# Image Engineering Fundamentals Report

Group 1

June 17, 2018

## 1 Introduction

Images are represented by pixels on a computer screen. Each image has dimensions $n \times m$. These images are stored as a corresponding $n \times m$ matrix where each entry is one of the pixels on the screen. The values in this matrix then relate to the representation of that pixel on the screen. This can be in grey scale or RGB to name a few possible image representation schemes. As each can be coded in the $n \times m$ matrix it is possible to perform operations on the image to manipulate the image. A simple example of this would be to apply the standard general clockwise rotational matrix transformation to the matrix, this would result in the image being rotated. Six of these image manipulation methods will be discussed from their implementation to practicality.

The rest of the report is organized as follows: Averaging, Median Filtering, Linear Grey-level Transform, Histogram Equalization, Sharpening and Thresholding.

## 2 Averaging

Image averaging is a technique used to remove granular noise of an image (more specifically one or two-pixel granular noise). Suppose for each pixel we define a $a \times a$ matrix around it. In this case we will look at a $3 \times 3$ matrix with entries $a_1, a_2 \ldots . a_9$ . Each of these pixels has a given value, the mean value of these nine entries is calculated and then applied to the filtered pixel. This method is extremely effective n removing such granular noise it does however blur the overall image to some degree. Examples of this can be seen in Fig.1 and Fig.2.

To test the following method in a more efficient way, we used an noise-adding Algorithm to apply noise to the original pictures.

Figure 1: Result of Averaging I

Figure 2: Result of Averaging II

## 3 Median Filtering

An improvement to basic image averaging is median filtering. A similar approach is taken to image averaging. A window is created around each pixel, the $a \times a$ matrix as described earlier. This time however instead of taking the average of the surrounding pixels in the window the median value of $a_1, a_2 \ldots . a_9$ is calculated and then applied to the filtered pixel. This proves to be way to remove granular noise while also retaining the image definition and not adding any blur. The implementation is slightly more complicated than image averaging if the $a \times a$ contains an even number of entries as the median value is not an actual value within the window. The value of median filtering is that as the images edges are preserved through the filtering we it can be used in images where there is text as the text remains legible. The results can be easily compared with those of image averaging as seen below in Fig.3 and Fig.4.

Figure 3: Result of Median Filtering I

Figure 4: Result of Median Filtering II

## 4 Linear Grey-level Transform

As stated in the introduction, images can be represented in either a given color scheme of in grey scale. In grey scale the pixel takes a value which corresponds to an intensity of the color black such that every pixel ranges between white and black. If there was an image with very low contrast that we wished to make clearer there are a few possible methods to do this. One such is the liner grey scale transform. Suppose there is a pixel that has a given intensity $I_1$. If we let $f(x) = ax + b$ be a function that maps $I_1$ to some new intensity vale $I_2$. It must also ensure that the new value is within the desired range. This results in a higher contrast image, a can be seen in the examples given though, as there is some upper limit on the intensity value a significant number of pixels can be transformed to the maximum intensity value. This results in a large amount of black in the image. This constraint on the method causes the nonlinear grey scale transformation to be effect for images which have some narrow greyscale intensity band.

Figure 5: Result of Linear Grey-level Transform I

Figure 6: Result of Linear Grey-level Transform II


Figure 7: Result of Linear Grey-level Transform III

Figure 8: Result of Linear Grey-level Transform IV

## 5 Histogram Equalization

For images that have a wide range of greyscale values but also suffer from poor contrast nonlinear greyscale transforms will not work as they will cause a large portion of the pixels to be capped out at the maximum intensity values. In situations such as these histogram equalization is used. Suppose each value of greyscale intensity was stored in a histogram to represent their frequency within the image. The goal of histogram equalization is to take the histogram representing the pixels' intensity and transform it to a flat histogram such that all intensities occur with equal probability. To do this a cumulative histogram is taken of the values. From this histogram you can get some function $g(x)$ that represents the cumulative distribution function of the original histogram. In histogram equalization a transformation $T$ is applied to $g(x)$ such that $g(x) \cdot T \rightarrow h(x)$ where $h(x)$ is the new cdf corresponding to a flat histogram. The results of this when applied to a low contrast image with a wide greyscale intensity range can be seen below.

