|  |
| --- |
| ­ |
| ECSE 490-Experiment 3 |
| Image Processing |
|  |
| **Isabel Helleur (260350713) and Loren Lugosch (260404057)** |
| **3/13/2013** |

|  |
| --- |
|  |

# Background

This lab introduced us to 2-dimensional signal processing through its most prevalent application: image processing. Humans rely more heavily on vision than on any other way of obtaining information about the world, so it makes sense that image processing is a well-developed field in signal processing and that the theory handling 2-dimensional signals should have sprung up around it. For those reasons, MATLAB is already well-equipped to handle images and 2-d filtering/convolution.

# 2D DTFT

2D DTFT of checkerboard.tiff

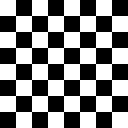


Figure 1 - Checkerboard

Using MATLAB’s fft2 function, we computed the 2D-DFT of the 2-dimensional signal representing the checkerboard image. Since the checkerboard is the 2-dimensional equivalent of the traditional 1-dimensional square wave, the frequency spectrum is a sinc function in both dimensions (as derived in the pre-lab). The image below is that frequency spectrum. Note the average value of 128 (scaled) at (0 Hz, 0 Hz), which we expect (since the checkerboard at any pixel is either 0 or 255, and both values occur equally often). Just as in sampled frequency spectra in one dimension, there is symmetry about the Nyquist frequency (here, 63 Hz).

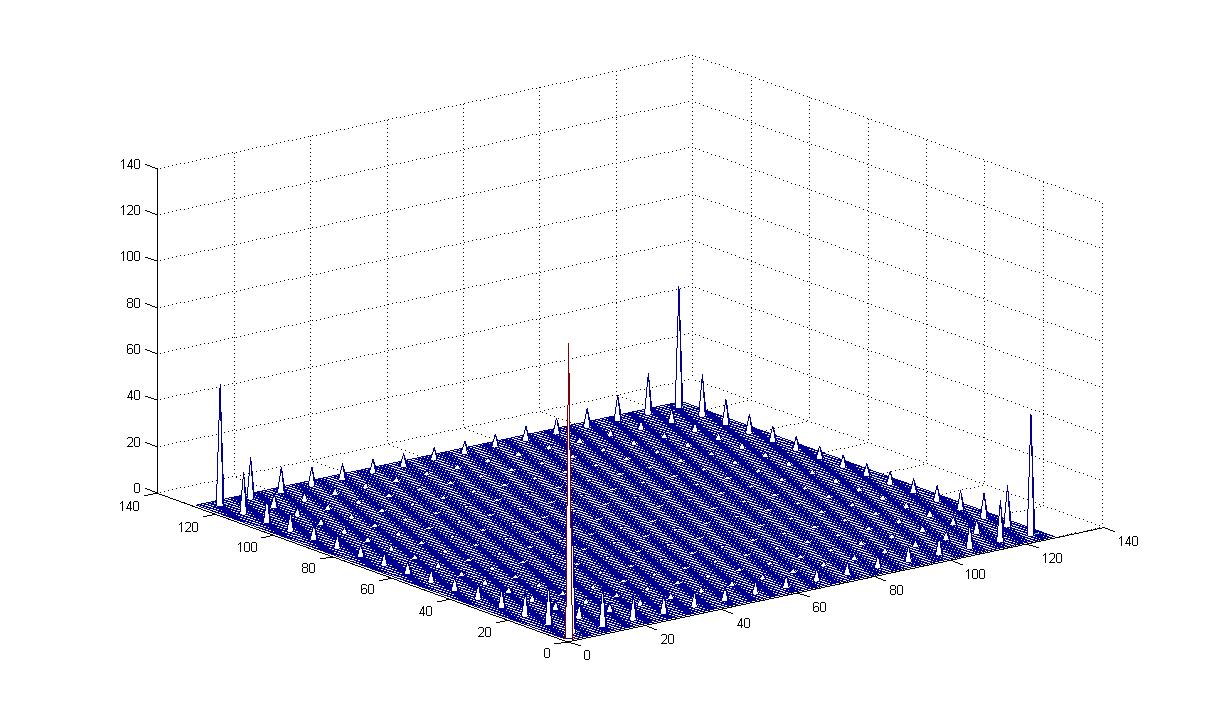
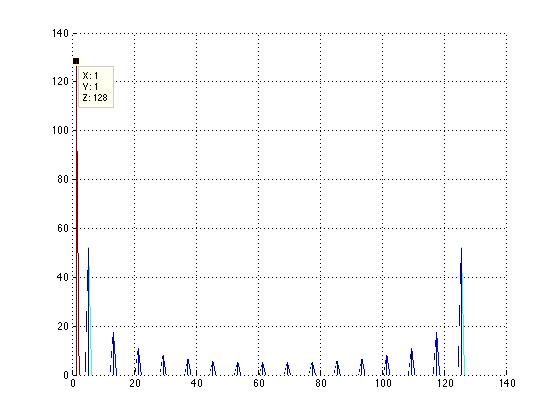


Figure 2 - 2D DFT of checkerboard.tiff, viewed from

Figure 3 - 2D DFT of checkerboard.tiff with average value 128 at (1,1)

If we use more points for this DFT, we see more resolution in the frequency spectrum (although the frequencies have to be normalized):

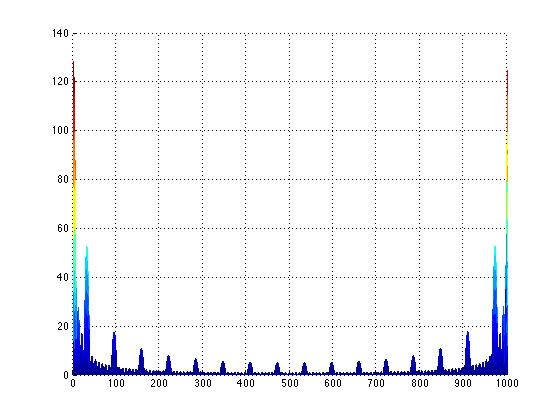


Figure 4 1000-point DFT of checkerboard

## 2D DTFT of Claire.tiff



The next image we processed was a woman in a blazer. Again, we took the 2-D DFT:

