Spring 2013

Image pre-processing to improve data matrix barcode read rates

Nathan P. Brouwer
University of New Hampshire, Durham

IMAGE PRE-PROCESSING TO IMPROVE DATA MATRIX BARCODE READ RATES

BY

NATHAN P. BROUWER

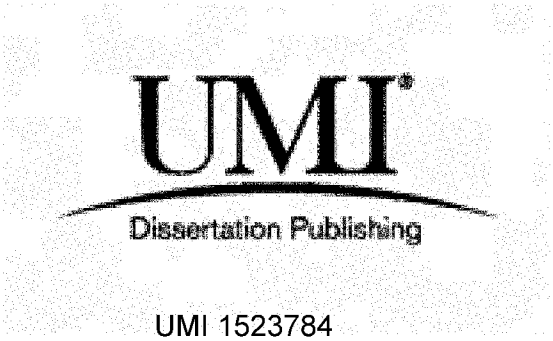
B.S. Electrical Engineering 2011
University of New Hampshire

THESIS

Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Master of Science
in
Electrical Engineering

May 2013



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

This thesis has been examined and approved.

Thesis Director, Richard A. Messner
Associate Professor, Electrical and Computer Engineering

Michael J. Carter
Associate Professor, Electrical and Computer Engineering

Wayne J. Smith
Senior Lecturer, Electrical and Computer Engineering

Dragan Vidacic
IDEXX Laboratories

Date: _____

Acknowledgements

I would like to express my gratitude to my research adviser Dr. Richard Messner who constantly pushed me to take ownership on this thesis and make it my own. His insights allowed the project to develop and evolve into something I can really be proud of.

Many thanks to the faculty and staff of the UNH Department of Electrical Engineering for taking me in as an undergraduate, allowing me to grow and develop as a student, and offering encouragement all along the way. I am very appreciative to be funded for my graduate career and for the opportunity to teach an undergraduate class; a most humbling and gratifying experience. I would like to thank my thesis committee members: Dr. Mike Carter, Dr. Wayne Smith, and Dr. Dragan Vidacic for their careful review of my thesis document. I would also like to acknowledge Dr. Kent Chamberlin for giving me his full support and many book recommendations throughout my graduate career. Thanks to the saint Mrs. Kathy Reynolds for countless help with all the inevitable administrative matters.

Special thanks to my parents and brothers, whose unending love and support carried me through even the most trying times. Loving shout out to the UNH Men's Volleyball team, who have become family and preserved my sanity.

Table of Contents

List of Figures

List of Tables

Abstract

IMAGE PRE-PROCESSING TO IMPROVE
DATA MATRIX BARCODE READ RATES

BY

Nathan Brouwer
University of New Hampshire, May 2013

The main goal of this study is to research image processing methods in attempts to develop a robust approach to image pre-preprocessing of Data Matrix barcode images that will improve barcode read rates in an open source fashion. This is demonstrated by element state classification to re-create the ideal binary matrix corresponding to the intended barcode layout through pattern recognition theory.

The research consisted of implementing and evaluating the effectiveness of many image processing algorithms types, as well as evaluating key features that clearly delineate different element states. The algorithms developed highlight the use of morphological erosion and region growing for object segmentation and edge analysis and Fisher's Linear Discriminant as a means for element classification.

The results demonstrate successful barcode binarization for ideal barcodes with improved read rates in most cases. The techniques developed here provide ground work for a test bed environment to continue improvements by analyzing non-ideal barcodes for additional robustness.

Chapter 1

Introduction

1.1 Background

We truly live in a technology age. Today, over half of the US population owns a smart phone (16), an individualized personal computer loaded with features and applications ("apps") carried around at all times as a convenient way to stay connected to the world. To certain business strategists, this means over 150 million cameras flooding the country, ready to snap a picture in a moments notice at anything worthwhile. Companies all over have already begun adding 2D barcodes to merchandise tags, posters, billboards, and other printed advertisements. Incentives are offered such as coupons, discounts, and additional information to those individuals who snap a picture of the barcode, use an app to decode its contents, and receive the benefit directly or as a link to a website. To keep customer interest levels elevated, an imperative part of the process is high levels of successful barcode decoding.

Original linear barcodes were read by optical scanners, but newer two dimensional barcode reading requires quality camera hardware for image acquisition and robust software processing to extract the encoded barcode message from the image. Cameras in these smartphones are improving constantly, but due to size limitations, they will always be behind the cutting edge camera technology. Furthermore, continual advancements in processing power clearly indicate that software gains will develop much more quickly than hardware gains. This thesis involves development of image processing software algorithms and theory to improve read rates for Data Matrix barcodes, a commonly used 2D barcode scheme.

In the simplest sense, a barcode is a series of black and white elements that correspond to logical ones and zeros. These form code words that in turn code an intended message. This message is usually used to label the item the barcode is printed on or provide a link to an advertisement or discount. In computer vision system applications, barcodes are used as machine readable labels to convey information about moving parts in the system. It is a tool that allows the design of more complex machine vision systems by allowing the processor direct access to additional input that would not be readily available otherwise. Barcodes began as a linear or one-dimensional form of vertical white and black lines, but now have evolved into a two-dimensional format for greater efficiency. Figure 1.1 shows some examples of both 1D and 2D barcodes. Current technology for decoding the raw binary bit data of barcode schemes is mature, but cost requirements have led away from optical scanners to cheaper CCD and CMOS transistor array sensors for camera acquisition. New image processing methods are evolving to extract the binary string of code words from the image pixel intensity data.

