April 2016

Digits Recognition on Medical Device

Chang Liu
The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science

Abstract

With the rapid development of mobile health, mechanisms for automatic data input are becoming increasingly important for mobile health apps. In these apps, users are often required to input data frequently, especially numbers, from medical devices such as glucometers and blood pressure meters. However, these simple tasks are tedious and prone to error. Even though some Bluetooth devices can make those input operations easier, they are not popular enough due to being expensive and requiring complicated protocol support. Therefore, we propose an automatic procedure to recognize the digits on the screen of medical devices with smartphone cameras.

The whole procedure includes several "standard" components in computer vision: image enhancement, the region-of-interest detection, and text recognition. Previous works existed for each component, but they have various weaknesses that lead to a low recognition rate. We proposed several novel enhancements in each component.

Experiment results suggest that our enhanced procedure outperforms the procedure of applying optical character recognition directly from 6.2% to 62.1%. This procedure can be adopted (with human verification) to recognize the digits on the screen of medical devices with smartphone cameras.

Keywords: Image Enhancement, Stroke Width Transform, Tesseract-OCR Engine

Acknowlegements

I am heartily grateful to my supervisor, Dr. Charles Ling, who has assisted me through all the stages of the thesis, guiding me and encouraging me. I am so grateful that he looked at every details of my experiments, pointed out the problems and gave me suggestions and advices. His seriousness towards researches, as well as his great passions, impressed me a lot and I learnt much from him. Besides, he even sacrificed his own holiday time to review the writing of the thesis, tuning every details, which certainly made this thesis much more logical and clear than before. Without his consistent guidance and support, this thesis would not have been completed on time.

Also, I would like to provide my appreciation to my colleague and friend Yan Luo, who always takes care of me, brotherly, both on my study and life here in Western. He encouraged me a lot to the experiment and raised me up when I was down.

I would like to thank my colleagues and friend Shuang Ao and Xiang Li, who gave me many useful suggestions to my experiments. They also provided valuable resources, in particular on the useful skills of using latex, as well as enough concerns in the writing of the thesis.

Besides, I would like to thank my colleague and friend Robin Liu, who works and always fights with me during my study here.

I would also like to thank other friends, who brought about loads of joys and delights to me through all the stages of the thesis. Life could be much more boring without them.

In the end, I am greatly thankful to my parents, who always support me, physically and mentally, though far away. Without them, I even cannot have the chance to study here.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we briefly introduce the overall content of this thesis. The first section describes the background and problem definition of this topic. In the second section, we discuss the dataset we used and the challenges. The third section is a short description of the method we adopted. At last, the thesis outline is introduced.

1.1 Backgrounds

With the development of mobile health, how to log all kinds of data in a more convenient way becomes the hottest topic for all mobile application companies and health researchers. Let us imagine this: if a mobile application can fetch data automatically from the user's body without any extra operations, that would be a groundbreaking event for the mobile health industry. Unfortunately, right now no such application exists.

As machine learning and data mining techniques get developed drastically in these years, many huge IT companies such as Google, Apple or Microsoft provide some alternative solutions based on these techniques. Such as, use of large amounts of exercise data to train a model which lets mobile devices estimate the exercise data of users through the movement of mobile devices itself. Inspired by their idea, we combine the computer vision techniques with a practical problem and try to make the data input from medical devices easier.

For most mobile health applications, especially for diabetic health applications, users need to input the data from medical devices such as glucometers and blood pressure meters frequently. Logging data always needs at least three steps: first, users read the digits from the medical devices; then open the app and go to the log interface; and last users can log the data by typing the correct digits into the dialogues or text boxes. Between the first and the last steps, users often forget the digits they read and may need to check those digits again. We even have

not considered the extra time for users to find their cell phones or transform the data into the correct form. These kinds of "simple" tasks are inconvenient and prone to error. They will also decrease the users' desires of using the application.

However, if we can simplify those steps into only one click, it will significantly improve the user experiences of these kind of operations. The idea is to use the mobile device to read and record the data automatically. Some medical devices provide Bluetooth interface for sharing data with other devices through their protocols. Other devices can connect to those medical devices to export all the data logged in them, which makes the data logging operation much easier. The extra costs for Bluetooth hardware and protocol design increase the price. Moreover, for competitive reasons, the complexity and variety of protocols also narrow the usage of them. After all, they are not popular enough to solve this problem.

As most smartphones have cameras for users to take photos, users can use the camera to take a picture of the screen of the medical device. After we get the photo, we can make use of computer vision techniques to recognize the digits on the picture and thus record the data. This procedure do not need to consider extra expense or complex protocol issues, which can be a common solution for logging the data. In this paper, we will focus on one kind of medical device, namely glucometers, whose data is required by health applications most frequently.

1.2 Standard Methods

Reading or recognizing characters or numbers from a picture can be categorized as the problem of Optical Character Recognition (OCR). Most OCR engines require the input image to have a clean background and clear foreground. However, the source images for our tasks are taken under various conditions, which extremely increase the difficulties to recognize the target digits. Besides, for typical natural scene images, there are lots of useless objects and unpredictable noises, which will highly affect the result of all kinds of OCR engines.

As shown in Figure 1.1 we need to use image enhancement techniques to improve the quality of source image first and then segment the image into small regions and get the ones of interest so as to eliminate the useless objects.

Therefore, we decided to design a computer vision process including several standard computer vision steps: image enhancement, the region of interest detection and optical character recognition. Previous works exist for each, but they have various weaknesses that lead to a low recognition rate. We apply several novel enhancements in each step which generates a much better result.

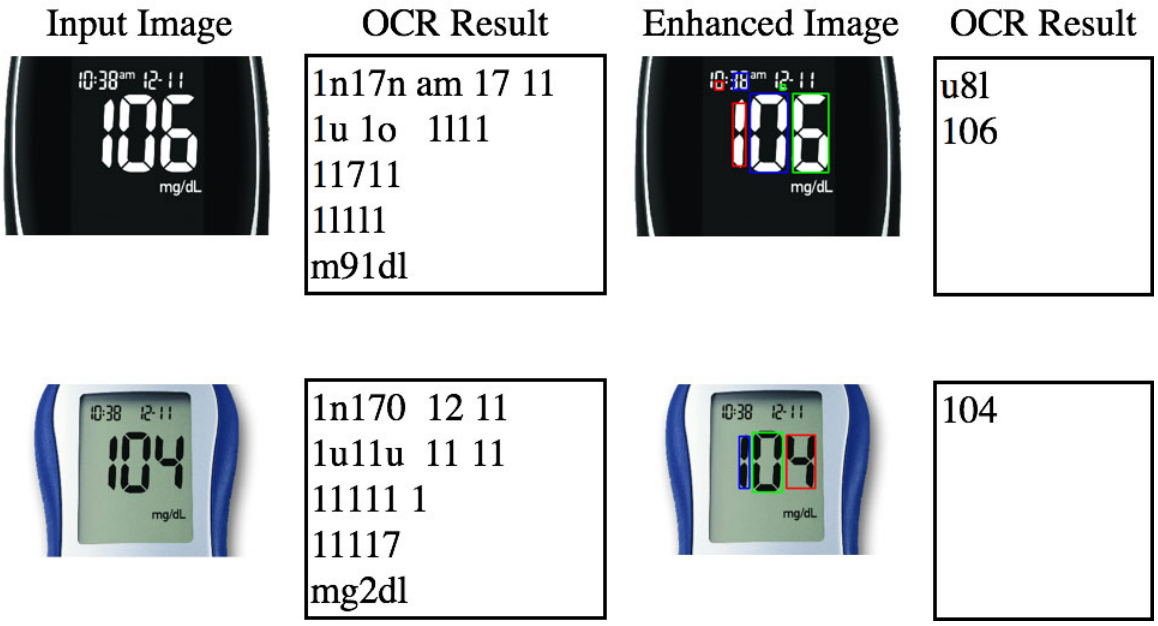


Figure 1.1: Optical Character Recognition Results between enhanced images and original images

1.3 Proposed Methods

In the image enhancement step, we adopted the combination of basic filtering algorithms to decrease the noises and improve the image quality. For example, we use the homomorphic filter to balance the light effects, median filter to eliminate the noises introduced by weak light source and modified histogram equalization to increase the contrast, and morphological scale the image to a standard form for next step. The experiment results show that there is no certain best composition of these algorithms and also their parameters, we can only find out one relatively good solution for a group of input images.

For better recognition result, we adopted the procedure named stroke width transform to detect the regions of interest. However, the original procedure is used for text detection, the experiment on our datasets generated very low recall rate. Therefore, we adjust the filtering and chaining part of it to get a high recall rate but a low precision rate on average, which provided a group of image candidates for optical character recognition. To improve the precision and eliminate the number of image candidates, we used a simple linear support vector machine to filter those wrong ones. We collected 9161 image candidates as the training set, 266 images as the test set, and used the histogram of oriented gradients as the feature and finally got an acceptable precision rate.

The optical character recognition (OCR) part is the core of our procedure. For effectiveness and compatibility, we decided to adopt the Tesseract OCR engine which is the state of the art in OCR. Firstly, the Tesseract OCR engine is retrained by ten kinds of seven segment fonts to fit the requirement. Moreover, then all the image candidates are scaled to a standard width or height, and modified by morphological dilation. Lastly, the modified images are passed to the Tesseract OCR engine to get the final recognition result.

The final recall and precision rate was 68.84% and 61.87% respectively, which out performed the original OCR engine 64% and 59% respectively.

1.4 Data Analysis

To adjust the overall procedure and evaluate it, we need to build up a specific dataset for glucometers. After investigating all kinds of glucometers, we found several patterns for them:

1. The Largest digits on the screen of glucometers are the blood glucose readings, shown in Figure 1.2.

2. The blood glucoses have two kinds of measurement (mg/dl ranging from 40 to 500 or $mmol/l$ ranging from 2.2 to 28), shown in Figure 1.3.

3. The digits have two kinds of displays (seven segment display or dot matrix display), shown in Figure 1.4.

4. The digits and the background can be light and dark respectively (or opposite), shown in Figure 1.5.

5. The contrasts of most digits are not strong, shown in Figure 1.6.

6. The glucometers have time information, shown in Figure 1.7.

7. The glucometers have brands labels, shown in Figure 1.8.

8. Some glucometers may contain charts, shown in Figure 1.9.

9. Most glucometers do not have any backlit on the screen, shown in Figure 1.10.

The blood glucose readings are the only data that we need to recognize, but the glucometers will provide more information which distracts the recognition method (Pattern 6, Pattern 7, Pattern 8), so we need to segment the digits first (Pattern 1). The two measurements of blood glucose have a significant difference in values (Pattern 2), which means we do not need to recognize the measurements. The two kinds of displays (Pattern 3) cause an essential problem, Because the seven segment digits shared completely different features than the typical digits, even some popular OCR engines such as Tesseract cannot generate acceptable results. This will be discussed in both Region of Interest Detection component and Text Recognition component. Whether background is dark or light (Pattern 4) is also an essential problem, which

Figure 1.2: Example images

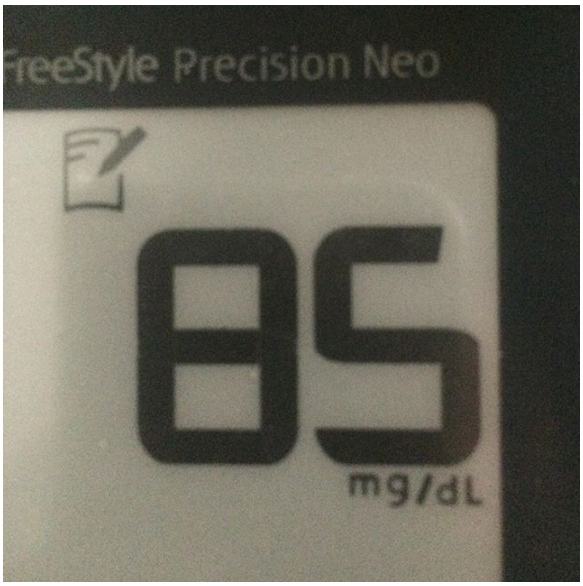
Figure 1.3: Example images with two kinds of measurement

(a) seven segment display

(b) dot matrix display

Figure 1.4: Example images with two kinds of displays

(a) the glucometer with dark background



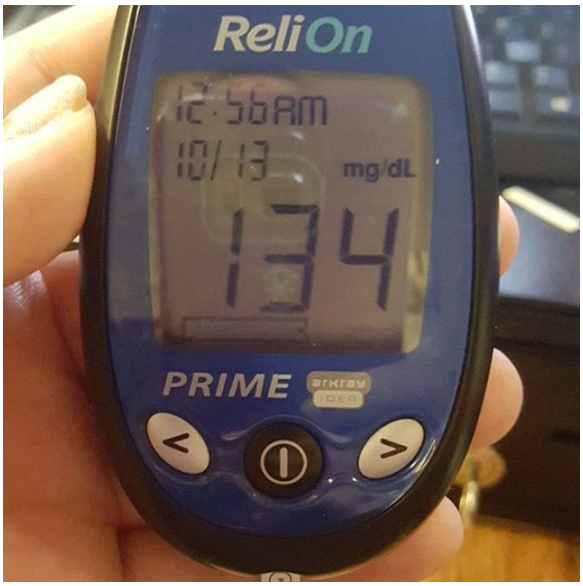
(b) the glucometer with light background

Figure 1.5: Example images with different background

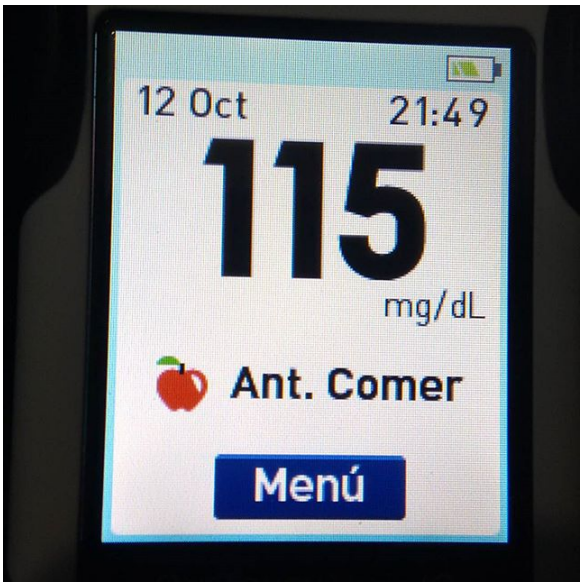
(a) Strong contrast

(b) Weak contrast

Figure 1.6: Example images with different contrasts



(a) 12-hour time format



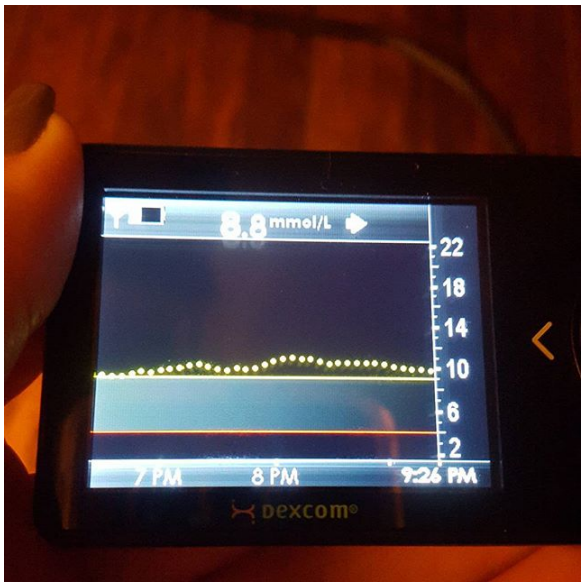
(b) 24-hour time format

Figure 1.7: Example images with time information

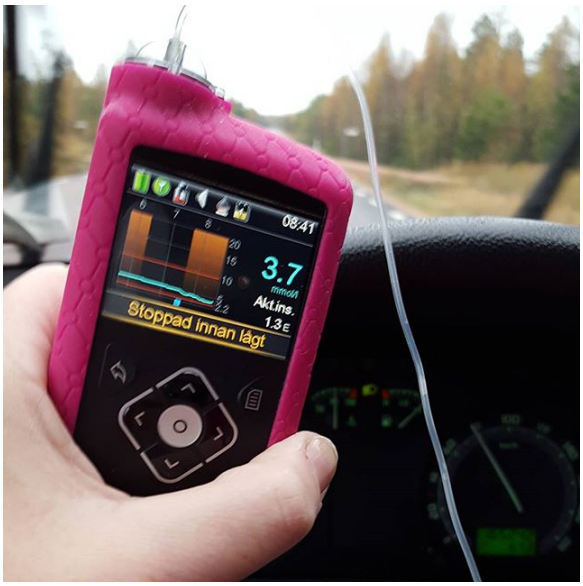
(a) the glucometer with top brand label

(b) the glucometer with bottom brand label

Figure 1.8: Example images with brand labels

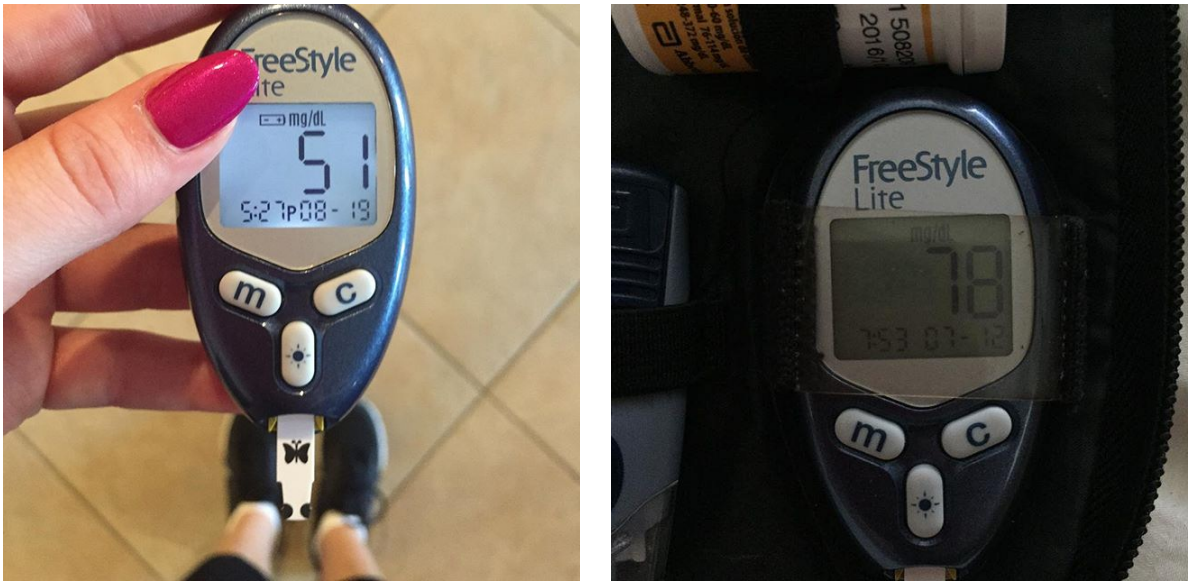


(a) the glucometer with line chart



(b) the glucometer with combined charts

Figure 1.9: Example images with charts



(a) the glucometer with backlit (b) the glucometer without backlit

Figure 1.10: Example images with or without backlit

will be discussed in Region of Interest Detection component. The digits are hard to be distinguished from the background (Pattern 5, Pattern 9). This problem will be discussed in Image Enhancement component.

To build up the dataset, we collected 266 glucometer images containing glucometers of all these nine patterns. Among them, 200 images were captured from the web, and 66 images are taken from cameras or mobile phones. We found out that 85.6% of these images used seven segment display and only 14.4% used dot matrix displays, 84.7% of these images are with light background and only 15.3% with dark background, 71.2% of these images are in measurement of mg/dl and 28.8% in measurement of $mmol/l$.

Among those images, we randomly chose 200 of them as training dataset and 66 of them as the test dataset. For the training dataset, it was used in experiments in Image Enhancement component and Region of Interest Detection component to find out the best parameters. Meanwhile, we sampled the 9161 sub-images and 266 sub-images from those images as the training dataset and testing dataset respectively for the support vector machine in region filtering part within the Region of Interest Detection component. For testing dataset, it was used to evaluate the performance of the overall procedure.

1.5 Thesis Outline

The thesis is organized as follows: Chapter 1 is the introduction which gives a brief view of our method. Chapter 2 discusses the image enhancement method that we adopt to improve the qualities of camera-taken images. Chapter 3 proposed the region of interest detection method along with the method to improve the precision. In Chapter 4, we introduce the Tesseract optical character recognition engine and our improvement on it to get better performance on our dataset. Chapter 5 is the conclusion chapter, and future works are also discussed in this chapter.

Chapter 2

Image Enhancement

In this chapter, we mainly discuss the component of image enhancement. Since the camera-taken images may have various qualities under different light sources and conditions, we adopted the combination of basic filtering algorithms as to improve their qualities, say homomorphic filter, median filter, modified histogram equalization, and morphological transform to achieve the best results. After parameter searching, we got the optimal parameters for our training images which were validated on test images.

In the first section, we define the problem and describe the challenges we face, and discuss the image enhancement techniques that we adopt. In the next section, we introduce the related work of three image enhancement techniques respectively. In the third section, we discuss our modification and implementation of these techniques. In the last section, we show the experiment result and the final determined parameters.

2.1 Problem Description

General Optical Character Recognition (OCR) engines can produce good results from clean documents. Specifically, those OCR engines require high-resolution, high-quality input images with a fairly simple structure (black text on a white background to give out a good recognition result). Unfortunately, for camera-based systems, these requirements are not typically met. Liang et al.[45] introduced 12 kinds of factors that affect the camera-based systems: low resolution, uneven lighting, perspective distortion, nonplanar surface, wide-angle-lens distortion, complex backgrounds, zooming and focusing, moving objects, intensity and colour quantization, sensor noise, compression, and lightweight algorithms.

These factors can be categorized into two types. The first type of factors is determined by hardware device conditions (low resolutions, wide-angle-lens, intensity and colour quan-

tization, sensor noise, compression, and lightweight algorithms). The second type of factors is introduced by the environment conditions (uneven lighting, perspective distortion, nonplanar surface, complex backgrounds, zooming and focusing, and moving objects). Nowadays, most device hardware conditions have been improved greatly with the development of digital cameras techniques, but still cannot overcome the problems introduced by the environment conditions.

For our system, the environmental conditions can be simplified to three categories: uneven lighting, complex backgrounds, zooming and focusing. Figure 2.1 shows four example images affected by these environment conditions. As image (a) shows, the weak light source will introduce Gaussian noises into the original image, and decrease the contrast between the background and the target object. The image (b) tells us that overexposure may also decrease the contrast between the target object and background while increase the contrast of the identical object. The image (c) gives us a sense that losing focus will blur the whole image. The image (d) shows the reflection effects, which introduce more information to the background. In this chapter, we focus on using image enhancement techniques to process the image with uneven lighting and zooming and focusing.

Image enhancement techniques, e.g. contrast adjustment, filtering, morphological filtering, and deblurring, can bring out the detail in an image that is obscured or highlight certain features of interest in an image. Those techniques are frequently used as a preprocessing step to improve the difference between the target object and the background for image segmentation. To generate proper source images for next step, We adopt a combination of image enhancement steps, including illumination normalization, contrast adjustment, and noise reduction.

2.2 Related Work

In this section, we introduce the related work of the most popular solutions for illumination normalization, contrast adjustment, and noise reduction. They are homomorphic filtering, histogram equalization, and median filtering, respectively.

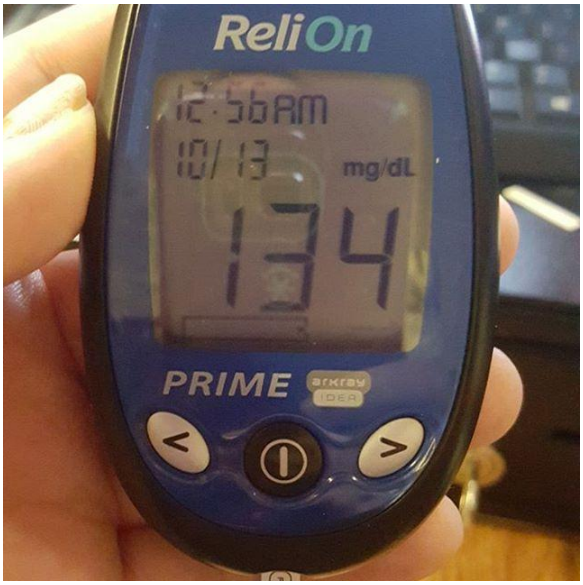
2.2.1 Illumination Normalization

Considering the problem we discussed above, one essential factor of the problem is the uneven lighting. According to Liang et al.[45]'s definition, uneven lighting is a common uncontrollable environment condition, due to both the physical environment (shadows, reflection, fluorescents) and uneven response from the devices, which generates poor illumination.