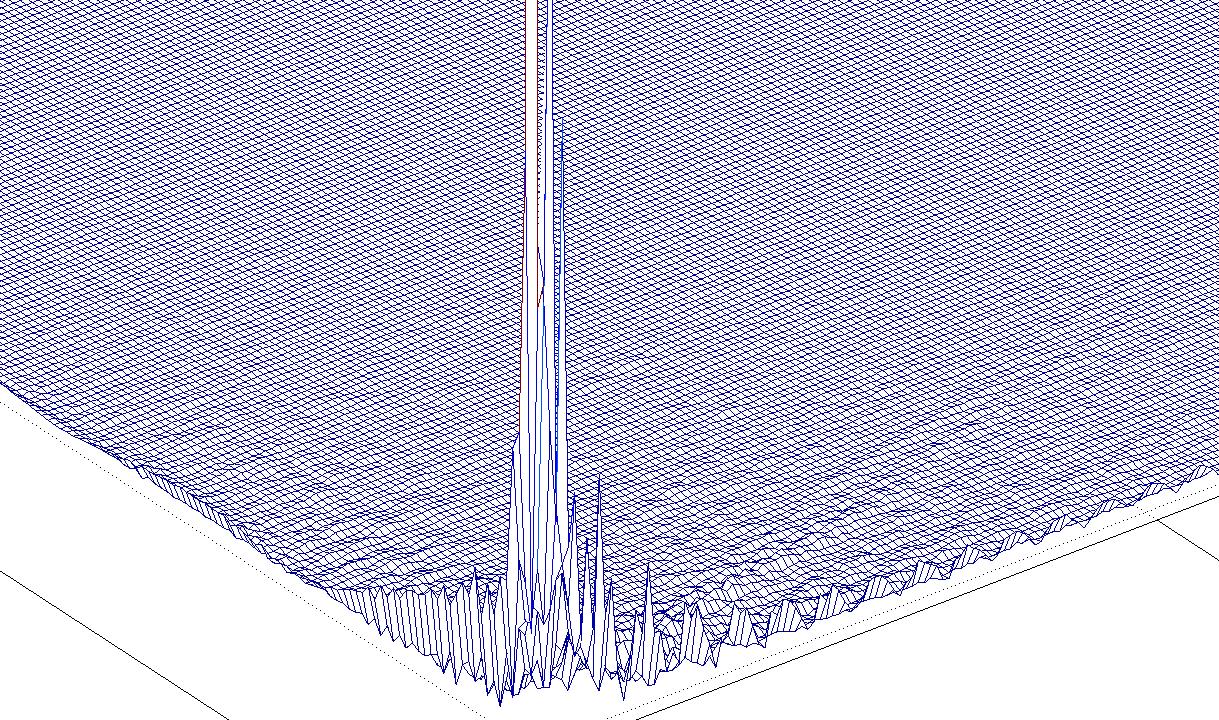


Figure 5 – 2-d DFT of Claire.tiff

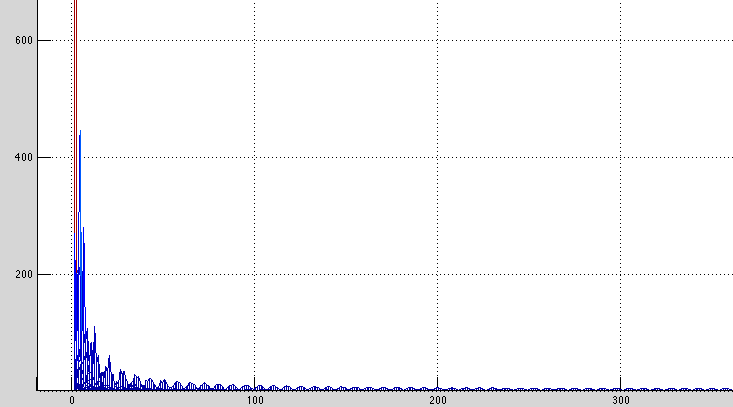
The 

Figure 6 - DFT of Claire, side view (in x)

Obviously this image lent itself to more interesting analysis; the sharp edges of clothing and background together with the softer changes in skin tone produced an interesting frequency spectrum when fft2’d. The unusual diagonal ripples are interesting patterns which the simple checkerboard DTFT does not include.

## STFT

We wrote a script to compute the Short-Time Fourier Transform of an image. The script divides an image into squares and takes the fft2 of each square. Since each square is itself a small image/signal with all the properties of a normal 2-dimensional signal, it can be Fourier transformed, and that spectrum can be used to reconstruct the small image. Each small image can then be concatenated to form the larger image. We can display the STFT in a similar way as the concatenation of each of the small spectra using the mesh function.

For simplicity we first examine the STFT of the checkerboard. The checkerboard is 128 pixels by 128 pixels and 8 squares by 8 squares, meaning that every 32 pixels is one cycle of a square wave. When we take the STFT using 32 by 32 squares, each square’s spectrum will be an identical sinc function in two dimensions:

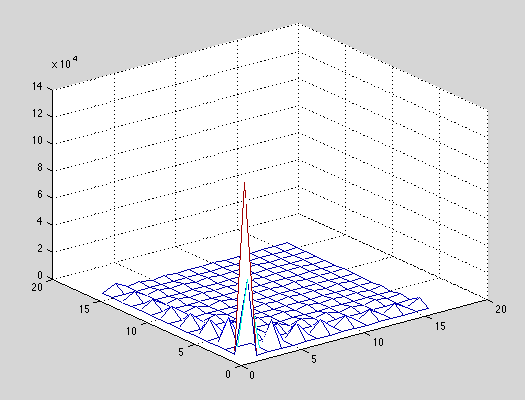


Figure 7 - One section of the checkerboard's STFT (symmetrical part not shown)

When we concatenate each spectrum, we get 16 of these squares as the complete STFT:

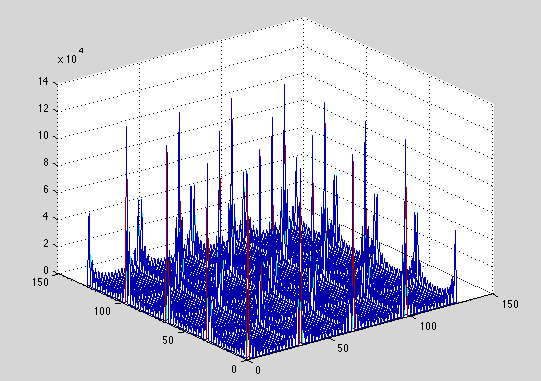


Figure 8 - Complete STFT of checkerboard

With that in mind, we can make more sense of Claire.tiff’s STFT in 16x16, 8x8, and 4x4 sections. Each peak represents the “DC offset” – the average value of the pixels in that square.

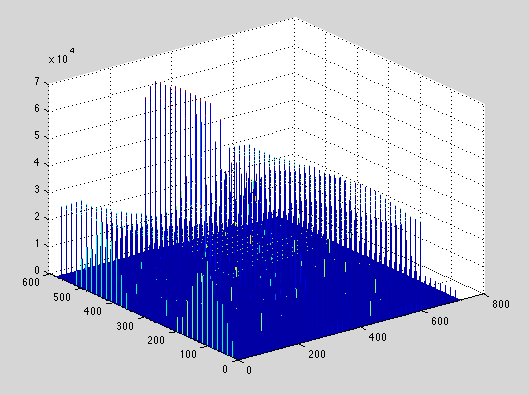


Figure 9 - 16 by 16 STFT

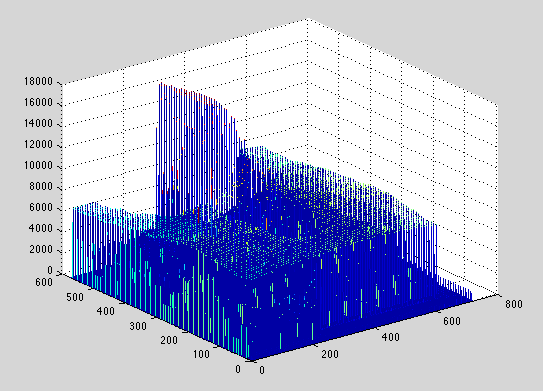


Figure 10 - 8 by 8 STFT

## :clairestft4.png

Figure 11 - 4 by 4 STFT

## Random Noise

We created an image of white Gaussian noise:

