not shift any color right now, by distributing the pixel intensity in regards to hue, and leave the saturation channel untouched as well. Let's leave the blue waterbed, as saturated blue as it is! Merging the flamingo image, with only the 'value' channel equalized gave us the following result (for reasons explained before).

Figure 17: Flamingo image before and after applying histogram equalization only to the Value channel

By equalizing only the value channel of the deer image, we obtained the following outcome: The elements within the image became more distinct, allowing for better visibility. For instance, the tree in the background is now more clearly observable due to the increased contrast that arises from equalizing the value channel, as we previously explained. The colors and saturation channels were untouched.

Figure 18: Deer image before and after applying equalization only to the Value channel

Edge Detection

First Order Spatial Edge Detection

Edges correspond to high changes in intensity along line [3]. By applying convolution with an appropriate kernel, we are able to detect these. In this case, a first order kernel for 45-degree edge detection was build. It can be derived from the orientation of the numbers within the matrix which diagonal is being detected.

$$\begin{bmatrix} -1 & -1 & 0 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}$$

Figure 19: First order 45-degree edge detection kernel

Rotating the previously mentioned matrix by 90-degrees allows us to detect 135 degrees edges. This gives us the following kernel.

$$\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

Figure 20: First order 135-degree edge detection kernel

After applying our kernels of interest on an image with prominent 45-degree and 135-degree lines/edges, and subsequently applying binarization by forcing pixel intensities to either 0 or 255 based on a predetermined threshold, we obtained the following results.

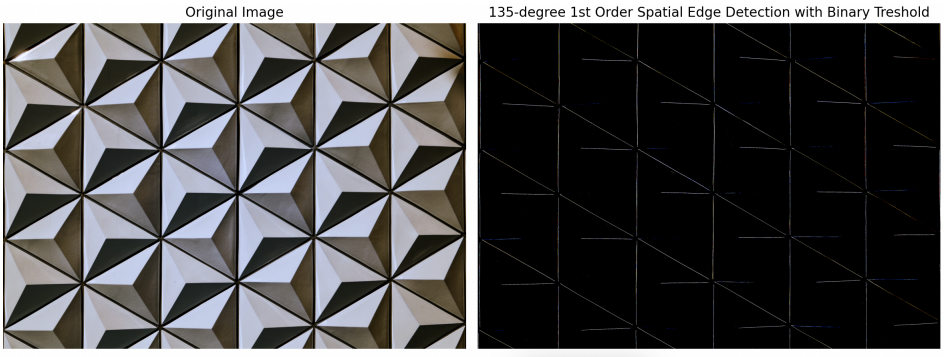

Figure 21: First order 135-deg spatial edge detection filter with binarization on threshold 100

Edge Detection and Impulse Noise

If an image contains impulse noise, edge detection is likely to be less effective. This is because edge pixels are typically defined as pixels that exhibit sharp changes in intensity [3]. Hence, when an image is heavily corrupted with impulse/salt-and-pepper noise - which introduces many pure white (255) and pure black (0) pixels - the strong intensity differences that correspond to edges become disrupted. This phenomenon is evident in the following images.

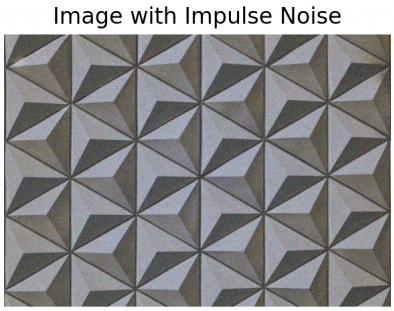

Figure 22: Image with impulse noise added

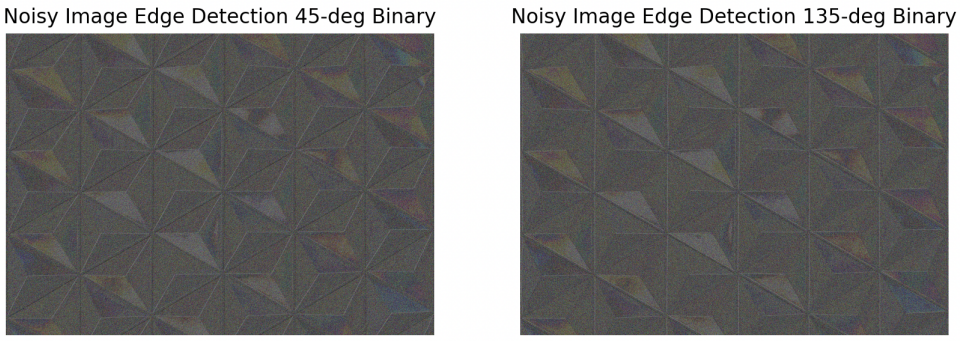

Figure 23: Result of applying 45-deg and 135-deg edge detection kernels to image with added impulse noise, with binarization applied afterwards

According to [3], the non-linear median spatial filter is a well-known method for effectively removing impulse noise. In this study, the built-in OpenCV median blur function with a kernel size of 5 was used to achieve this objective. Fortunately, the edge detection algorithm works well even after the application of the filter.

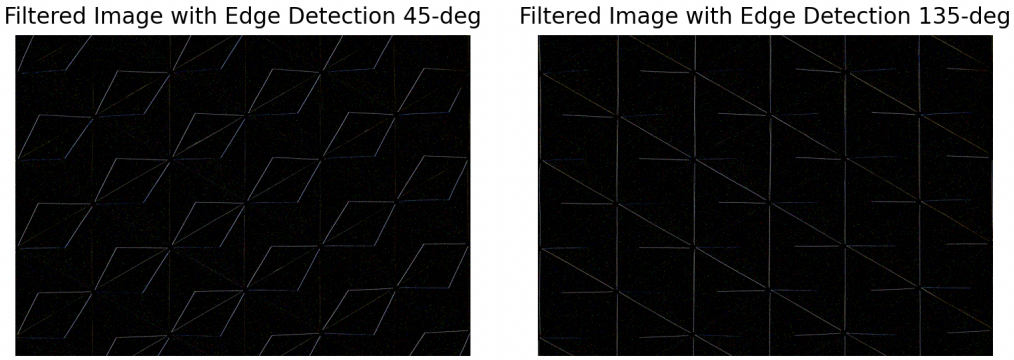

Figure 24: Noisy and denoised images using openCV median blur (kernel size 5) with the edge detection kernels applied

Edge Detection and Gaussian Noise

We will investigate the performance of the edge detection kernels in presence of Gaussian noise. Does adding Gaussian noise allow the edge detection to still perform well? More importantly, if we manage to remove the Gaussian noise, how does the edge detection perform afterwards?