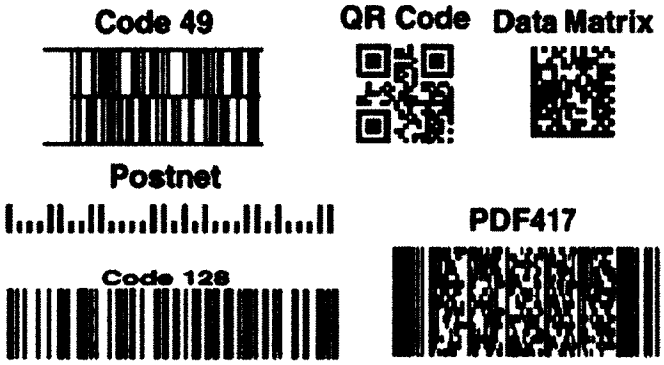


Figure 1.1: Examples of Various Barcodes

Barcode technology is a machine readable representation of data usually encoding a string of characters and/or numbers. Their purpose is to aid in tasks generically lumped into a broad category called automatic identification and data capture (AIDC). Back

in 1974, the first commercial barcode was an optically machine readable series of white and black vertical lines to represent the identification of a pack of Wrigleys chewing gum for supermarket sales (18). Barcodes have evolved from these linear codes into geometrical patterns in two dimensions including arrangements of rectangles, dots, or hexagons. This relatively newer coding scheme allows for a much greater amount of information per unit area, granting the possibility for numerous novel applications such as universal labeling for printed media or identification for commercial and industrial components.

Data Matrix is one of the prominent two-dimensional coding schemes, consisting of a square or rectangular barcode format that can encode up to 1556 bytes or 2335 alphanumeric characters. A key feature that can be exploited is the Data Matrix border coding scheme that is always identical. Disregarding image inversion, the left and bottom edges will always be white, and the top and right edges will always be alternating white and black. These border regions are standard reference objects called the L-shape finder pattern and the timing pattern (8). The entire set of technical standards for the Data Matrix scheme is called ECC 200 (14), which describes the bit encoding process and necessitates the use of Reed-Solomon error correction codes that allow for a successful recovery and reconstruction of the original code with up to 30% damage to the barcode. For these reasons, the Data Matrix coding scheme is very useful and often is the choice for many commercial and industrial applications.

The technology age brought explosive leaps to computing speed and camera technology. For this reason, barcode readers have departed from traditional optical scanners and have gravitated towards using cameras and computer imaging software to acquire image representations of a printed barcode and decode or "read" the barcode in the image. Camera technological specifics can vary but essentially light enters the camera lens which causes charge to build up on an array of sensors. The array is then sampled to create a two dimensional matrix of pixels called an image with pixel intensity proportional to the accumulated charge. Barcode reading software will then process an image to determine the

intended layout of the white and black coded dots or squares called elements, finally applying algorithms according to the ECC 200 standards to convert the binary representation back into the original alphanumeric string.

Algorithms to decode an ideal Data Matrix barcode (perfect binary data) are mature and no longer require significant improvement, but the steps to accurately convert the visual pixel information representing the physically printed dots into the coded binary array is an unperfected process. Few existing image analysis methods are capable of identifying codes under uncertain conditions, such as varying illumination, different sizes, or orientations. Several closed source commercial products are available that successfully decode barcodes from images in many conditions, but the intent here is to design techniques for improving this conversion process for non-ideal conditions in an open source fashion (15). Besides the choice of camera, another design aspect of great importance to consider is how to choose light for illuminating the barcode label on a moving part. These design aspects will be discussed in the methodology section.

1.2 Current Work

"It's all in the software" states Voosen (19). He means that because of the advances in main stream processors, the trends in machine vision lead towards innovations in the software processing over imaging hardware. Every machine vision system must acquire images, process images, and communicate results. While image hardware is evolving linearly but slowly, developing sophisticated machine vision niche applications can be tackled using higher-level image processing packages without the need to resort to low-level programming and hardware optimization. Every imaging application will require acquisition, but instead of focusing on the front end, the increasing speed of processing and communication of results can exceed hardware speed-up capabilities. Today's easy-to-use vision software usually consists of an application prototyping environment, automatic code

generation, and an intuitive graphical programming interface; and this type of software can handle complex problems with the ease of user customization and control. In summary, while image hardware developments are not over, the future of evolving machine vision applications will primarily be handled in the software world.

Falas (2) discusses the potential of using quick processing barcode reading software to enable quick data acquisition. Normally data entry in phones is limited by the keypad, or keyboard in today's smart phones, but by allowing them to scan a barcode quickly, information such as contact info, warnings, deals, and coupons can be conveyed. Even links to the internet allow for the posting of larger information bulletins or allow for marketing various websites. By using cheaper cameras on phones for image acquisition, it forces several common problems to be encountered such as various types of noise, blur, uneven lighting, extraneous background information, image tilt and skew, and rotational deformities (9). Given these problems, barcode reading software must rise to the challenge and be more robust to handle the many problems encountered in unstructured environments.

When acquiring image data via a cell phone camera it is necessary to pay attention to the pre-processing stages. Some of the steps involved in pre-processing include gray-scaling quantization, histogram stretching, local (sometimes global) adaptive thresholding, noise filtering, cropping, rotation correction, and tilt correction. While all these steps may not be crucial in every situation, it is likely that including some form of these processes will add a level of robustness to any barcode decoding algorithm (20). Empirical results show that the trade off that exists with any amount of image pre-processing implementation is decoding ability versus decoding time. They are inversely related, but due to timing constraints of barcode systems, decoding time should be minimized as much as possible while maintaining successful barcode reads. This is possible through optimization of each pre-processing step.

