Image Enhancement by different Histogram Equalization Techniques

-Neha Pande (201202023)

Sonali Goyal (201201042)

Introduction

Histogram equalization is a method in image processing of contrast adjustment using the image's histogram. Various histogram equalization techniques that have been implemented are:

A. Histogram Equalization (HE)
B. Adaptive Histogram Equalization (AHE)
C. Clipped Local Adaptive Histogram Equalization (CLAHE)

Throughout the project we have used 6 different images including images of noise, bands of extremely high contrast, flat image, crowd image, and others. There is a large volume of images due to the nature of the assignment. Due to this large volume we will address an image (or set of the same image) per each section. The rest are supplied for reference. Also all of the images have been scaled down in size to save in the size of the report.

A. Histogram equalization

1. histogram(I, n, min, max)

The program takes in an input image, the number of bins, and a min and a max value. The range (min - max + 1) is equally divided by the number of bins and rounded off to the nearest integer using the ceil function. The pixel intensity values in the range are allocated to the respective bins and then divided by the total number of pixels in the bins, to get the relative frequency in every bin. The program outputs the relative frequency of intensity values in each bin.

2. histoeq(I, n, min, max)

As with histogram, this program too takes in the same parameters and outputs an image with histogram equalization performed on it. This function first calls the function histogram to create the pdf's/ relative frequency for every bin. The function histoeq then uses these pdf values to calculate the cdf for an intensity value and maps it to an intensity value between the input min and max. The program outputs an image with histogram equalization performed on it.

PROCEDURE AND RESULTS

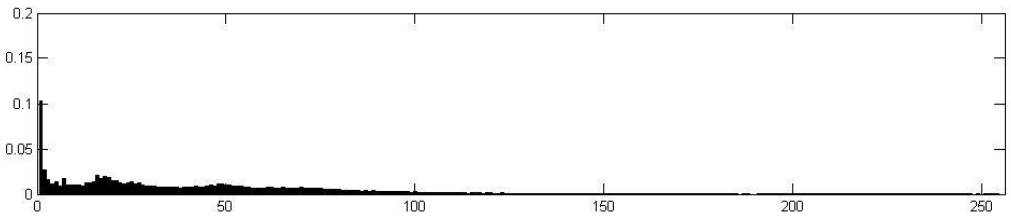
Algorithm that was specified has been implemented. We generated a pdf (histogram), built a cdf from that. And then scaled the cdf by N-1. The result for the images were exactly as expected. For each of the images we get a new image that visually is a lot easier to see. In terms of the histogram, we notice it does what it can to flatten them out.

1. **Nightsky**

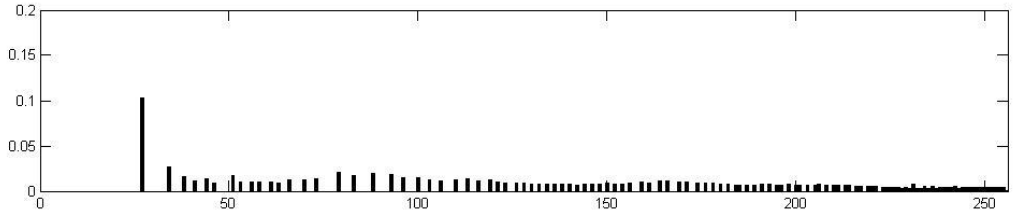
Original Image

Equalized Image

Original Histogram



Equalized Histogram



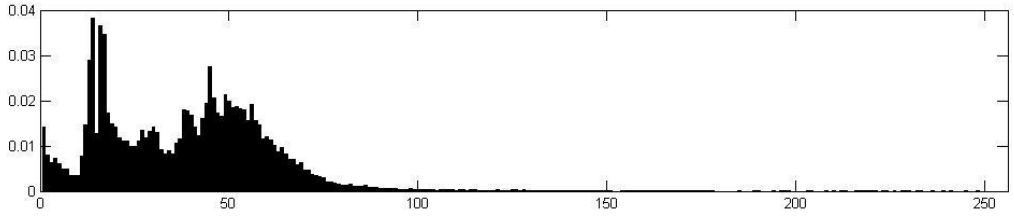
Running time to equalize: ***0.047000 seconds***