(a) weak-light-source

(b) over-exposure

(c) lose-focus



(d) reflection

Figure 2.1: Example images affected by the environment conditions

Further complications occur when the image generated from the picture taken under artificial light or the surface of the objects is reflective. As Fisher[22] found, if on-camera flash is used, the center of the view is the brightest, and then lighting decays outward.

Under this condition, the same uniform region will appear brighter on some areas or darker on others. This undesired situation will lead to several severe problem in computer vision based system. The pixels might be misclassified, leading to wrong segmentation results, and thus contribute to inaccurate evaluation or analysis from the system. Therefore, it is very crucial to process this type of image first before they are fed into the system. One of the popular methods used to enhance or restore the degraded images by uneven illumination is called homomorphic filtering.

Homomorphic filtering applies illumination-reflectance model to process the image. This model characterize the image as two elementary components. The first one is the amount of source illumination incident on the sight being viewed $i(x, y)$. The other is the reflectance component of the objects on the sight $r(x, y)$. The image $f(x, y)$ is then defined as [24, 55, 25]:

$$f(x, y) = i(x, y)r(x, y) \tag{2.1}$$

In this model, the intensity of $i(x, y)$ varies slower than $r(x, y)$. As a result, $i(x, y)$ is regarded to have more low frequency components than $r(x, y)$. Based on this fact, homomorphic filtering process targets to reduce the significance of $i(x, y)$ by decreasing the low-frequency components of the image. This can be obtained by performing filtering in the frequency domain. Since we want to process an image in the frequency domain, the requisite is to transform the image from spatial domain to frequency domain. This can be achieved by using transformation functions, such as Fourier transform. What is worth noticing is that before the transformation, logarithm function should be applied to change the multiplication operation of $r(x, y)$ with $i(x, y)$ in 2.2 into addition operation.

In general, homomorphic filtering process can be implemented through five stages, as described as follows:

1. Use a natural logarithm of both sides to decouple $i(x, y)$ and $r(x, y)$ elements:

$$z(x, y) = ln(i(x, y))ln(r(x, y)) \tag{2.2}$$

2. Apply the Fourier transform to transfer the image into frequency domain:

$$\Im\{z(x, y)\} = \Im\{ln(i(x, y))\} + \Im\{ln(r(x, y))\} \tag{2.3}$$

or

$$Z(u, v) = F_i(u, v) + F_r(u, v) \tag{2.4}$$

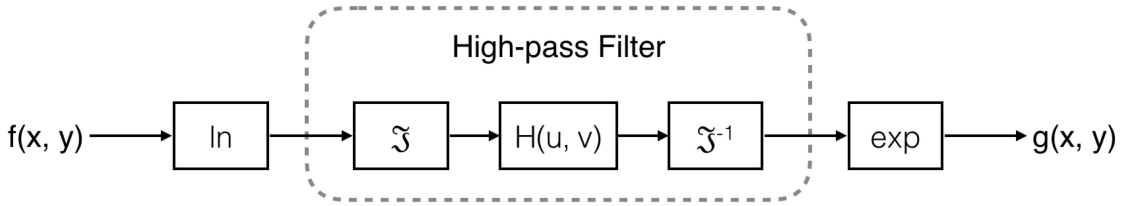


Figure 2.2: Homomorphic Filter Process

3. Use a filter function $H(u,v)$ to high pass the $Z(u,v)$ in frequency domain, and then achieve a filtered version $S(u,v)$ as:

$$S(u,v) = H(u,v)Z(u,v) = H(u,v)F_i(u,v) + H(u,v)F_r(u,v) \tag{2.5}$$

4. Perform an inverse Fourier transform to obtain the filtered image in the spatial domain:

$$s(x,y) = \mathfrak{F}^{-1}\{S(u,v)\} = \mathfrak{F}^{-1}\{H(u,v)F_i(u,v) + H(u,v)F_r(u,v)\} \tag{2.6}$$

5. After filtering, the enhanced image g(x,y) can be gained via:

$$g(x,y) = exp\{s(x,y)\} \tag{2.7}$$

According to the fundamental of homomorphic filtering, an essential issue is to choose an appropriate high pass filter. Saleh et al.[59] recommended several high pass filters for homomorphic filtering. Those high pass filters are mostly Gaussian high-pass filter or Butterworth high-pass filter or their derivatives and achieve various results in different situations (using different parameters).

The most popular filter for homomorphic filtering is the Gaussian high-pass filter. It has been presented in many image processing text books[24, 55, 25, 54, 68]. The character of this filter is to have circularly symmetric curve shape, which centred at $(u,v) = (0,0)$ coordinates in frequency domain. The transfer function for this filter is stated as:

$$H(u,v) = (\gamma_H - \gamma_L)\left[1 - exp\left\{-c\left(\frac{D(u,v)}{D_0}\right)^2\right\}\right] - \gamma_L \tag{2.8}$$

in which constant c has been defined to control the steepness of the slope, D_0 is the cut-off frequency, $D(u,v)$ is the distance between coordinates (u,v) and the center of frequency at $(0,0)$.

This filter requires three parameters to be set by the user. They are the high frequency gain γ_H, the low frequency gains γ_L, and the cut-off frequency D_0 respectively. If γ_H is a number greater than 1, and γ_L is a number lower than 1, the filter function is supposed to decrease the effect produced by the illumination (which contains mostly the low-frequency components) and amplify the effect produced by the reflectance (which contains most of the high-frequency components). In the end, the net result becomes a simultaneous dynamic range compression and contrast enhancement. The value of the low-frequency gain should be set such as $\gamma_L = 0.5$ which can be half the spectral energy of the illumination, and the value of high-frequency gain is set such as $\gamma_H = 2$ which can double the spectral energy of the reflectance components [24]. In [20], the value of c is recommended to be 0.5. Practically, all three parameter values are often chosen empirically, and there is no fixed approach to determine exact values for these parameters.

The Butterworth high-pass filter has been previously mentioned in the work of [2]. It assessed the performance of homomorphi filter by comparing it with some other advanced image enhancement method. Two equations have been introduced for homomorphic filtering process in that paper:

$$H(u,v) = (\gamma_H - \gamma_L)\Big[1 - exp\Big\{ - a\Big(D(u,v)\Big)\Big\}\Big] - \gamma_L \tag{2.9}$$

$$H(u,v) = (\gamma_H - \gamma_L)\left[1 - \frac{1}{1 + \Big[D(u,v)/a\Big]^n}\right] - \gamma_L \tag{2.10}$$

Equation 2.9 is the Gaussian high-pass filter, and Equation 2.10 is the modified Butterworth high-pass filter. In Equation 2.10, the term $D(u,v)/a$ decides the transition point and Butterworth power, parameter n indicates the steepness of the transition slope. Experimentally, when applying the homomorphic filtering process onto different images, the author discovered that a $(\gamma_H - \gamma_L) \geq 1.5$ maximal amplification value may be too much for many images. Furthermore, it is also found that the modified high-pass Butterworth equation is better performed than Gaussian high-pass filter for homomorphic filtering process as it permits a separate setting of the transition point from the transition slope.

A simple modification to the Butterworth high-pass filter has been proposed in [15]. This modification dramatically improves the results for face recognition tasks. The modified Butterworth equation applied by it is shown as following:

$$H(u,v) = \frac{1}{1 + \Big[D_0/D(u,v)\Big]^{2n}} \tag{2.11}$$

The recommended value of D_0 is 0.25 and n is 1.

Another homomorphic filter equation has been applied in [55]. This filter take a linear function as the high-pass filter. The transfer function of this filter is defined as:

$$H(u,v) = \frac{1}{1 + exp\Big\{ - a\Big(D(u,v) - D_0\Big)\Big\}} + A \tag{2.12}$$

From this equation, the high frequency gain and the low frequency gain are given as:

$$\gamma_H = 1 + A \text{ and } \gamma_L = \frac{1}{1 + exp\{aD_0\}} + A \tag{2.13}$$

The recommended values for this transfer function are $a = 1$, $D_0 = 128$, and $A = 10$.

2.2.2 Contrast Adjustment

Contrast adjustment is a process involving changing the pixels' intensity of the input image so that the output image should subjectively look better [24]. The purpose of contrast adjustment is to improve the interpretability or perception of information contained in the image for human viewers or to provide a "better" input for other automated image processing systems. Many contrast adjustment methods have been proposed. A very popular technique for contrast adjustment is histogram equalization (HE). This technique is commonly employed for contrast adjustment because of its simplicity and comparatively better performance on almost all types of images.

The operation of Standard Histogram Equalization (SHE) is performed by remapping the gray levels of the image based on the probability distribution of the input gray levels. It flattens and stretches the dynamic range of the image's histogram and resulting in overall contrast enhancement [38].

For a given image X, the probability density function $p(X_k)$ is defined as

$$p(X_k) = \frac{n^k}{n} \tag{2.14}$$

For $k = 0, 1 \ldots L - 1$, where represents the number of times that the level (appears in the input image X and n is the total number of samples in the input image. Note that $p(X_k)$ is associated with the histogram of the input image which represents the number of pixels that have a specific intensity. In fact, a plot of vs. is known histogram of X. Based on the probability density function, the cumulative density function is defined as

$$c(X) = \sum_{j=0}^{k} pX_j \tag{2.15}$$

Where $X_k = x$, for $k = 0, 1 \ldots L-1$. Note that $c(X_{L-1}) - 1$ by definition. SHE is a scheme that maps the input image into the entire dynamic range (X_0, X_{L-1}), by using the cumulative density function as a transformation function. The transformation function $f(x)$ based on the cumulative density function can be defined as

$$f(x) = X_0 + (X_{L-1} - X_0)c(x) \tag{2.16}$$

Then the output image of the SHE, $Y = \{Y(i, j\}$, can be expressed as

$$Y = f(X) \tag{2.17}$$

$$Y = \{f(X(i, j)|\forall X(i, j) \in X)\} \tag{2.18}$$

The high performance of the SHE in enhancing the contrast of an image as a consequence of the dynamic range expansion. Besides, SHE also flattens a histogram. Base on information theory, the entropy of message source will get the maximum value when the message has uniform distribution property [72]. As addressed previously, SHE can introduce a significant change in brightness of an image, which hesitates the direct application of SHE scheme in consumer electronics.

Standard histogram equalization(SHE) is subjected to major defects especially when implemented to process digital images.

Firstly, SHE transforms the histogram of the original image into a flat uniform histogram with a mean value that is in the middle of the gray level range. Therefore, the mean brightness of the output image is always at the middle (or close to it under discrete implementation) regardless of the mean of the input image. For images with high and low mean brightness values, the image outlook can have a significant change in the price of enhancing the contrast.

Secondly, SHE enhances the image based on the global content and in its discrete version large bins cannot be broken and redistributed to generate the desired uniform histogram. In other words, histogram equalization is useful in highlighting the borders and edges between different objects. However the local details may decrease within these objects, especially smooth and small ones. Another consequence of this change between large and small bins is the production of over-enhancement and saturation artifacts [32].

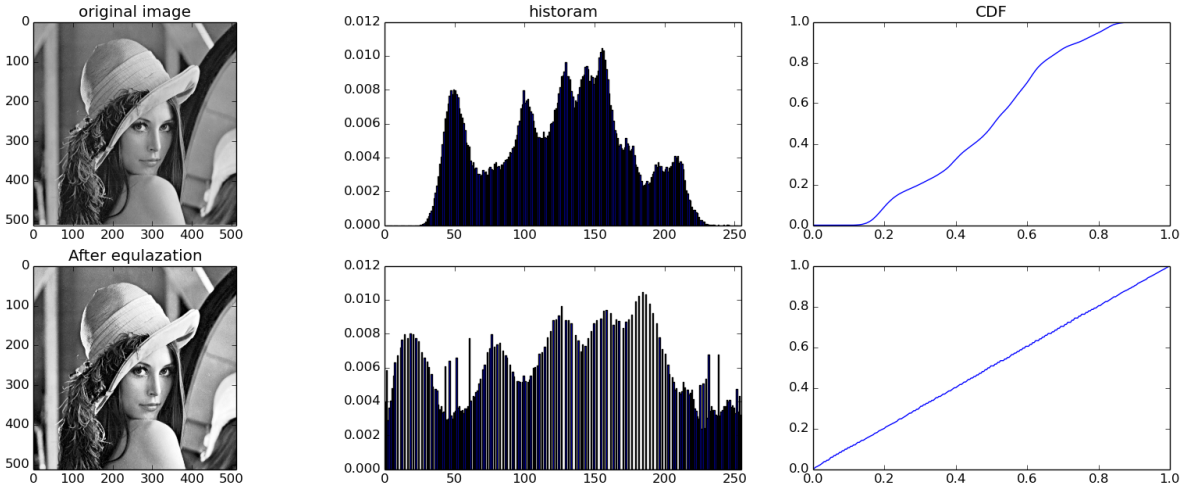


Figure 2.3: Example SHE images

There are two categories of research for solving those two defects. One is concentrating on the first drawback by preserving the mean brightness of the image while increasing the contrast, the other approach is trying to solve the second drawback by increasing the contrast as much as possible with introducing the least noises.

The typical method for the first group is trying to divide the input image histogram into different parts, and each part is equalized independently. For example, mean preserving bi-histogram equalization (BBHE) [74], equal area dualistic sub-image histogram equalization (DSIHE) [42] and minimum mean brightness error bi-histogram equalization (MMBEBHE) [61, 41].

BBHE divides the input image histogram into two parts based on input mean. Each part is then equalized independently. This method tries to overcome the brightness preservation problem. DSIHE method applies entropy value for histogram separation. MMBEBHE is the extension of BBHE method that provides maximal brightness preservation. Though these methods can conduct good contrast enhancement, they also cause more annoying side effects depending on the variation of gray level distribution in the histogram [60]. Recursive Mean-Separate Histogram Equalization (RMSHE) [61] is another improvement of BBHE. However, it also does not avoid side effects [1].

To get rid of the side effects of HE, Dynamic Histogram Equalization (DHE) technique takes control over the effect of conventional HE so that it enhances the image without losing any details in it. DHE partitions the image histogram based on local minimal and assigns specific gray level ranges for each partition before equalizing them separately. These partitions go

through a repartitioning test to determine whether any dominating portions have been missing. This method outperforms other present approaches by enhancing the contrast well without introducing noticeable side effects, such as washed out appearance, checkerboard effects, etc., or undesirable artifacts [1]. The brightness preserving dynamic histogram equalization (BPDHE), which is an extension to DHE that can generate the output image with the mean intensity almost the same as the mean intensity of the input, therefore satisfying the requirement of keeping the mean brightness of the image [31].

For the second group, the common approach is to divide the original image into several grids and apply histogram equalization separately. One popular solution named Adaptive Histogram Equalization (AHE) was proposed in [36] and [28]. In its simplest form, each pixel is transformed based on the histogram of a square surrounding the pixel, as in Figure 2.4.

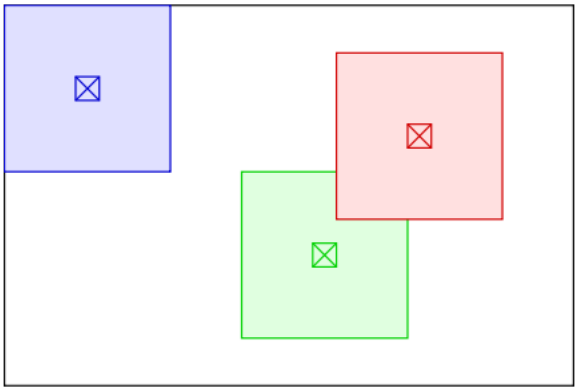


Image source: *https : //en.wikipedia.org/wiki/Adaptive_histogram_equalization*

Figure 2.4: Examples for Adaptive Histogram Equalization

However, AHE may over-amplify noise in relatively homogeneous regions of an image. A variant of adaptive histogram equalization called Contrast Limited Adaptive Histogram Equalization (CLAHE) was created[3] to prevent this by limiting the amplification. In the case of CLAHE, the contrast limiting procedure is applied for each neighbourhood from which a transformation function is derived. CLAHE was developed [56] to overcome the over-amplification of noise that adaptive histogram equalization can give rise to.

As in AHE, the contrast amplification in the vicinity of a given pixel value is determined by the slope of the transformation function. This is proportional to the slope of the neighbourhood cumulative distribution function (CDF) and therefore to the value of the histogram for that pixel value. CLAHE restricts the amplification by clipping the histogram at a predefined value before calculating the CDF. This restricted the slope of the CDF and the transformation function. The value at which the histogram is clipped, the so-called clip limit, relies on the normalization of

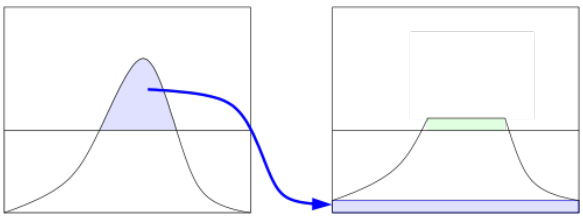


Image source: *https* : *//en.wikipedia.org/wiki/Adaptive_histogram_equalization*

Figure 2.5: Example for Contrast Limited Adaptive Histogram Equalization

the histogram and thereby on the size of the neighbourhood area. Common values limit the resulting amplification to between 3 and 4.

It is advantageous not just to discard the part of the histogram value that more than the contrast limit but to redistribute it equally among all histogram bins [56], as Figure 2.5 shows. The redistribution will push some bins over the clip limit again (region shaded green in the figure), resulting in an effective clip limit that is larger than the prescribed limit and the exact value of which depends on the image. To avoid this, the redistribution procedure can be repeated recursively until the excess is negligible.

2.2.3 Noise Reduction

Noise is the result of errors in the image acquisition process that result in pixel values that do not reflect the true intensities of the real scene. Because all factors mentioned above as well as former steps of image enhancement methods can introduce noises. We adopted another important step which was the noise reduction.

According to the research of Farooque et al.[21], there are certain types of noises that can be introduced into images. Those noises include Gaussian noise, salt-and-pepper noise (impulse noise), Poisson noise, shot noise, speckle noise, quantization noise, film grain, and anisotropic noise. Regarding the input images for this component, which is the result of image enhancement, Gaussian noises, and salt-and-pepper noises appear most frequently.

To eliminate these two noises, several noise reduction algorithms have also been introduced in [21]. Among these algorithms, we use the median filter to eliminate the two noises in our images for its effectiveness and simplicity.

The standard median filter (SMF) [6] is a simple rank selection filter which is also called as median smoother. SMF was introduced by tukey in 1971 that devotes to get rid of impulse noise by changing the luminance value of the center pixel of the filtering window with the median of the luminance values of the pixels contained within the window. The median filter is simple

and performs noise removal reasonably; it also removes thin lines and blurs image details even at low noise densities. The filtered image $S = \{S(i, j)\}$ from SMF can be interpreted by the Equation 2.19:

$$S(i, j) = Median(k, l) \in W_{m,n}\{D(i + k, j + l)\} \tag{2.19}$$

Where $W_{m,n}$ is a sliding window of size $m \times n$ pixels centered at coordinates (i, j). The median value is determined by using equation 2.19 with $n_s = m \times n$. Even though SMF can significantly decrease the level of corruption by impulse noise, uncorrupted pixel intensity values are also changed by SMF. This unexpected situation occurs because SMF is not capable of differentiating between uncorrupted from corrupted pixels. Moreover, SMF needs a large filter size when there is a high corruption level. However, a large filter of SMF will bring a significant distortion into the image [75].

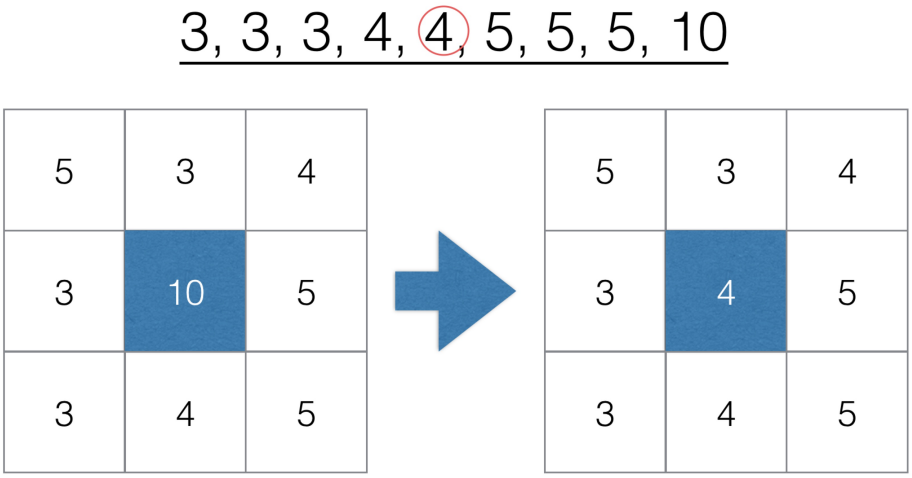


Figure 2.6: Standard Median Filter

It is worth noting that equation 2.19 is typically using a sorting algorithm such as quicksort or bubble-sort to organize the samples in ascending or descending order. Although sorting algorithm can be easily implemented, sorting procedure needs long computational time when $W_{m,n}$ is a large filter because the number of samples (i.e., $n_s = m \times n$) is large. Therefore, to avoid from using any sorting algorithm directly, local histograms have been applied for median value calculation. The required time for forming local histogram can be shortened by using a method proposed by Huang et al. [27], in which instead of updating $m \times n$ samples, only $2m$ samples need to be updated in each sliding-iteration.

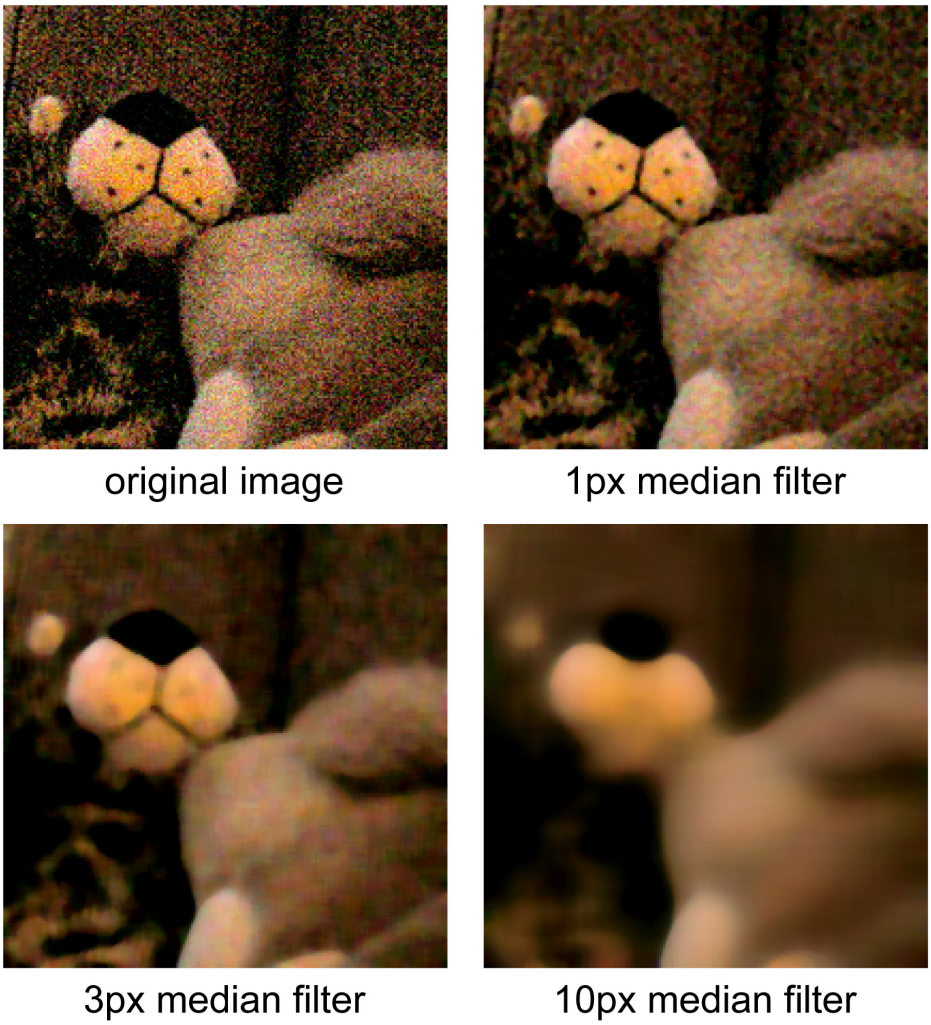


Image source: *https* : *//en.wikipedia.org/wiki/Median_filter*

Figure 2.7: Standard Median Filter Examples

A much wider range of algorithms selection can be found to filter the digital images from the Gaussian and impulse noise, besides Standard Median Filter(SMF).