Microscan (14) has released a technology paper that compares quality parameters for Data Matrix symbol verification and they also describe many of the common

parameters and distortion possibilities when these variables are adjusted. This is significant because it allows the decoding expert to know ahead of time many of the anticipated problems and by using this advanced knowledge it is possible to tune algorithms to deal with these issues beforehand. Some of the parameters mentioned in the Microscan document, shown in figure 1.2, include angle of distortion, axial non-uniformity, cell contrast, and dot-center offset. It becomes clear that a simple technique such as static thresholding may not be particularly effective when dealing with images with low contrast.

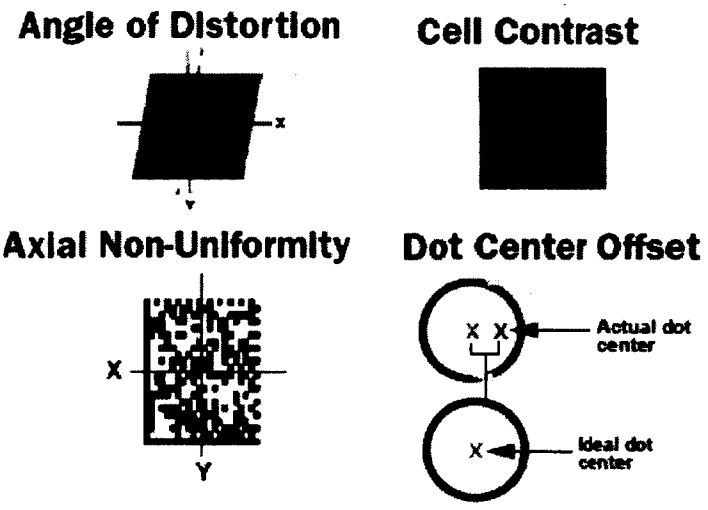


Figure 1.2: Examples of Parameter Distortions

Attempting better methods for decoding Data Matrix barcodes is not a new idea. Pârvu (17) created a method for fast detection and decoding using contour tracing techniques for identification followed by a couple fitting steps. Independent components are identified and then by using contour corner analysis, the accepted components merge to form L-shaped markers around element corners which are progressively built up using consecutive fitting and re-fitting steps. Though a successful method, a major drawback of the algorithm method was the weakness in handling inverted contrast codes. It was suggested to use the same method but search for inner contours, but the approach was not

successfully tested.

Other researchers have claimed various levels of success with barcodes as well. Liu (13) developed a method of barcode localization using boundary tracking and the Radon transform. Their method touts speed and robustness of localization, but their barcode pixel classification for accurate decoding concedes room for improvement. Formiga (4) has also developed techniques to recognize Data Matrix barcodes under scaling, rotation, and cylindrical warping. While not directly aiding in the decoding process, these pre-processing methods are invaluable since they add levels of robustness to decoding algorithms in the face of various parameter distortions.

1.3 Proposed Methods

There is commercially available Data Matrix decoding software on the market, but cost and timing considerations again make the software operating independently unfeasible. The overall goal of this work is to develop image pre-processing strategies to enable a barcoding system to be as robust and as fast as possible.

The hypothesis is that it is possible to create a robust barcoding system to read quality barcode images by employing image pre-processing algorithms coupled with an inexpensive or open source Data Matrix decoding library. The proposed method will capitalize on pattern recognition theory by employing and adapting standard image processing and feature extraction algorithms to pre-process the barcode image. The algorithms will include a classifier that uses statistical analysis methods to reliably and accurately recreate the perfect idealized barcode from the raw acquisition. Utilizing standard Data Matrix decoding software to decode the synthetically created images makes the decoding trivial and will improve reading capabilities while reducing barcode read times.

After performing basic image processing algorithms to processes the images into a standard format, it is then possible to take advantage of the specific barcode geometry

to locate the center of each element, and through a segmentation method, grow a region of pixels that best describes that element. Using the new region of interest (ROI), it will be a matter of extracting descriptive and separable features that can be used in statistical analysis. By applying Bayes Theorem which describes the ideal classifier and estimating probabilistic parameters, Fishers Linear Discriminant will provide a method in determining a valuable feature space boundary. Using this boundary as the classifier threshold, it will be possible to accurately determine the state of each element in the barcode and therefore recreate the perfect binarized barcode for each case.

The previous methods describe the barcode binarization process from a clean ideal environment. Unfortunately, the reader needs to work in a realistic "dirty" environment in which additional challenges will likely include blur, non-uniform lighting, differing contrast, obstructions, and physical damage to the barcode. A significant goal for this thesis is to lay the ground work for a test bed to explore many of the possible deformities with real barcode images and begin to develop strategies to combat the most common problems. By developing ideal condition pre-processing binarization methods, it will set the stage for future work in realistic non-ideal conditions.

1.4 Research Objectives

There are numerous non-trivial steps to robustly acquire and accurately decode image representations of 2D barcodes. There are several described methods to process images after acquisition for localization purposes (13). Liu, et al., proposed a technique for Data Matrix localization based on boundary tracking and the Radon Transform from a heuristic approach. Furthermore, many researchers have discovered robust methods to find barcodes in complex backgrounds that are rotation and scale invariant (12). The system proposed by Lin is comprised of three stages: barcode course-to-fine segmentation extraction, max-min differencing for barcode enhancement, and multi-directional decoding. GS1

publishes clearly established technical guidelines for the conversion of the binary element data to the message string (8). The ECC 200 Standards describe exactly how the encoding process works and how the Reed-Solomon error correction code should function.