2. **Crowd**

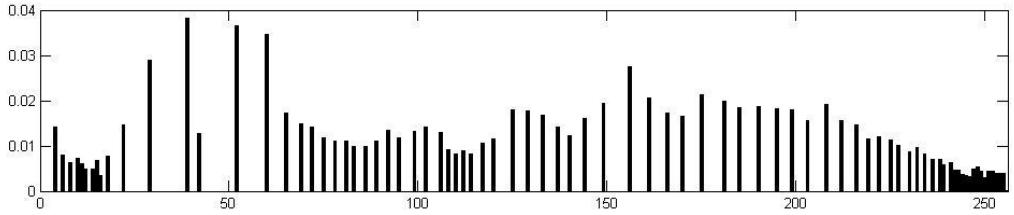
Original Image

Equalized Image

Original Histogram



Equalized Histogram



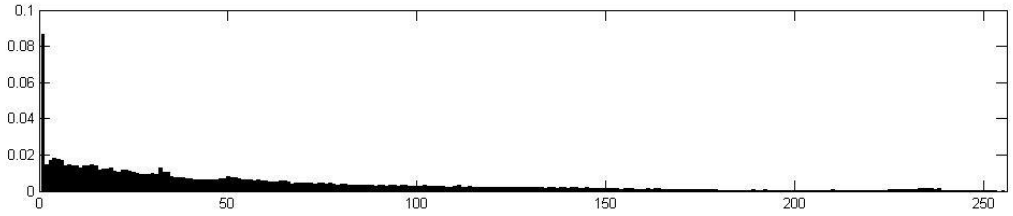
Running time to equalize: ***0.049000 seconds***

3. Batman

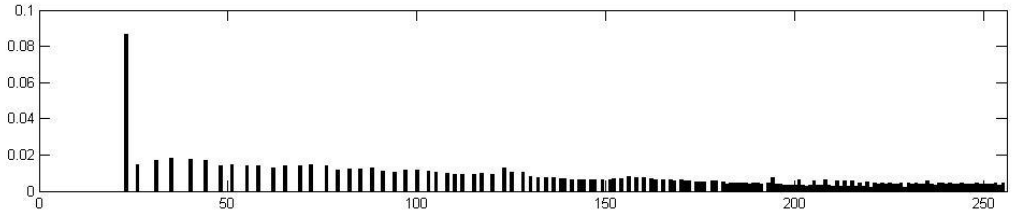
Original Image

Equalized Image

Original Histogram



Equalized Histogram



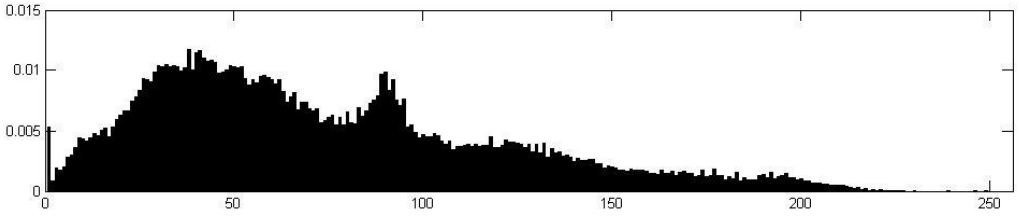
Running time to equalize: ***0.048000 seconds***

4. **Mountains**

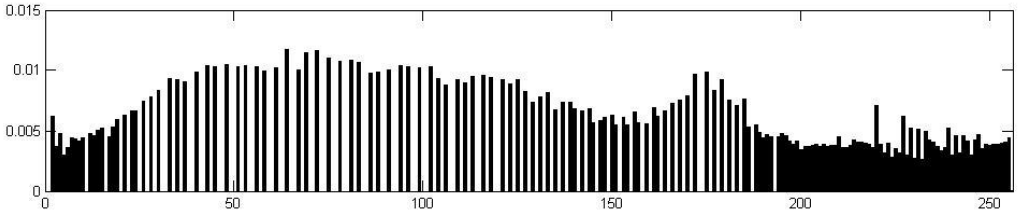
Original Image

Equalized Image

Original Histogram



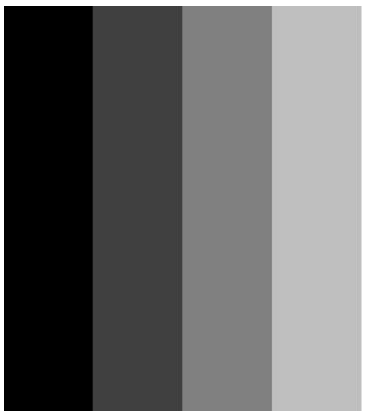
Equalized Histogram



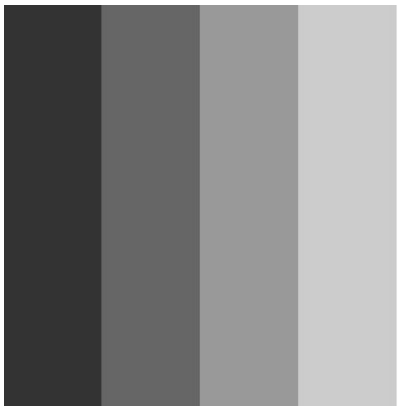
Running time to equalize: ***0.048000 seconds***