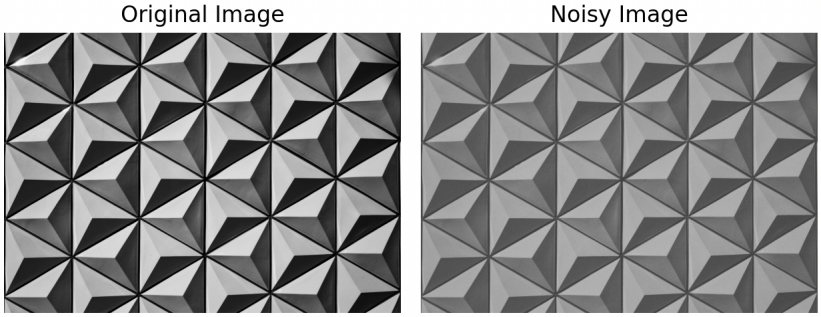

Figure 25: Original image and image with Gaussian noise added with mean 0 and standard deviation 0.1

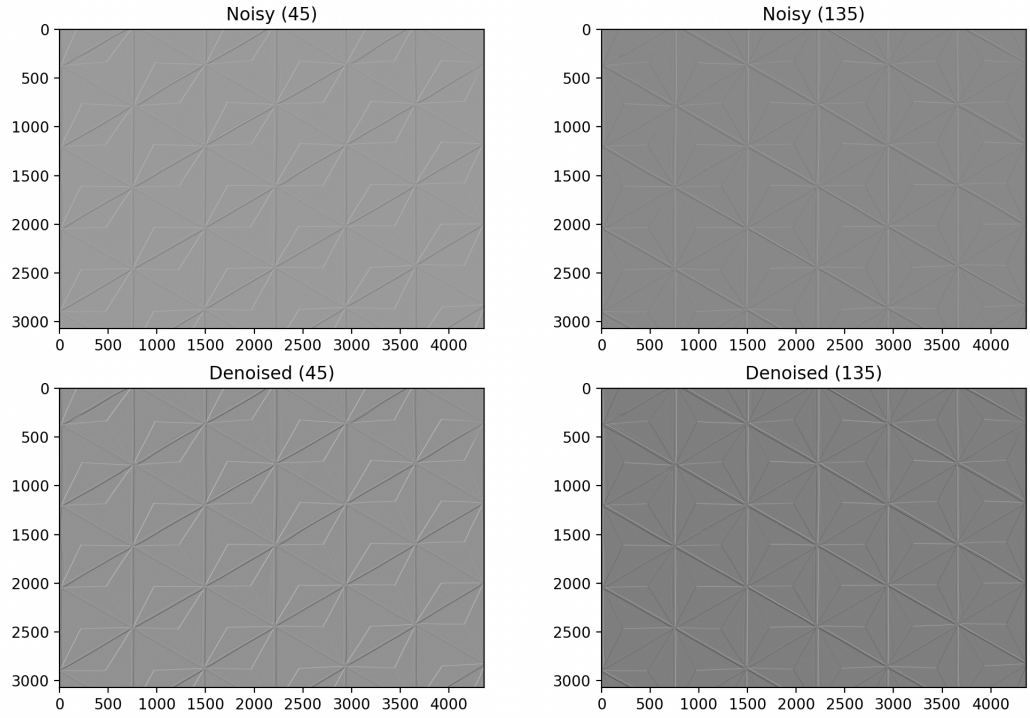

Figure 26: Noisy and denoised images using openCV Gaussian blur (kernel size 5) with the edge detection kernels applied

Clearly, the edge detection for both diagonals, performs worse in the presence of Gaussian noise. This makes sense, since edges are large difference in intensities [3], and adding Gaussian noise disrupts that.

After, Gaussian blur was applied, the edge detection had a clear improvement. The reason why this experiment is valuable is that Gaussian filtering is widely recognized as an effective pre-processing technique for edge detection [2]. However, if it is applied too excessively, important details may be lost. When applied appropriately, however, only the most significant edges are detected.

Fourier Transformation

2D Sinusoid using Meshgrid

In order to add periodic noise in a controlled manner to an image, we build a 2d mesh-grid sinusoid. The formula used for this is:

$$f(x, y) = \sin\left(\frac{2\pi x}{\lambda}\right)$$

. The wavelength was set to 0.5 which corresponds to a frequency of 2 and our amplitude was set to 10. Later, the amplitude was changed.

We want the sinusoid to be either vertical or horizontal. By using either X or Y in the equation for the mesh-grid sinusoid, we can control the orientation of the wave.

Our meshgrid sinusoid 2D and 3D projections look as follows.

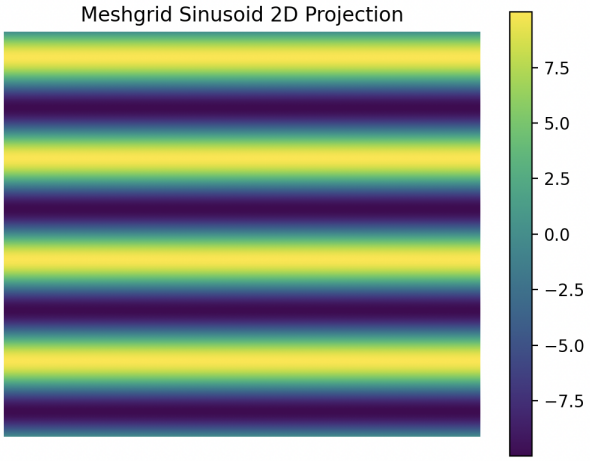

Figure 27: 2D Projection of horizontal meshgrid sine with frequency 2 and amplitude 10

Here brighter areas correspond to higher intensities/magnitudes.

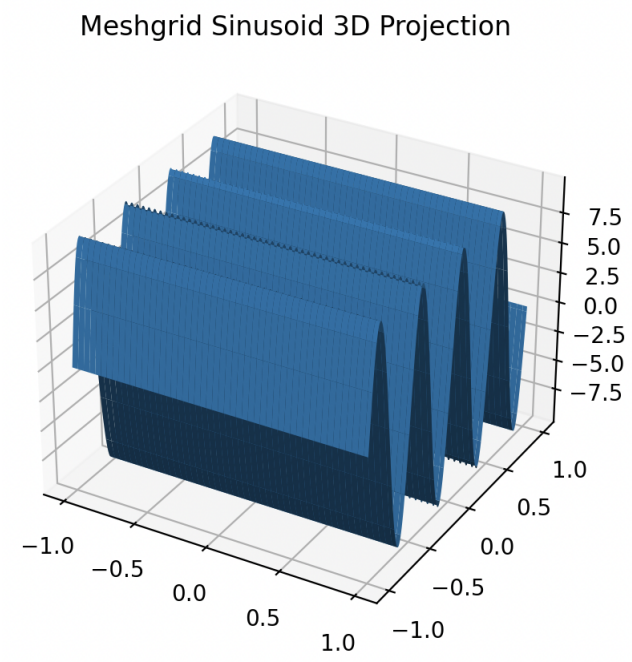

Figure 28: Horizontal 3D sine with frequency 2 and amplitude 10

2D Sinusoid FFT

The next step in our journey to understanding periodic noise and it's removal, is to further investigating the frequency domain content of our 2D sinusoid.