The scope of this project, therefore, involves the image pre-processing methods to convert image data into binary element data to make the decoding process trivial. The investigation is to determine defining characteristics of the cells unique to black and white elements in the image as a whole. The improved techniques developed here will correctly recreate the intended barcode layout for decoding purposes in a robust manner to decrease or eliminate reading errors. The goal of the project then is to effectively recreate synthetic barcode images that contain the same message as that of the original sample barcodes.

1.5 Organization

This thesis is organized into five chapters. This first chapter serves as an introduction to the problem, an overview of current trends in barcode technology, and sets the stage for this study. The second chapter serves as an introduction to pattern recognition theory and provides a background for the many classification and image processing methods that are evaluated. The third chapter describes the algorithm development process and offers an interpretation of the overall effectiveness of many of the techniques. The third chapter will conclude with the final design implementation and offer specific reasoning behind the decision making process. The fourth chapter will analyze the specific read rate results of the image processing algorithms and present conclusions. Finally, the fifth chapter will provide a framework for a test bed for future work and give a definite direction to develop a test bed to analyze "dirty" environment images.

Chapter 2

Background

2.1 Pattern Classification Theory

2.1.1 Pattern Recognition Systems

Pattern recognition systems have been developed to aid in the automization of data processing. A system that can intelligently process and analyze raw input into a system of "categories" is useful because it saves significant time and man power. When the input data increases significantly, an efficient and robust system drastically reduces computational time when compared to a brute force method. By consolidating an overflow of raw data into a simple collected report, the user is now able to do more with his/her time. The following flow chart shows the basic order of processes for an automatic pattern recognition system.

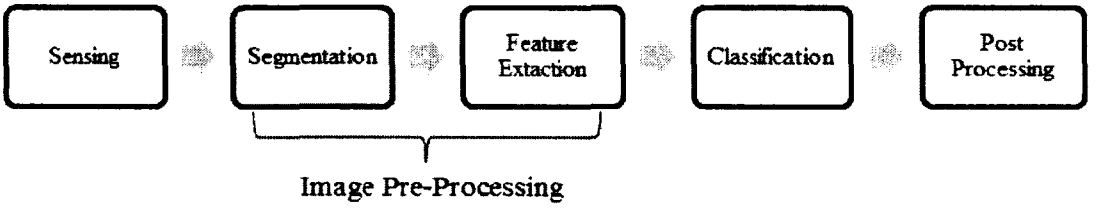


Figure 2.1: Anticipated Pattern Recognition System Flow Chart

2.1.1.1 Pattern Recognition

A pattern is an arrangement of descriptors. A feature is often used to denote a descriptor which can have a broad variety from boundary/border descriptors to

regional or textural descriptors. A pattern class is a family of patterns that share some common properties. Pattern recognition by machines involve techniques for assigning patterns to their respective classes automatically and with as little human intervention as possible (7). In machine learning, pattern recognition is the assignment of a label to a given input value.

Before developing a theory for a machine-vision pattern recognition system, it is worthwhile to briefly assess one of the greatest natural pattern recognition systems, the human brain. It is believed that through evolution, the neural pathways have developed a complex process to naturally and automatically extract features from visual information and other sense stimuli, and use a developed neural network of connections to form correlations between similar data. In particular, it has been shown that the human brain is highly skilled at classifying patterns (5). In regards to developing a machine vision application, an obvious yet non-trivial approach would be to create a neural network model for deciphering barcode element layouts. However, this may be over complicated and more computationally intensive than necessary. In addition, a neural network approach would require extensive training of data sets. After all, the brain takes several years of "training" data before the individual enters society as a peer.

In machine processing, the goal of pattern recognition is to build a model that can accurately classify a set of test data. This process usually involves at least some form of a three step approach including segmentation, feature extraction, and classification. The thesis goal is to perform some pre-processing to a given set of input that unlocks (or extracts) additional information beyond the raw data itself. In this barcode problem the input will be a set of gray scale images (i.e. a matrix of 8 bit pixel information) and the pre-processing will require some combination of segmentation and feature extraction using the pixel information and other known constraints. The ultimate pattern to be recognized is the matrix layout of black and white elements in a given barcode image. If the pattern is already known, a synthetic binary image can be created and decoded with high efficiency

by the barcode reader.

2.1.1.2 Sensing

Sensing has to deal with the input to a pattern recognition system. It generally involves a transducer to convert a type of physical form into electrical energy. An example of this would be to convert the number of photons hitting an area into a voltage equivalent that is proportional to the number of photons impingent on the sensor area. The focus here will be on using a camera to convert visual representations of barcodes into an image of that barcode. The reader should be aware that quality of the sensing is a physical limitation because the rest of the process deals with the data synthesized here (7). There are many ways to improve the quality, sensitivity, accuracy of a camera, but this is beyond the scope of the project. The camera that was chosen was a mid-quality USB snake camera with built in LED lights. Initial testing revealed the built-in lights caused significant specular reflection, and therefore a diffuse red LED bulb was used in its place held in position by a helping hands stand. The camera acquisition system that is used in this thesis is adequate for our investigation. The image data acquired is stored per pixel as 8 bit binary code word. Thus, there are only 256 values of gray levels that can be represented. The range is from 00 Hex (representing black) to FF Hex (representing white).