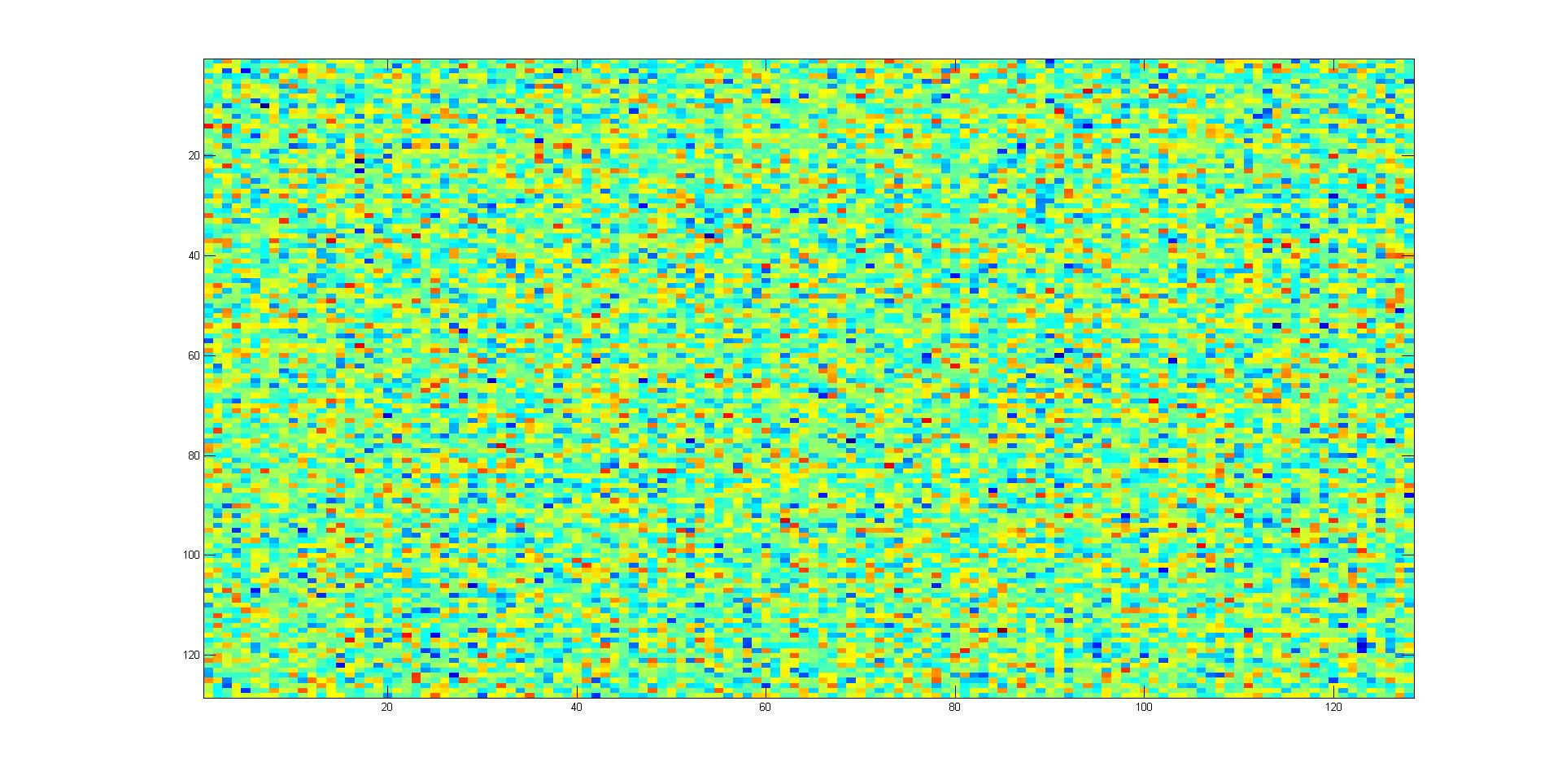


Figure 12 - White Gaussian Noise Process (default colormap)

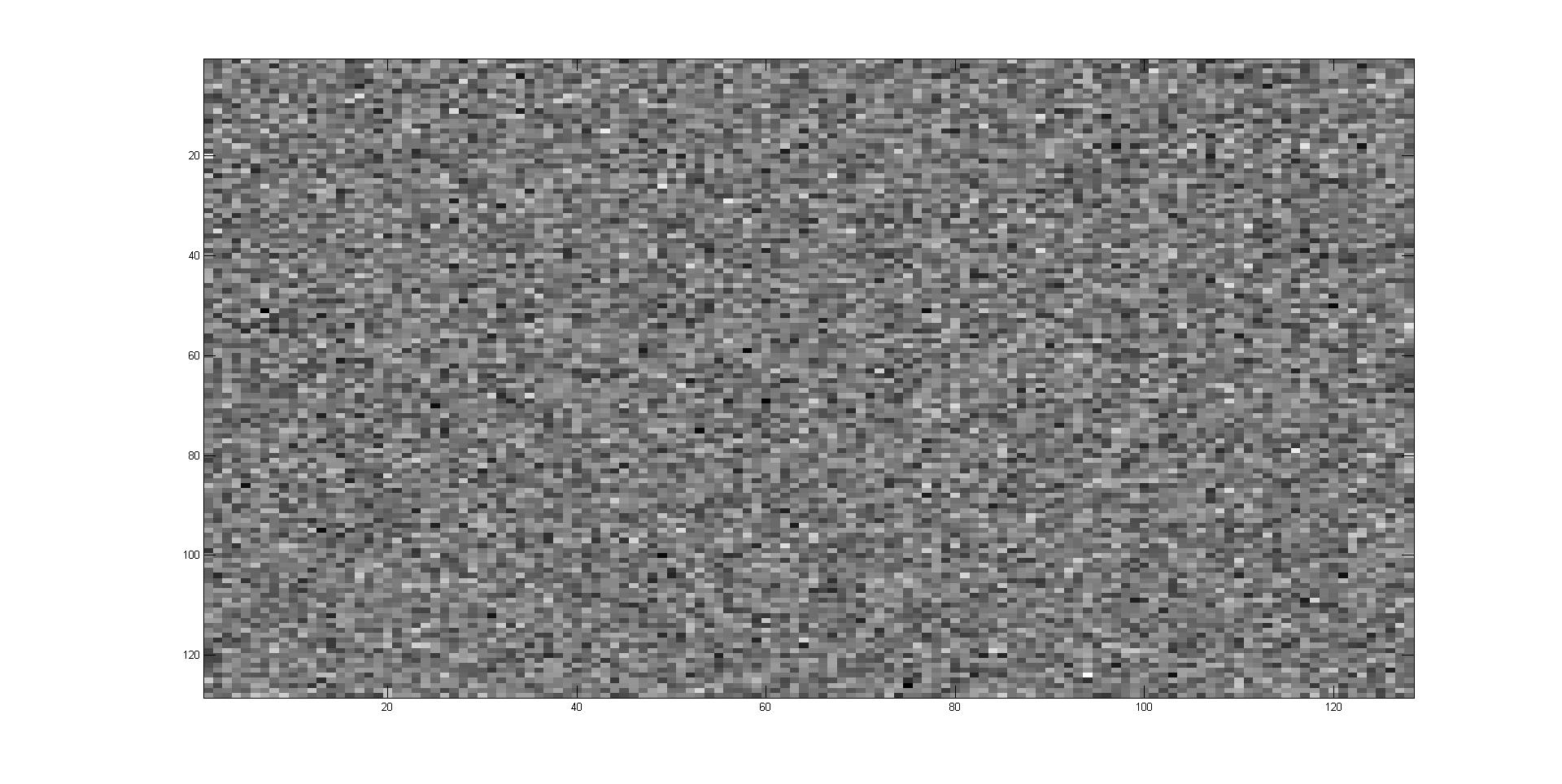


Figure 13 - White Gaussian Noise Process in Grayscale colormap

Next we tampered with Claire.tiff’s frequency spectrum by randomizing the phase and then the magnitude.