The direction of our sinusoid, will determine, whether the symmetric pair of peaks will appear in the 'middle row' or 'middle column' plot. The results show the following.

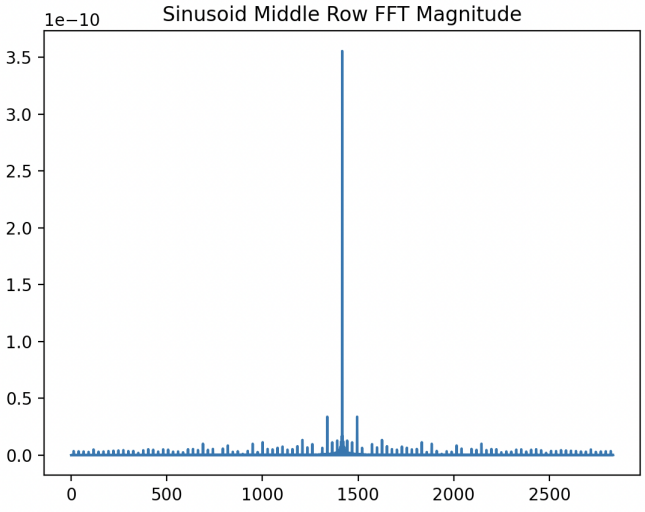

Figure 29: 1D Slice 'middle row' of meshgrid sine with frequency 2 and amplitude 10

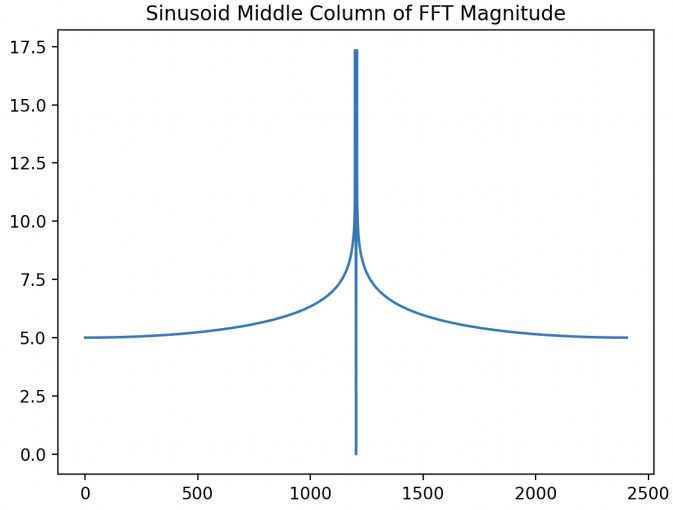

Figure 30: 1D Slice 'middle column' of meshgrid sine with frequency 2 and amplitude 10

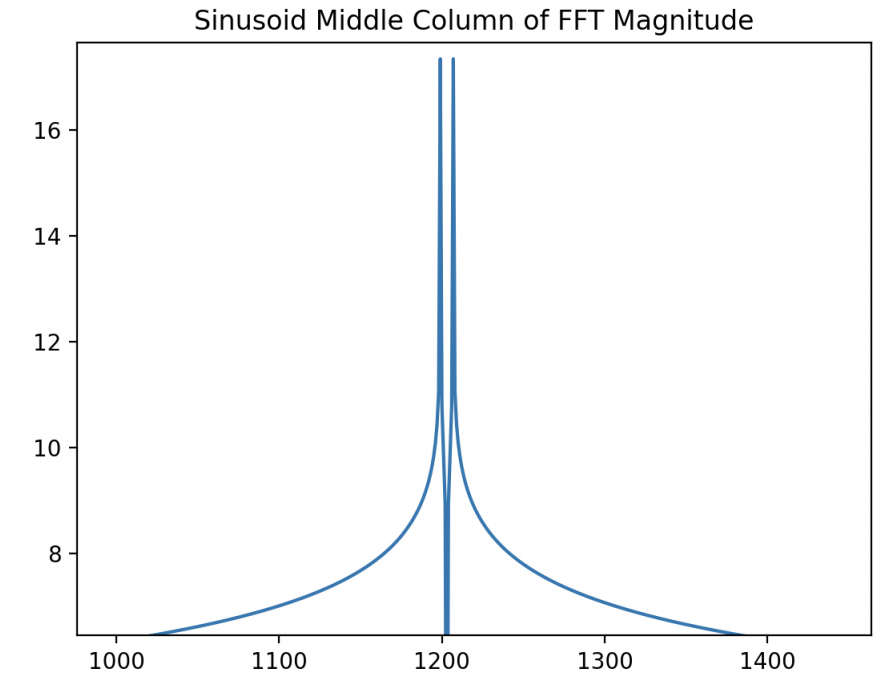

Figure 31: Zoomed 1D Slice 'middle column' of meshgrid sine with frequency 2 and amplitude 10 with better visible symmetric peaks

Since the sine has only one direction, we indeed observe the symmetric peaks only in the middle column slice.

A zoomed 2D magnitude spectrum was also plotted, which has practical engineering value that will be explained later.

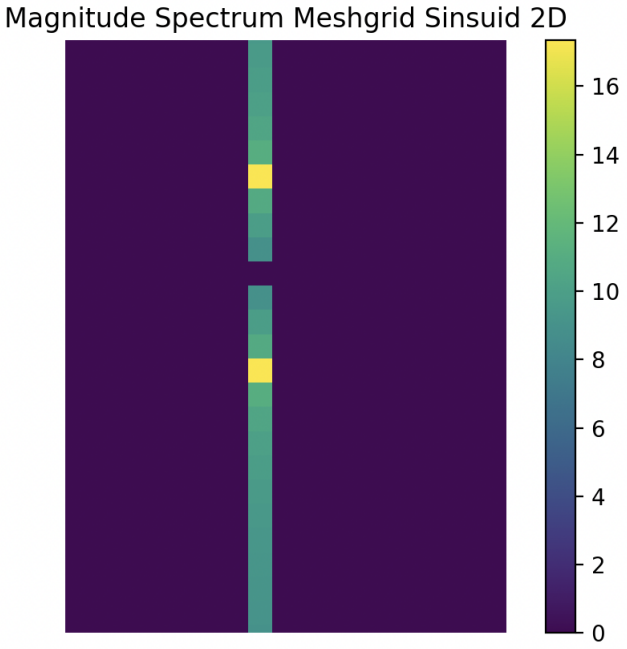

Figure 32: Zoomed 2D FFT of meshgrid sine with frequency 2 and amplitude 10

Looking at where the dots appear could be a technique, to guide us in setting notch center frequencies [3] and by understanding this spectrum, we could try pattern match when being given a spectrum of a image corrupted with periodic noise.