2.1.1.3 Segmentation

A typical image appears on a complex background, usually with multiple objects to be identified. Thus, the classification system needs to be able to segment the image by recognizing individual patterns and identifying the physical limits of where the object exists in the image. In general, classification is only as good as both segmentation (to reduce data) and the features that can be identified in the segmented region. Segmentation is one of the most prolific problems in pattern recognition in that there is a wide variety and combination of pixel differences between similar objects in an image. Therefore, higher-

level methods and approaches are required. It is believed the human brain is so successful in segmenting pieces of images is because it can focus on chunks of the image and quickly correlate it against a vast memory bank to see if it seems like a recognizable object (5). This requires that significant prior knowledge about the problem be known in the classification system design, and it makes it very difficult to handle an unanticipated change to the environment. Several image processing techniques involving segmentation will be described later in this thesis in the methodology section.

2.1.1.4 Feature Extraction

Once objects have been segmented into various regions or categories, all similar object types will have identifying patterns, or combinations of descriptors. It is in this combination of features that hold the key to the objects categorical membership because each object will have its own set of features with individual differences. Objects from the same category will inevitably have some amount of varying feature values. A major research goal is identify specific quantifiable feature data that can be used for feature extraction because the in-class feature differences are small compared to those of between-class categories. A design aspect for a classifier is the determination of a set of features to be extracted that will highlight individual patterns and create a well separated set of states in feature space. Features can be anything from the length of a tail of a fish to the average intensity of a certain area in an image. All states must have measurable features, but it is possible and often desirable to have certain states with unique feature values to aid in the classification (1). These distinguishing features are invariant to irrelevant transformations such as absolute location, scale, or orientation and even differences in element shape, contrast intensity, or uniformity.

Three common types of pattern arrangements used for quantitative descriptors are vectors, strings, and trees. Although strings of associated features and trees of hierarchical descriptors have their uses, vector notation holds the most promise in this appli-

cation. This is the case due to the ease of representation of multiple features using a single feature vector and then applying simple linear algebra to calculate a decision boundary in feature space. This will be discussed further in section 2.1.2.2 Fishers Linear Discriminant. Pattern vectors are represented in the following form

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

where each component, x_i represents the ith descriptor and n is the total number of such descriptors associated with the pattern. This form allows for a powerful classification technique called discriminant analysis, which will be discussed later.

Feature extraction represents a loss of information. In general, image processing techniques modify an input image to produce an output image, either preserving original information or at the very least producing a modified version of the pixel data. Feature extraction, such as finding peaks and valleys of intensity via a histogram, cannot tell you about the spatial locations of those pixels. Thus, each object becomes tied to its associated features rather than its spatial pixel values.

2.1.1.5 Classification

Classification theory is a broad area which describes a decision making process to place inputs into distinct categories. Classification presents an even further loss of information than feature extraction. A classifier takes necessary features and makes a crucial decision about whether a pattern exists or what category to assign to the input. It is fundamentally an information reduction process because one cannot reconstruct the pattern only knowing its categorical membership (1) because of the variability in individual

members in a broad category.

In this case, classification will reduce thousands of pixels down to a series of a few binary elements. This is preferable although it is important to realize the ramifications of incorrect classification. It is therefore imperative to be very thorough at this stage. Bayesian Decision Theory is a fundamental statistical approach to pattern classification by taking known information about the system to create a probability model for maximum success in classification. This theory along with linear discriminant analysis is used extensively to develop a robust classifier. In the case of unsuccessful reads, classification errors must exist. A robust approach could be to return to the pre-processing stages and tweak the classifier parameters, resulting in the inversion of certain element values that correspond to the level of least confidence of the classifier. This process could continue iterative classification and decoding attempts until the reader is successful.

2.1.1.6 Post Processing

Post processing involves sorting through all the processed data and classification results to display the information to the user in the most efficiently desired manner. The methods used will vary significantly depending on the application and the user's specifications. A barcode reading application simply requires the output string message be presented to the next stage of the system.

2.1.2 Statistical Methods

2.1.2.1 Bayes Formula

The probabilistic approach to maximize correct classification is described by Bayes Theorem which states that the posterior probability is equal to the likelihood function times the prior probability scaled by the evidence factor (1). This is mathematically stated in equation 2.1. It is an ideal equation that can never be fully implemented because the posterior probability cannot be known ahead of time. However, it is still useful to

discuss the theory because the posterior probability can often be estimated to obtain good classification results.

$$P(w_j|x) = \frac{p(x|w_j)P(w_j)}{P(x)} \tag{2.1}$$

$p(x|w_j)$ is the likelihood that feature x will be observed, given that the true state is w_j and $P(w_j)$ is the priori probability that the next sample will be of state w_j. $P(w_j|x)$ is the probability that the true state is w_j given that feature x has been observed. It follows logically that in a two state system, given x, whichever state probability is greater should be the decision rule. In terms of a classification theory, the goal is to minimize incorrect decisions because they incur a loss to the system. The loss function can be stated as $\lambda(\alpha_i|w_j)$; the loss incurred for taking action α_i given that the true state is w_j. The conditional loss R of making decision α_i given feature x can be described in equation 2.2 as the Bayes decision rule.

$$R(\alpha_i|x) = \sum_{j=0}^{c} \lambda(\alpha_i|w_j)P(w_j|x) \tag{2.2}$$

In a two-category or binary classification system, the likelihood ratio is expressed as $\frac{p(x|w_1)}{p(x|w_2)}$, which focuses on the x-dependence of the probability densities. Using the likelihood ratio as well as experimentally attained prior probability densities, a decision boundary of equal probability can be created to minimize our classification error.