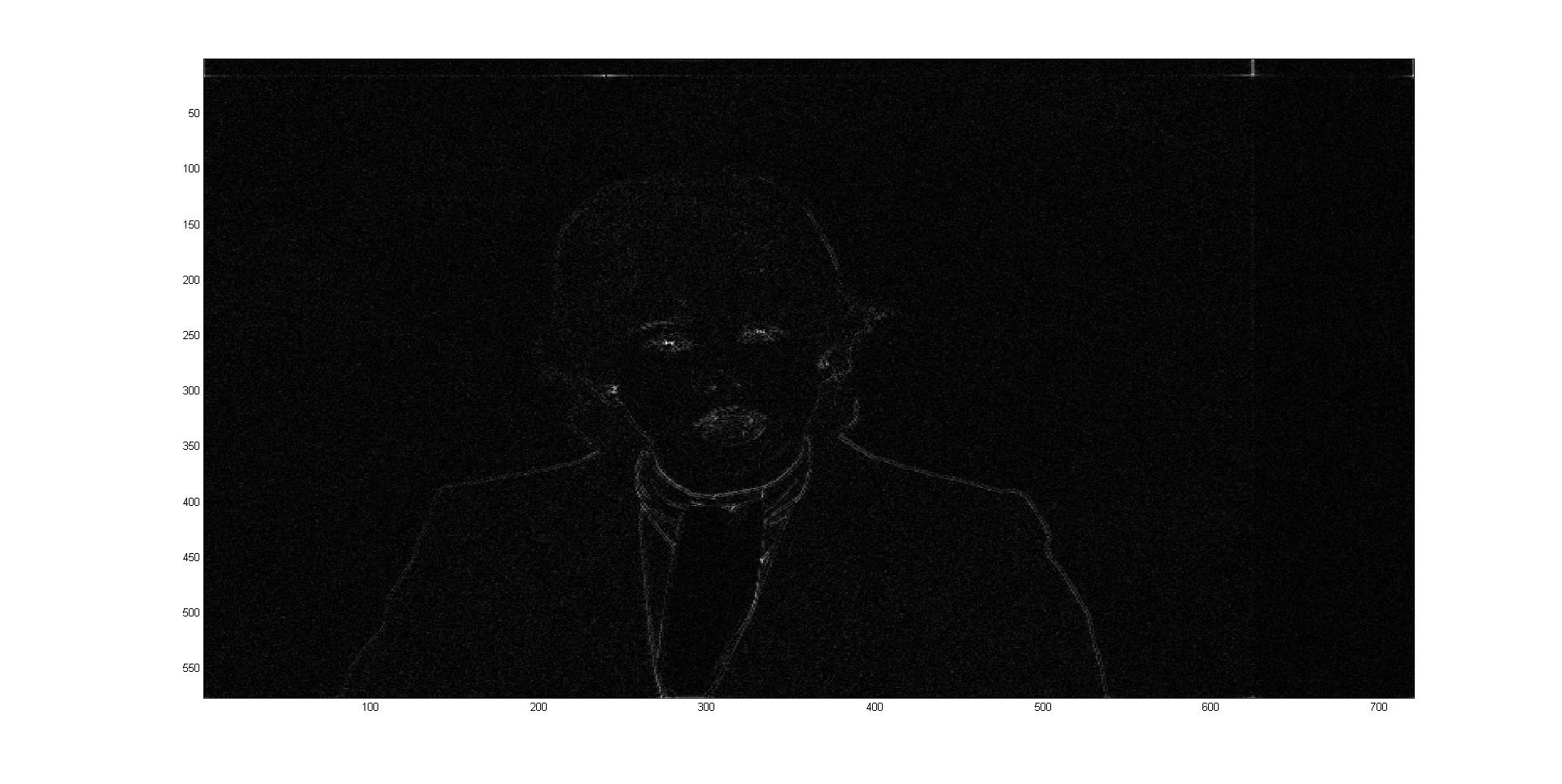


Figure 14 - Random Magnitude

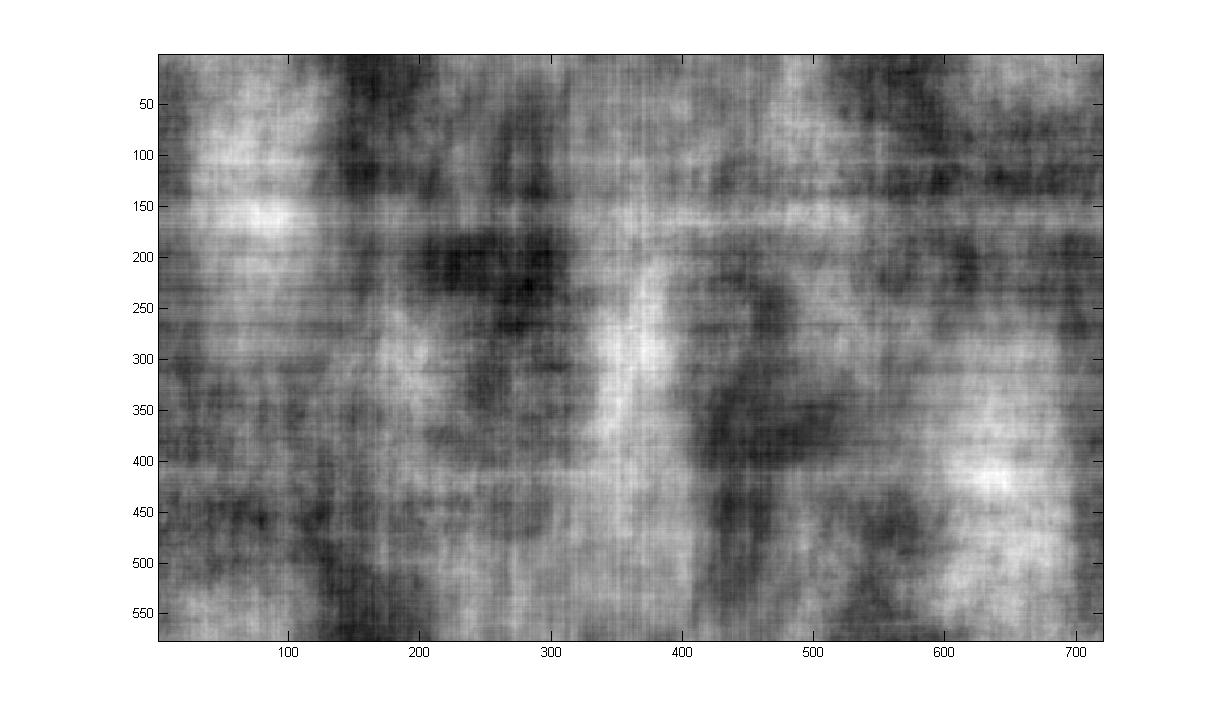
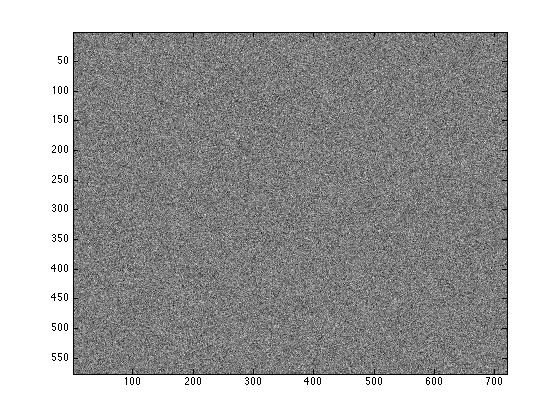
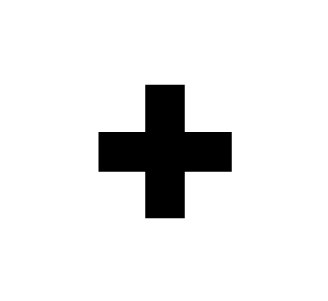


Figure 15 - Random phase

The image whose frequencies have random magnitude still contains a visible person, but the image with randomized phase is gibberish. The reason for that contrast makes sense from a musical analogy: if you change the volume of each of the notes in a song (i.e. randomize the magnitude of the frequencies), the song may sound very strange, but the melody will still be recognizable. If you change the time at which each note occurs (i.e. randomize the phase of the frequencies), the song could be totally different, even though the volume of the notes is all the same.