5. **Bands**

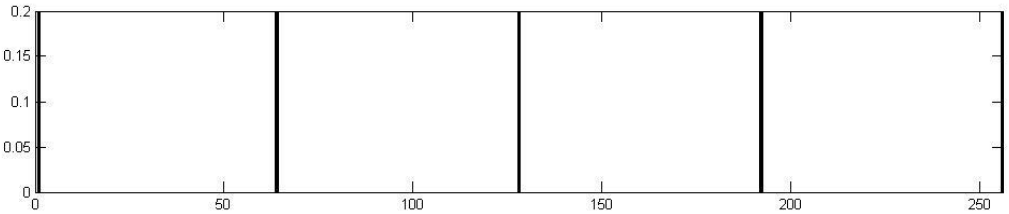
Original Image



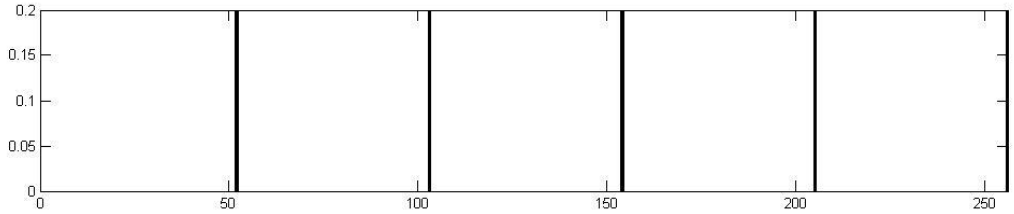
Equalized Image



Original Histogram



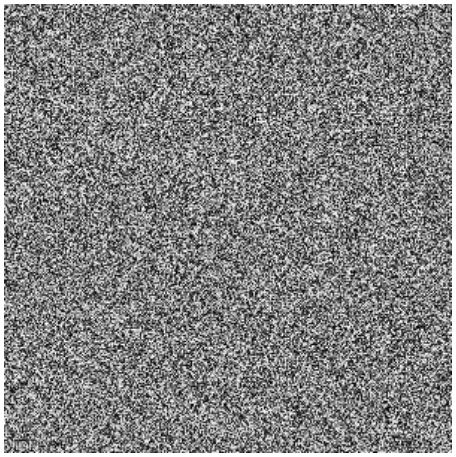
Equalized Histogram



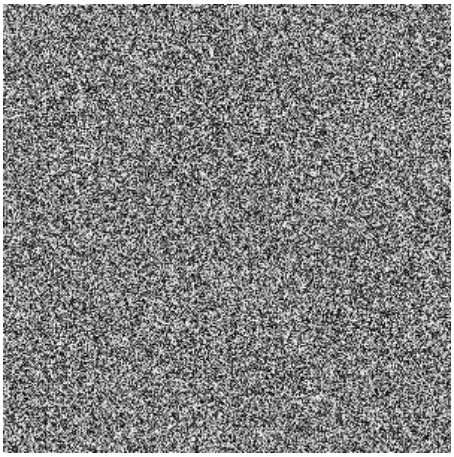
Running time to equalize: ***0.042000 seconds***