Bayes Theorem is an ideal situation that cannot be described exactly because the likelihood and prior probabilities cannot be known, but they can be estimated with prior data. If the likelihood function $p(x|w_j)$ was known, then the posterior $P(w_j|x)$ could be calculated (The probability that if feature x is observed, the true state is actually w_j). This will be applied later, by using border element pixel probability distributions to estimate the posterior ($P(w_j|x)$) distribution, and then attempt to determine the most likely state of each element w based on features x in the present case.

2.1.2.2 Fishers Linear Discriminant

In a fundamental paper published in 1939, Fisher (3) describes the original framework for statistical methods of discriminant analysis. He recognized three types of iris flowers (Iris *setosa*, *virginica*, and *versicolor*) by measuring the width and lengths of their petals. Essentially each flower was described only by the two measurements, and using a minimum distance classifier and discriminant functions, boundaries in feature space can be created for classification. Figure 2.2 shows quadratic boundaries created based on Fisher's data. These boundaries are based on minimum distance classification which maximizes the distance between feature means, and in practice works very well given that the feature means are large compared to the variance or randomness of each class with respect to its mean.

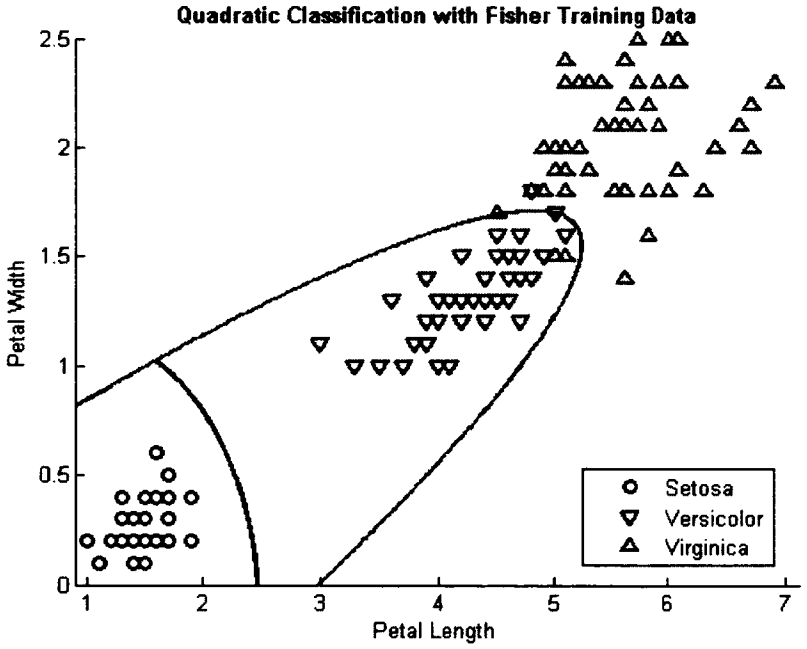


Figure 2.2: Example of Discriminant Analysis

With a set of features and estimated underlying probability densities, all that remains is the task of choosing a decision boundary in feature space as the classifier.

To avoid several computational difficulties, it is assumed that density distributions are Gaussian in nature and can be entirely described by their mean and a variance. In general, the criterion function for classification will be to minimize the sample risk, the average loss incurred in classifying a set of training samples. Overall, we are looking to gain the best possible classification with all types of input. By looking at the linear system of equations $y = w^t x$, where y is the set of samples, w^t is a row vector of possible states or categories, and x is a column vector of features, we will see each feature has something to offer about the current state of sample y. Solving the system of equations simultaneously, we arrive at a covariance matrix for all features (called a scatter matrix) that describes the variance relationship between features and states. If features are statistically independent (which is generally desirable for classification), it is optimal to maximize the within-class scatter (only values down the top left to bottom right diagonal, like an identity matrix) and minimize the between-class scatter (all off diagonal variances). In the ideal independent case, it is a form of an eigenvalue equation. In feature space, this will maximize distance between clusters of feature samples, while minimizing the amount of scatter or variance in each state cluster. Essentially, it is important to limit the effect features have on each other, or the feature-to-feature dependence, while maximizing the feature to state dependence. In doing so, we create a linear boundary in feature space that is the minimum squared-error boundary that minimizes risk (incorrect classifications). This multidimensional boundary maximizes the distance between feature means, but includes boundary shifts for unequal within-class variance differences.

In a two state system, Fishers linear discriminant boundary can be described as $w^t x + w_0 = 0$ where $w = \Sigma^{-1}(\mu_1 - \mu_2)$. w_0 is a constant involving w and the prior probabilities, while we use sample means and the sample covariance matrix to estimate μ_i and Σ. This means that given a set of features x with known distributions, a likely decision boundary can be estimated to effectively classify any sample y into the most likely state w. This is powerful classification tool because it allows for a state to be described by

multiple features, but is often difficult or computationally expensive to implement. Fishers Linear Discriminant (FLD) is used as the one of the classification methods in determining barcode elements.

2.1.3 Image Pre-Processing

Image pre-processing can involve any number of steps, but essentially the goal is to aid the classification process. Pre-processing generally includes image segmentation and feature extraction and it may be used to process and extract additional information, or format and change the data to enable more successful future processing. For example, it may be necessary to filter raw data to get rid of unnecessary spikes or noise variations. By processing the image to determine what kind of objects exist and what patterns they exhibit, the analysis can help determine the appropriate classification method or approach, which may be different depending on what is contained in the image. In this case, the goal is image binarization therefore pre-processing will most likely result in feature extraction for the direct purpose of element state classification.

