CSMS 630 Project 1- Image Processing

By: Kirtan Shah

This project is combination of different filters and conversions. It is written in Python and process 499 images in a span of 2.5 hours. The GitHub link for this project is here. There is a configuration .ini file *(userConfiguration.ini)* attached with the project which must be place in the same folder as the main python file. The user my edit the .ini file to receive different results. To get the most optimum results please follow the format of the already established values within the ini file.

The image path must complete path to the images:

- C:/Users/user/Cancerous cell smears/*

The output path must complete path to the directory to store the images:

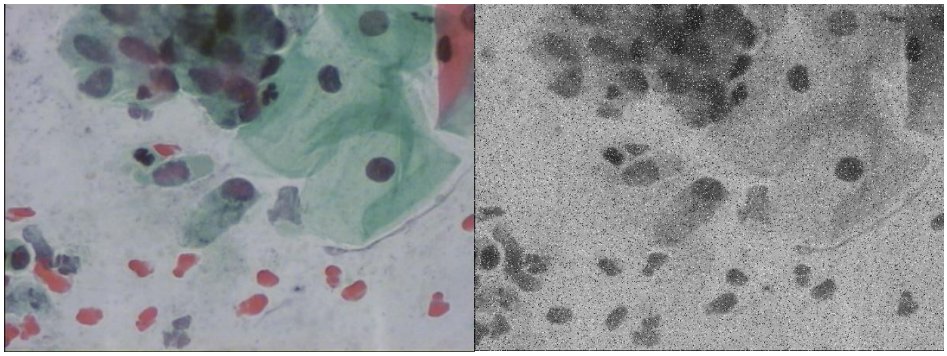
- C:/Users/user/Documents/output

Noise Addition methods:

The addition of noise to an image is crucial way to obscure any point with in the image or the image itself.

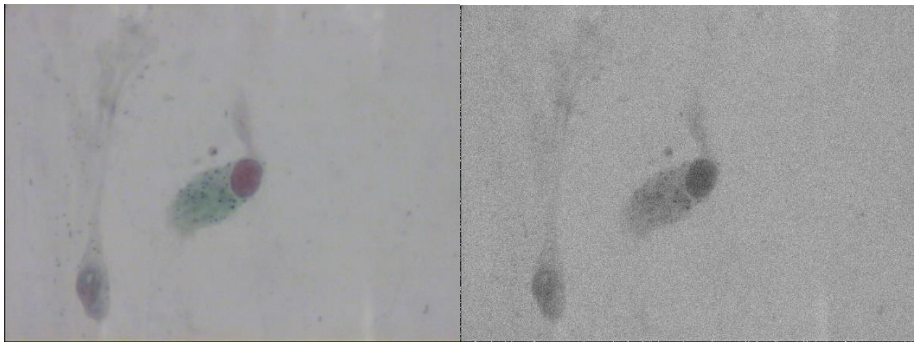
- **Salt and Pepper**:

A probability is established and based on a random salt (0) and pepper (255), noise is assigned to the image. The user assigned strength is use to generate a probability. If the pixel value is within this is probability, then the pixel value will change to 255 if not then to 0. You can find the complete set in */saltNpeper* folder.



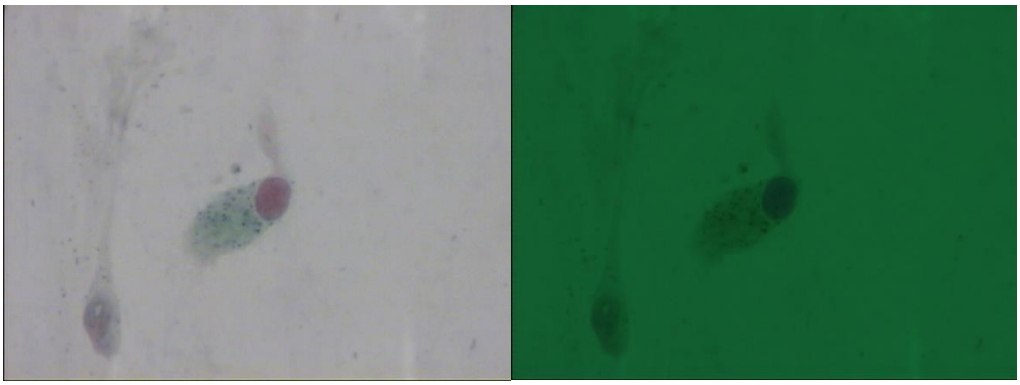
- **Gaussian**:

After determining the standard deviation and mean the Gaussian noise can be added to the image based on the user inputs. Mean is default to 0 and Standard Deviation to be visible has to be above 15. You can find the complete set in */gaussian* folder.