6. Noise

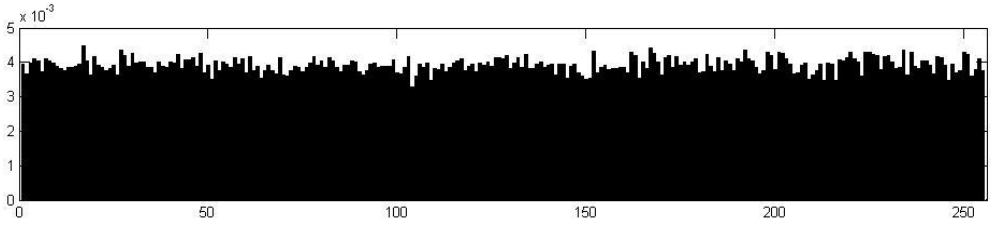
Original Image



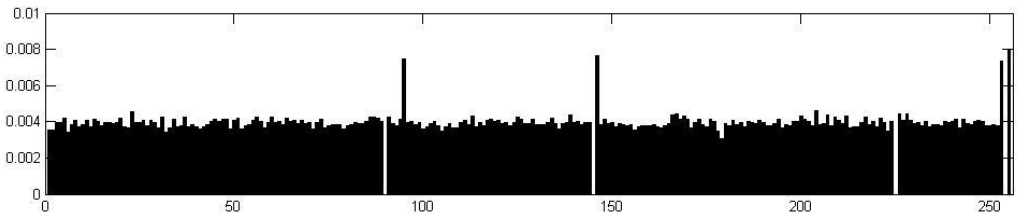
Equalized Image



Original Histogram



Equalized Histogram



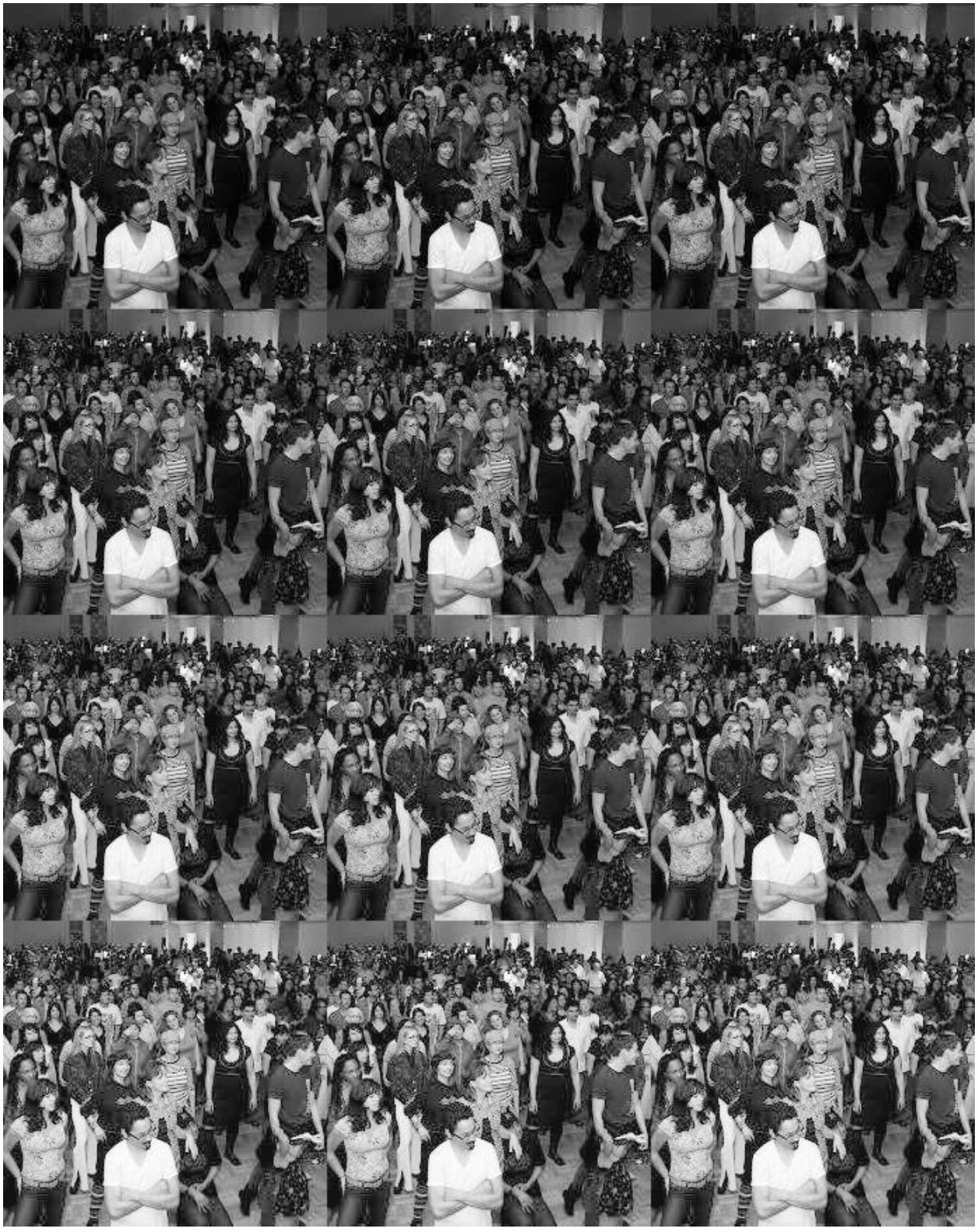
Running time to equalize: ***0.016000 seconds***

As we can see, when there are large spikes of intensity the image can't break them apart so it makes it hard to achieve a perfectly flat histograms. But we do notice that it spreads them out a fair bit, and uses the available space quite a bitter better.

As expected from the bands, the result doesn't seem to modify them much. It changes the values slightly due to the nature of the algorithm, but we still get bands of relatively the same color. With the noise we see little change, as expected; this histogram is nearly already flat. Something interesting to note is in the right histogram. Because a few intensities DO get shifted, and because it already is mostly flat, we get a few peaks that are nearly twice as tall as the original image. Visually it doesn't make a difference, but for an almost flag image, histogram equalization can have such artifacts.

Histogram equalization seems highly effective for each of these images except the two special cases (the bands and the noise). We do notice here though that histogram equalization gives a sort of night vision effect. The images are easier to see, but due to scaling some things appear less realistic.