Justusson introduced Weighted median filter(WMF) as one of the branches of median filter in 1981, and further improved by Brownrigg. The works involved in WMF are similar to SMF, apart from that WMF has weight associated with each of its filter element. These weights correspond to the number of sample duplications for the calculation of median value. The filtered image $S = \{S(i, j)\}$ from WMF can be denoted by the following equation [75]:

$$S(i, j) = Median(k, l) \in W_{m,n}\{W_{m,n}(k, l) \otimes D(i + k, j + l)\} \tag{2.20}$$

Where operator $\otimes$ indicates repetition operation. The median value is calculated based on equation 2.19 with n_s is equal to the total of $W_{m,n}(k, l)$. Normally, the filter weight $W_{m,n}$ is set such that it will reduce when it is located away from the center of the filtering window. This allows the filter give more emphasis to the central pixel and thus improve the noise suppression ability while preserving image details [47, 43, 40, 65]. However, the performance of the weighted median filter in preserving image details is highly relied on the weighting coefficients, and the nature of the input image itself. Unfortunately, in practical conditions, to find the suitable weighting coefficients for this filter is tough and requires high computational time when the weights are large [3, 5, 4]. There are two popular extensions of WMF; Central Weighted Median Filter (CWMF) [40] and Adaptive Weighted Median Filters (AWMF)[52].

Directional median filter (DMF), or so-called stick median filter, works by separating its 2-D filter into several 1-D filter components [17, 29, 13]. Each filter component or stick, presented as a straight line, is related to a certain direction or $angle\theta$. For a window of size $m \otimes n$ pixels, there are $m + n - 2$ sticks that will be applied. The calculated median values from these 1-D filters are then gathered to achieve the final result. In [13], the output intensity is defined as:

$$S(i, j) = max\{Median(k, l) \in W_\theta\{D(i + k, j + l)\}\} \tag{2.21}$$

Where W_θ is the stick. The output intensity is defined as the largest median value determined at each location. For images with randomized noise, it achieves near the same result as SMF while reduces the speed.

The Adaptive Median Filter(AMF) is designed to solve the problems encountered SMF. The fundamental difference between the two filters is that, in the Adaptive Median Filter, the size of the window surrounding each pixel is not fixed. This variation relies on the median of the pixels in the current window. If the median value is an impulse, then the size of the window

is extended. Otherwise, further processing is carried out on the part of the image within the current window specifications. Thus, AMF solves both the purpose of removing the noise from the image and the purpose of reducing distortion in the image. AMF is capable of handling the filtering operation of an image corrupted with impulse noise of probability greater than 0.2. This filter also smoothens out other types of noise, and produces better output images than SMF.[24] However, this method is very time-consuming, because it needs additional operations on every pixel of the image.

One of the conventional median filtering approaches recently is the Switching Median Filter(SWMF), or also known as the decision-based median filter. This approach has been used in recent works, such as [51, 26, 34, 30]. Switching median filter attempts to minimize the unwanted alteration of uncorrupted pixels by the filter. Thus, to solve this problem, switching median filter evaluates each input pixel whether it has been corrupted by noise or not. Then it alters only the intensity of noisy pixel candidates while leaving the other pixels unchanged. Usually, switching median filter is built from two stages. The first stage is for noise detection, and the second stage is for noise reduction. The output of the noise detection stage is a noise mask M. This mask is a binary mask. Next, mask M will be applied in the noise reduction stage, where only pixels with $M = 1$ are processed by the median filter. For the computation of median, only "noise-free" pixels (i.e., pixels with $M = 0$) are chosen as the sample. The noise detection method is crucial, if cannot detect the noise properly, this method will have no difference with SMF.

2.3 Our Enhanced Methods

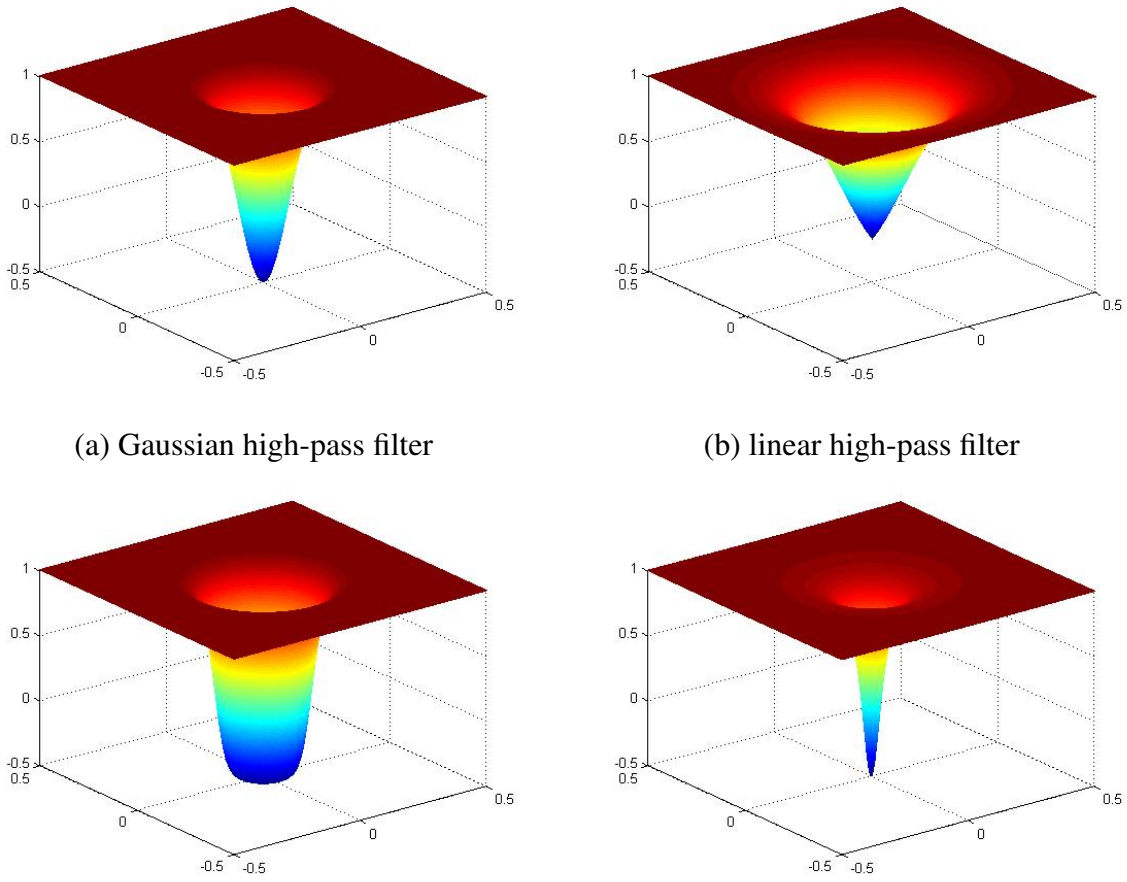
In this section, we introduce which of the image enhancement methods we chose for our scenarios. The final decision of these methods is the homomorphic filtering with the Gaussian high pass filter, the contrast limited adaptive histogram equalization, and the standard median filter.

2.3.1 Homomorphic Filtering

In this step, we desired to use homomorphic filtering to solve the problem of over-exposure and reflection. As previous work showed to us, we have several high pass filter to choose: the Gaussian high-pass filter, the Butterworth high-pass filter, the modified Butterworth high-pass filter, and the linear high-pass filter.

We implemented all those high-pass filters and process homomorphic filtering on the test images with those filters respectively. Figure 2.8 shows the 3D-plot images of those filters, and

Figure 2.9 displays the result images generated by homomorphic filtering with those filters.



(a) Gaussian high-pass filter

(b) linear high-pass filter

(c) Butterworth high-pass filter

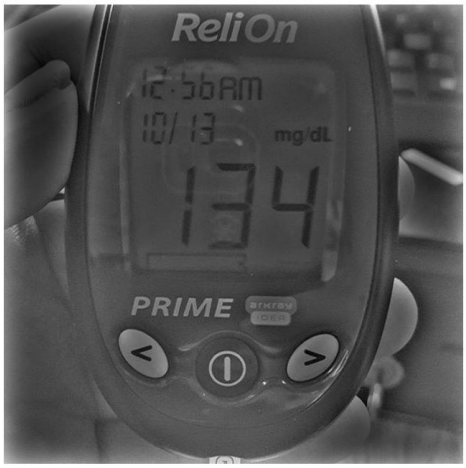
(d) modifed Butterworth high-pass filter

Figure 2.8: Example 3D-plot images of different high-pass filters

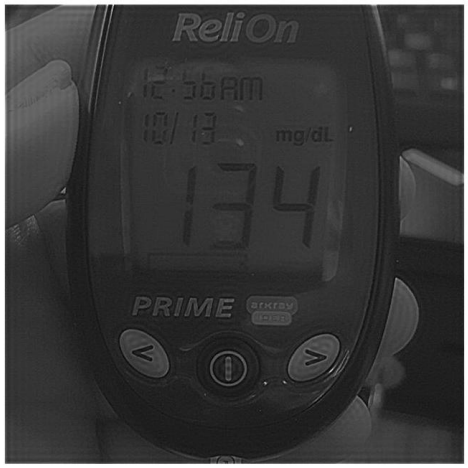
In Figure 2.9, image (b) has introduced other noises into the original image, image (c) has eliminated too much low-frequency information which looks like an edge image, and image (d) eliminated too little low-frequency information which will do little help with illumination normalization. Therefore, image (a) is the best result image.

According to the experiment result, we finally chose the Gaussian high pass filter as our homomorphic filter. Figure2.10 shows the result of our homomorphic filter, we set the parameter as $\gamma_L = 0.25$, $\gamma_H = 2$, and $D_0 = 80$.

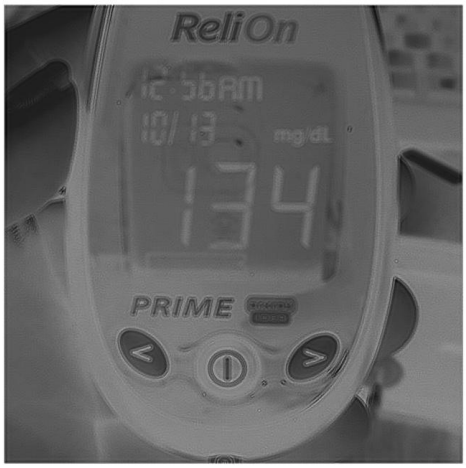
From the result images, we can see that the digits became clearer after processing, but the contrast is slightly decreased. The next step will fix this problem.



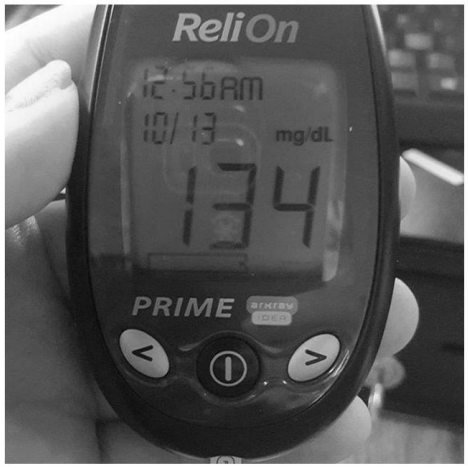
(a) Gaussian high-pass filter



(b) linear high-pass filter



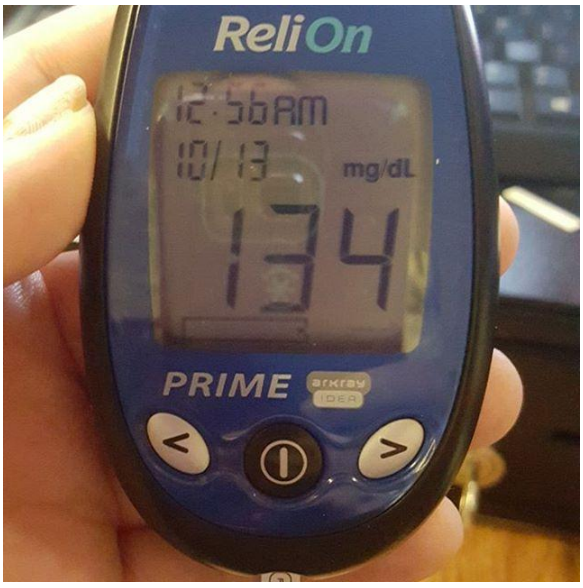
(c) Butterworth high-pass filter



(d) modifed Butterworth high-pass filter

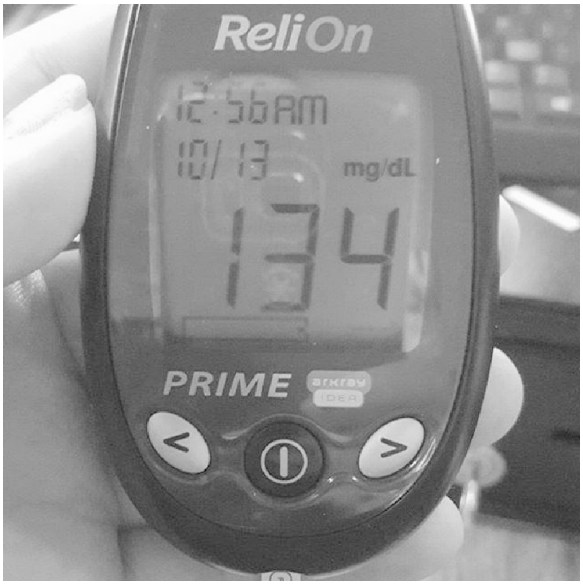
Figure 2.9: Result images of homomorphic filtering with different high-pass filters

(a) original over-exposure image



(b) original reflection image

(c) processed over-exposure image



(d) processed reflection image

Figure 2.10: Test images processed by homomorphic filtering

2.3.2 Histogram Equalization

In this step, we desire to use histogram equalization to enhance the image with weak-light-source and the post processed image by homomorphic filtering. Figure 2.11 are the result images generated by different histogram equalization method. Image (b) is the test image processed by the Standard Histogram Equalization (SHE). It suffers from the two drawbacks of SHE. The background's contrast has been improved significantly while the target digit is still not clear enough. Image (c) is the test image processed by the Brightness Preserving Dynamic Histogram Equalization (BPDHE), to preserve the mean brightness, the target digits get corrupted by the background. Image (d) is the test image processed by the Contrast Limited Adaptive Histogram Equalization (CLAHE). The contrast of target digits, as well as background, has been improved significantly, even though this method introduces a little more noises.

Considering all the result in Figure 2.11, we decide to use the CLAHE as our contrast adjust method. Figure 2.12 shows the result of our CLAHE method. We set the clip limit to 10 and window size as 40. As the same as the test image, there are more noises. We will eliminate them as much as possible at next step.

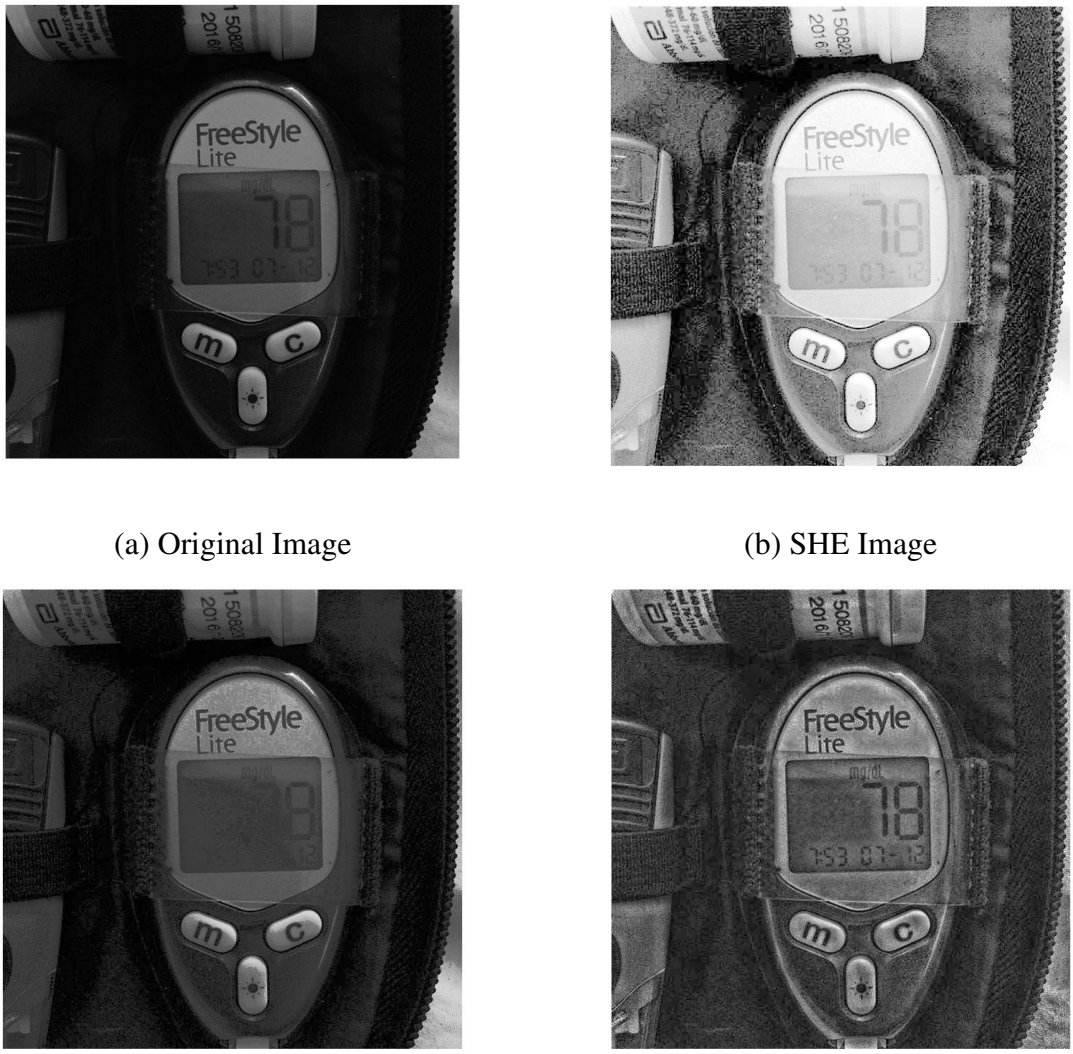
2.3.3 Median Filter

The original images may contain various noises, and former steps also introduce extra noise. These conditions increase the hardship of adopting advanced median filter. Figure 2.13 shows the result of different median filters applied on test images. Image (b) is processed by the Standard Median Filter(SMF). Image (c) is processed by the Adaptive Median Filter(AMF). Image (d) is processed by the Switching Median Filter (SWMF). From the result images, we can see that there is no significant difference between the result of these methods. For speed concern, we decide to choose the SMF.

2.4 Experiment Results and Summary

Since evaluating the performance of each step of image enhancement component is very hard, we evaluated those steps by evaluating the result in the region of interest detection component. The evaluate method is as follows.

The output of region of interest detection component is a set of rectangular designating regions for detected digits. We call this set the estimate set. A set of ground truth regions, we call it the target set is provided in the dataset.



(a) Original Image

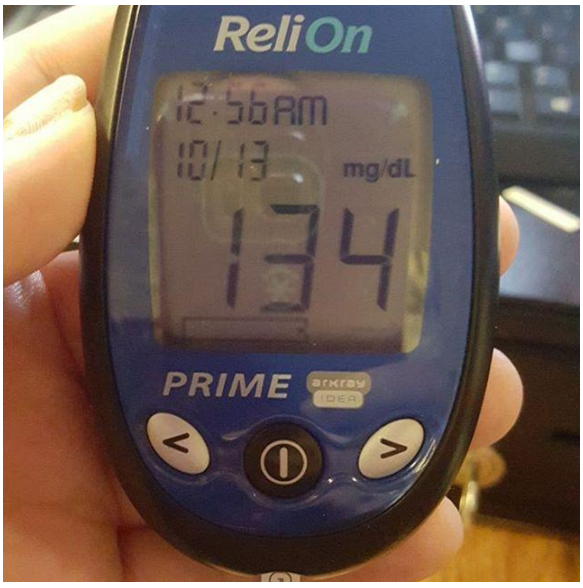
(b) SHE Image

(c) BPDHE Image

(d) CLAHE Image

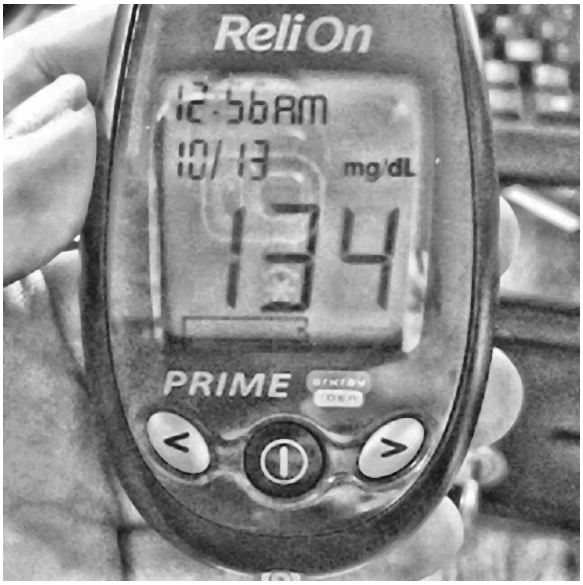
Figure 2.11: Examples of different histogram equalization method