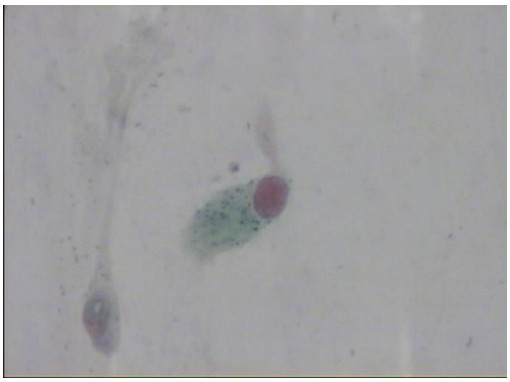
RGB to Single color spectrum

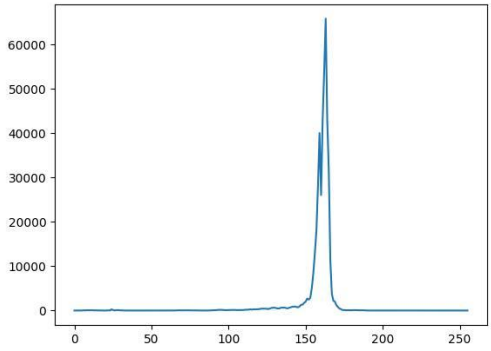
From the image each color channel Red, Green and Blue will be isolated and averaged and then the user specific weights will be multiplied to generate a R, G, or B image. You can find the complete set in */colorSpecturm* folder.



Individual Histogram per image

The image is first converted into gray scale image. Each pixel intensity is first counted and mapped to a list 256 values. Based on the index of the list a histogram is created by plotting the pixel intensity on a X, Y graph. You can find the complete set in */histogram* folder.





Individual Histogram Equalization

Graphical representation or the cumulative probability is calculated by find using the cumulative distribution functions. For each pixel intensity multiply it by the 1/total number of pixels. Then keep a running tally of this and add it to the previous tally. Establish a CDF for each pixel intensity value and this will allow for image smoothing. You can find the complete set in */histogramEqualizations* folder.

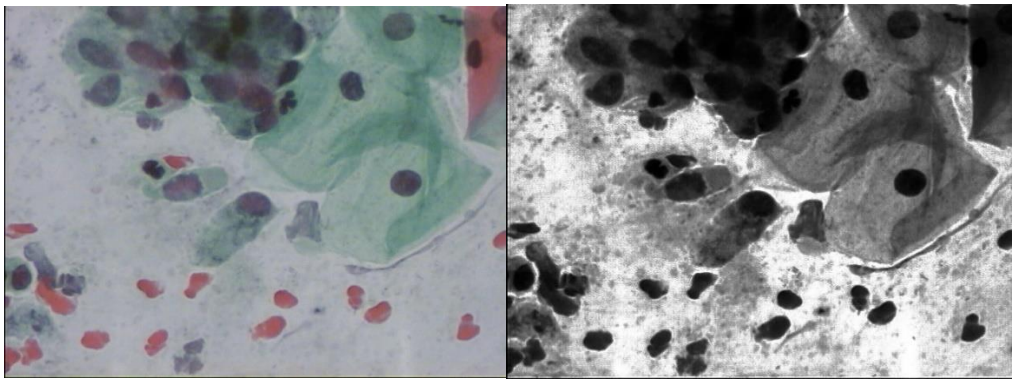
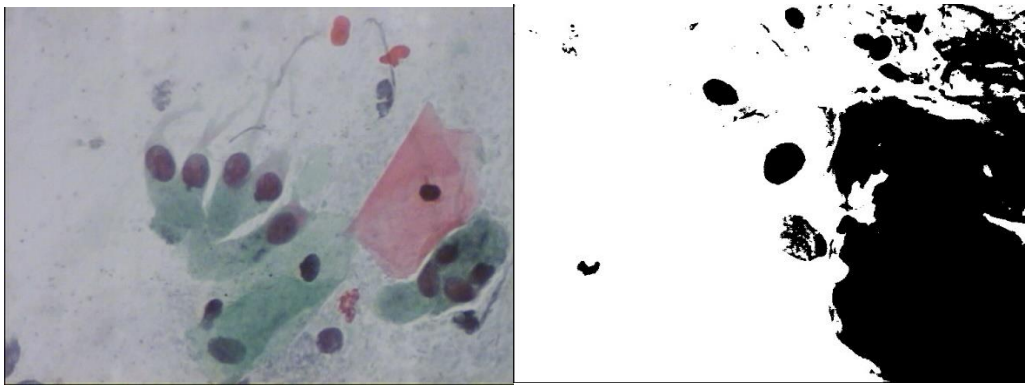


Image Quantization

First the image is converted into gray scale then based on the quantization level provided by the user the quantization level is calculated. First subtract the level by 1 and then divide the result by 255 to get the proper sampling level. Then divide the current grey scale pixel by the quant level and round it to the nearest whole number. I.E Quantized image – grey scale image. You can find the complete set in */imageQuantization* folder.