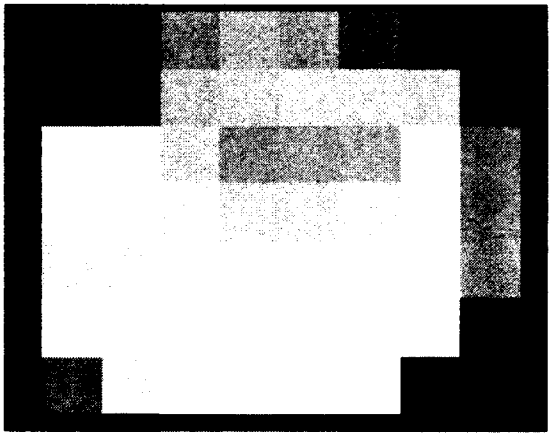
Juett (11) uses a bottom hat filter to pre-process images to locate barcodes in images. The author uses grayscale enhancements and contrast stretching, then a specialized high pass filter for bar highlighting, and finally a bottom hat filter to accentuate the barcode. These pre-processing steps are preliminary ones, but are essential to subsequent steps involving density region analysis and finally barcode localization, which require clear barcode lines created by the bottom hat filter. This example shows the necessity of pre-processing to prepare the image for later algorithms to ensure desired results.

2.2 Image Processing Techniques

The following subsections will describe the basic theory behind many of the image processing techniques used in subsequent chapters.

2.2.1 Neighborhood Operators

In image acquisition systems, sensor arrays measure the amount of incoming photons that are collected by a sensor of set area. A lens precedes that sensor area so that the image is properly focused onto the sensor. In this way, the arrays build up charge that corresponds to pixel intensity values. Unless the lens and sensor arrays are highly calibrated, there will be an unavoidable amount of randomness to the path of the photons, which can result in neighboring pixels accumulating the charge. Therefore, pixel values have a certain relationship of dependency with neighboring pixels, this is why supposedly uniform areas will have a fluctuation of pixel values across the region. This fact can be seen in figure 2.3, where a supposedly uniform white element in fact contains a wide range of pixel values.



104	131	181	192	186	164	147	63
113	157	195	200	205	206	204	136
236	231	203	186	191	193	234	183
226	218	215	203	205	214	226	182
214	214	221	236	226	218	255	184
223	253	254	255	255	255	216	96
169	209	233	251	253	244	41	38

Figure 2.3: Pixel Values for "Uniform" White Element

This introduces the reader to the concept of neighborhood processing. A pixel's neighbors are considered to be the pixels directly adjacent to the given pixel. For a given pixel at coordinates (x, y), a 4-neighbor adjacency considers pixels directly above, below, right, and left of a pixel, i.e. $(x+1, y), (x-1, y), (x, y+1), (x, y-1)$. An 8-neighbor adjacency also includes the 4 diagonal neighbors to the northeast, northwest, southeast,

and southwest $((x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1))$, as shown in 2.4.

Figure 2.4: Eight Neighborhood Adjacency

$(x-1, y+1)$	$(x, y+1)$	$(x+1, y+1)$
$(x-1, y)$	(x, y)	$(x+1, y)$
$(x-1, y-1)$	$(x, y-1)$	$(x+1, y-1)$

By using the neighborhood of a pixel, it is possible and often desirable, to preform processing by using neighborhood operators. In this way, we can do many different types of operations to improve the raw acquired data or extract additional information, such as finding borders and edges, or pixel connectivity.

2.2.2 Filtering

Image filtering at the most basic level takes the concept of neighborhood operators and acts on an image as a whole. In spatial filtering, the concept of a kernel is a type of transformation function, it describes the output's dependency on the input. For example, a common kernel or mask is a $3X3$ neighborhood, which describes the center output pixel as a function of that pixel's value and its corresponding 8-neighbor adjacency. The filtering operation would be carried out by starting the kernel in one corner of the image and preforming a neighborhood operation, such as pixel by pixel multiplication and summing, and then repeating the operation at every pixel by sliding the kernel across the entire image. Mathematically, it is the convolution of the image I with the kernel k.

Convolution:

$$I_{filtered} = (I \otimes k)[n] = \sum_{m=-\infty}^{\infty} I[m] * k[n-m]$$

One example of a basic linear filtering operation is an averaging filter shown in table 2.1. The 3x3 kernel systematically moves through an image (convolution in mathematical terms) and returns the average value of each pixel with its 8-neighborhood adjacency.

Table 2.1: 3X3 Averaging Filter

1/3	1/3	1/3
1/3	1/3	1/3
1/3	1/3	1/3

This is called a low-pass filter, which means it passes low spatial frequency information while tending to block higher frequency information. In the spatial domain, high frequency refers to large differences in pixel intensity between pixels with small amount of distance between them. Therefore it tends to leave largely uniform regions relatively unchanged while smoothing out irregularities as well as blurring quick transitions and edges. The amount of severity of the smoothing will be determined by the size and weighting values of the kernel being used, although it is generally desirable to keep the weighting total unity so as not to brighten or darken the image as a whole.

The basic low-pass filter is an example of a linear operation. For a general operator H that produces an output image g(x,y), where $H[f(x,y)] = g(x,y)$. A linear

filtering operation is defined if

$$\mathrm{H}[\mathrm{a}_i\mathrm{f}_i(x,y) + \mathrm{a}_j\mathrm{f}_j(x,y)] = \mathrm{a}_i\mathrm{H}[\mathrm{f}_i(x,y)] + \mathrm{a}_j H[\mathrm{f}_j(x,y)] = \mathrm{a}_i\mathrm{g}_i(x,y) + \mathrm{a}_j\mathrm{g}_j(x,y) \qquad (2.3)$$

where $\mathrm{a}_i, \mathrm{a}_j, \mathrm{f}_i(x,y), \mathrm{f}_j(x,y)$ are arbitrary constants and images of the same size. Equation 2.3 indicates that the output of a linear operation due to the sum of two inputs is the same as performing the operation on the inputs individually and then summing the results. In addition, the output of a linear operation to a constant times an input is the same as the output of the operation due to the original input multiplied by that constant (7). This explains the two properties that define a linear operation: additivity and homogeneity.