Adding Periodic Noise to Images

In order to make things more interesting and of practical value, we finally start adding our periodic noise to an image, that does not contain strong horizontal or vertical (spatial) elements.

Assume, there was some hypothetical 2 Hz sine 'hum' corrupting our measuring device. What will it look like when we obtain our images? For demonstrative purposes, we picked the amplitude of our meshgrid sinusoid sufficiently large. This helped to make the periodic noise clearly visible.

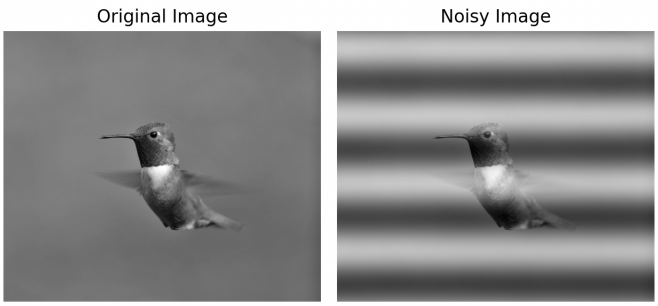

Figure 33: Original image and noisy image with sine wave added having frequency 2 and amplitude 10

We investigate the 2D FFT and 1D FFT slices of the corrupted image. Since the image doesn't contain any strong line components, we should not observe any strong lines, orthogonal in our 2D FFT, therefore, making our investigation easier.

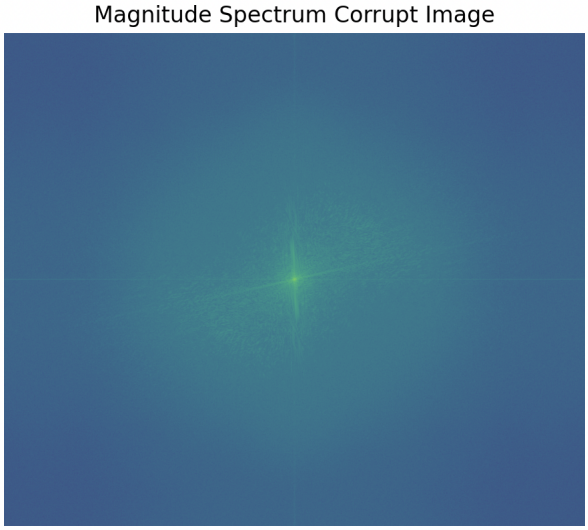

Figure 34: 2D FFT of noisy image with sine wave noise with frequency 2 and amplitude 10 added

The 2D spectrum in this case does not reveal very much, zooming in could help us reveal more information. It is not easy to spot any obvious dots, corresponding to periodic noise.

The 1D column slice, clearly shows the symmetric peaks equivalent to these observed of the sinusoid isolated, of the periodic noise when zooming in.

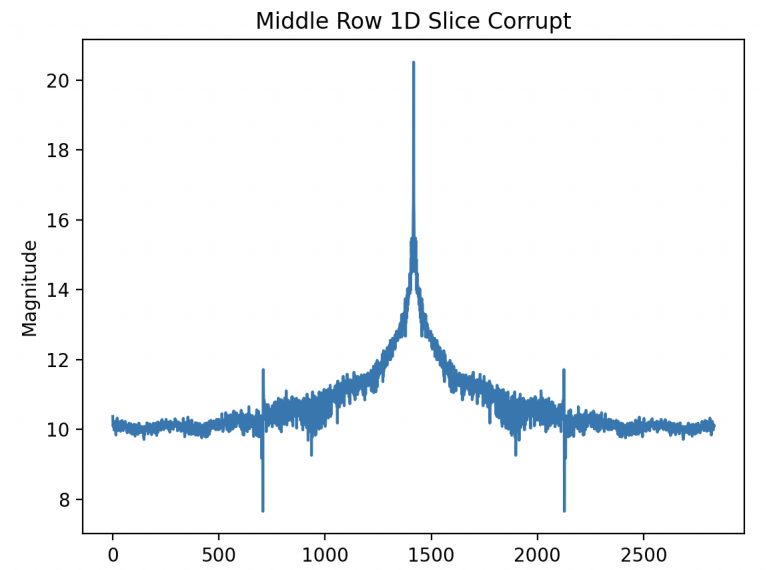

Figure 35: 1D Slice 'middle row' of noisy image with noise characteristics being frequency 2 and amplitude 100

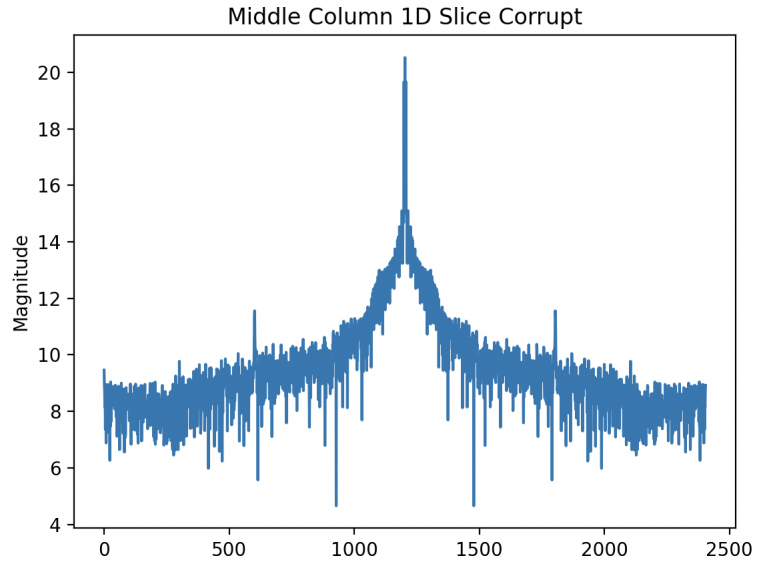

Figure 36: 1D Slice 'middle column' of noisy image with noise characteristics being frequency 2 and amplitude 100