# Noise Removal

We created a noisy version of the picture of Claire by creating an image of the same size of random noise and adding it to the original:



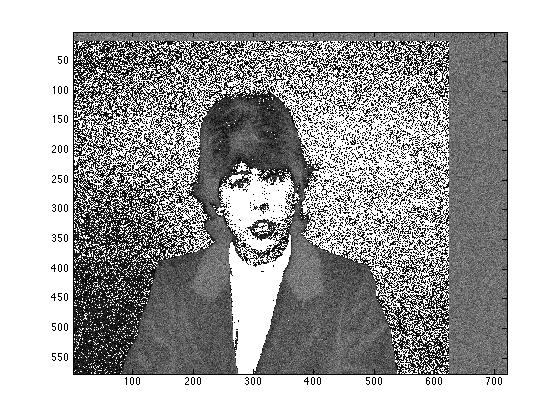
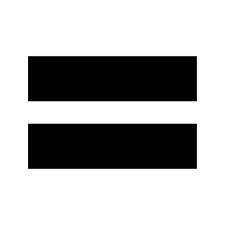


Figure 16 - Corrupted Image

## Noise Removal via STFT

We used our STFT from above on the corrupted image and on the noise. From these spectra we calculated the power spectra and divided them to get the SNR. To reconstruct the image, wherever the SNR was below a certain threshold, we set a new spectrum to zero; otherwise we set that new spectrum to the value of the signal’s spectrum. Then we IFFT’d the reconstructed spectrum to get the image. Varying the threshold gave varying amounts of noise in the picture.

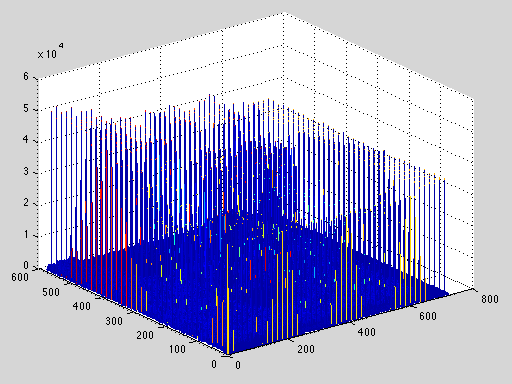


Figure 17 - STFT of corrupted image

## ::::Desktop:noisespectrum.png

Figure 18 - STFT of noise

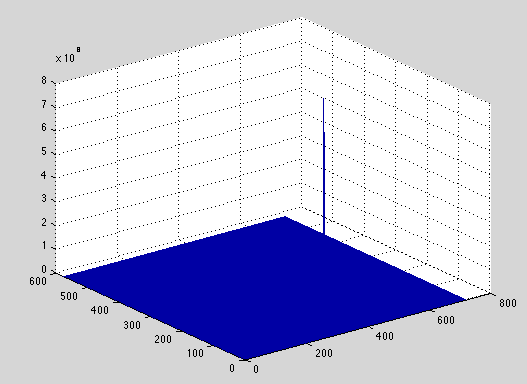


Figure 19 - SNR

## :SNRsideview.jpg

Figure 20 - SNR side view

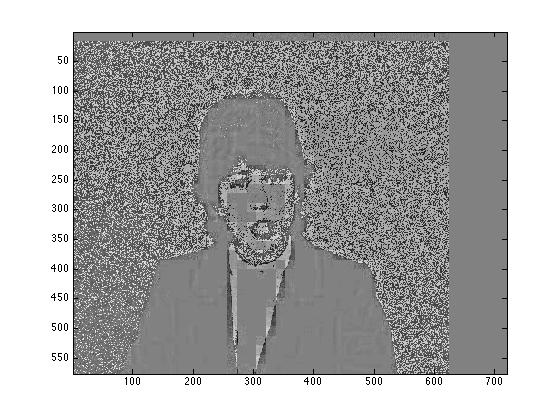


Figure 21 - Reconstructed image with threshold 5



Figure 22 - Reconstructed image with threshold 1

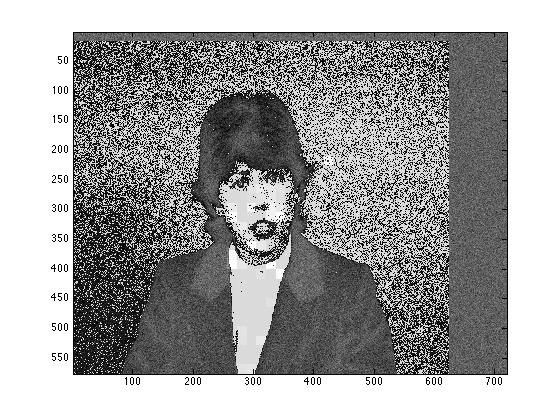


Figure 23 - Reconstructed image with threshold one-eighth

## Noise Removal via Low Pass Filter

We used a low-pass filter to remove noise from the image. Here first for comparison are versions of the original image low-pass-filtered with different cutoff frequencies:



Figure 24 – Lowpass, cutoff at 50 in x and y

Figure - Lowpass 100100



Figure 27 - Lowpass 500500

Here are the filtered versions of the corrupted image. Using 100 as the cutoff frequencies seems to recover some of the facial features, like the sides of the nostrils, which are invisible in the noisy image. Since the noise is at low frequencies as well as high frequencies, it is impossible to recover the original image perfectly using a low-pass filter.

Figure - Corrupt Image, filtered with cutoff at 50



Figure 29 - Filtered with cutoff at 100

# Edge Detection

The one-bit quantizer is a simple way of mapping to black and white. Observe when we use it on an unfiltered image of Claire:

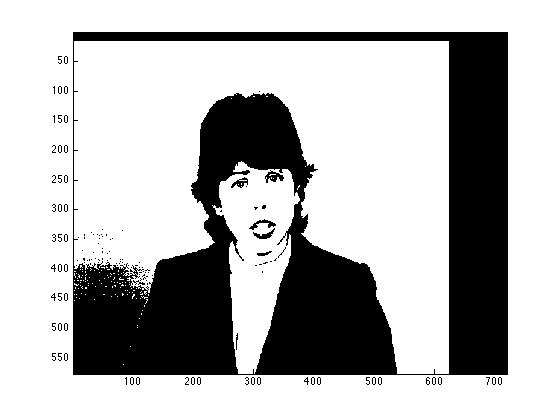


Figure 30 - One-bit quantizer

The Sobel filter is a differentiator. When we convolve it with the image we see where the image changes sharply more obviously.