(a) original over-exposure image



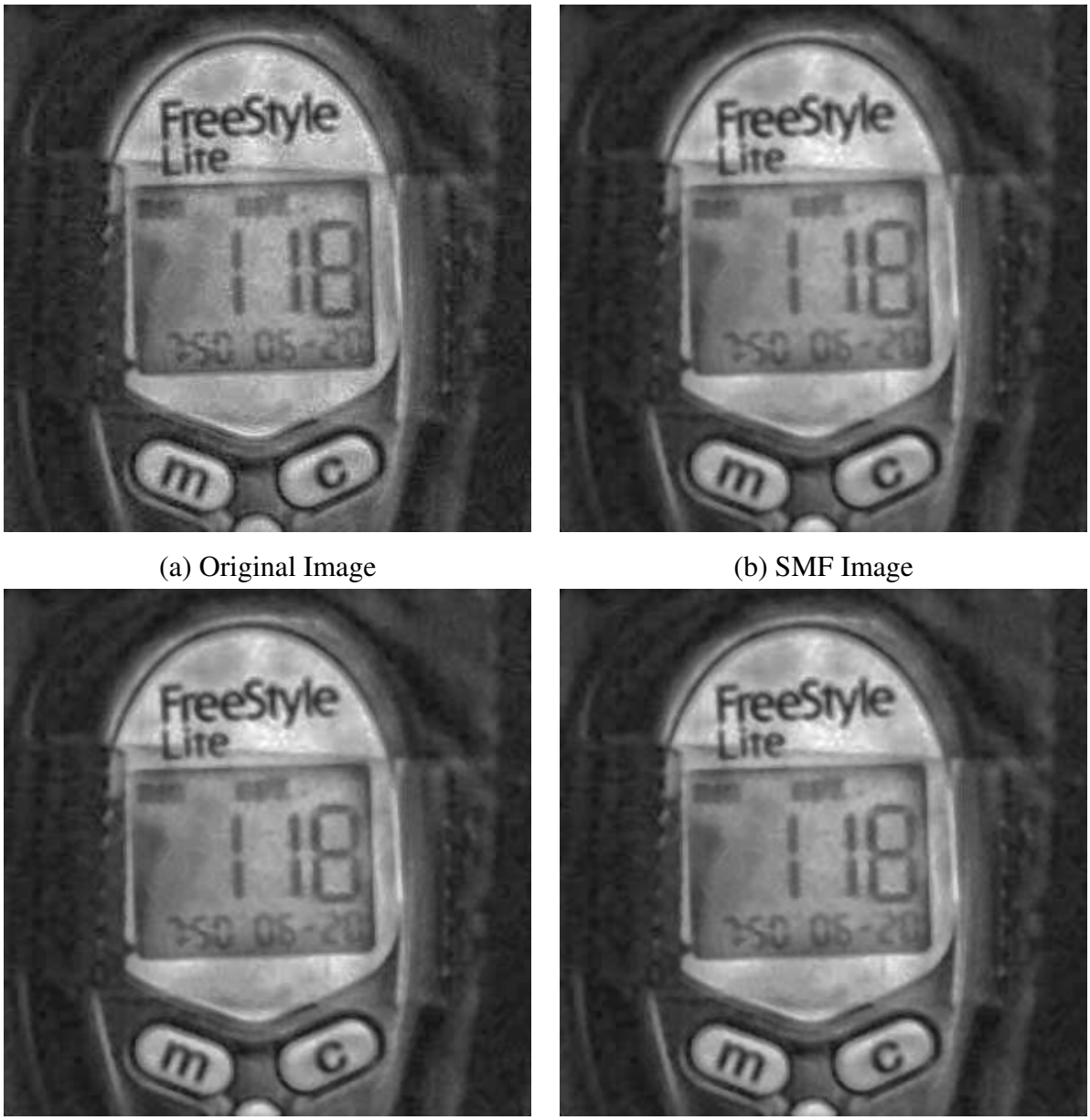
(b) original reflection image

(c) processed over-exposure image



(d) processed reflection image

Figure 2.12: Test images processed by histogram equalization method

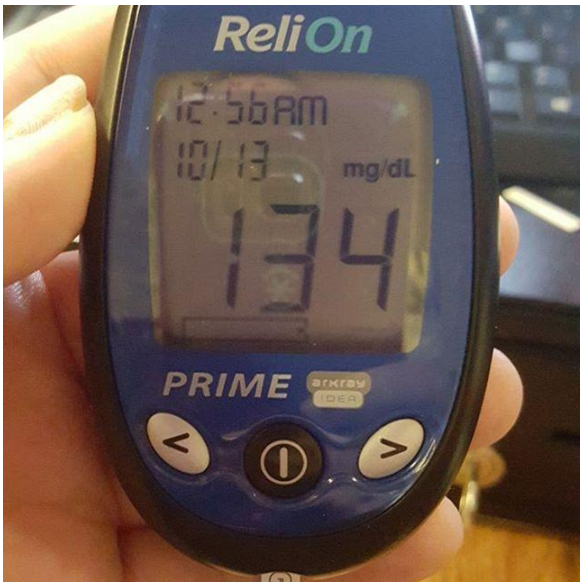


(a) Original Image (b) SMF Image

(c) AMF Image (d) SWMF Image

Figure 2.13: Test images processed by the different median filters

(a) original over-exposure image



(b) original reflection image

(c) processed over-exposure image

(d) processed reflection image

Figure 2.14: Test images processed by median filter

The matchup between two regions is defined as the area of intersection divided by the area of their unions. This value will be one for the identical region and zero for regions that have no intersection. For each estimated region, the closest match was found in the set of targets, and vice versa. Hence, the best match $m(r;R)$ for a region r in a set of regions R is defined by Formula 2.22.

$$m(r;R) = max\{m_p(r;r_0)|r_0 \in R\} \tag{2.22}$$

Meanwhile, the Precision and Recall can be defined as Formula 2.23 and Formula 2.24 respectively, where T and E are the sets of ground-truth and estimated regions respectively.

$$Precision = \frac{\sum_{r_e \in E} m(r_e;T)}{|E|} \tag{2.23}$$

$$Recall = \frac{\sum_{r_t \in T} m(r_t;E)}{|T|} \tag{2.24}$$

After we decided all the image enhancement steps and the evaluation methods, we performed the image enhancement component with or without a certain step combined with the region of interest detection component. The tables and figures below show all the recall rates of the procedures. Inside the table, the titles of "y1y2", "y1n2","n1y2" , and "n1n2" refer to "with both homomorphic filtering and histogram equalization", "with only homomorphic filtering", "with only histogram equalization", and "without both" respectively. Each row refers to a different window size of median filters.

median	y1y2	y1n2	n1y2	n1n2
5	0.534878	0.532892	0.516189	0.553113
9	0.502834	0.436243	0.448298	0.435315
13	0.432527	0.357313	0.397098	0.394581
17	0.35281	0.3001	0.329823	0.309849
21	0.31917	0.221021	0.316837	0.263065
25	0.238333	0.161826	0.281278	0.192066

Table 2.1: Performances with or without certain steps

In Table 2.1, the highest value is the procedure without both. However, when we checked the Figure 2.15, the plot of "y1y2" keeps the highest on average, under each window size of the median filter. Since we have not adjusted the parameter for the next step, the highest on average will be the most acceptable one. We also select 3 as the median filter window size.

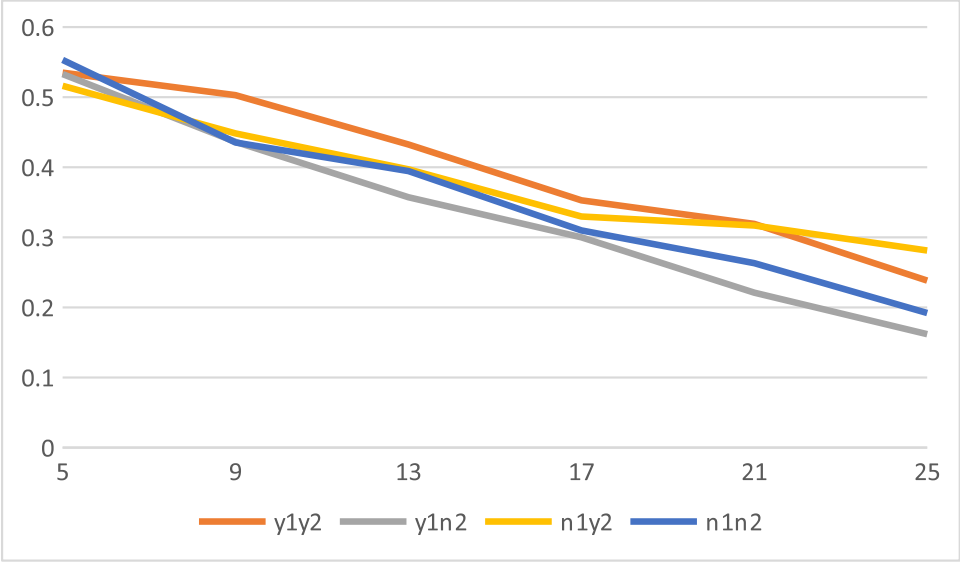


Figure 2.15: Performances with or without certain steps

Chapter 3

Region of Interest Detection

In this chapter, we propose the region of interest detection component. Typical natural scene images tend to have lots of useless and unpredictable objects, which will largely affect the result of OCR engines. Thus, we need to segment the region of interest, i.e. the digits. We adopted the procedure named Stroke Width Transform, owing to its output enables fast and dependable detection of text. We adjust the filtering and the chaining part of the algorithm to get a higher recall rate for our dataset and use a Linear support vector machine to improve the precision rate.

In the first section, we define the problem and describe the challenges we face. We propose the related work of region detection algorithm and region filtering method, in the second one. In the third section, we introduce our enhanced method of region detection and filtering. At last, we show the experiment result and the final determined parameters.

3.1 Problem Description

As the previous chapter discussed, to detect the text in camera-taken images is much harder than dealing with scans of printed pages, faxes and business cards. Liang et al.[45] introduced 12 kinds of factors that affect the camera-based systems: low resolution, uneven lighting, perspective distortion, nonplanar surface, wide-angle lens distortion, complex backgrounds, zooming and focusing, moving objects, intensity and colour quantization, sensor noise, compression, and lightweight algorithms. Among those factors, apart from those determined by hardware device conditions, the rest of them are introduced by the environment conditions, including uneven lighting, perspective distortion, nonplanar surface, complex backgrounds, zooming and focusing, and moving objects. The environment conditions for our system can be simplified to three categories: uneven lighting, complex backgrounds, zooming and focusing.

In the previous chapter, we solved the problems caused by uneven lighting, zooming and focusing. In this chapter, we will deal with the issue of complex background which is the most difficult part of all text detection method. For traditional OCR engines, even the state-of-art ones, it is hard to recognize the text directly. There are several reasons for this. First, the majority of OCR engines are designed for scanned text and so depend on segmentation which correctly separates text from background pixels, and this is usually simple for scanned text with pure white background and black text. Second, camera-taken images contain way more information than the scanned text images which is hard for OCR to distinguish. Finally, while the page layout for traditional OCR is simple and structured, in camera-taken images, it is much harder, because there is far less text, and there exists less overall structure with high variability both in geometry and appearance. When we apply the OCR to our test images, the success rates drop drastically, as shown in Figure 3.1.

Under these circumstances, limiting the information that passed to OCR engines becomes the most important part. To accomplish this goal, first we can find suitable regions of interest which divide original images into small image candidates. We then filter the image candidates under some certain conditions, make sure the filtered ones are text regions. Finally, we pass the text regions to the OCR engine to do the recognition. This process is what we will propose in the next two sections.

3.2 Related Work

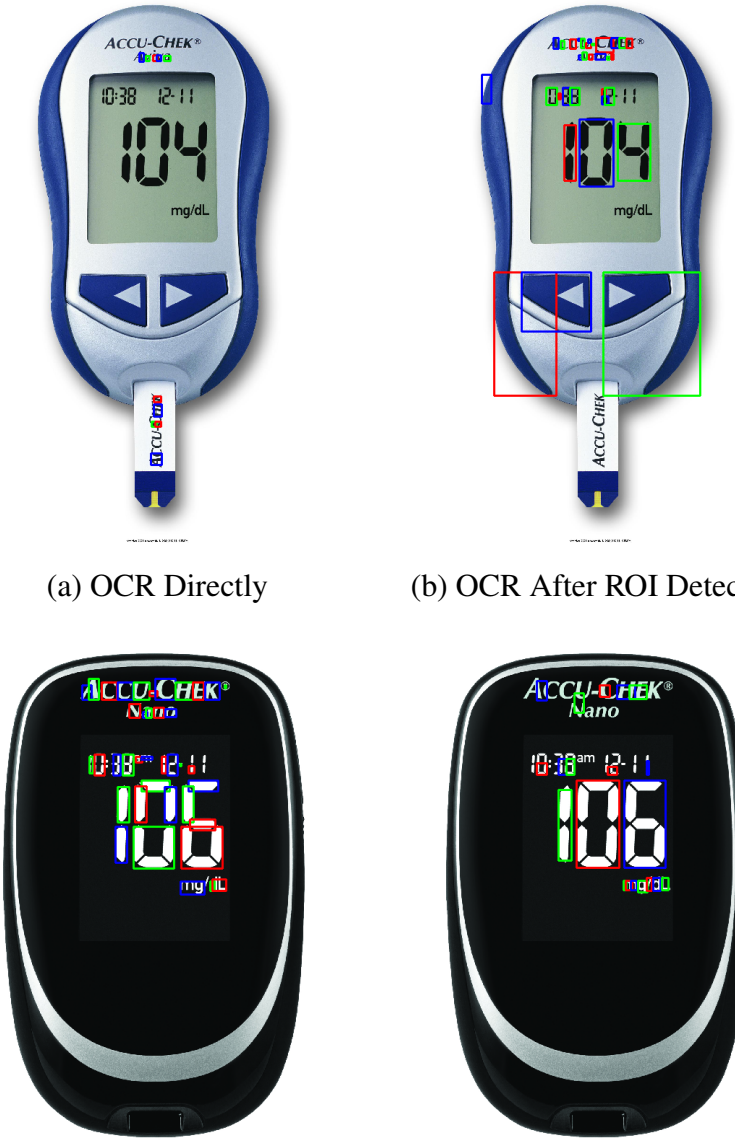
In this section, we introduce the related work of the most popular solutions on text region detection and text region filtering.

3.2.1 Text Region Detection

In this section, we desire to detect all possible text regions for input images, which means we are not generating one correct region but returning the correct region in a group of regions. For our dataset with labeled bounding box, we are focusing on generating the recall rate as high as possible of the overall process.

In general, the methods for detecting text regions can be broadly categorized into two groups: texture- based methods and region-based methods.

Texture-based [11, 46, 23, 44, 77] methods scan the image at different scales, classifying neighbourhoods of pixels based on some specific text properties, such as high density of edges, low gradients above and below text, a significant variance of intensity, distribution of wavelet or DCT coefficients, etc.



(a) OCR Directly (b) OCR After ROI Detection

(c) OCR Directly (d) OCR After ROI Detection

Figure 3.1: Examples of applying OCR directly or after ROI Detection

Another group of text detection algorithms is based on regions [33, 37, 76]. In these methods, pixels exhibiting certain properties, such as approximately constant colour, are grouped together. The resulting connected components (CCs) are then filtered geometrically and using texture properties to exclude CCs that certainly cannot be letters.

For this step, we decided to use region-based methods, because the texture-based methods have some limitations that we were concerned with. The first one is the high computational complexity due to the need for scanning the image at several scales. Besides, the problems with integration of information from different scales and lack of precision due to the inherent fact that only small (or sufficiently scaled down) text exhibits the properties required by the algorithm are also essential. Additionally, these algorithms are typically unable to detect sufficiently slanted text.

Typical Region-based methods use the properties of the color or gray scale in a text region or their differences with the corresponding properties of the background. These methods can be further divided into two sub-approaches: connected component (CC)-based and edge-based. These two approaches work in a bottom-up fashion; by identifying sub-structures, such as CCs or edges, and then merging these sub-structures to mark bounding boxes for text. Note that some approaches use a combination of both CC-based and edge-based methods.

A typical Region-based method contains the following five steps; namely feature extraction, sub-structure identifying, sub-structure filtering, sub-structure merging, bounding box marking. The architecture of this process can be visualized in Figure 3.2.

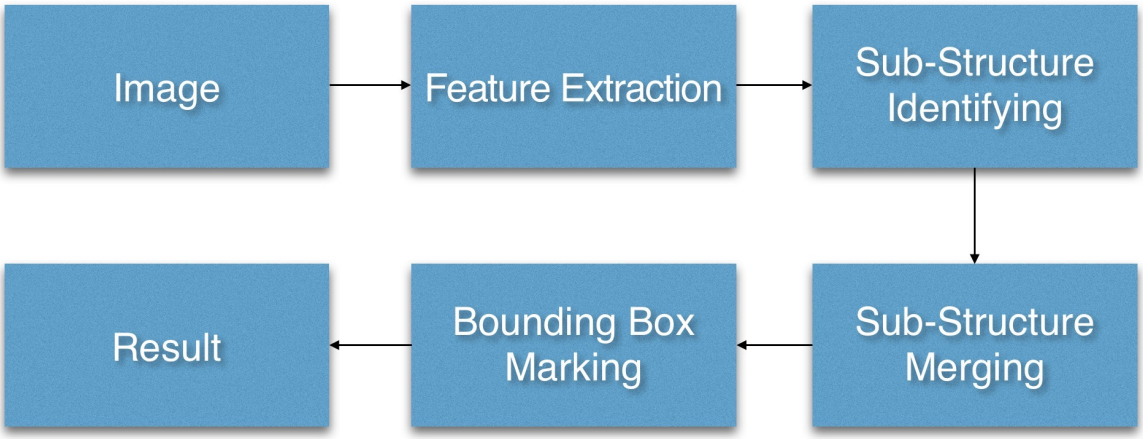


Figure 3.2: Architecture of a typical region-based method

Feature Extraction: This phase takes image or video frame as input and transfers it into a feature like image (colour value, edge value, or some other features).

Sub-structure Identifying: This phase uses some kinds of methods (e.g., connected compo-

nents) to identify the sub-structures that may be text.

Sub-structure Filtering: This phase adopts certain conditions to filter the sub-structures that are certainly not text.

Sub-structure Merging: This step is used to merge sub-structures together based on their localization relationships or some other rules.

Bounding Box Marking: The last module of this process is to mark the final output bounding boxes. Some of the merged sub-structures may need to combine to a bounding box due to some reason(e.g., overlapping on each other). This step is usually along with the last step.

Here we take a newly proposed feature called Stroke Width. Because text can be distinguished from other elements of a scene by its nearly constant stroke width, we can leverage this fact to recover regions that are likely to contain text.

A great number of works deals directly with detection of text from natural images and video frames. Related works from other domains study the extraction of linear features.

The work that uses a somewhat similar idea of detecting character strokes is presented in [64]. The method, however, differs drastically from the algorithm developed in this paper. The algorithm proposed in [64] scans an image horizontally, looking for pairs of sudden changes of intensity (assuming dark text on bright background). Then the regions between changes of intensity are examined for colour constancy and stroke width (a range of stroke widths is assumed to be known). Surviving regions are grouped into a vertical window of size W, and if enough regions are found, a stroke is declared to be present. The limitations of this method include some specific parameters tuned to the scale of the text to be found (such as vertical window size W), inability to detect horizontal strokes, and the fact that detected strokes are not grouped into letter candidates, words and sentences. Consequently, the algorithm is only able to detect near-horizontal text.

Another method [16] also uses the idea of stroke width similarity, but is restricted to finding horizontal lines of small text, due to the traversal along horizontal scan lines to detect vertical strokes, and the use of morphological dilation to connect candidate pixels into connected regions. The algorithm would not be able to deal with arbitrary directions of strokes.

Finally, the work [35] uses the idea of stroke width consistency for detecting text overlays in video sequences. The limitations of the method include the need for integration over scales and orientations of the filter, and, again, the inherent attenuation to horizontal texts.

Epshtein et al.[19] introduced a brand new usage on stroke width. Their definition of stroke is related to linear features which are commonly dealt with in two domains: remote sensing (extraction of road networks) and medical imaging (blood vessel segmentation).

In road detection, the range of road widths in an aerial or satellite photo is known and limited, whereas texts appearing in natural images can vary in scale drastically. Additionally,

roads are typically elongated linear structures with low curvature, which is again not true for text. Most techniques for road detection rely on the assumptions listed above and thus are not directly applicable for text detection.

For a survey of techniques, see [57]. The closest work is [18], which uses the fact that road edges are antiparallel for detecting points lying on the centerlines of the roads, then groups these center points together. No attempt is made to use constant road width to facilitate grouping. Epshtein et al.[19] uses dense voting on each pixel of the stroke, thus resulting in a much more reliable identification of strokes without requiring a difficult and brittle process of grouping center points. Another method [6] uses lines extracted from low-res images and border edges extracted from hi-res images to find road candidates. In the case of text detection, a whole multi-scale pyramid of images would be required for a similar strategy; moreover, the small or thin text still is unlikely to be detected using this method.

For a survey on blood vessel segmentation, see [39]. Related works use model fitting (snakes, generalized cylinders), ridge finding (ridge operators, binarization followed by thinning, wavelets) and other methods. Studies that use vessel width as an additional feature for tracking vessels starting from a user-designated seed include [53, 66]. None of the existing works try to detect vessels directly, in a bottom-up fashion, using the low variance of widths.

Epshtein et al.[19] proposed an operator named the Stroke Width Transform (SWT). Because SWT transforms the image data from containing colour values per pixel to containing the most likely stroke width, and the resulting system can detect text regardless of its scale, direction, font and language, we adopt and improve it for our dataset.

Their method differs from previous approaches in that it does not look for a separating feature per pixel, like gradient or colour. Instead, they collect enough information to enable smart grouping of pixels. In their approach, a pixel gradient is only necessary if it has a corresponding opposing gradient. This geometric verification significantly reduces the number of detected pixels, as a stroke forces the co-occurrence of many similarly matched pairs in a small region. Another notable difference of their approach from previous work is the absence of scanning window over a multi-scale pyramid. Instead, they perform a bottom-up integration of information, merging pixels of similar stroke width into connected components, which allows them to detect letters across a wide range of scales in the same image. Since they do not use a filter bank of a few discrete orientations, they detect strokes (and, consequently, text lines) of any direction. They also do not require the stroke width to be constant throughout a letter but allow slowly bounded variations instead.

Additionally, they do not use any language-specific filtering mechanisms, such as OCR filtering stage [11] or statistics of gradient directions in a certain candidate window. This allows them to come up with a truly multilingual text detection algorithm.

3.2.2 Text Region Filtering

In the last section we propose the method of detecting all possible text regions. However, some non-text patterns on image may be ambiguous and can be mistreated as text by SWT algorithm, so the results can not be used directly. For those candidates, we then use classification algorithm to further filter the images with regions of interest out of the whole candidates. Some wrong regions that do not contain any text or contain undesired text are shown in Figure 3.3.

Figure 3.3: Examples of wrong regions

Considering the region candidates fulfill the features we introduced in last section, those sub-images cropped by those regions will be in simple shapes or, at least, similar to text. The task of identifying the correct one from those similar sub-images is very similar to the task of classifying the Text Characters because we need to distinguish different types of texts. However, our task is much simpler, because we only need to judge whether it is correct or not.

For this section, we decide to use a binary classifier. One popular machine learning method, namely Support Vector Machine (SVM), can generate very high precision result under certain features and kernels. It will be used as the simple classifier to accomplish this goal.

Text Character Classification (TCC), which includes feature representation to model character structure and multi-class classification to predict label and score of character class, mostly plays a significant role in word-level text recognition.

TCC is a multi-class classification which can be processed in three stages, as a typical classifying procedure. First, character-like features extracted from both training images and the testing images. Second, those features extracted from training images are passed to a certain classifier to train that classifier. Third, the features of testing images are used to test the classifier, through the test result we can see if we need to adjust the classifier or change the features. Above procedure shows that the features and the classifiers would always play a significant role in TCC. Therefore, accuracy improvement of TCC will result in how we choose the features and the classifiers.

A variety of feature representations and classifiers for Text Character Classification (TCC) were proposed. In [73], Gabor filter responses were employed to extract features of character appearance. In [62], similarity expert was built from Scale Invariant Feature Transform (SIFT) descriptors to compute the character similarity. In [71], Histogram of Oriented Gradient (HOG) descriptors were densely extracted and cascaded as feature representations of character patches for the nearest neighbour classifier. In [70], Random Ferns algorithm was adopted for character detection. In [49], HOG feature was extracted for a multi-class Support Vector Machine(SVM) with the Radial Basis Function (RBF) kernel. In [12], local features of character patches were extracted by an unsupervised learning method related to a variant of K-means clustering and spatially pooled by cascading sub-patch features. In [50], feature extraction for an SVM with the RBF kernel was generated from Maximally Stable Extremal Regions (MSER), which is split into eight levels by MSER boundary orientations. In [79], TCC for Chinese, Japanese and Korean characters was performed by SIFT feature matching to template character patches.

One survey proposed by Yi et al.[78] discussed the works mentioned above, and also made a comparison on choosing features and classifiers.To learn a robust character classifier, they adopted the state-of-the-art SVM learning model. For feature selection, they did experiments on both local sampling and global sampling.

For global sampling, they used the whole character patch as a feature window to extract features. It skips keypoint detection, coding and pooling process to reduce information loss. They extracted only Histogram of Oriented Gradient (HOG) descriptor [14] as character structure features.

To obtain feature representation from local sampling, they detected keypoints, computed local descriptors, built the dictionary of visual words, and performed feature coding and pooling to get a histogram of visual words, i.e., bag-of-words. They adopted 6 state-of-the-art feature descriptors which had been extensively used in the general visual recognitions, including HOG,

SIFT [48], Speed Up Robust Features (SURF) [7], DAISY [67], Binary Robust Independent Elementary Features (BRIEF) [9], and Oriented Fast and Rotated BRIEF (ORB) [58].