To improve contrast without giving it that almost surreal feel to the pictures we can blend the two algorithms. The process can be seen from the images below. We decided to use the crowd image since the result seemed a little washed out. We get the following. The alpha values were 0.2, 0.4, 0.6, 0.8 (0.0 meaning no change to original, 1.0 meaning exactly like histogram equalization. The first image is of Crowd.



We also did a special case to show how important binning is. We used the outside example to show the effect of using less bins. The bin sizes were: 128, 64, 32, 16.

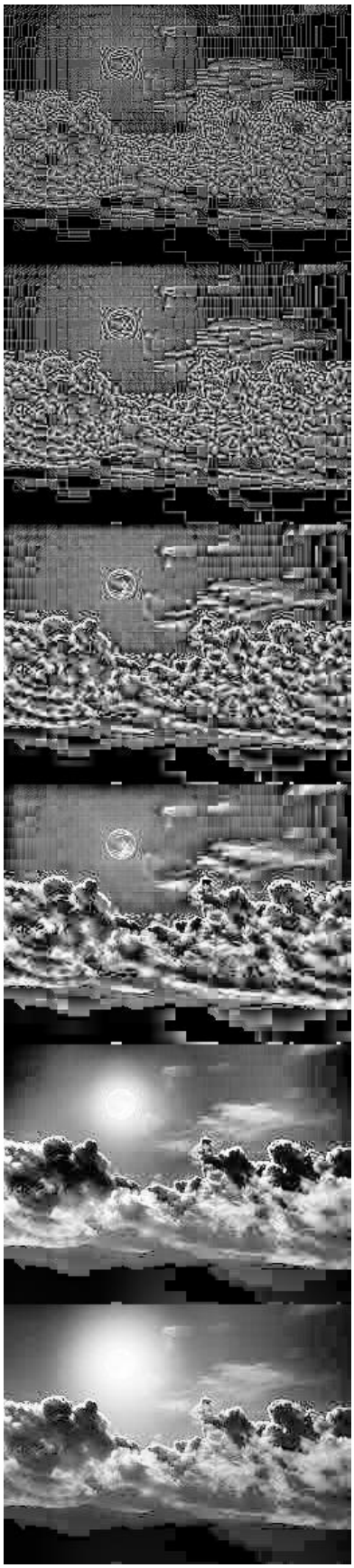
Nightsky Image

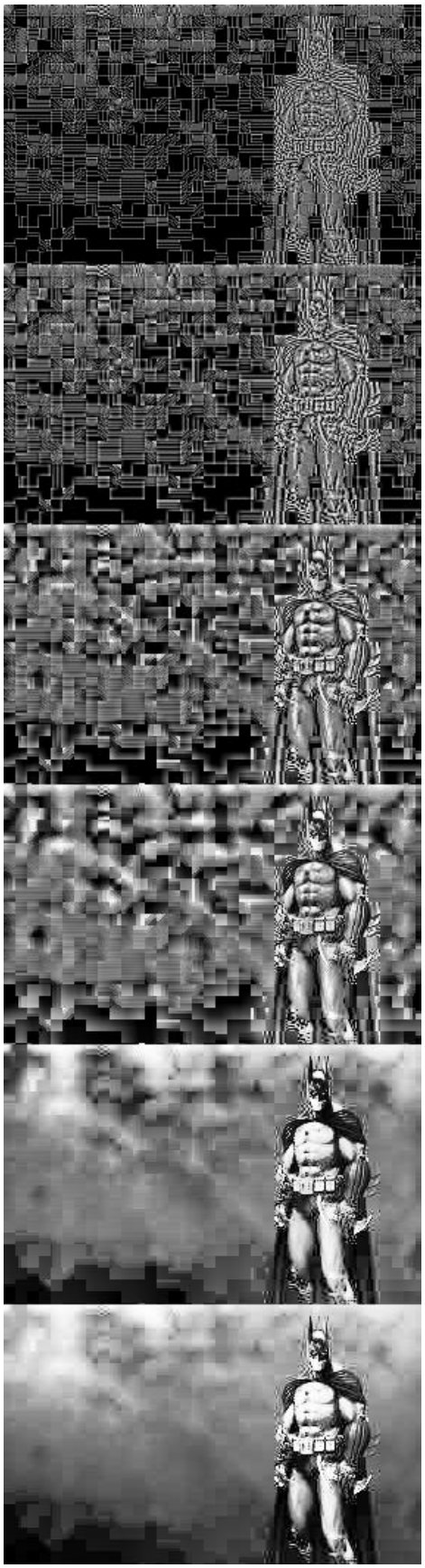
B. Adaptive Histogram Equalization (AHE)