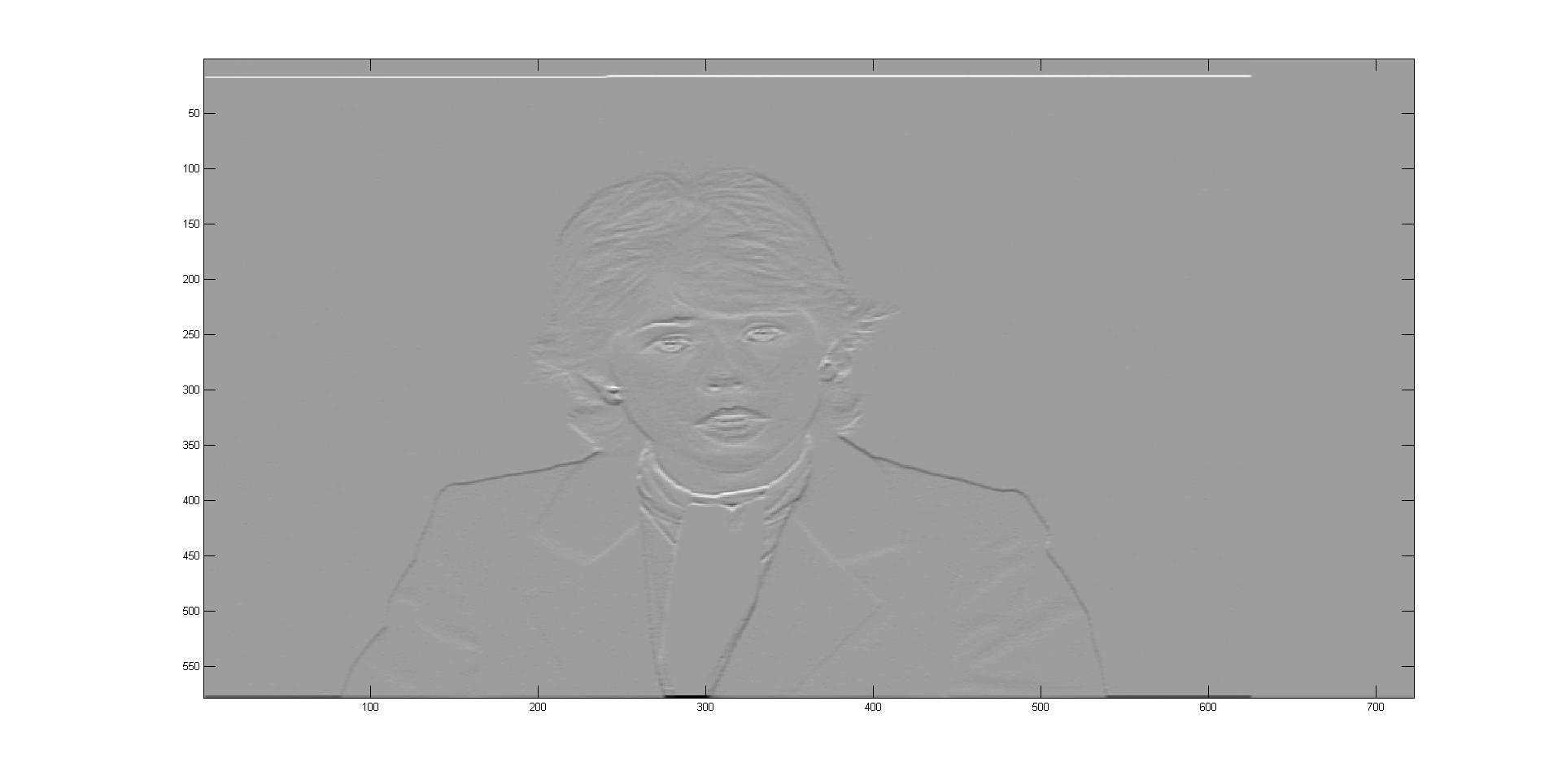


Figure 31 - Vertical gradient

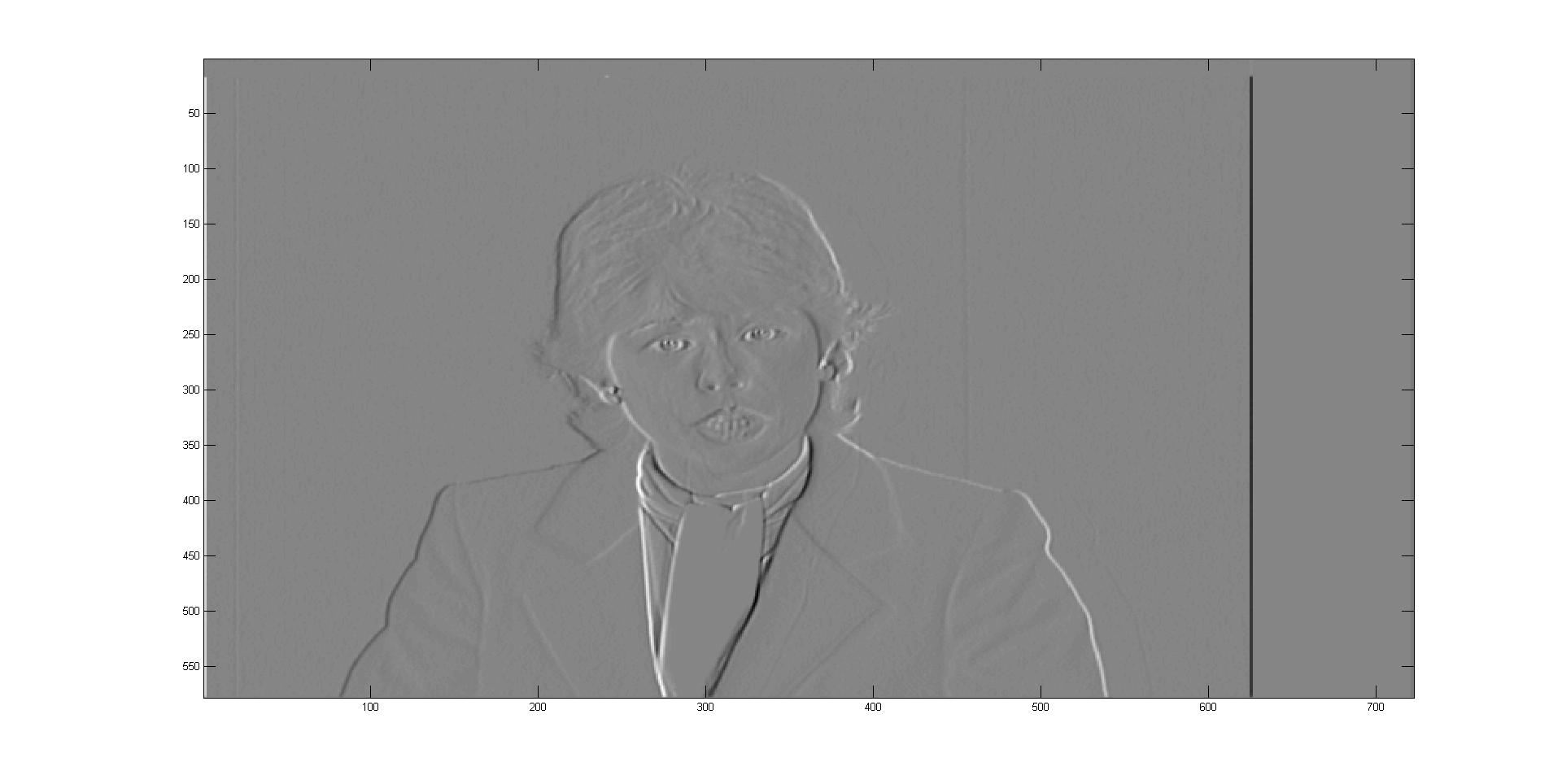


Figure 32 - Horizontal gradient

Using a one-bit quantizer on this derivative shows us where the derivative is very high by taking only values above a certain value; this is edge-finding. In the images below we look at derivatives in both dimensions with different threshold values. Raising T makes the edges thinner by only showing very high values of the derivative. Lowering T makes more subtle edges visible.