The evaluation results on two datasets CHARS74K and ICDAR2003 demonstrated that Histogram of Oriented Gradient (HOG) descriptor, soft-assignment coding, max pooling, and Chi-Square [69] Support Vector Machines (SVM) obtain the best performance among local sampling based feature representations.

3.3 Our Enhanced Methods

In this section, we discribe our enhanced methods for both region detection and region filtering. We adopted the stroke width transform for region detection and text character classification for region filtering.

3.3.1 Region-based Detection

As we propose our improvement on the work of Epshtein et al.[19], to make the entire process understandable, we will introduce their theories and our improvement together. This method includes five steps, like the basic procedure of region-based text detection method. First, we define the notion of a stroke and then explain the Stroke Width Transform. Second we explain how it is used for grouping pixels into letter candidates. Third, we filter the candidates. Fourth, we merge them into chains. Finally, we describe the mechanism for grouping letters into bounding boxes which enable further filtering. The flowchart of the algorithm is shown on Figure 3.4.

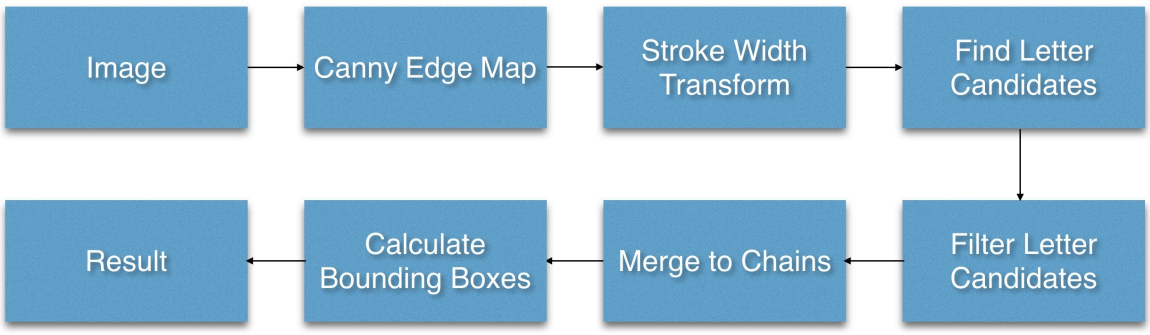
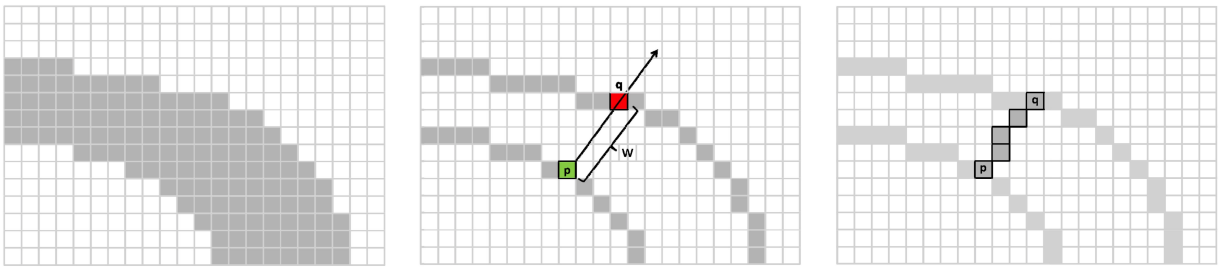


Figure 3.4: Flowchart of Epshtein et al.[19]'s work

The Stroke Width Transform

The Stroke Width Transform (SWT for short) is a local image operator which computes per-pixel the width of the most likely stroke containing the pixel. The output of the SWT is an image of the size equal to the size of the input image where each element contains the width of the stroke associated with the pixel. The stroke is defined to be a contiguous part of an image that forms a band of a nearly constant width, as depicted in Figure 3.5a.[19]



(a)A Stroke (b) Finding Stroke Width (c) The Stroke Width

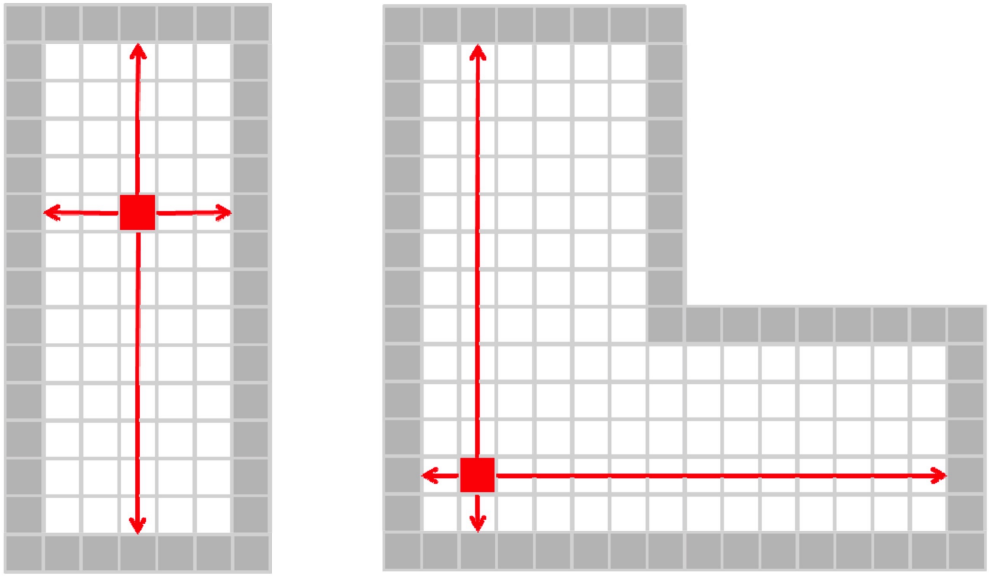
Figure 3.5: Steps of SWT from [19]

The initial value of each element of the SWT is set to ∞. In order to recover strokes, we first compute edges in the image using Canny edge detector [10]. After that, a gradient direction d_p of each edge pixel p is considered (Figure 3.5b). If p lies on a stroke boundary, then d_p must be roughly perpendicular to the orientation of the stroke. We follow the ray $r = p + n \cdot d_p, n > 0$ until another edge pixel q is found. We consider then the gradient direction d_q at pixel q. If d_q is roughly opposite to $d_p (d_q = -d_p \pm \pi/6)$, each element s of the SWT output image corresponding to the pixels along the segment $[p, q]$ is assigned the width $\|\overrightarrow{p - q}\|$ unless it already has a lower value (Figure 3.6a). Otherwise, if the matching pixel q is not found, or if d_q is not opposite to d_p, the ray is discarded. Figure 3.5 shows the process of SWT computation.[19]

As shown in Figure 3.6b, the SWT values in more complex situations, like corners, will not be true stroke widths after the first pass described above. Therefore, we pass along each non-discarded ray again, compute median SWT value m of all its pixels, and then set all the pixels of the ray with SWT values above m to be equal to m.[19]

Because the performance of SWT highly relies on the quality of edge image, how to generate the best canny edged images through the thresholds remains a complex question, we will discuss on this parameter in the last section.

The SWT operator described here is linear in the number of edge pixels in the image and also linear in the maximal stroke width. The output of the SWT is an image where each pixel contains the width of the most likely stroke it belongs to. The examples of before and after



(a) Pixels with two Stroke Width (b) Complex situation

Figure 3.6: Special conditions from [19]

applying SWT on test images are shown in Figure 3.7.

Finding generalized letter candidates

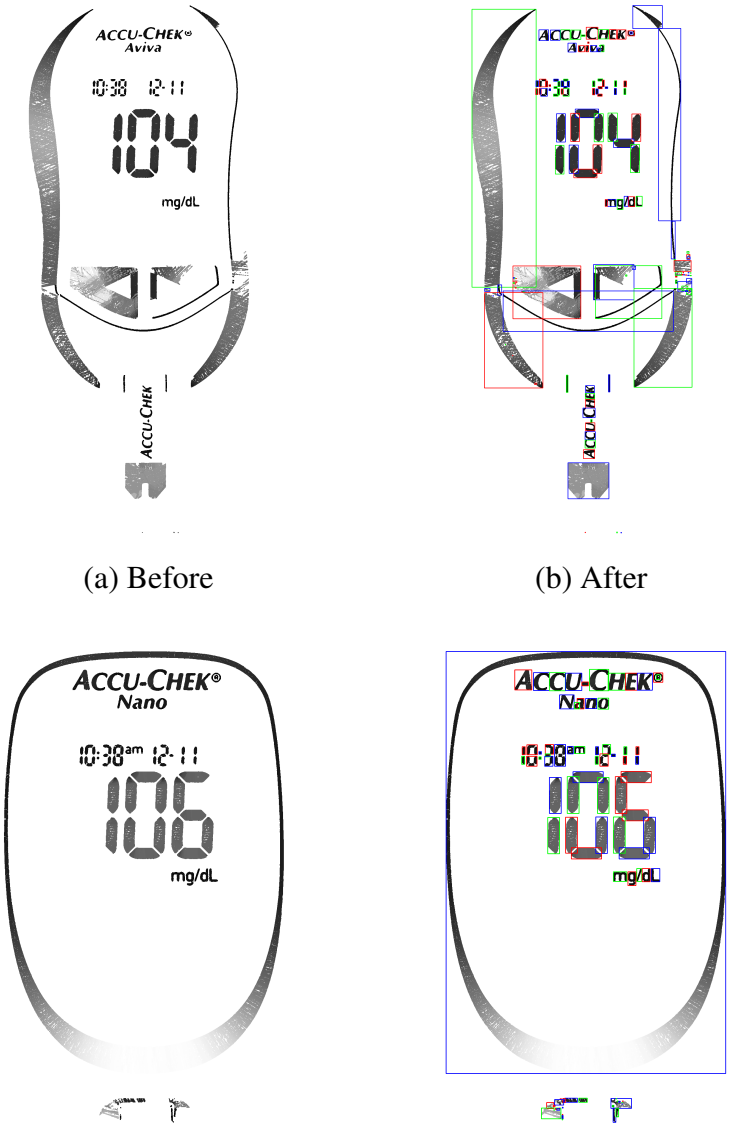
In our case, because our dataset contains the seven-segment digits (discussed in section 1.3), the ”so-called” letter candidates may contain those segments separately. Here we treat those segments as letter candidates, and we call them the generalized letter candidates. To group these pixels into the generalized letter candidates, we employ a set of rules which is different with the rules in[19].

After SWT, we can get the output image of all the stroke width of each pixel. Right now, a modified Connected Component algorithm should be applied to the out image. Because two neighbouring pixels may be grouped together if they have similar stroke width. We change the association rule from a binary mask to a predicate that compares the SWT values of pixels, as Epshtein et al did in [19]. They let the two neighbouring pixels group together when their SWT ratio does not exceed 3.0. However, our choice is more conservative with the value of 2. Because the stroke width of the digits in glucometers remains almost constant, to accommodate both bright text on dark background and vice-versa, we apply the algorithm twice, once along d_p and once along $-d_p$. The output of this step will be a group of connected components shown in Figure 3.8.

(a) Before (b) After

(c) Before (d) After

Figure 3.7: Examples of before and after applying SWT



(a) Before (b) After

(c) Before (d) After

Figure 3.8: Examples of before and after applying letter candidate localizing

Filtering generalized letter candidates

To identify components that may contain ideal segments, we employ a small set of fairly flexible rules. For our training dataset, the parameters of each rule are also different with [19].

The first parameter in [19] is the variance of the stroke width of each connected component. Their learned threshold is half the average stroke width of a particular connected component which ours is the same.

The second parameter they concern is the aspect ratio (the longer side length divided by, the shorter side length) of these components, because many natural processes may generate long and narrow components that may be mistaken for possible digits or segments. They limit the aspect ratio to be less than 10 times. We use the same value.

Thirdly, they concern the ratio between the diameter of the connected component and its median stroke width. Their value should be less than 10, and we eliminate this value to be less than 5.

The fourth one is the overlap counts of different components because the connected components may surround by text, such as sign frames. They eliminate those by ensuring that the bounding box of a component will include not more than two other components while we chose one as the number of components.

Lastly, they ignore the components whose size is too small or too large for the height to be limited in the range of 10 to 300 pixels. For this parameter, we use the same as theirs.

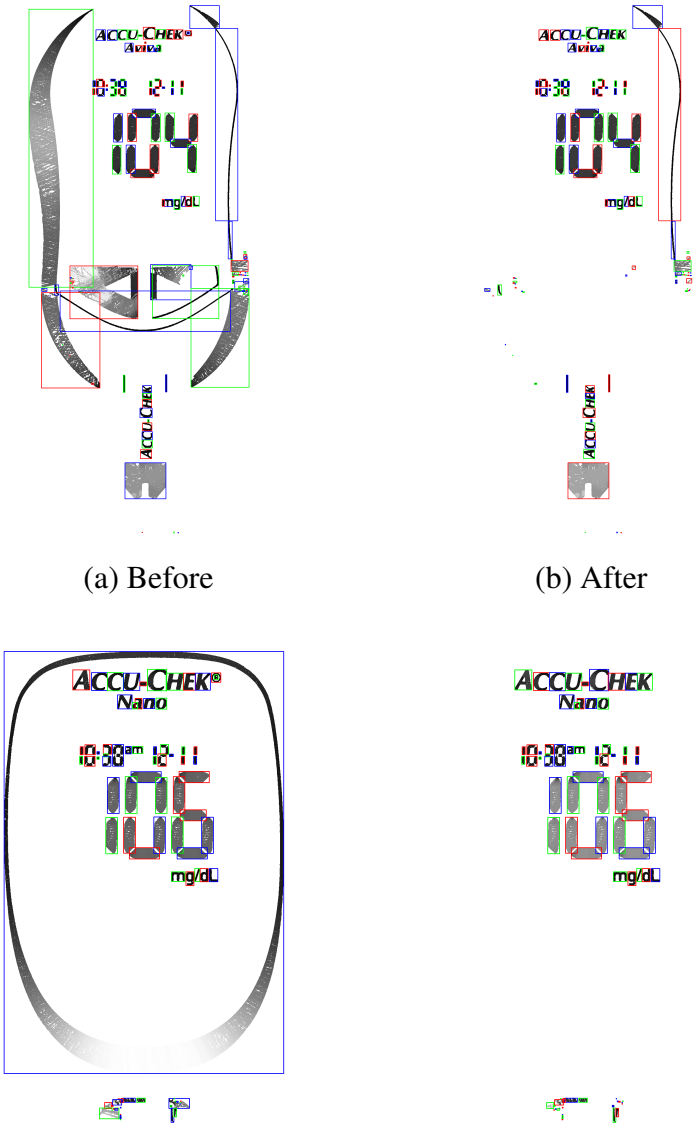
Remaining components are considered generalized letter candidates. In the next section, we will combine those digits or segments together by merging them to chains in next section. The output examples on test images are shown in Figure 3.9.

Merging to chains

To get the possible digits region, we concern to group the generalized letter candidates. Finding such groups is important as single one digit or segment do not usually appear in images, which allows us to remove randomly scattered noise.

For those typical digits, they may appear in a linear form as typical text forms. Like text on a line that is expected to have similarities, the digits in a line also shares similar stroke width, letter width, height and spaces between the single digits.

For those segments, they will appear in a more complex form than the typical digits, but they also share some similarities, and those similarities are different with typical digits. For example, they often shares the similar shape (they typically are a long and thin rectangle) with similar aspect ratios and similar areas, they often are closer to each other from the same digits than from different ones, and they should also have similar stroke width.



(a) Before (b) After

(c) Before (d) After

Figure 3.9: Examples of before and after applying letter candidate filtering

Under these two conditions, we consider each pair of letter candidates should obey some specific rules.

First, two candidates should have similar stroke width (ratio between the median stroke widths has to be less than 2.0).

Second, the aspect ratio of the candidates must not exceed 2.0 (due to the difference between capital and lower-case letters).

Third, The distance between letters must not be greater than a certain value will be finally determined in the last section.

Fourth, average colours of candidates for pairing are compared, as letters in the same word are typically expected to be written in the same colour, this value will be finally determined in the last section.

After applying those rules, the candidates are merged into candidate pairs. Next we need to cluster those candidate pairs together into chains. Initially, each chain consists of a single pair of letter candidates. Two chains can be merged if they share one end and have similar direction. The process ends when no chains can be merged. The output is a group of chains shown in Figure 3.10.

Calculating Bounding Boxes

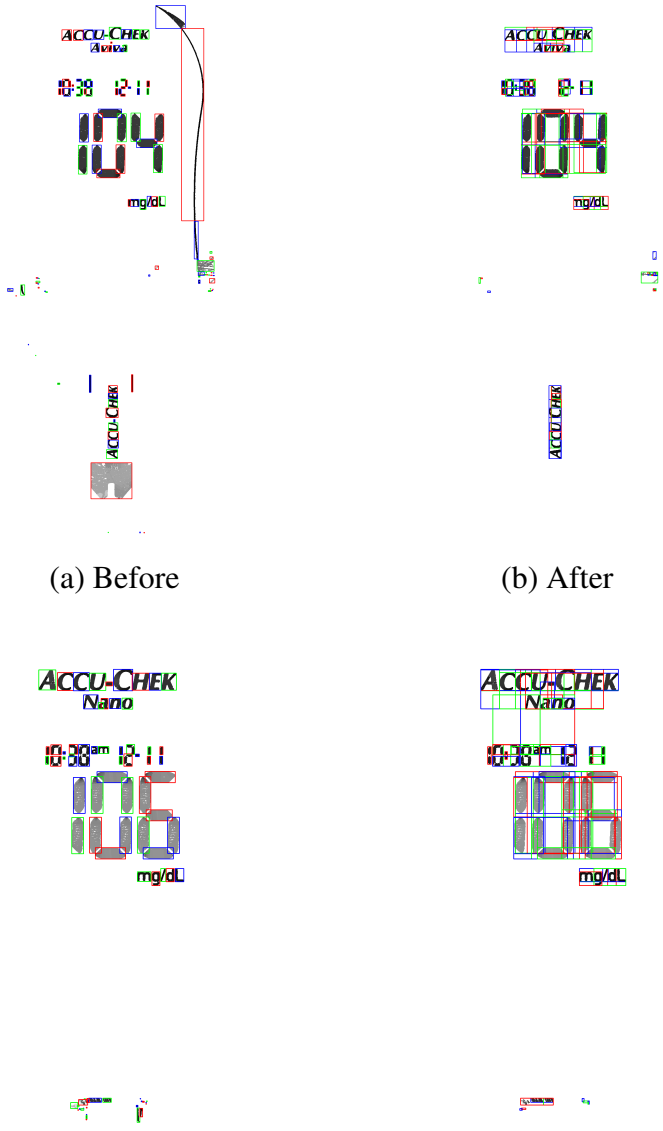
To get the location of the final bounding boxes, the very top left point and the very bottom right point of each chain are recorded as the top left point and bottom right point of the bounding boxes. However, when the chains get overlapped by some other chains, we will get bounding boxes with the same area, which drastically increase the number of final output bounding boxes. Therefore, before we perform the bounding box calculation, we merge the overlapped chains and get the final possible text region, shown in Figure 3.11

3.3.2 Text Character Classification

Similar to Yi et al.[78]'s work, we also train an support vector machine for filtering the text region. Firstly, we collect data for training and testing. Secondly, we calculate HOG features for both training and testing data. Finally, we evaluate the model through cross-validation and testing data.

Data collection

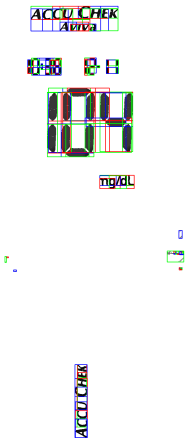
We create the training dataset through the all the image candidates that we generate through above steps. We collect 9161 images as training samples through text region detections with



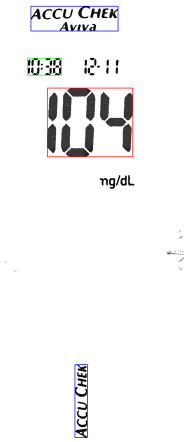
(a) Before (b) After

(c) Before (d) After

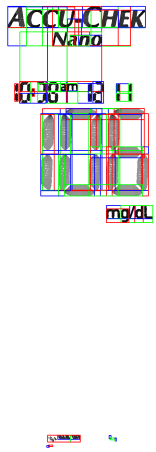
Figure 3.10: Examples of before and after applying chains merging



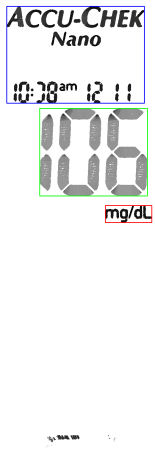
(a) Before



(b) After



(c) Before



(d) After

Figure 3.11: Examples of before and after applying bounding box calculating

various parameter conditions. Each image in the training samples is labeled by 1 or 0 for being valid or not respectively. To make sure the dataset function works well, we label every image by hand. Figure 3.12 shows some examples of the training dataset.

Figure 3.12: Examples of training samples

Some images are hard to decide whether is valid or not. We label them with a basic rule: if the image contains any character or pattern we label it as 0; if the image only contains digits but the digits are hard to be recognized by human, we also label it as 0; otherwise, we label them as 1. Figure 3.13 shows some confusing training samples.

For testing dataset, we use the sub-images cropped by the ground true bounding boxes of the original dataset. Thus, we have 266 testing images, and all of them will be labeled as 1. Figure 3.14 shows some examples of the test dataset.

At last, we get a training dataset with 9161 samples and testing dataset with 266 samples.