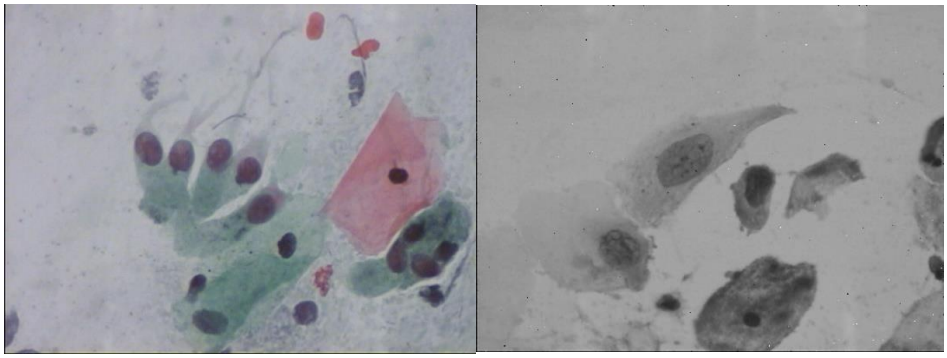
Mean Squared Quantization Error

First the quantized image and the original grey scal image is subtracted. The result is then squared. After the result has been squared I multiply the result 1/total number of pixel. Once that matrix has been established, I add the array elements to a list and the sum value of that list is the Mean Squared Quantized Error of that image. I have written the error for each image in *MSQE.txt*

Filters

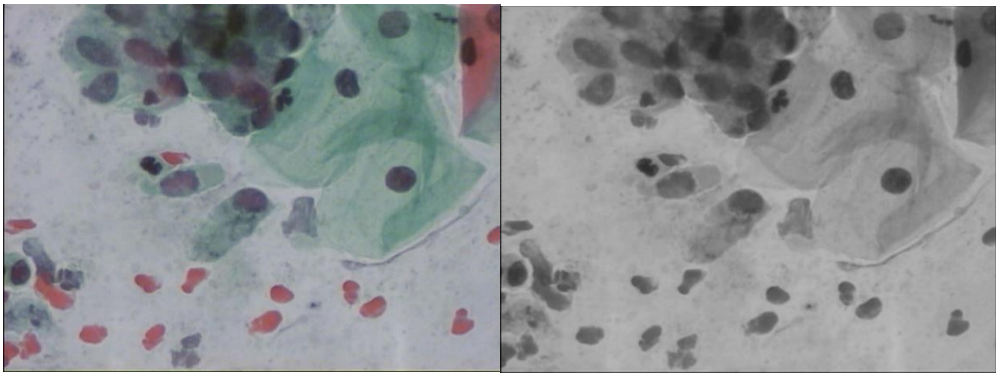
- **Median Filter**

After the image has been converted in to grey scale and the weights have been obtained. The image will be padded to account for borders. A mask kernel (I.E 3x3) is generate to test all pixels. Each pixel is appended to a list. The times appended is determined by the associated weight value. Once the list comprised it is sorted and the median value of that list is added to current pixel value of the image. You can find the complete set in */medianFilter* folder.



- **Linear Filter**

After the image has been converted in to grey scale and the weights have been obtained. The image will be padded to account for borders. A mask kernel (I.E 3x3) is generate to test all pixels. All the values will be multiplied by 1/filterSize^2. This generates a weights array which can be multiplied to the mask kernel. The values of the kernel will be appended to a list where the sum of the list will replace the current pixel value. You can find the complete set in */linearFilter* folder.



Metrics

Method	Average Time per image in seconds	Total Time in Seconds
Grey Scale Conversion	2.6701528056112225	1332.859375
RGB Conversion	0.02094814629258517	11.96875
Salt and Pepper	2.8839867234468937	1439.875
Gaussian	4.953564541457286	2260.78125
Histogram	3.95069998	1965.456821378
Histogram Equalization	3.5898584066164156	1971.39929002
Image Quantization	5.223003695999995	2606.2788443039976
Median Filter	15.6757577655	7822.203125
Linear Filter	27.963458166332664	13954.312
MSQE	7.458388	3721.735612

The average time per image in seconds: 7.438981823

The total Process time in seconds: 37086.87007