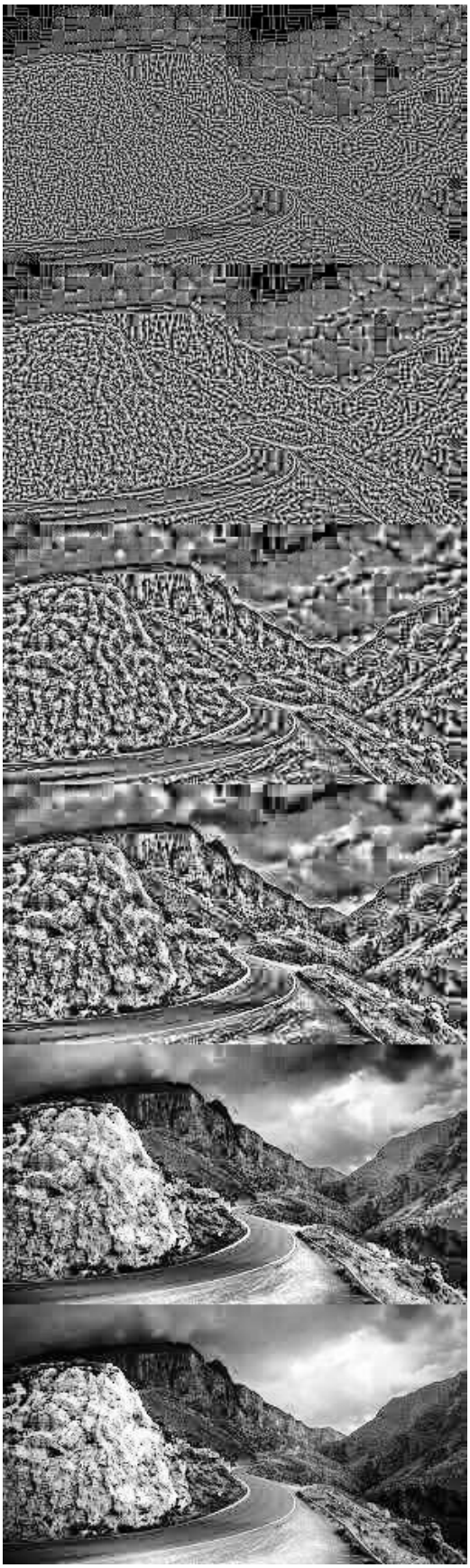
For images which contain local regions of low contrast bright or dark regions, global histogram equalization won't work effectively. A modification of histogram equalization called the Adaptive Histogram Equalization can be used on such images for better results. Adaptive histogram equalization works by considering only small regions and based on their local cdf, performs contrast enhancement of those regions.

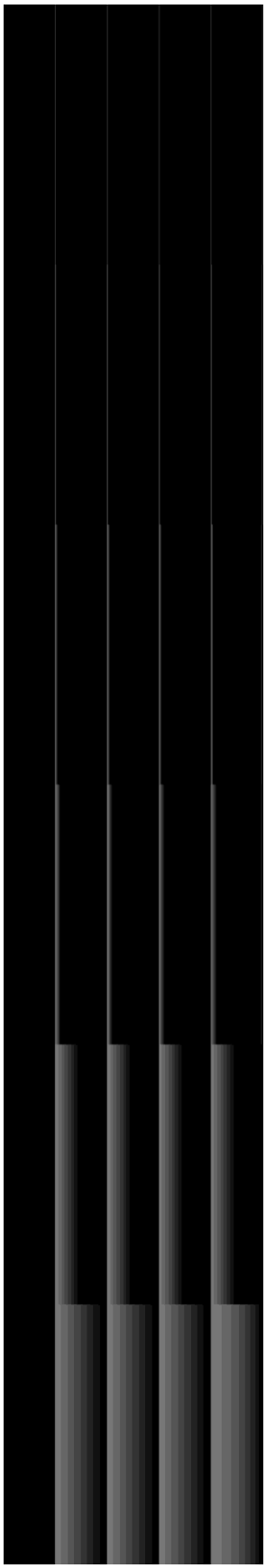
Adaptive histogram equalization can be implemented by various methods and each of those methods has multiple variations. We decided to use the pixel by pixel approach. We knew the results would have more artifacts than the tiling approach but I figured the general implementation would be easier. We tried our best and made decisions to bring down the overall running time.