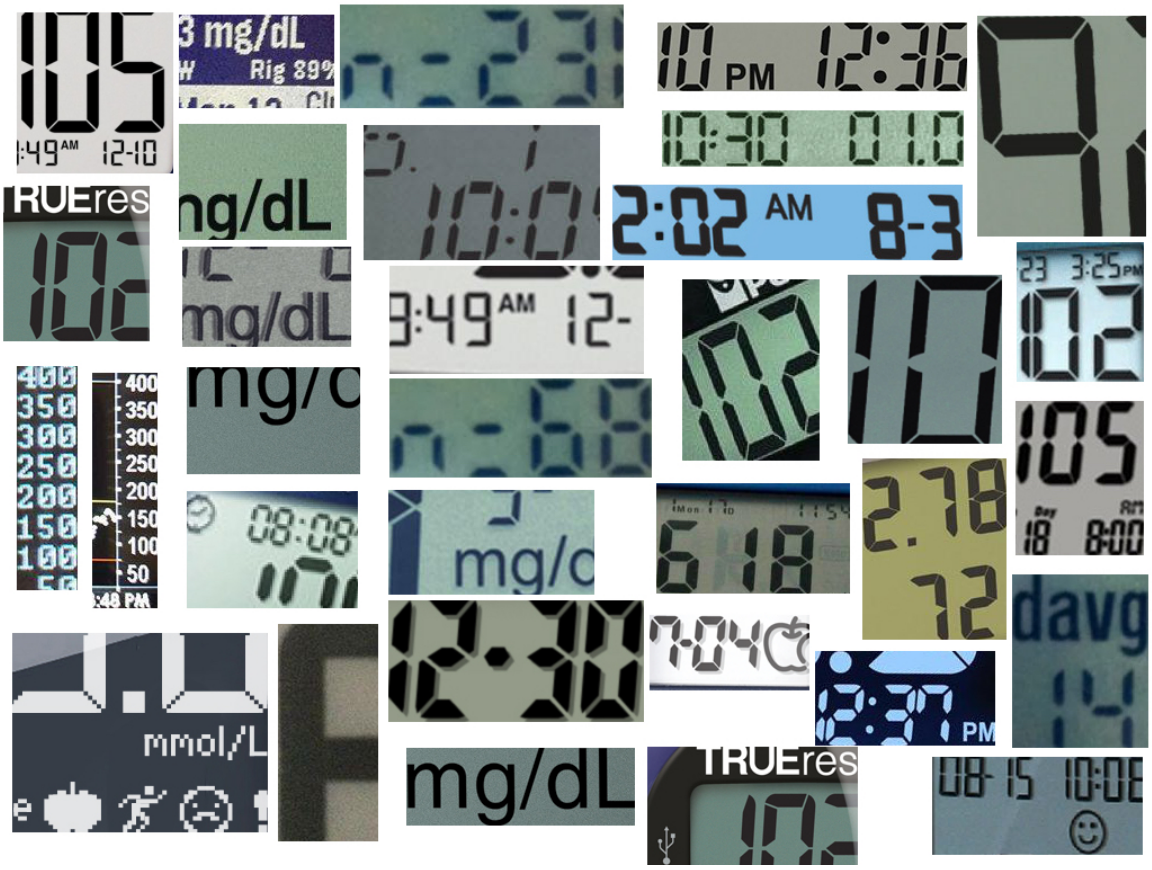


Figure 3.13: Examples of confusing samples

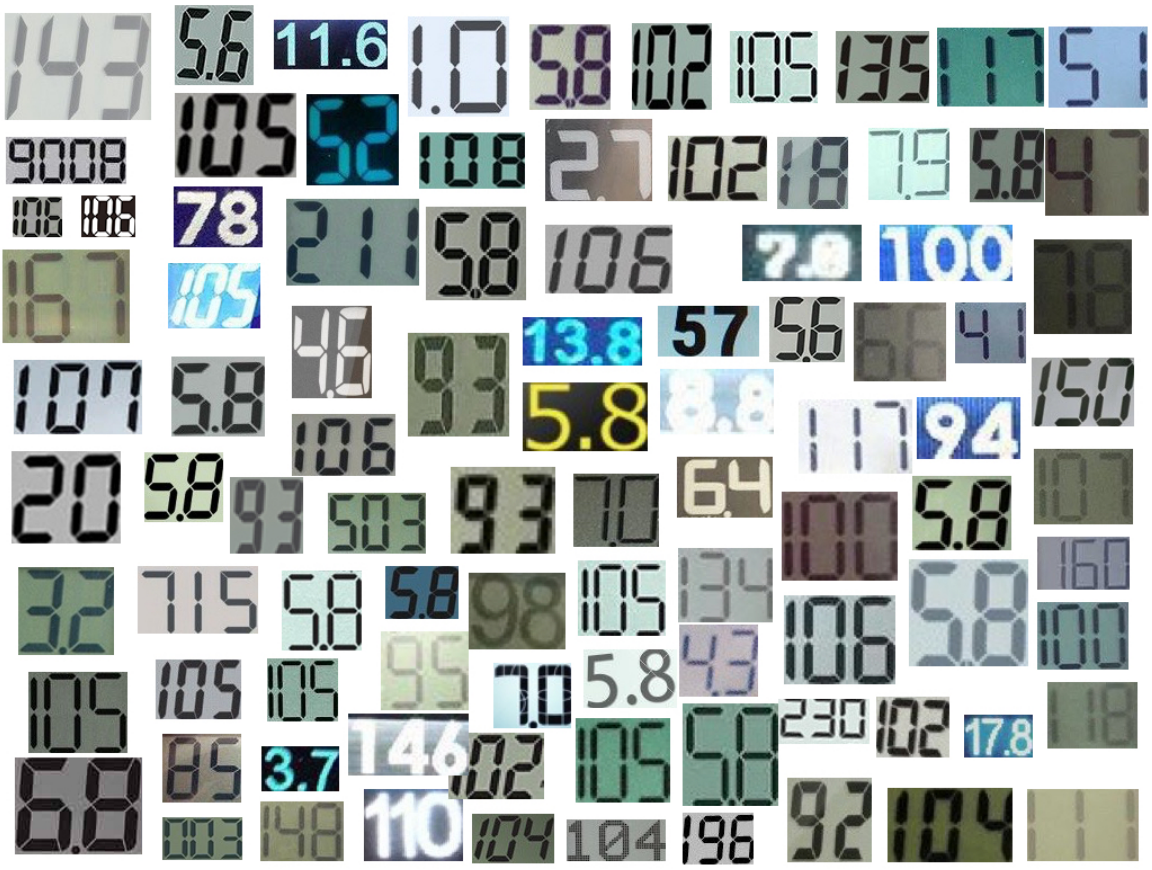


Figure 3.14: Examples of test samples

Feature Calculation

For all the images from training dataset and testing dataset, we compute their Histogram of Oriented Gradience(HOG) Descriptors.

First, we calculate the gradient image of the input image as G.

Second, we resize G to a rectangle patch whose width and height are the multiple of W, we can get width and height equal to mW and nW respectively. In the $mW \times nW$ patch, sub-patch in $W \times W$ was generated as the size of the feature window to extract feature descriptor, as shown in Figure 3.15. Inside the window, square blocks are used with the block width of B_w which typically should be the multiple of 8, sliding from top-left to horizontal and vertical directions with the block stride of B_s which typically should be half of B_w, as shown in the Figure 3.16. Within the block, we define the cell as a 8×8 square, which should contain 64 pixels as shown in the Figure 3.17.

Third, the histogram of oriented gradients is calculated in each cell with the number of bins N_{bin}. N_{bin} is the number of directions we define in the histogram. Such as, $N_{bin} = 9$ means histogram contains nine directions, and each direction takes 20 degrees out of 180 degrees in angular measurement.

At last, we connect all the histogram into one vector as the feature vector. The feature should have a length calculated by Formula 4.1.

$$L = nmN_{bin}(B_w/8)^2[(W - B_w)/B_s + 1]^2 \tag{3.1}$$

Here we set $W = 64, B_w = 16, B_s = 8, N_{bin} = 9$ and set the m and n as variables to get a better result.

Training and Testing

In this part, we use our training data with different m and n to train Support Vector Machines with linear, RBF and Chi-Square kernels separately. After cross validation and testing, we can get a result shown in Table 3.1

From the Table 3.1 we can see that the linear kernel under the resize size of 64×64 can generate the highest result. For this step, we adopt the linear kernel with the resize size of 64×64.

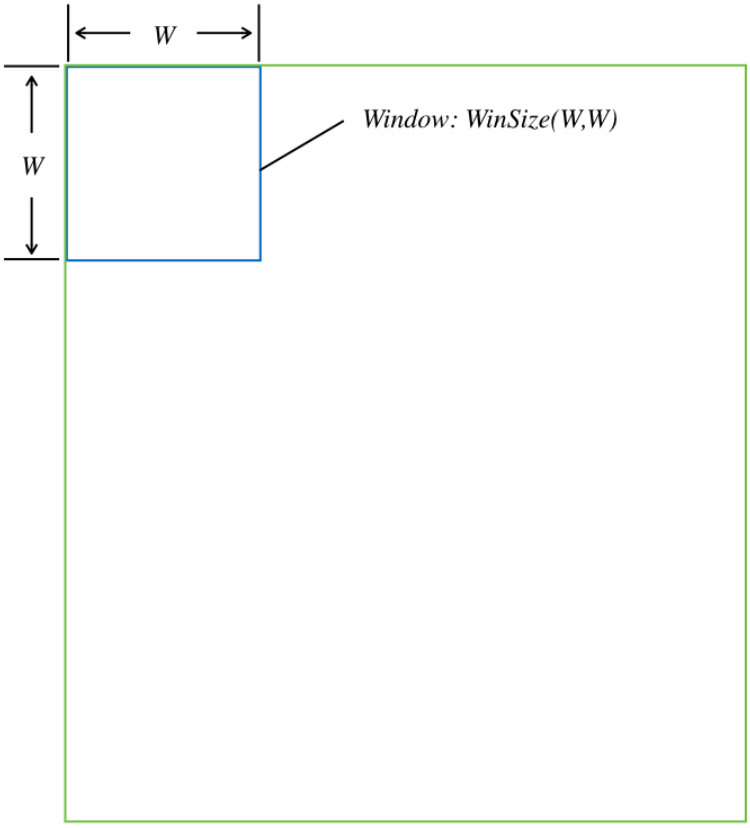


Figure 3.15: Window definition of HOG

kernel	Linear	RBF	Chi-Square
m=1,n=1	93.23	90.50	86.03
m=1,n=2	91.92	89.63	89.38
m=2,n=1	90.72	90.06	90.07
m=2,n=2	91.21	90.15	90.50

Table 3.1: SVM results

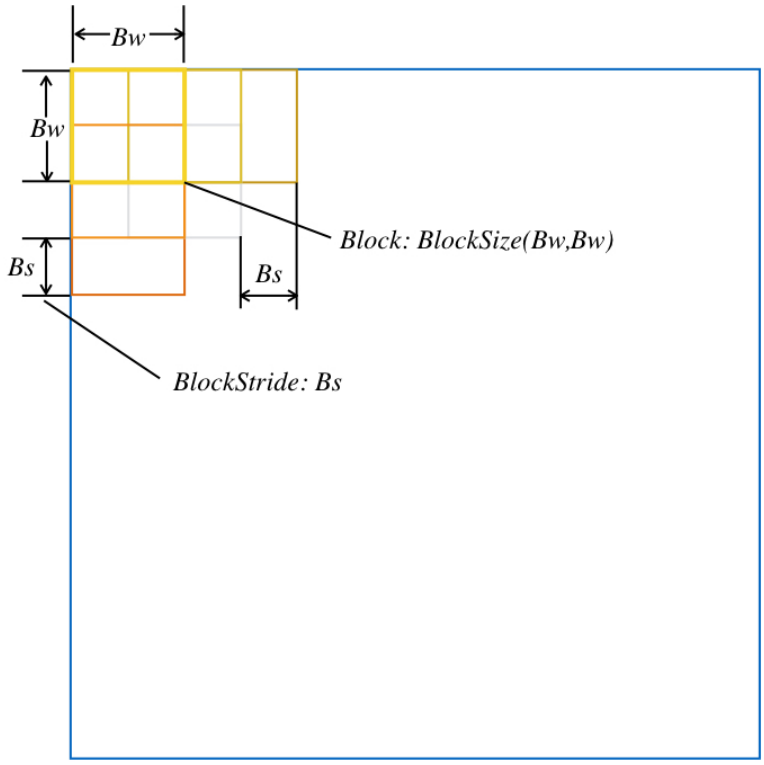


Figure 3.16: Block definition of HOG

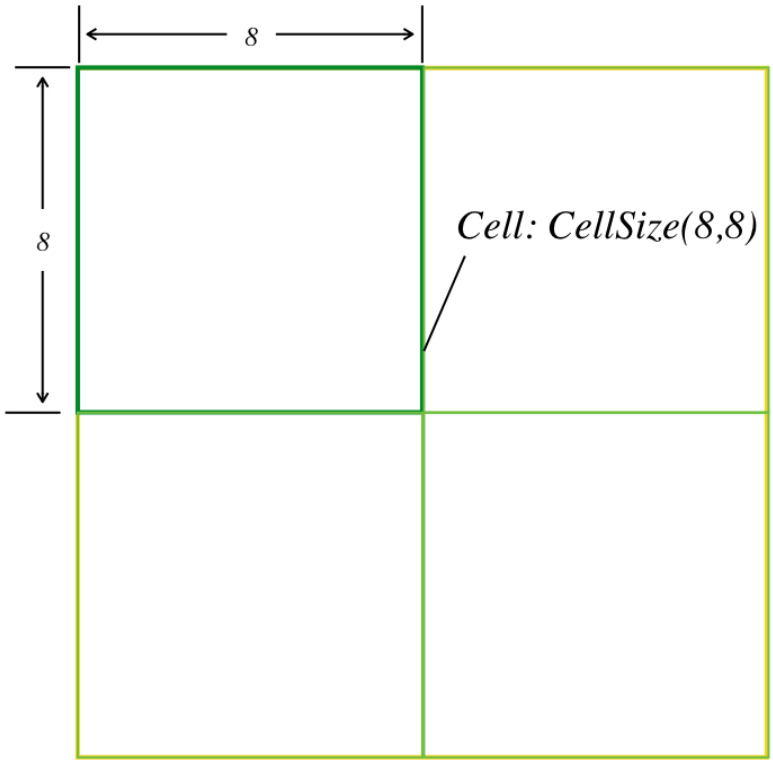


Figure 3.17: Cell definition of HOG

3.4 Experiment Results and Summary

Our evaluation method is the same as we defined in the last chapter. We experimented on three important parameters which we discussed in the last section: canny threshold, component distance multiplier, color distances.

canny	(10,30)	(35,105)	(60,180)	(85,255)	(110,330)	(135,405)	(160,480)	(185,555)
y1y2	0.640574	0.705947	0.694367	0.658665	0.534897	0.542848	0.40765	0.201315

Table 3.2: Performances with different canny threasholds

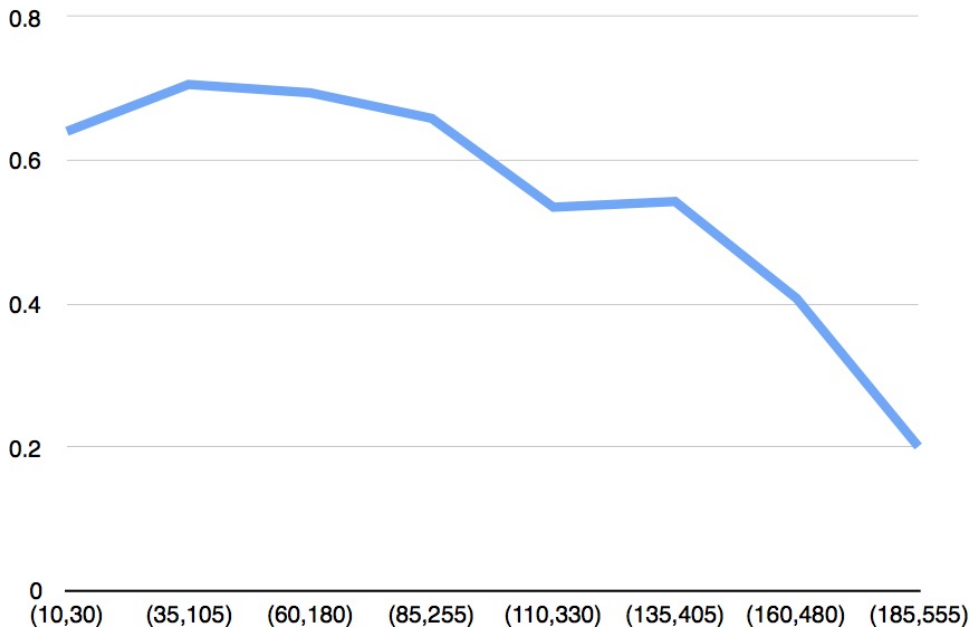


Figure 3.18: Performances with different canny threasholds

dist	5	10	15	20	25
y1y2	0.525907	0.664931	0.721518	0.711841	0.635215

Table 3.3: Performances with different component distance multipliers

In Table 3.2, the highest value appears when the canny threshold equals $(35, 105)$. Table 3.3 shows the best component distance is 20. Table 3.4 reminds us that when color distance is 500, we can generate the highest value.

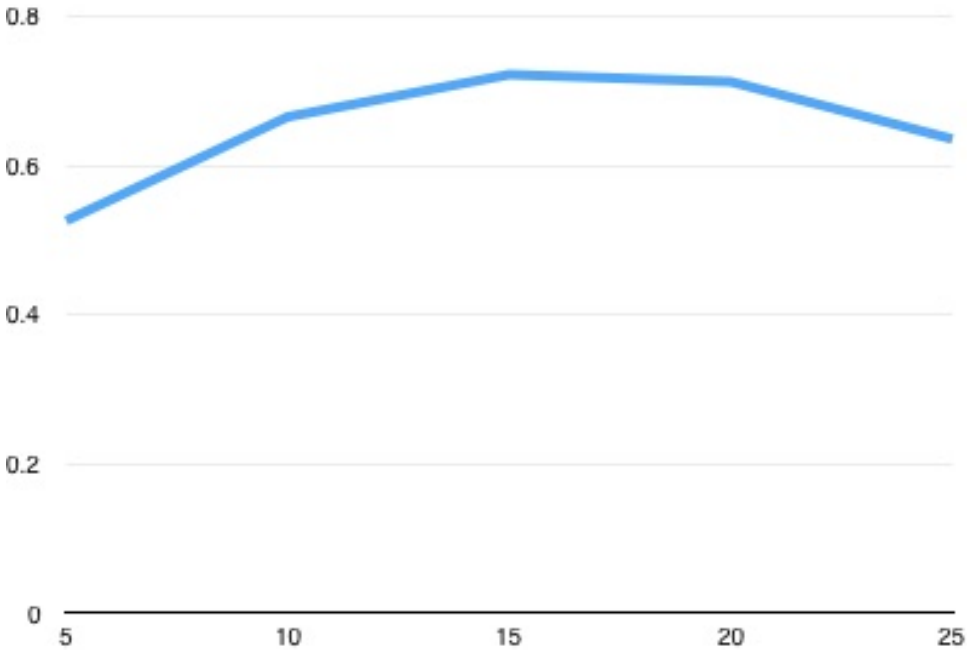


Figure 3.19: Performances with different component distance multipliers

color	50	500	5000	50000
y1y2	0.731157	0.761851	0.635183	0.484772

Table 3.4: Performances with different color distances

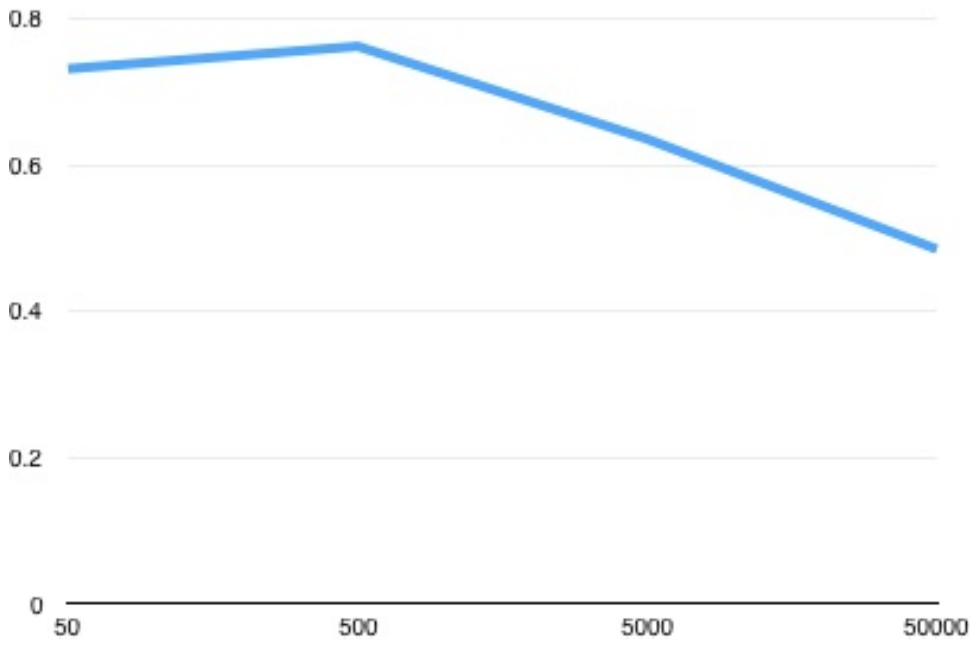


Figure 3.20: Performances with different color distances

Chapter 4

Text Recognition

In this chapter, we discuss the text recognition component. Typically, we call the method of recognizing the text in the image as Optical Character Recognition (OCR) method. Many OCR engines can recognize text in a high accuracy, and the most popular one is named Tesseract OCR engine. As it is the most popular and the state-of-art in OCR, we use it to accomplish our goal. However, it can only generate accurate results on devices with dot matrix displays. For those with seven-segment displays, it cannot locate the correct digits. Most of the data we collected were seven-segment digits, and thus we needed to improve it to make it functional well. We created a dataset of 10 kinds of seven segment fonts and retrained the Tesseract OCR with this dataset. These improvements in using Tesseract significantly improved the accuracy.

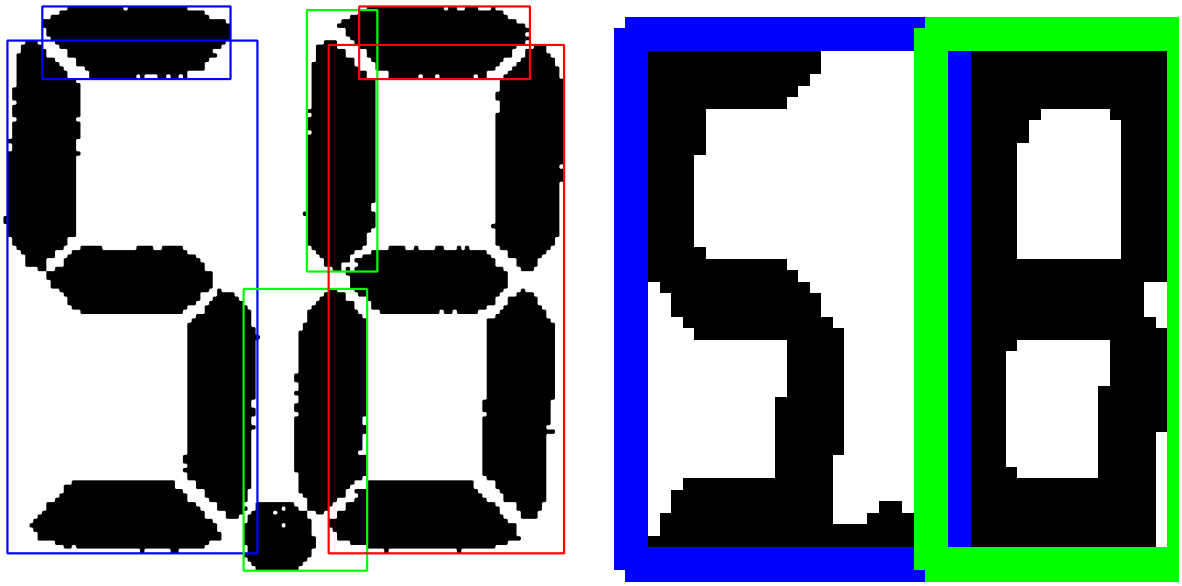
In the first section, we describe the challenges and the problem of this component. In the second one, we propose the related work of optical character recognition method. In the third section, we propose our enhancement of Optical Character Recognition as our text detection method. At last, we show the experiment result and the final determined parameters.

4.1 Problem Description

Upon the outcome of above components, we get the desired text regions of the original image. To get the final recognition result, we need to perform the text recognition based on those desired text regions. However text regions are only a group of bounding boxes, we still needed to crop the expected sub-images from the original images. It is evident that those desired sub-images are not post-processed images, which may contain all the problems we discussed in Chapter 2. Those problems still strongly affect the OCR engines, which means we need to do the image enhancement again. Fortunately, because the text regions helped us a lot on eliminating irrelevant information, this time, the image enhancement only required a few steps

in noise reduction.

Additionally, in Chapter 1 we discussed the two kinds of digit displays, which were also discussed in Chapter 3. The dot matrix display is more like our traditional digits because it can show the digits in more complex form, which can be easily captured by OCR engine to recognize. However, The seven segment display gives an entirely different form of digits, and we call it seven segment digits (SSD) here. The SSD is much harder for OCR engine to recognize or even locate the correct character because it is segmented. The OCR engine always locates those separated segments separately. Figure 4.1a shows a example.



(a) Bad Result Image (b) Good Result Image

Figure 4.1: Examples of recognition result on the connected segment image and the separate segment image

If we accidentally connected those segments together, the result changes a lot, as shown in Figure 4.1b. At least, the OCR engine can locate the right character. Therefore, we need to add a new step before passing the sub-images to the OCR engine, namely Morphology Transformations.

4.2 Related Work

According to Borovikov et al.[8]'s definition, Optical character recognition (OCR) is the process of converting scanned images of machine-printed or handwritten text (numerals, letters, and symbols), into machine readable character streams, plain (e.g., text files) or formatted (e.g.,

HTML files). As shown in Figure 1, the data path in a typical OCR system consists of three major stages:

1. document digitization
2. character/word recognition
3. output distribution

In the first stage, the scanner optically captures text in documents and produces document images. Here, the document image is the image we generated from above steps.

The second (and the most interesting) stage is responsible for character and word recognition in document images. The process involves four operations. The first one is the image analysis, including image quality assessment, text line detection, word and character extraction, etc. The second is the image enhancement, including removing speckle and other image noise, filling holes and breaks, etc. The third operation is character/word recognition. It is usually based on their shapes and other features. The last one is optional contextual processing, which limits the feature search space.