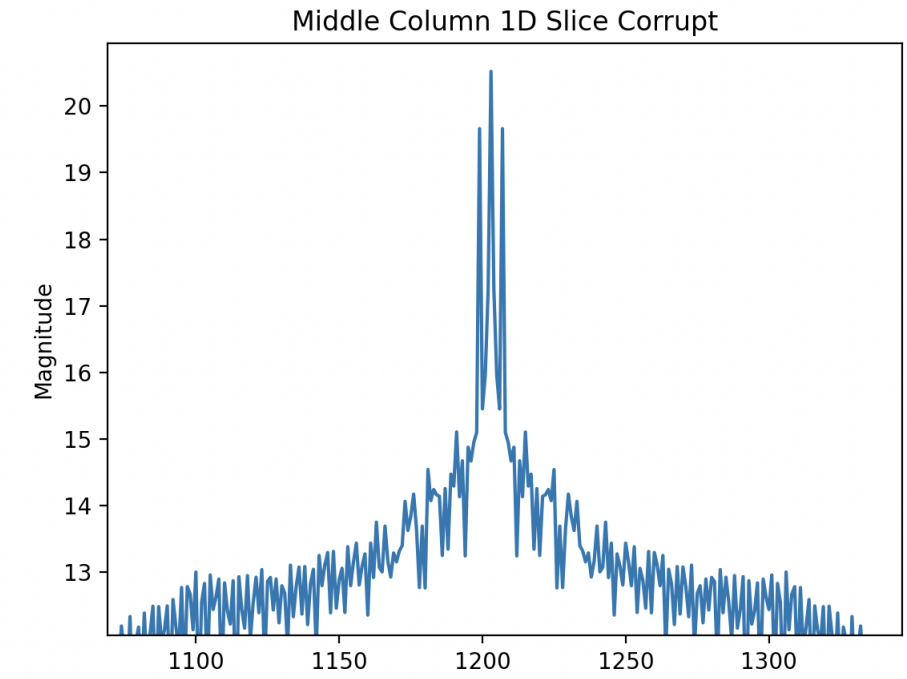

Figure 37: Zoomed 1D Slice 'middle row' of noisy image with noise characteristics being frequency 2 and amplitude 100, with symmetric peaks more clearly visible

The question begs, can we filter this noise out?

Filtering Periodic Noise

The next challenge, requires us to remove the periodic noise added to the image. We will do this in the frequency domain. Butterworth notch reject filters, seemed like an appropriate choice, since they offer a lot of flexibility [3].

Firstly, we look into the filter properties by investigating the 1D and 2D FT magnitude plots.

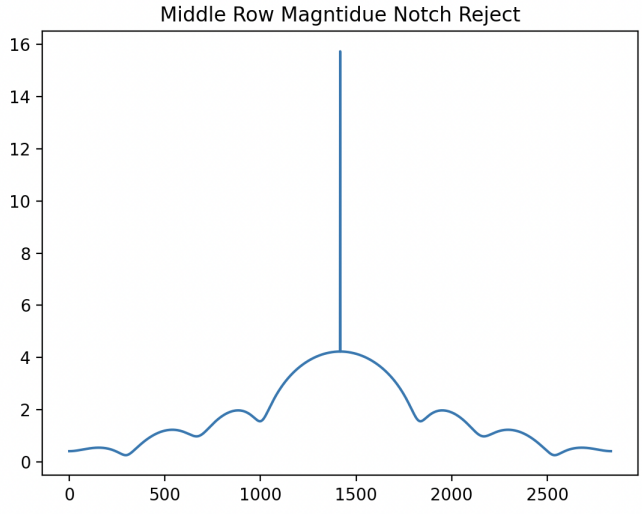

Figure 38: 1D FT slice 'middle row' of Butterworth notch reject filter with radius 3 and order 8

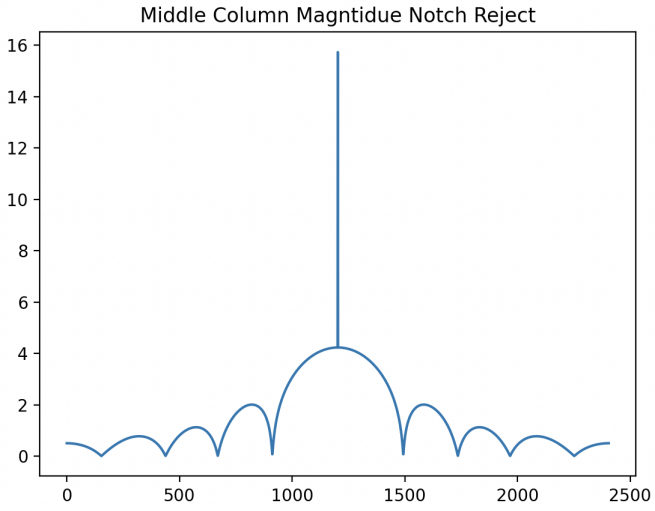

Figure 39: 1D FT slice 'middle column' of Butterworth notch reject filter with radius 3 and order 8

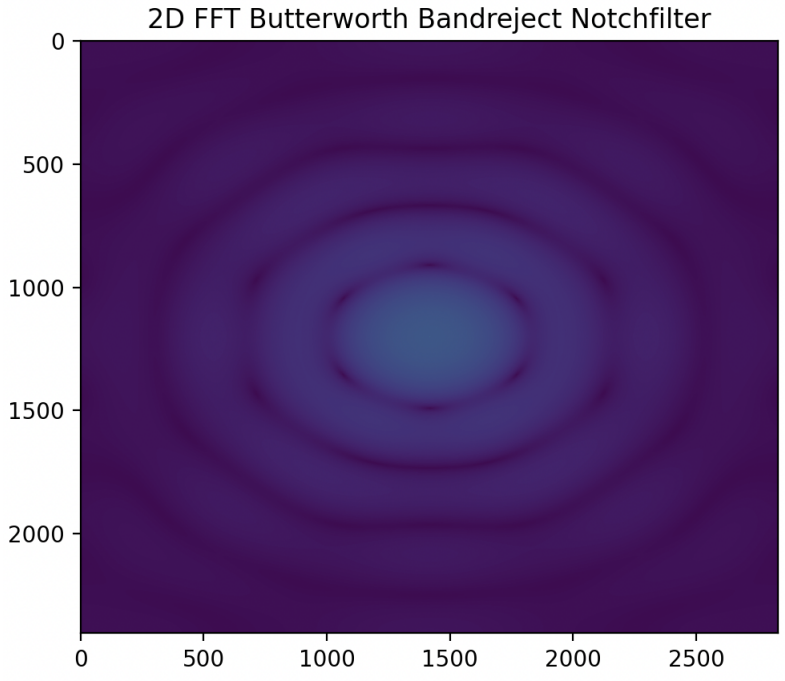

Figure 40: 2D FT magnitude of Butterworth notch reject filter with radius 3 and order 8

In order to really observe what the filter does in the frequency spectrum, it is very use full to plot the filtered image 1D slices. It can be observed, frequencies, that correspond to the filter 1D plots (and 2D), get notched out. Which is of course, the purpose of the notch filter!