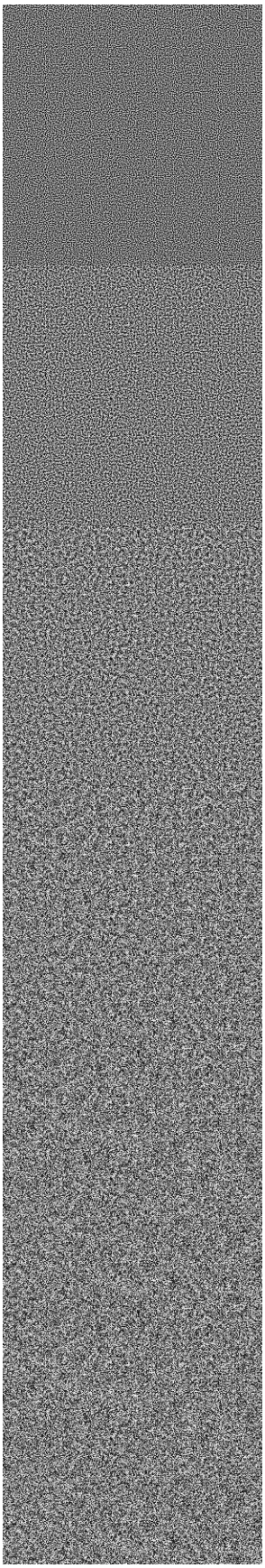
The window sizes for the below images are: [3 6 12 24 99 198]. Due to scaling the lowest values look a bit like noise. But in the full version when zooming in you definitely get crisp lines and shapes. This goes to show how crucial different window sizes are.











Two interesting examples here are the bands example and noise example. We can easily see the size of window increasing in the amount of change in our images. Because it is only effected by changes in it's neighborhood each of the bands turns to be the same color.

In the noise case it seems to make the image feel less random. This makes sense, because with small enough window sizes even the small areas of near consistent (but random) intensities will be broken apart. In the end we get a still random, but more uniform image.

We notice bands and chunks where we can definitely see where our window was. The plus here is that we can see some definite structures (visually) that were couldn't see before. At a first glance, this may not look as visually pleasing as straight histogram equalization, but it does bring out different structures.

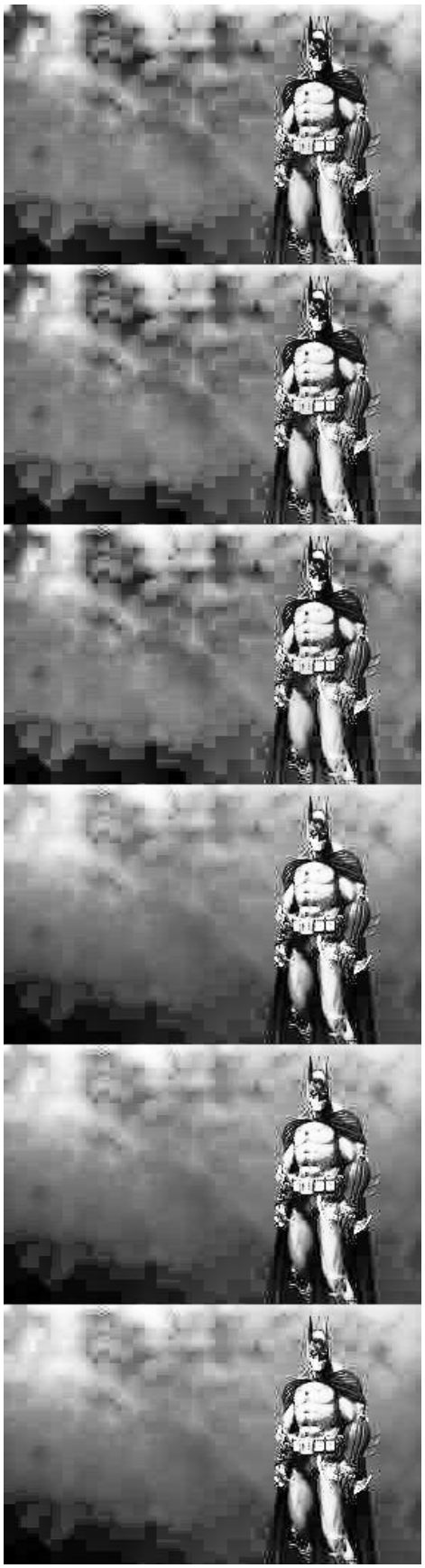
The below table shows the time to run AHE with each window size on each image. All the times are expressed in seconds. The below confirms the belief that doing the non-tile approach is O(N*M), where N is the size (N=H*W) of our image and M is the size of our window. (This is obtained since we have to iterate over the pixels, and recalculate the histogram which we do in a smart way which requires is only 2*M).