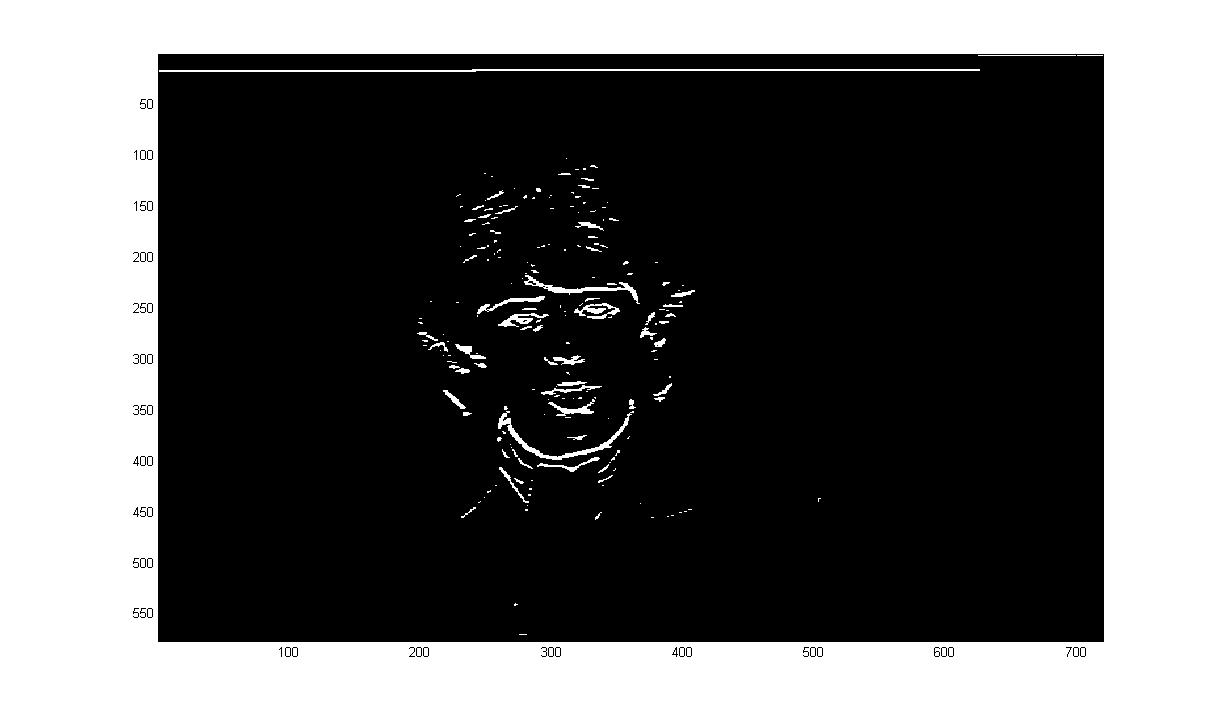


Figure 33 - Vertical Sobel 50

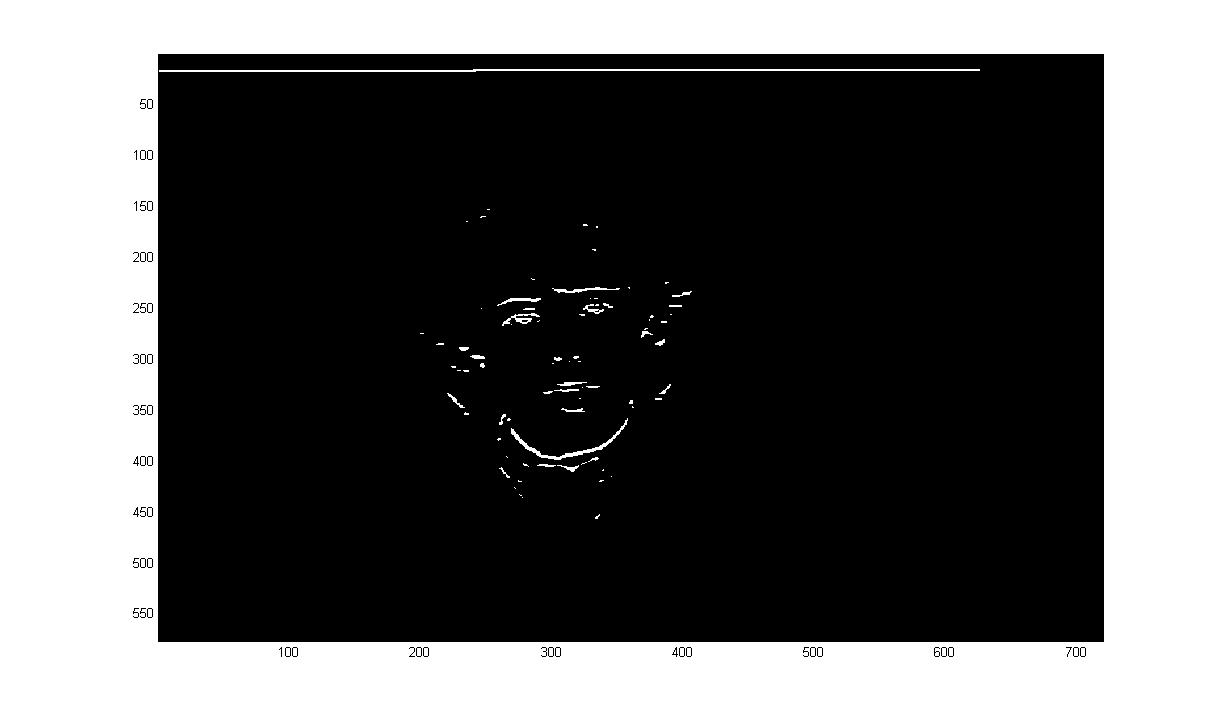


Figure 34 - Vertical Sobel 100

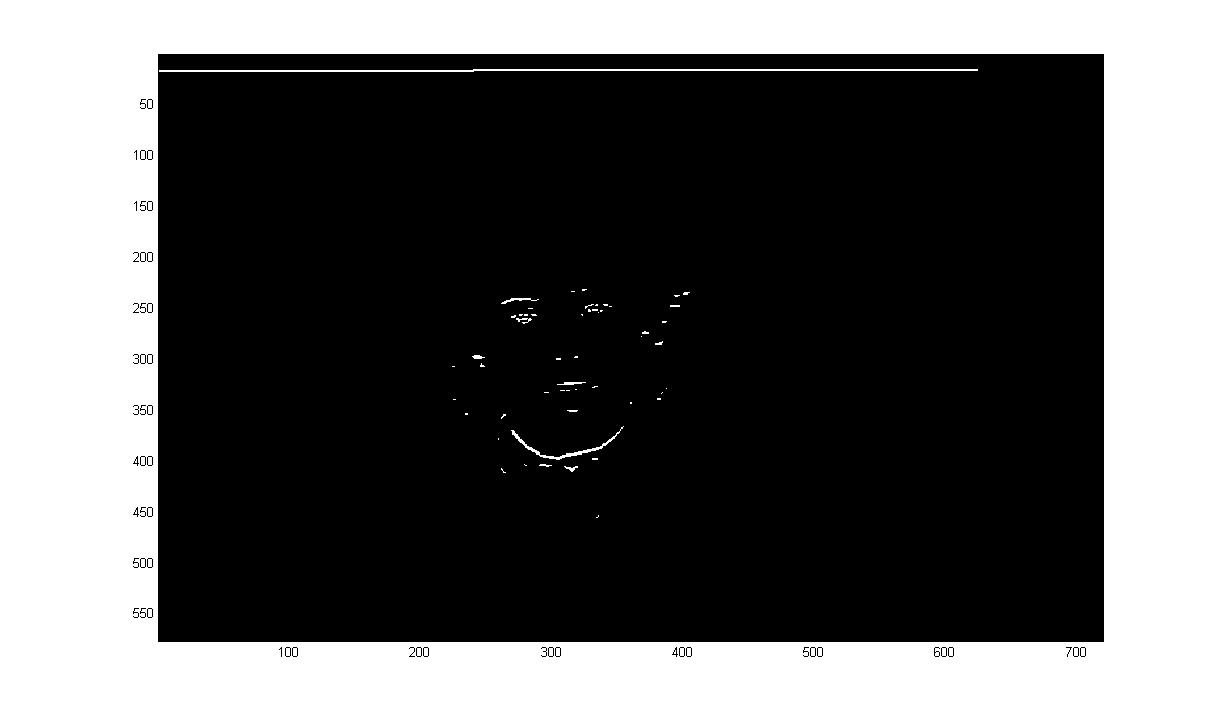


Figure 35 - Vertical Sobel 150