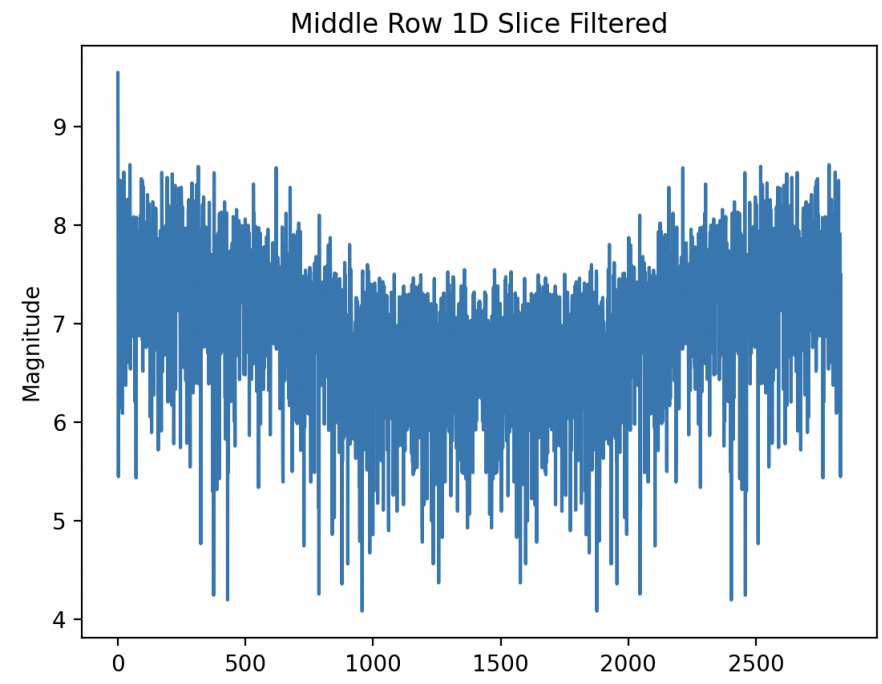

Figure 41: 1D FT slice 'middle row' of denoised image by applying Butterworth notch reject filter with radius 3 and order 8

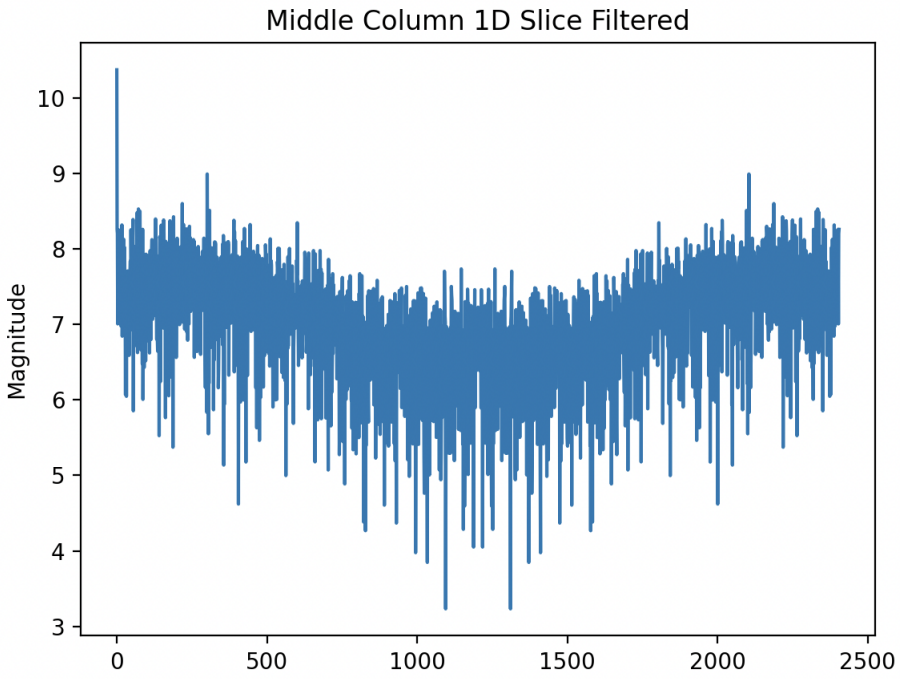

Figure 42: 1D FT slice 'middle column' of denoised image by applying Butterworth notch reject filter with radius 3 and order 8

We can observe in the plots before, that the notch filter decreases the magnitude of certain frequencies. Which obviously corresponds, to filtering in frequency domain, as we desired to do.

Since the notch filter works in both directions (2D) the filtering can be observed in both 1D slices, as said before.

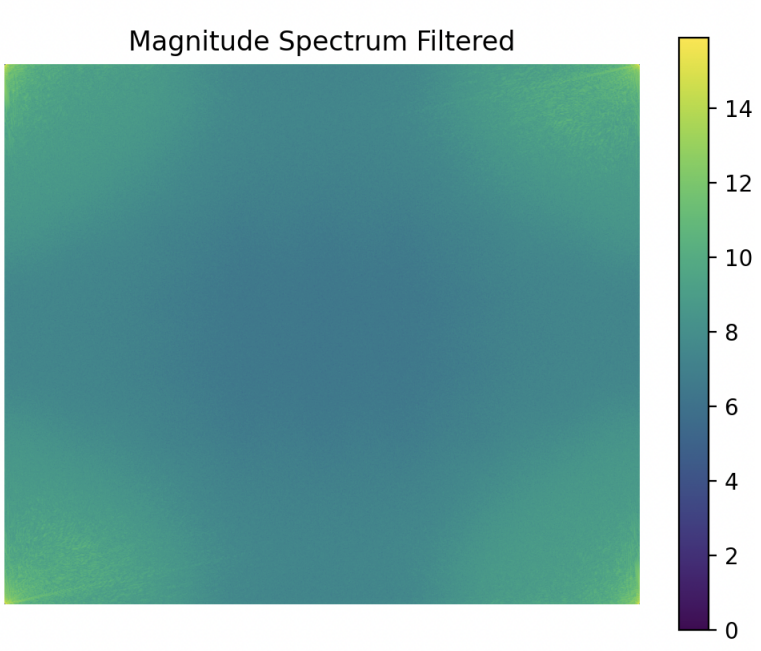

Figure 43: 2D FFT of noise image after applying the Butterworth notch reject filter with radius 3 and order 6

After applying the filter through multiplication in the frequency domain, inverse fast Fourier was applied, which give us the filtered image. This showcases the performance of our notch filter.