Figure 9: Histogram Equalization I

Figure 10: Histogram Equalization II

Figure 11: Histogram Equalization III

Figure 12: Histogram Equalization IV

## 6 Sharpening

As mentioned when discussing image averaging, the drawback of the method is that it can result in image edges being blurred. Image sharpening is the process of sharpening of image edges. To achieve this one such method is as follows. Let $f(x, y)$ be the greyscale intensity values of a given pixel $(x, y)$. Two copies of this image are taken the original $f$ and then a second $f$. To the second the Laplacian is applied.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

The resulting image $g(x, y)$ is created by $g = f - \nabla^2 f$ . This works as when the Laplacian is applied to the image, the areas of greater difference in intensity then result in a greater

value than the original. This causes the original image's edges to be enhanced once the operation has been performed as is shown in the images below.

Figure 13: Result of Sharpening I

Figure 14: Result of Sharpening II

## 7 Thresholding

As has been discussed, each pixel in greyscale representation has some value that corresponds to their intensity. If the there was some image $I_1$ and it is to be broken down into distinct segemets, the simplest way of applying segmentation to the image is through thresholding. Thresholding is when for each pixel in $I_1$ there is a threshold value for the intensity that if it is within that threshold it is transformed to a new intensity level. The simplest example of this would be if the intensity was high enough the value in the pixel is

then replaced such that the pixel is now black and if it fails it is then converted to white. This results in a high contrast black and white image. Several examples of this can be seen the images below.

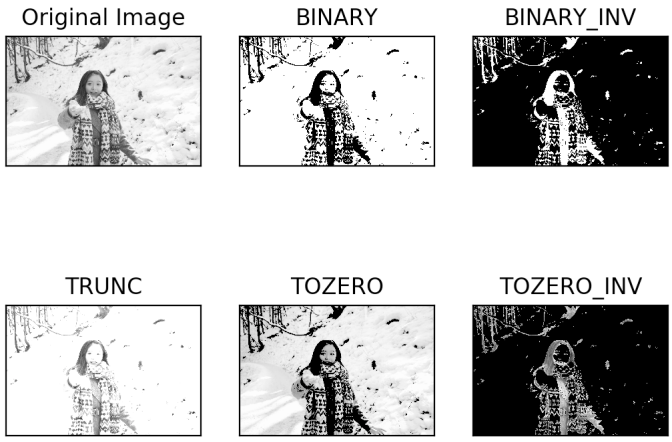


Figure 15: Result of Sharpening I

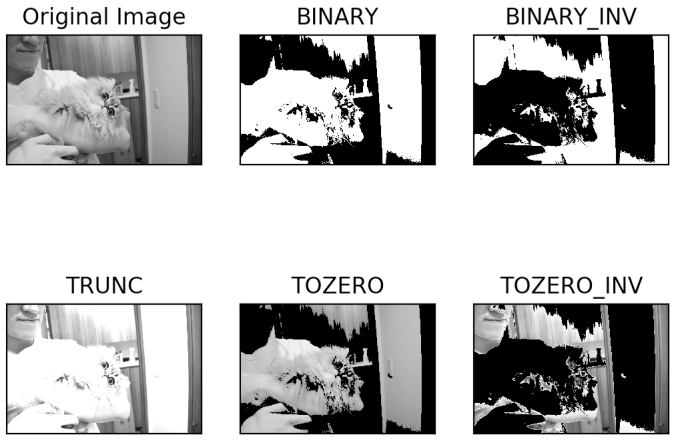


Figure 16: Result of Sharpening II

# 8 Contribution

Alexander Ashcroft 1W15BG01-2 Report compilation, presentation, Q1 100%
Chen Yuxuan 1W15BG12-1 Python Code, Formatting, Q2 100%
Pratishtha Singh - 1W18CG81-7 Q3,Q4 100%
Lan Dongwon 1X15BG08-6 Q5 100%
Zhang Zhaowei 1X15BG03-8 Q6, slides 100%