In the third (final) stage, the output interface communicates the OCR results to the outside world. For our system, the output stream will connect to the file for evaluation.

Obviously, the second stage is the most important and may be different in various OCR engine. Here we adopt the Tesseract OCR engine because it is the most popular and the state-of-art in OCR.

The OCR processing of Tesseract OCR engine follows a traditional step-by-step pipeline, but some of the stages were unusual in their day, and possibly remain so even now. The first step is a connected component analysis in which outlines of the components are stored. This was a computationally expensive design decision at the time, but had a significant advantage: by inspection of the nesting of outlines, and the number of child and grandchild outlines, it is simple to detect inverse text and recognize it as easily as black-on-white text. Tesseract was probably the first OCR engine able to handle white-on-black text so trivially. At this stage, outlines are gathered together, purely by nesting, into Blobs.[63]

Blobs are organized into text lines, and the lines and regions are analyzed for fixed pitch or proportional text. Text lines are broken into words differently according to the kind of character spacing. Fixed pitch text is chopped immediately by character cells. Proportional text is broken into words using definite spaces and fuzzy spaces.[63]

Recognition then proceeds as a two-pass process. In the first pass, an attempt is made to recognize each word in turn. We pass each word that is satisfactory to an adaptive classifier as training data. The adaptive classifier then gets a chance to recognize text lower down the page more accurately. Since the adaptive classifier may have learned something useful too late to make a contribution near the top of the page, we run a second pass over the page, in which

words that were not recognized well enough can be recognized again.[63]

A final phase resolves fuzzy spaces, and checks alternative hypotheses for the x-height to locate small cap text.[63]

4.3 Our Enhanced Methods

In this section, we discuss our implemented methods on text recognition. These methods include image preprocessing before optical character recognition (OCR) and some modification on the Tesseract-OCR engine.

4.3.1 Image Preprocessing

Here, we propose a sequence of Image preprocessing steps ahead of applying the final OCR, including image resizing, image enhancement, image binarization, and morphology transform.

Image Resizing

To keep consistency on choosing the window size on every filter, we first change the size of the input sub-images. Because the sub-images are all formed in a line of digits, we fix the height of the resized image to 50 pixels to make all the digits in the image to have the same size. Figure 4.2a shows the example images.

Image Enhancement

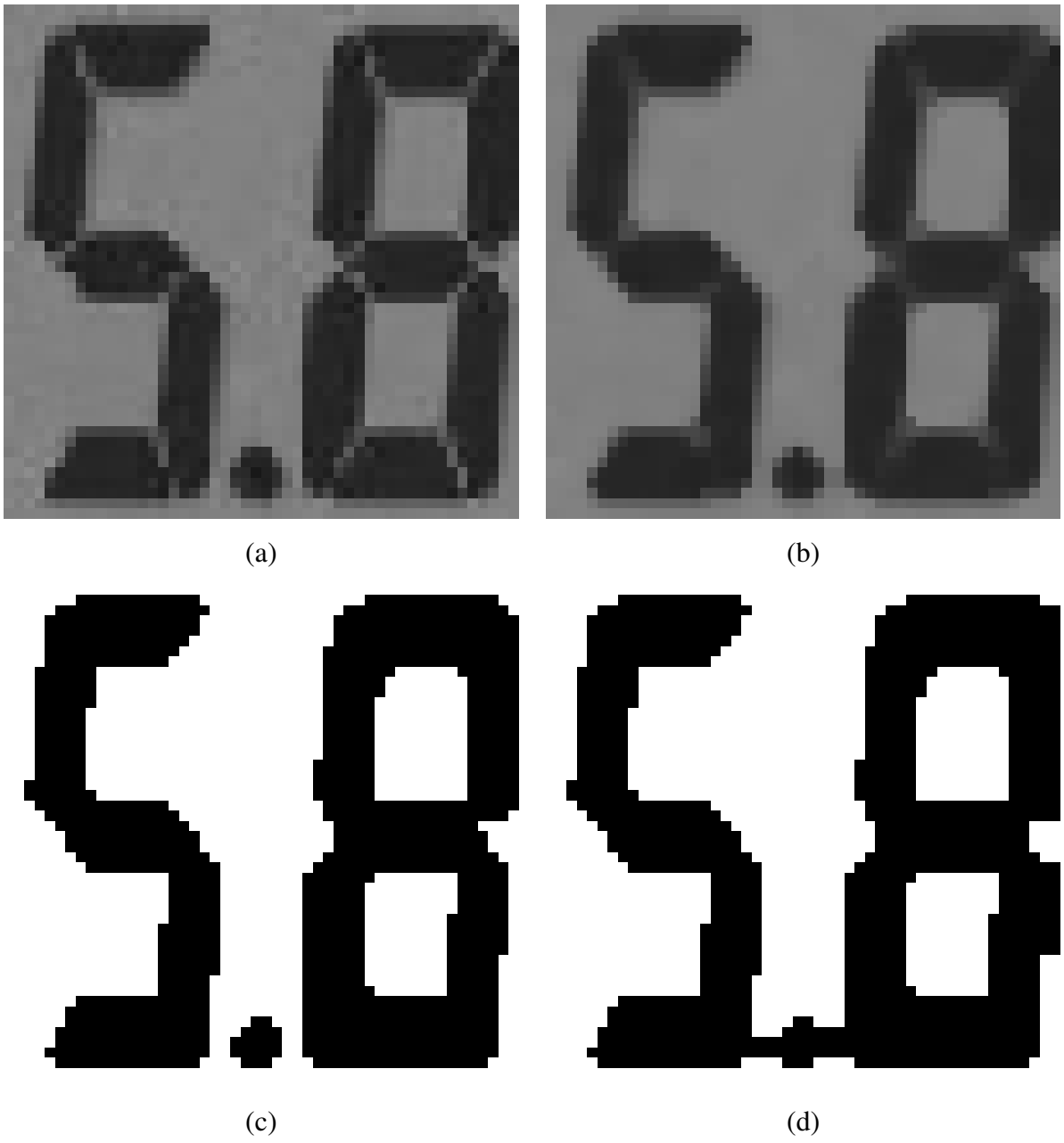
We also need to eliminate the noises from the original images, as we discussed in the last section. Here we simply apply the median filter with the window size of 3 to remove the Gaussian noises. Figure 4.2b shows the example images.

Image Binarization

To get the desired output for OCR engine and better input for morphology transform, we need to transfer the image into a binary image, by using Otsu Binarization algorithm. Figure 4.2c shows the example images.

Morphology Transform

To connect those segments into one digit, we adopt the closing operation by first eroding the image then dilating the eroded image with the same window size of 3. Figure 4.2d shows the example images.



(a) (b)

(c) (d)

Figure 4.2: Result Images of each step

4.3.2 Tesseract OCR

The Tesseract-OCR engine can recognize English texts and typical digits in a very high precision rate. However, for seven segment digits, it needs to be retrained. We collected ten kinds of different seven segment digits fonts and use them to train the Tesseract-OCR engine. The training samples are shown in Figure 4.3.

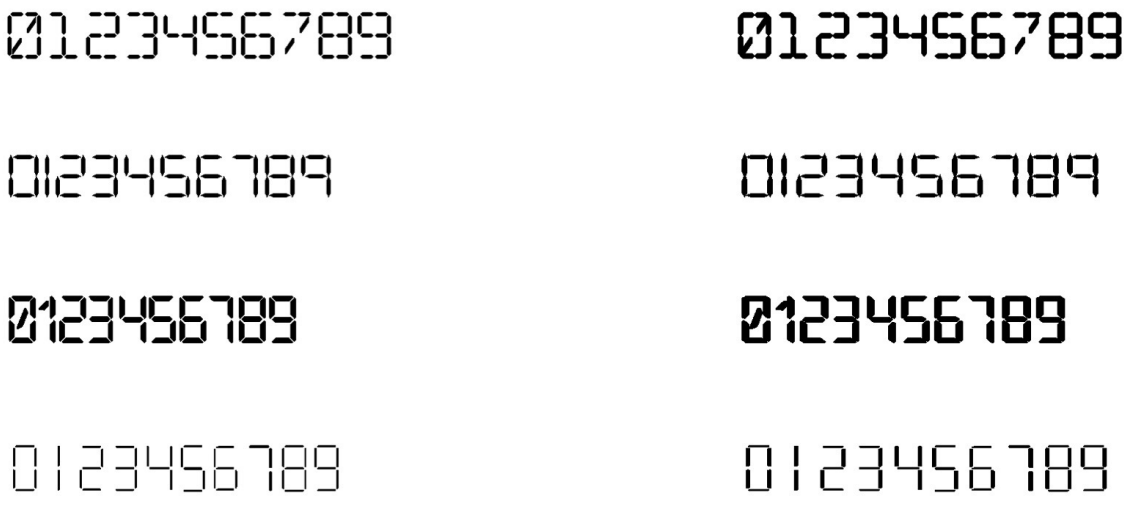


Figure 4.3: Example of training digits fonts

We also use our test samples (as we created in section) to test the engine, and we defined a method to evaluate the precision rate of the output. We defined the digit difference to measure the distance of two digits, and the calculation is as follows:

For each digit, we can treat it as a combination of seven segments. Therefore, we can define the similarities of digits by the number of the overlapping segments (the intersections) while can set the differences by the number of the separated segments (the unions subtracted by intersections). Table 4.1, Table 4.2 and Table 4.3 show the calculations of those values.

Based on the Table 4.3 we can measure the result of Tesseract-OCR engine for our test dataset. The result is shown in Table 4.4, the average difference of each digit is much higher than our retrained OCR output difference of each digit.

4.4 Experiment Results and Summary

Until this section, we have proposed the complete procedure of our digits recognition method. Right now, we can finally evaluate all the parts of our method. The evaluation includes the recall rate and precision rate of regions and the digits, whose Formula we discussed in Section

Similarities	0	1	2	3	4	5	6	7	8	9
0	6	2	4	4	3	4	5	3	6	5
1	2	2	1	2	2	1	1	2	2	2
2	4	1	5	4	2	3	4	2	5	4
3	4	2	4	5	3	4	4	3	5	5
4	3	2	2	3	4	3	3	2	4	4
5	4	1	3	4	3	5	5	2	5	5
6	5	1	4	4	3	5	6	2	6	5
7	3	2	2	3	2	2	2	3	3	3
8	6	2	5	5	4	5	6	3	7	6
9	5	2	4	5	4	5	5	3	6	6

Table 4.1: Digits similarities

Unions	0	1	2	3	4	5	6	7	8	9
0	6	6	6	6	6	6	6	6	7	6
1	6	2	6	5	4	6	7	3	7	6
2	6	6	5	6	7	7	7	6	7	7
3	6	5	6	5	6	6	7	5	7	6
4	6	4	7	6	4	6	7	5	7	6
5	6	6	7	6	6	5	6	6	7	6
6	6	7	7	7	7	6	6	7	7	7
7	6	3	6	5	5	6	7	3	7	6
8	7	7	7	7	7	7	7	7	7	7
9	6	6	7	6	6	6	7	6	7	6

Table 4.2: Digits unions

Differences	0	1	2	3	4	5	6	7	8	9
0	0	4	2	2	3	2	1	3	1	1
1	4	0	5	3	2	5	6	1	5	4
2	2	5	0	2	5	4	3	4	2	3
3	2	3	2	0	3	2	3	2	2	1
4	3	2	5	3	0	3	4	3	3	2
5	2	5	4	2	3	0	1	4	2	1
6	1	6	3	3	4	1	0	5	1	2
7	3	1	4	2	3	4	5	0	4	3
8	1	5	2	2	3	2	1	4	0	1
9	1	4	3	1	2	1	2	3	1	0

Table 4.3: Digits differences

Differences	0	1	2	3	4	5	6	7	8	9
average	1.9	3.5	3.0	2.0	2.8	2.4	2.6	2.9	2.1	1.8
output	0.77	1.15	1.73	0.28	0.21	0.0	0.78	0.5	0.66	0.82

Table 4.4: Difference output compared to the average difference

2.4. We also use the standard f measure to combine the precision and recall figures into a single measurement of quality. The relative weights of these are controlled by a parameter α , which we set to 0.5 to give equal weight to precision and recall:

$$f = \frac{1}{\frac{\alpha}{precision} + \frac{1-\alpha}{recall}} \tag{4.1}$$

Our digits recognition algorithm combines three components together. These components are image enhancement (discussed in chapter 2), the region of interest detection (in chapter 3) and enhancement for text recognition (in section 4.3 for morphology transform and 4.4 for retrained Tesseract-OCR engine). To compare the influence of each component on the final result, we apply the following experiments. The results are shown in Table 4.5 and Figure 4.4. The row titles are defined as:

"y1y2y3" refers to "our entire method";

"y1y2n3" refers to "method with image enhancement and region of interest detection without text recognition enhancement";

"n1y2y3" refers to "method with region of interest detection and text recognition enhancement without image enhancement";

"n1y2n3" refers to "method with region of interest detection without image enhancement and text recognition enhancement";

"n1n2y3" refers to "method with retrained OCR without morphology transform and image enhancement and region of interest detection";

"n1n2y3-2" refers to "method with both retrained OCR and morphology transform without image enhancement and region of interest detection";

"n1n2n3" refers to "method of applying the original OCR engine directly";

From the Figure 4.4 we can see that our method outperforms others in all types of evaluation. Additionally, each enhancement step will improve the result by applying them.

Final results	Region recall	Region precision	Digits recall	Digits precision	Digits F1score
y1y2y3	0.656984	0.789216	0.688377	0.618709	0.621194
y1y2n3	0.656984	0.789216	0.42626	0.420833	0.406301
n1y2y3	0.619906	0.730392	0.676899	0.585049	0.607991
n1y2n3	0.619906	0.730392	0.41651	0.367402	0.37952
n1n2y3	0.031276	0.004902	0.251109	0.139297	0.167484
n1n2y3-2	0.031276	0.004902	0.141961	0.530474	0.185022
n1n2n3	0.031276	0.004902	0.040655	0.305147	0.061793