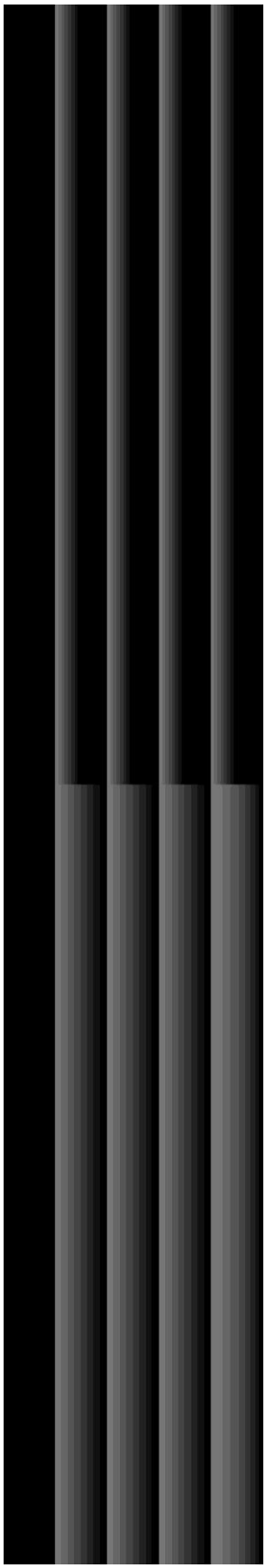
	3 pixels	6 pixels	12 pixels	24 pixels	99 pixels	198 pixels
Nightsky	0.23	0.26	0.37	0.59	1.74	3.01
Crowd	0.5	0.58	0.81	1.29	3.94	7.14
Batman	0.97	1.12	1.56	2.46	7.38	13.3
Mountains	0.24	0.27	0.35	0.52	1.44	2.53
Bands	0.64	0.72	0.94	1.4	4.07	7.22
Noise	0.25	0.28	0.38	0.57	1.7	2.92

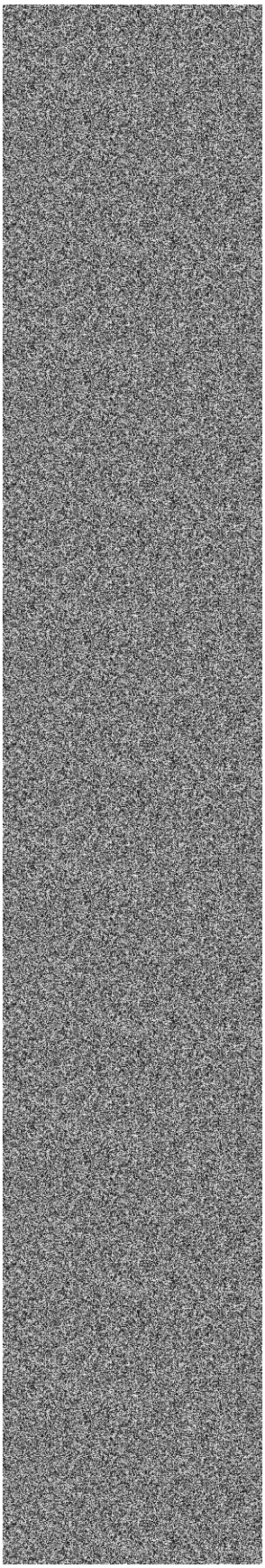
We expect the first column to be somewhere around 3 times slower than the regular histogram speeds. Indeed as we seem to range from about 2-4 times slower. This agrees with our projected running time.

C. Clipped Local Adaptive Histogram Equalization

The below images are generated clipped local adaptive histogram equalization (CLAHE). There are 6 images for each of the given images. There are 6 images for each of the given images. The top three have a window size of 99, and the bottom three have a window size of 199. In the three we then try 3 different clipping levels. We decided it would be easiest to have a percentage clip instead of a raw number. Percentage clip takes the values of the image (in this case between 0-256) and multiplies it be a number between (0.0-1.0) to give us our clip threshold. The thresholds are [0.4, 0.6, 0.8] which translate to be [102.4 153.6 204.8]. This means that for a larger window, the relative threshold will be larger, hence the reason the images are darker.







CLAHE has the result of decreasing the rate of change heavily. This shows up visually as a smoothing effect applied to the images.

The bands example now looks fairly neat. Since it's not allowing us to focus too largely on the areas of change now, we start to see them as distinct bands again. With a small enough thresholds we get a shading effect around the borders. We also see in the noise example that as the window gets larger we start to get more histograms that are clipped. And by clipping and spreading that value throughout the rest of the histogram, we begin to get a gray haze around the image. It discourages the heavy change, the very nature of the noise image. This explains why the lower images are generally darker, because we're spreading clipped values back into the mass of the image. One drawback we see from the noise is that since we use a hard threshold value we get a spherical haze. This happens since there are fewer values to add to our histogram near the edges. Having more values to add to the histogram gives us a greater chance of clipping.

CONCLUSION

Image enhancement has been implemented by different histogram equalization techniques on all different types of images.