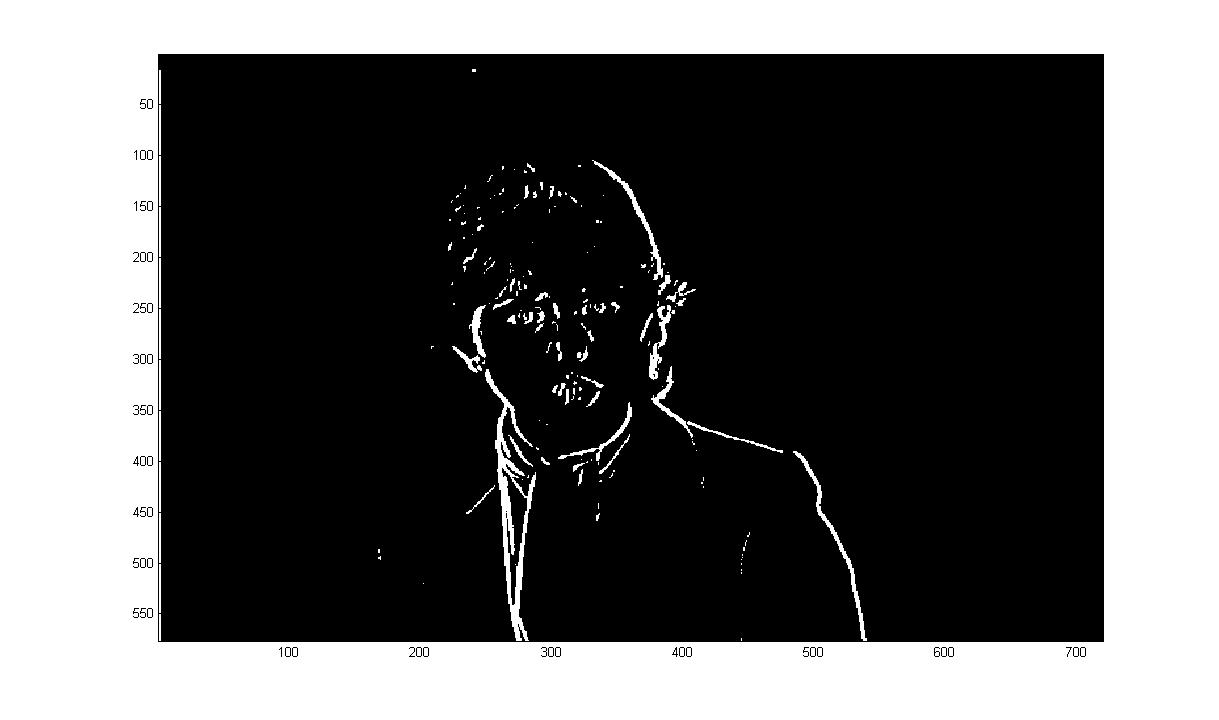


Figure 36 - Horizontal Sobel 50

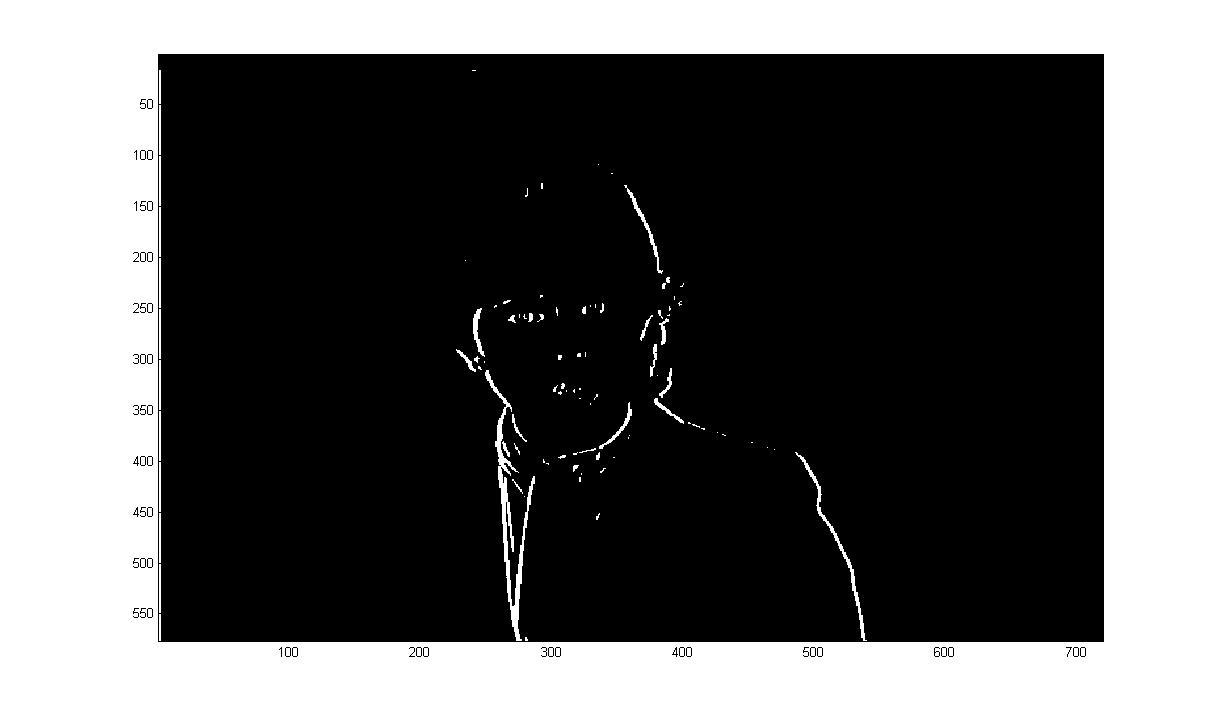


Figure 37 - Horizontal Sobel 100

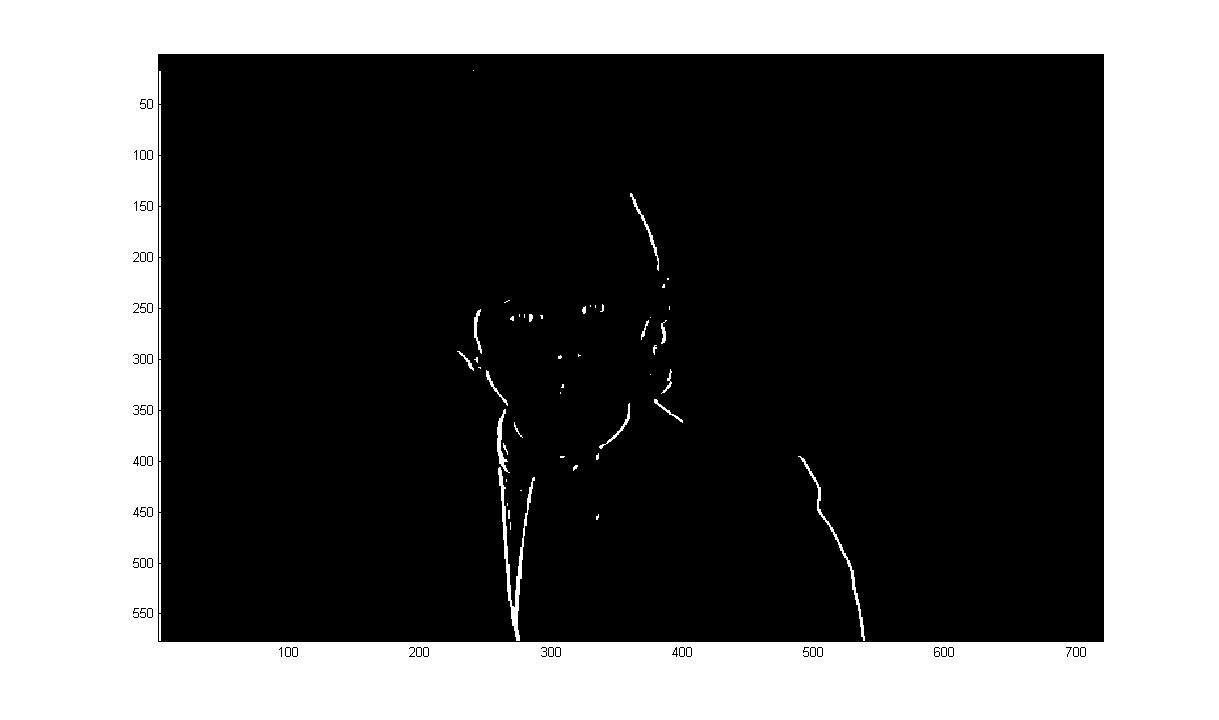


Figure 38 - Horizontal Sobel 150