Table 4.5: Final Results

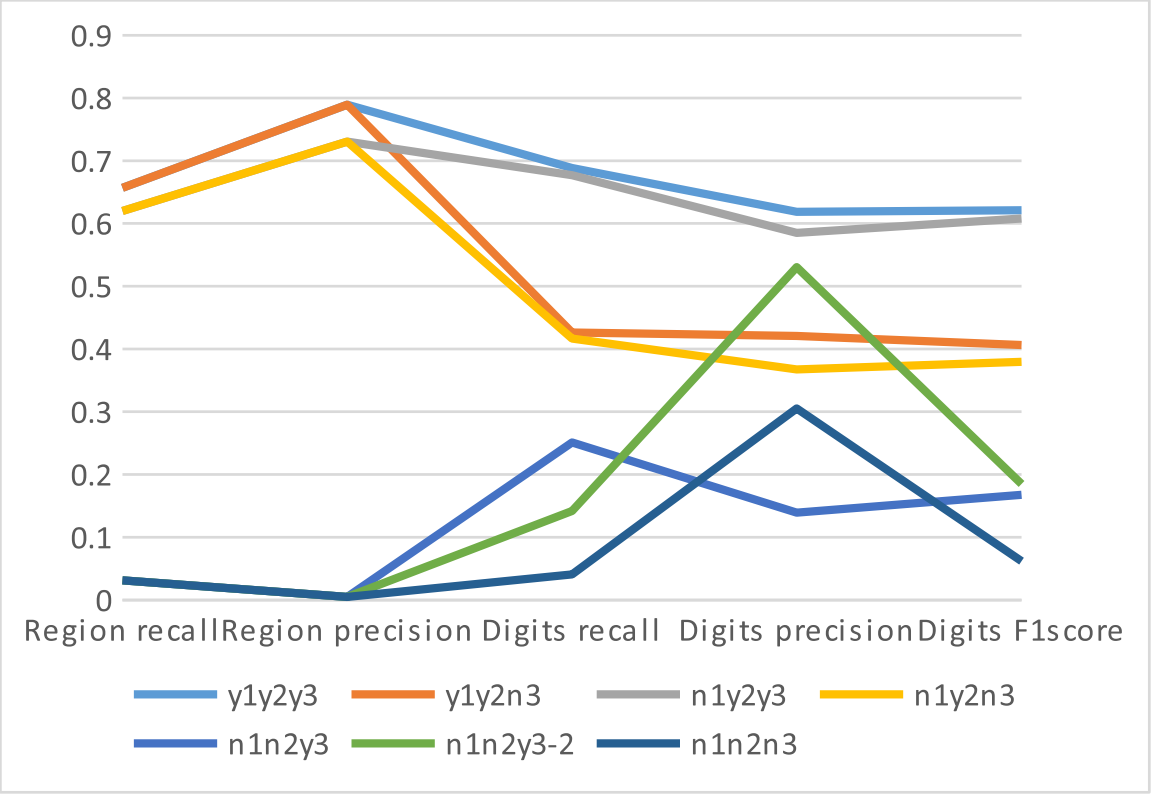


Figure 4.4: Final Results

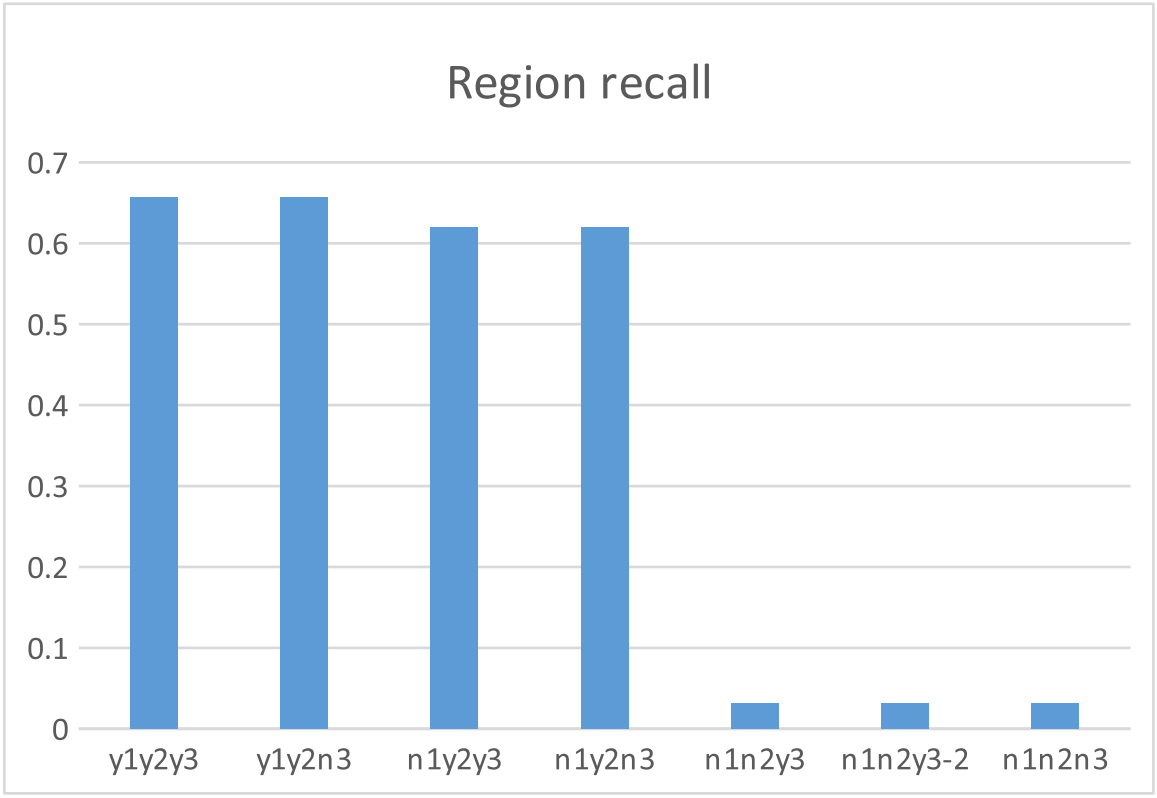


Figure 4.5: Regions recall

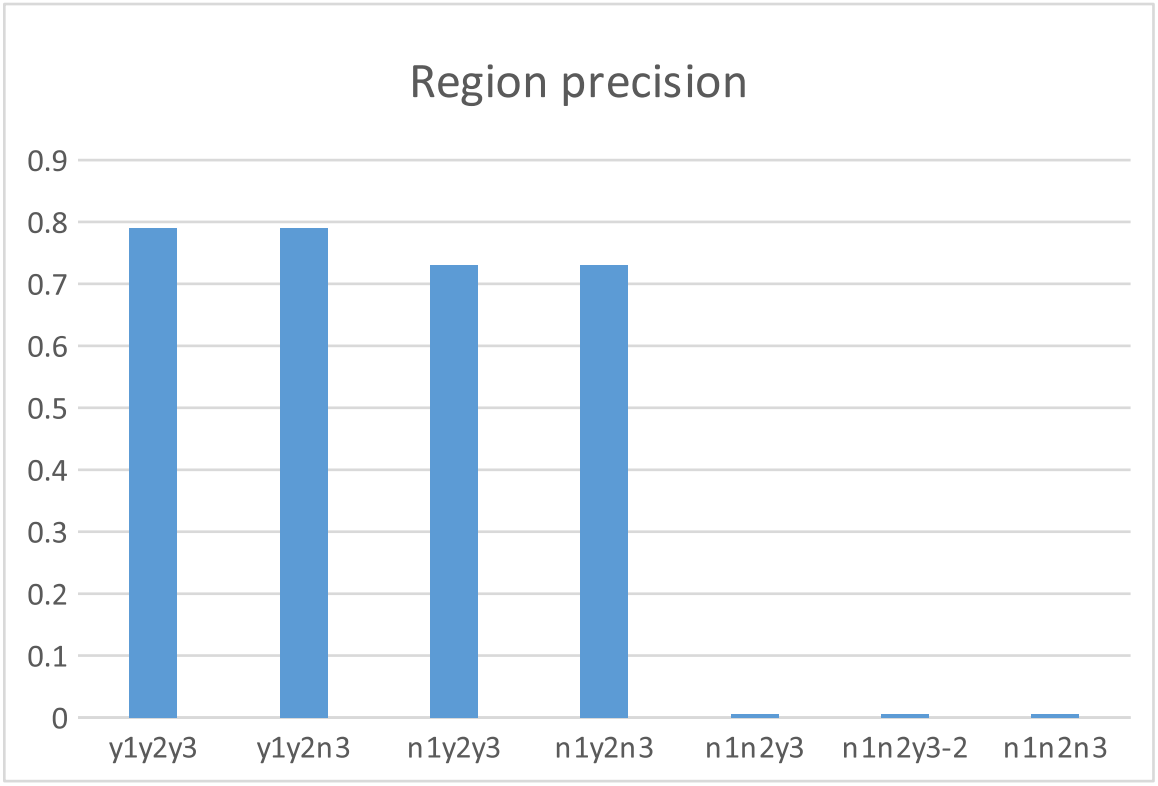


Figure 4.6: Regions precision

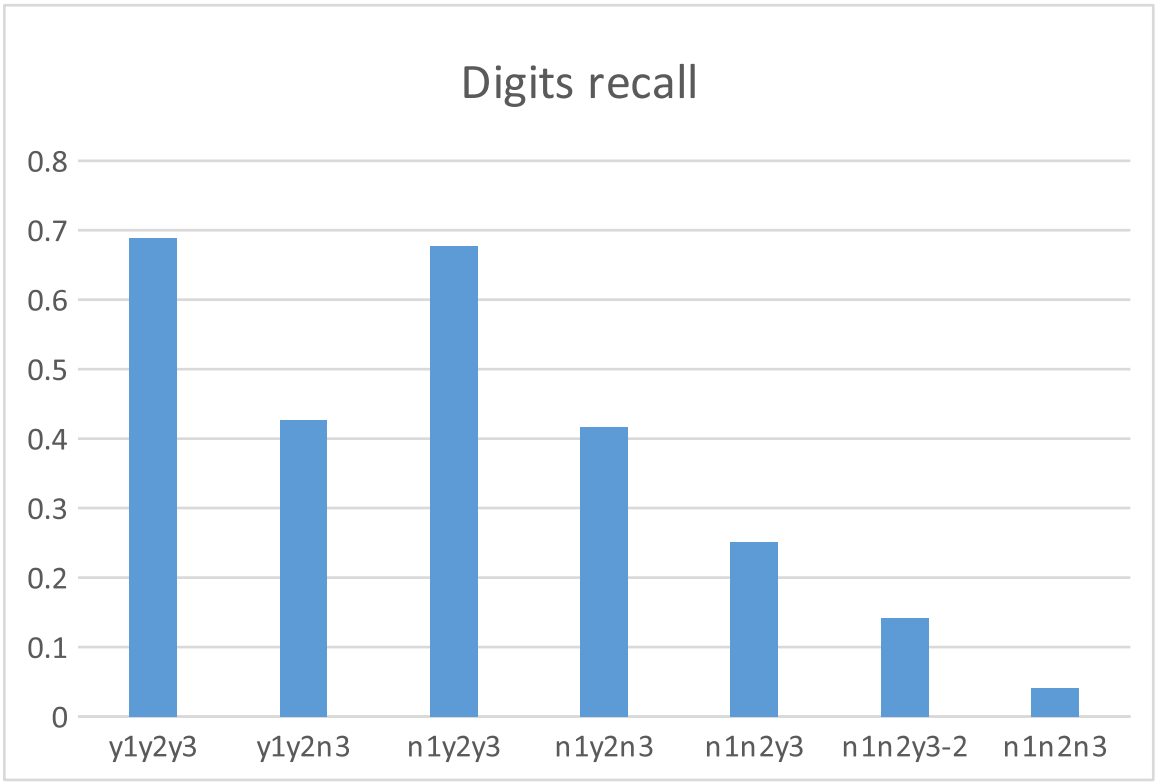


Figure 4.7: Digits recall

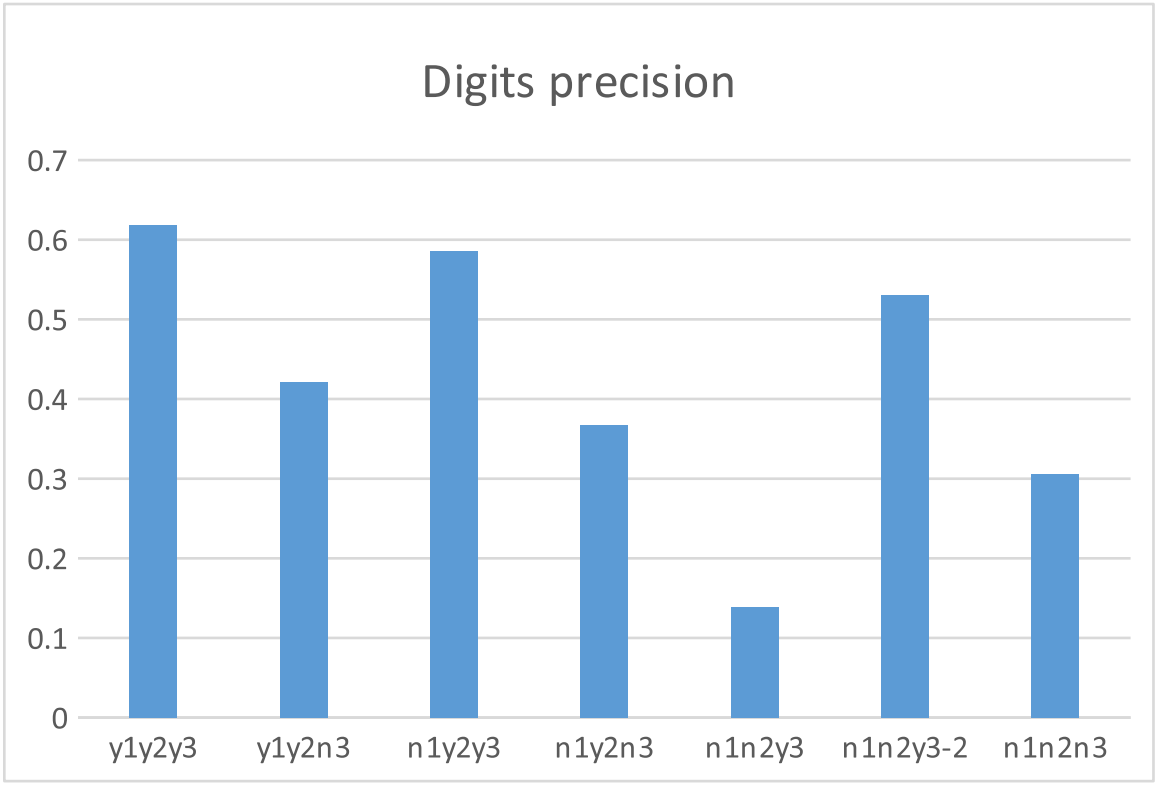


Figure 4.8: Digits precision

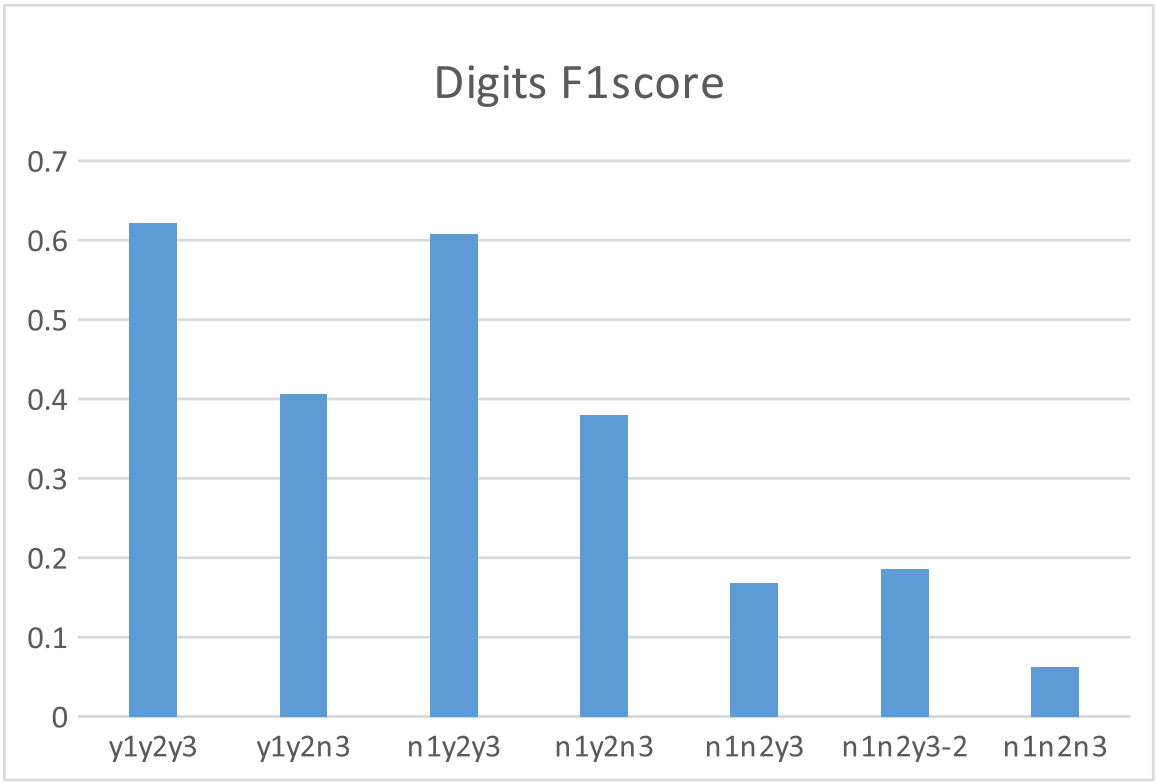


Figure 4.9: Digits F1-Score

Chapter 5

Conclusions

In this chapter, we will summarize all the experiment results and make a conclusion in section one, and the discuss the future works in section two.

5.1 Conclusions

In this thesis, we proposed an automatic procedure to recognize the digits on the screen of medical devices with smartphone cameras.

The overall procedure includes several ”standard” components in computer vision: image enhancement, the region of interest detection, and optical character recognition. Previous works existed for each component, but they have various weaknesses that lead to a low recognition rate. We proposed several novel enhancements in each component.

The first component is image enhancement. Since images may have various qualities under different light sources and conditions, we adopt the combination of basic filtering algorithms as to improve their qualities. After parameter searching and optimization, we used the combination of homomorphic filter, median filter, modified histogram equalization, and morphological transform to achieve the best results.

The second component is the region of interest detection. Typical natural scene images tend to have lots of useless objects and unpredictable noises, which largely affects the result of OCR engines. Thus, we need to segment the region of interest, i.e., the digits. We adopted the procedure named Stroke Width Transform, owing to its high accuracy and speed in detecting text regions. However, it generated poor results in our dataset. We adjusted the filtering and the chaining part to get a higher recall rate and used a Linear support vector machine to improve the precision rate.

The optical character recognition is the last component. Because the Tesseract OCR engine

is the most popular and the state-of-art in OCR, we used it to accomplish our goal. However, it could only generate accurate results on devices with dot matrix displays. For those with seven-segment displays, it could not locate the correct digits. Most of the data we collected were seven-segment digits, and thus we needed to improve it to make it functional well. We created a dataset of 10 kinds of seven segment fonts and retrained the Tesseract OCR with this dataset. These improvements in using Tesseract significantly improved the accuracy.

Experiment results suggested that our enhanced procedure outperforms the original processes from 6.18% to 62.12%. This procedure could be adopted (with human verification) to recognize the digits on the screen of medical devices with smartphone cameras.

5.2 Future Works

There are several possible extensions of our work.

First, for parameter searching of each step, we have just optimized each single parameter, without considering the cross effects between them. If the global optimization can be found correctly, the precision can be improved.

Second, our parameters are optimized on our dataset, and our dataset may have biases on choosing training images and test images, further improvement can be made through a larger and standard dataset.

Third, our parameters are are decided by all the images in our dataset and our dataset contains images in all kinds of conditions. All parameters can be optimized under certain image conditions, which will also generate a better result.

Fourth, our research focus on recognizing the digits on glucometers, how to generalize it for all medical devices can be another useful topic.

Fifth, all data and results are generated under the experimental condition, further practical implementations are also needed for mobile applications, such as applications for iOS and Android system.

Bibliography

[1] Mohammad Abdullah-Al-Wadud, Md Hasanul Kabir, M Dewan, and Oksam Chae. A dynamic histogram equalization for image contrast enhancement. *Consumer Electronics, IEEE Transactions on*, 53(2):593–600, 2007.

[2] Holger G Adelmann. Butterworth equations for homomorphic filtering of images. *Computers in Biology and Medicine*, 28(2):169–181, 1998.

[3] Kaoru Arakawa. Median filter based on fuzzy rules and its application to image restoration. *Fuzzy sets and systems*, 77(1):3–13, 1996.

[4] Gonzalo R Arce and José L Paredes. Recursive weighted median filters admitting negative weights and their optimization. *Signal Processing, IEEE Transactions on*, 48(3):768–779, 2000.

[5] Akira Asano, Kazuyoshi Itoh, and Yoshiki Ichioka. Optimization of the weighted median filter by learning. *Optics letters*, 16(3):168–170, 1991.

[6] Albert Baumgartner, Carsten Steger, Helmut Mayer, Wolfgang Eckstein, and Heinrich Ebner. Automatic road extraction based on multi-scale, grouping, and context. *Photogrammetric Engineering and Remote Sensing*, 65:777–786, 1999.

[7] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[8] Eugene Borovikov. A survey of modern optical character recognition techniques. *arXiv preprint arXiv:1412.4183*, 2014.

[9] Michael Calonder, Vincent Lepetit, Mustafa Ozuysal, Tomasz Trzcinski, Christoph Strecha, and Pascal Fua. Brief: Computing a local binary descriptor very fast. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1281–1298, 2012.

[10] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[11] Xiangrong Chen and Alan L Yuille. Detecting and reading text in natural scenes. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–366. IEEE, 2004.

[12] Adam Coates, Blake Carpenter, Carl Case, Sanjeev Satheesh, Bipin Suresh, Tao Wang, David J Wu, and Andrew Y Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 440–445. IEEE, 2011.

[13] Richard N Czerwinski, Douglas L Jones, and William D O'Brien Jr. Ultrasound speckle reduction by directional median filtering. In *Image Processing, 1995. Proceedings., International Conference on*, volume 1, pages 358–361. IEEE, 1995.

[14] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[15] K Delac, M Grgic, and Tomislav Kos. Sub-image homomorphic filtering technique for improving facial identification under difficult illumination conditions. In *International Conference on Systems, Signals and Image Processing*, volume 1, pages 21–23, 2006.

[16] Viet Cuong Dinh, Seong Soo Chun, Seungwook Cha, Hanjin Ryu, and Sanghoon Sull. An efficient method for text detection in video based on stroke width similarity. In *Computer Vision–ACCV 2007*, pages 200–209. Springer, 2007.

[17] Yiqiu Dong and Shufang Xu. A new directional weighted median filter for removal of random-valued impulse noise. *Signal Processing Letters, IEEE*, 14(3):193–196, 2007.

[18] Pete Doucette, Peggy Agouris, and Anthony Stefanidis. Automated road extraction from high resolution multispectral imagery. *Photogrammetric Engineering & Remote Sensing*, 70(12):1405–1416, 2004.

[19] Boris Epshtein, Eyal Ofek, and Yonatan Wexler. Detecting text in natural scenes with stroke width transform. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2963–2970. IEEE, 2010.

[20] Chun-Nian Fan and Fu-Yan Zhang. Homomorphic filtering based illumination normalization method for face recognition. *Pattern Recognition Letters*, 32(10):1468–1479, 2011.

[21] Mohd Awais Farooque and Jayant S Rohankar. Survey on various noises and techniques for denoising the color image. *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, 2(11):217–221, 2013.

[22] F Fisher. Digital camera for document acquisition. In *Proc. symposium on document image understanding technology*, pages 75–83, 2001.

[23] Julinda Gllavata, Ralph Ewerth, and Bernd Freisleben. Text detection in images based on unsupervised classification of high-frequency wavelet coefficients. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 1, pages 425–428. IEEE, 2004.

[24] Rafael C Gonzalez. *Digital image processing*. Pearson Education India, 2009.

[25] Berthold Horn. *Robot vision*. MIT press, 1986.

[26] Cheng-Hsiung Hsieh, Po-Chin Huang, and Sheng-Yung Hung. Noisy image restoration based on boundary resetting bdnd and median filtering with smallest window. *WSEAS Transactions on Signal Processing*, 5(5):178–187, 2009.

[27] Thomas S Huang, George J Yang, and Gregory Y Tang. A fast two-dimensional median filtering algorithm. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 27(1):13–18, 1979.

[28] Robert Hummel. Image enhancement by histogram transformation. *Computer graphics and image processing*, 6(2):184–195, 1977.

[29] Ayyaz Hussain, M Arfan Jaffar, and Anwar M Mirza. A hybrid image restoration approach: fuzzy logic and directional weighted median based uniform impulse noise removal. *Knowledge and Information Systems*, 24(1):77–90, 2010.

[30] Haidi Ibrahim. Adaptive switching median filter utilizing quantized window size to remove impulse noise from digital images. *AsianTransactions on Fundamentals of Electronics, Communication andMultimedia*, 2(1):1–6, 2012.

[31] Haidi Ibrahim and Nicholas Sia Pik Kong. Brightness preserving dynamic histogram equalization for image contrast enhancement. *Consumer Electronics, IEEE Transactions on*, 53(4):1752–1758, 2007.

[32] Iyad Jafar and Hao Ying. Multilevel component-based histogram equalization for enhancing the quality of grayscale images. In *Electro/Information Technology, 2007 IEEE International Conference on*, pages 563–568. IEEE, 2007.

[33] Anil K Jain and Bin Yu. Automatic text location in images and video frames. *Pattern recognition*, 31(12):2055–2076, 1998.

[34] V Jayaraj, D Ebenezer, and VR Vijayakumar. A noise free estimation switching median filter for detection and removal of impulse noise in images. *European Journal of Scientific Research*, 51(4):563–581, 2011.

[35] Cheolkon Jung, Qifeng Liu, and Joongkyu Kim. A stroke filter and its application to text localization. *Pattern Recognition Letters*, 30(2):114–122, 2009.

[36] David J Ketcham, Roger W Lowe, and JW Weber. Image enhancement techniques for cockpit displays. Technical report, DTIC Document, 1974.

[37] Hae-Kwang Kim. Efficient automatic text location method and content-based indexing and structuring of video database. *Journal of Visual Communication and Image Representation*, 7(4):336–344, 1996.

[38] Yeong-Taeg Kim. Contrast enhancement using brightness preserving bi-histogram equalization. *Consumer Electronics, IEEE Transactions on*, 43(1):1–8, 1997.

[39] Cemil Kirbas and Francis Quek. A review of vessel extraction techniques and algorithms. *ACM Computing Surveys (CSUR)*, 36(2):81–121, 2004.

[40] Sung-Jea Ko and Yong Hoon Lee. Center weighted median filters and their applications to image enhancement. *Circuits and Systems, IEEE Transactions on*, 38(9):984–993, 1991.

[41] Nicholas Sia Pik Kong and Haidi Ibrahim. Color image enhancement using brightness preserving dynamic histogram equalization. *Consumer Electronics, IEEE Transactions on*, 54(4):1962–1968, 2008.

[42] Nicholas Sia Pik Kong and Haidi Ibrahim. Improving the visual quality of abdominal magnetic resonance images using histogram equalization. In *Information Technology and Applications in Biomedicine, 2008. ITAB 2008. International Conference on*, pages 138–139. IEEE, 2008.

[43] VR Vijay Kumar, S Manikandan, PT Vanathi, P Kanagasabapathy, and D Ebenezer. Adaptive window length recursive weighted median filter for removing impulse noise in images with details preservation. *ECTI Transactions on Electrical Eng., Electronics, and Communications*, 6(1):73–80, 2008.

[44] Huiping Li, David Doermann, and Omid Kia. Automatic text detection and tracking in digital video. *Image Processing, IEEE Transactions on*, 9(1):147–156, 2000.

[45] Jian Liang, David Doermann, and Huiping Li. Camera-based analysis of text and documents: a survey. *International Journal of Document Analysis and Recognition (IJDAR)*, 7(2-3):84–104, 2005.

[46] Rainer Lienhart and Axel Wernicke. Localizing and segmenting text in images and videos. *Circuits and Systems for Video Technology, IEEE Transactions on*, 12(4):256–268, 2002.

[47] Tzu-Chao Lin. A new adaptive center weighted median filter for suppressing impulsive noise in images. *Information Sciences*, 177(4):1073–1087, 2007.

[48] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[49] Anand Mishra, Karteek Alahari, and CV Jawahar. Top-down and bottom-up cues for scene text recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2687–2694. IEEE, 2012.

[50] Lukas Neumann and Jiri Matas. A method for text localization and recognition in real-world images. In *Computer Vision–ACCV 2010*, pages 770–783. Springer, 2011.

[51] Pei-Eng Ng and Kai-Kuang Ma. A switching median filter with boundary discriminative noise detection for extremely corrupted images. *Image Processing, IEEE Transactions on*, 15(6):1506–1516, 2006.

[52] Chandra Sekhar Panda and Srikanta Patnaik. Filtering corrupted image and edge detection in restored grayscale image using derivative filters. *International Journal of Image Processing (IJIP)*, 3(3):105–119, 2009.

[53] Seokil Park, Jongshil Lee, Jayl Koo, Ohsang Kwon, and Seunghclng Hong. Adaptive tracking algorithm based on direction field using ml estimation in angiogram. In *TENCON'97. IEEE Region 10 Annual Conference. Speech and Image Technologies for Computing and Telecommunications., Proceedings of IEEE*, volume 2, pages 671–675. IEEE, 1997.

[54] Jim R Parker. *Algorithms for image processing and computer vision*. John Wiley & Sons, 2010.

[55] Maria Petrou and Costas Petrou. *Image processing: the fundamentals*. John Wiley & Sons, 2010.

[56] Stephen M Pizer, E Philip Amburn, John D Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart ter Haar Romeny, John B Zimmerman, and Karel Zuiderveld. Adaptive histogram equalization and its variations. *Computer vision, graphics, and image processing*, 39(3):355–368, 1987.

[57] Lindi J Quackenbush. A review of techniques for extracting linear features from imagery. *Photogrammetric Engineering & Remote Sensing*, 70(12):1383–1392, 2004.

[58] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2564–2571. IEEE, 2011.

[59] Sami Abdulla Mohsen Saleh and Haidi Ibrahim. Mathematical equations for homomorphic filtering in frequency domain: a literature survey. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 74–77, 2012.

[60] Nyamlkhagva Sengee and Heng Kook Choi. Brightness preserving weight clustering histogram equalization. *Consumer Electronics, IEEE Transactions on*, 54(3):1329–1337, 2008.

[61] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[62] David L Smith, Jeff Field, and Erik Learned-Miller. Enforcing similarity constraints with integer programming for better scene text recognition. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 73–80. IEEE, 2011.

[63] Ray Smith. An overview of the tesseract ocr engine. In *icdar*, pages 629–633. IEEE, 2007.

[64] Krishna Subramanian, Prem Natarajan, Michael Decerbo, and David Castañòn. Character-stroke detection for text-localization and extraction. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 1, pages 33–37. IEEE, 2007.

[65] Tong Sun, Moncef Gabbouj, and Yrjö Neuvo. Center weighted median filters: some properties and their applications in image processing. *Signal Processing*, 35(3):213–229, 1994.

[66] Ying Sun. Automated identification of vessel contours in coronary arteriograms by an adaptive tracking algorithm. *Medical Imaging, IEEE Transactions on*, 8(1):78–88, 1989.

[67] Engin Tola, Andrea Fossati, Christoph Strecha, and Pascal Fua. Large occlusion completion using normal maps. In *Asian Conference on Computer Vision*, number EPFL-CONF-164033, 2010.

[68] Scott E Umbaugh. *Computer imaging: digital image analysis and processing*. CRC press, 2005.

[69] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3):480–492, 2012.

[70] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1457–1464. IEEE, 2011.

[71] Kai Wang and Serge Belongie. *Word spotting in the wild*. Springer, 2010.

[72] Yu Wang, Qian Chen, and Baomin Zhang. Image enhancement based on equal area dualistic sub-image histogram equalization method. *Consumer Electronics, IEEE Transactions on*, 45(1):68–75, 1999.

[73] Jerod J Weinman, Erik Learned-Miller, and Allen R Hanson. Scene text recognition using similarity and a lexicon with sparse belief propagation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(10):1733–1746, 2009.

[74] K Wongsritong, K Kittayaruasiriwat, F Cheevasuvit, K Dejhan, and A Somboonkaew. Contrast enhancement using multipeak histogram equalization with brightness preserving. In *Circuits and Systems, 1998. IEEE APCCAS 1998. The 1998 IEEE Asia-Pacific Conference on*, pages 455–458. IEEE, 1998.

[75] Ruikang Yang, Lin Yin, Moncef Gabbouj, Jaakko Astola, and Yrjo Neuvo. Optimal weighted median filtering under structural constraints. *Signal Processing, IEEE Transactions on*, 43(3):591–604, 1995.

[76] LIU Yangxing and Takeshi IKENAGA. A contour-based robust algorithm for text detection in color images. *IEICE transactions on information and systems*, 89(3):1221–1230, 2006.

[77] Qixiang Ye, Qingming Huang, Wen Gao, and Debin Zhao. Fast and robust text detection in images and video frames. *Image and Vision Computing*, 23(6):565–576, 2005.

[78] Chucai Yi, Xiaodong Yang, and Yingli Tian. Feature representations for scene text character recognition: A comparative study. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 907–911. IEEE, 2013.

[79] Qi Zheng, Kai Chen, Yi Zhou, Congcong Gu, and Haibing Guan. Text localization and recognition in complex scenes using local features. In *Computer Vision–ACCV 2010*, pages 121–132. Springer, 2011.

Curriculum Vitae

Name:	Chang Liu
Post-Secondary Education and Degrees:	Peking University Beijing, China 2011 - 2014 M.Sc. University of Western Ontario London, ON 2014 - 2016 M.Sc.
Honours and Awards:	WSGR 2014 - 2015
Related Work Experience:	Teaching Assistant The University of Western Ontario 2014 - 2015