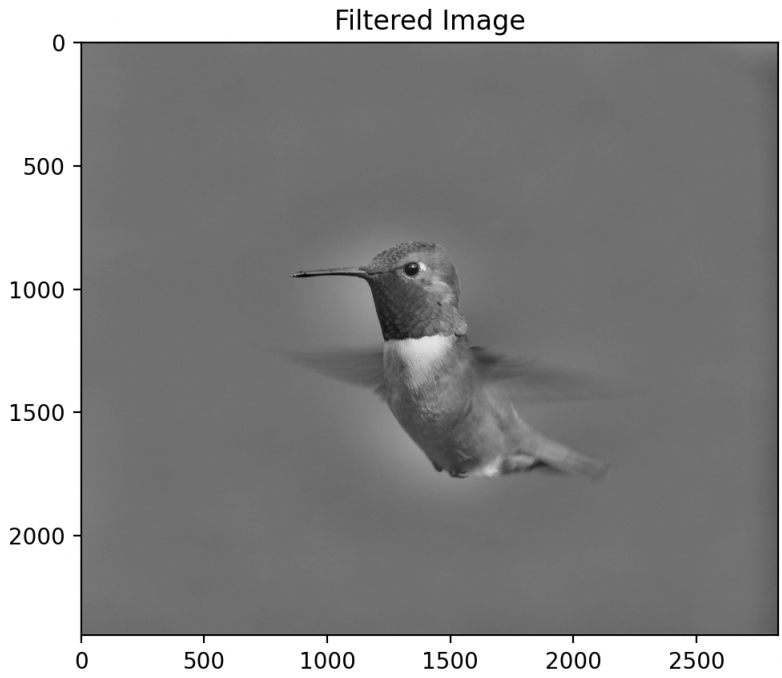

Figure 44: Filtered image

For completeness and in order to build more intuition, two additional plots show the Butterworth notch reject filters in 2D and 3D, with respectively radius 3 and 30. The order for both is 8. It shows most of the frequencies except the notch center frequencies are unaffected, which is exactly our intention when filtering. In the section 'Finding notch filter parameters', we described, how we found the optimal notch filter parameters to remove the periodic noise.

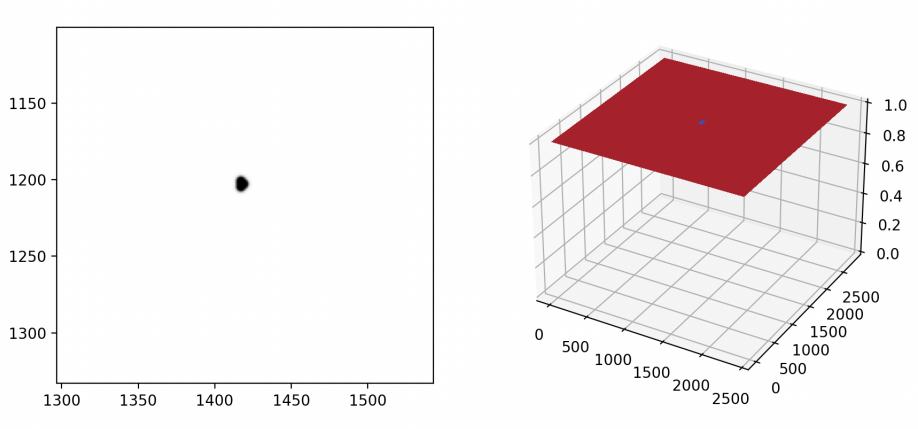

Figure 45: Butterworth notch reject filter with radius 3 and order 8

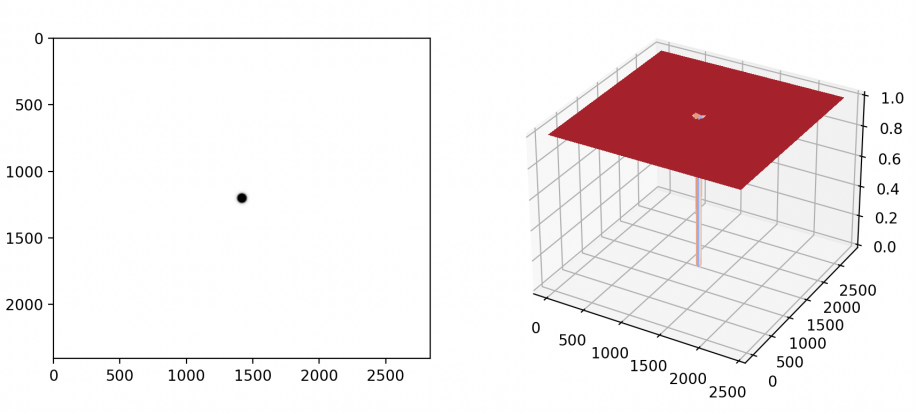

Figure 46: Butterworth notch reject filter with radius 30 and order 8

Increasing Periodic Noise Amplitude

In real-world applications, the amplitude of our source of noise could vary. Maybe, on some days, the satellite data get interfered more than on others. How will this affect the performance of our notch reject filter?

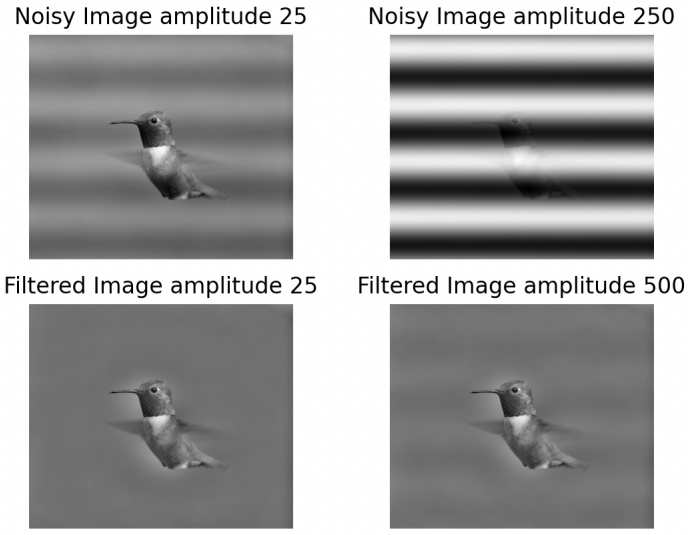

Figure 47: Varying the amplitude of the periodic noise with a factor 8 while filtering using Butterworth notch reject filter with radius 3 and order 8

As seen on the image, when keeping our notch reject filter consistent, the periodic noise is less suppressed. When the amplitude of the periodic noise increases, the magnitude of the noise in the frequency domain also increases. There is more magnitude/ energy for the filter to work on, which results in more periodic noise left in the filtered image.

The results in the 1D slices could be expected to have a dampening, though not filtering everything out. Since this seemed in line with the procedure before, the additional plots are left out.

Finding Notch Filter Parameters

In order to find the optimal radius and order for our Butterworth notch reject filter, different plots were generated to see the filters in action. Even number orders were investigated and different radius of increasing size. This gave us the following plots.

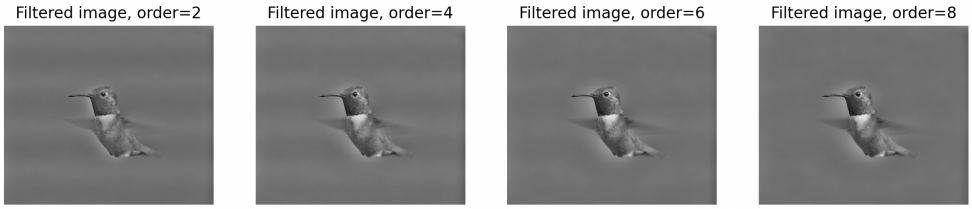

Figure 48: Varying the order of the notch filter, while keeping the radius at 3

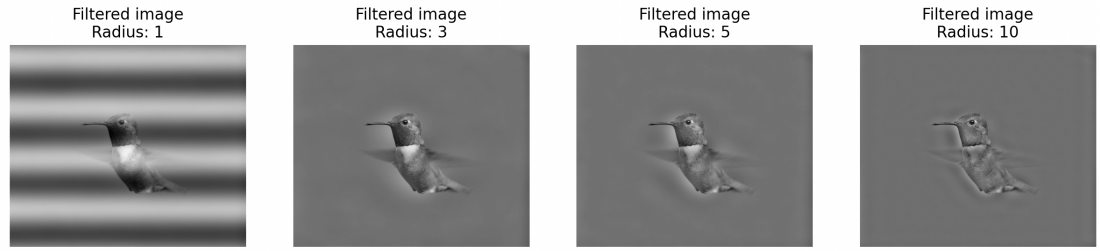

Figure 49: Varying the radius of the notch filter, while keeping the order at 8

After analyzing the plots, the preferred values for the parameters were determined to be $radius = 3$ and $order = 8$. A lower order was observed to result in a smoother output, similar to audio filters with less steep slopes resulting in smoother sounds. Conversely, a higher radius affects a broader range of frequencies, thereby increasing the likelihood of filtering out important information along with the noise. This motivated the choice of the selected parameter values.

Special Effects

Cartoon Images

Without special effects image processing can be mundane. However, we can add some creative flair to an image by making it look more like a cartoon. Here's one way to do it:

- Apply edge detection using a technique such as Sobel
- Invert the edges/take the negative to make them black and fat
- Apply k-means clustering with k=3 to create a painterly lo-fi color palette

- Finally, clip the saturation channel in the HSV color space to intensify the colors

These steps are all inspired by basic concepts, discussed in the book by Gonzalez. [3]

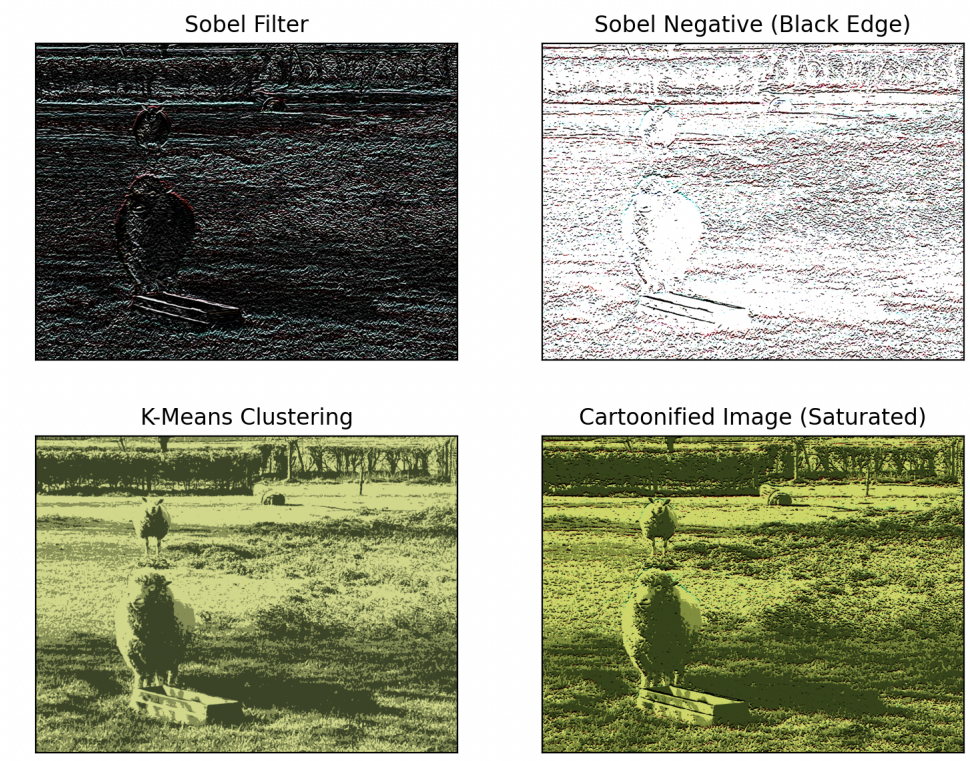

Figure 50: Final cartoon image and intermediate images (2,3,4) used to build the image

Geometric Transformations

The process of geometric transformations can be divided into two stages: spatial transformation of coordinates and geometric intensity interpolation, which assigns values to spatially transformed picture elements [3]. In this particular case, we used the following coordinate transformations:

- $\phi' = \phi$
- $\rho' = \sqrt{\rho}$

To perform the transformation, we used an inverse mapping technique with nearest neighbor interpolation. This approach was chosen due to its simplicity and computational efficiency, as compared to other methods like bicubic interpolation [3]. The resulting image after transformation can be seen in the figure below, which shows the original image and the transformed image with the aforementioned coordinate equations and nearest neighbor interpolation.

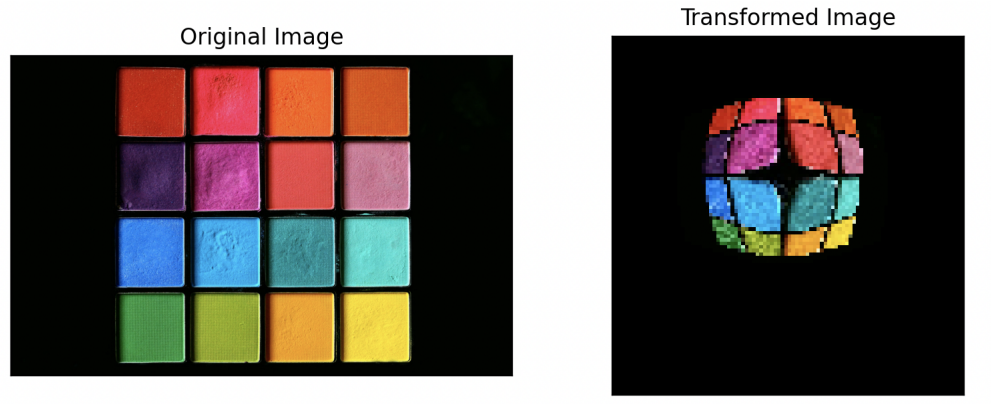

Figure 51: Original image and zoomed transformed image with coordinate equations $\phi' = \phi$ and $\rho' = \sqrt{\rho}$ using nearest neighbors.

References

[1] Trong-An Bui et al. "Deep Learning for Landslide Recognition in Satellite Architecture". In: *IEEE Access* 8 (2020), pp. 143665–143678. DOI: 10.1109/ACCESS.2020.3014305.

[2] John Canny. "A Computational Approach to Edge Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

[3] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 2008.

[4] Jacob Rus. *HSL and HSV*. https://en.wikipedia.org/wiki/HSL_and_HSV. Accessed May 8, 2023. 2022.