Linear operations are directly invertible given the full nature of the operator is known. This means a forward transformation has a backward transformation that produced the original function. In contrast, non-linear operations tend to force a loss of information that permanently change the output pixel information, making non-linear operators dangerous to use recklessly. Although this is true, non-linear operations can sometimes be better at reaching the end goal for a specific application. A powerful non-linear example is a maximum value operator, in which the center mask value becomes the value of the largest neighbor pixel value. It is not invertible, but if a maximum operator is desired, then it is invaluable. There are many other types of filtering operations both linear and non-linear such as median filtering, top/bottom hat filtering, and sharp or high boost filtering. It is crucial therefore to be aware of what the image filtering is actually doing and whether the output is really an improvement for the given application.

2.2.3 Segmentation

In segmentation, it is necessary to extract objects or fragments of an image based on certain conditions and classify them into certain regions. The result is the assignment of labels to all pixels in various regions defined by certain properties. Region

growing is a type of image segmentation. In a region growing algorithm, the basic approach is to start from a set of "seed" points and grow regions by appending neighboring pixels that have predefined properties similar to the seed, such as intensity ranges. The result is a region of pixels that all share similar characteristics.

This approach takes both intensity and relative location into account to result in better segmentation, compared to that of a simple threshold which only compares intensity information. The result is segmented pixels that have similar intensity values and have close spatial neighbors to identical elements. It is especially useful in determining the border between white and black elements as well as eliminating isolated bright spots that can severely hamper classification.

Another type of segmentation is called image decomposition, which splits image regions into further subcategories and rebuilds the regions into ones that are more desirably segmented. This evenly spaced decomposition could prove a powerful tool in the barcode problem because of the size structure and layout rigidity that the elements of a barcode need to follow. Similar but non-uniform areas that are essential are only one of two possible regions (white or black). One approach called Quadtree decomposition breaks up an image into 4 regions called quadtrees. These regions are tested against some criteria like intensity region uniformity. If regions fail the test, they are further subdivided by four and tested again. The process repeats until all quadtrees or quadregions either pass the given test or are maximally divided for the given problem. This method, if applied correctly can break down an image into many small subregions, but the largest undivided blocks would occur over element centers. These blocks can be merged back into the most likely binary white and black region layout for the barcode.

2.2.4 Morphology

Morphology is a non-linear approach that offers unified and powerful solutions to numerous image processing problems (7). Morphology is based on sets of pixel

operations, in which set reflections and translations are used extensively. Essentially it uses a structuring element (SE), similar to a kernel, to run through an image or image region to either erode or dilate an image. This is key because erosion and dilation are the foundation for image morphology.

Erosion:

$$A \ominus B = \{z|(B)_z \subseteq A\}$$

Dialtion:

$$A \oplus B = \{z|[(\hat{B})_z \bigcap A] \subseteq A^c\}$$

In particular, erosion is a thinning process by which the result is only the overlap with the image and the structuring element, which means the resulting image will have to be smaller or thinner (eroded). Dilation on the other hand results in a thickening or growing because the output image consists of the structuring element anywhere there is even a single pixel overlap between the image and the SE.

Duality of Erosion and Dilation:

$$(A \ominus B) = A^c \oplus \hat{B}$$

$$(A \oplus B)^c = A^c \ominus \hat{B}$$

While erosion and dilation are not true inverse operations by themselves, there is a certain mathematical duality that can be exploited. Performing an erosion then a dilation (or vice versa) implicitly implies that the image should be unchanged, which in practice is almost true except for various edge pixel conditions. These are standard morphological operations: opening and closing. As described, dilation expands the components of an image and erosion shrinks them. Opening generally smoothes the contours of an object, breaks narrow isthmuses, and eliminates thin protrusions. Closing also tends to smooth contours, but it

generally fuses narrow breaks, fills in gaps, and eliminates small holes. Notice the only difference between the two operations is the order of morphological operations.

Opening $(A \circ B)$

$$A \circ B = (A \ominus B) \oplus B$$

Closing $(A \bullet B)$

$$A \bullet B = (A \oplus B) \ominus B$$

The opening of A by B is the erosion of A by B, followed by a dilation of the result by B. Similarly, the closing of A by B is the dilation of A by B, followed by an erosion of the result by B. Morphology as described can be used powerfully to manipulate an image in a desired way. Often morphological approaches will appear very similar to filtering, but morphology is at heart a non-linear approach and therefore original image reconstruction is impossible. In many cases, the original image is not nearly as important as a processed version for visual appeal or information extraction.

2.2.5 Dam Construction

Another morphological segmentation approach is the use of watersheds. The theory behind this is to imagine that each region of interest has a local minimum. One punches a hole through the bottom of the image at these minima and allows "water" to rise up at a constant rate. As these watershed regions "fill up," the boundaries between regions shrink. At any point where the watershed regions would overflow into another basin, a thin dam is constructed to prevent the overflow. As the water continues to rise, the dams will continue to be erected until there are closed contour dams that bound each watershed. This is particularly useful to separate regions of similar type yet spatially separated such as in blob analysis.

Figure 2.5 shows how a series of round objects can be segmented, by grow-

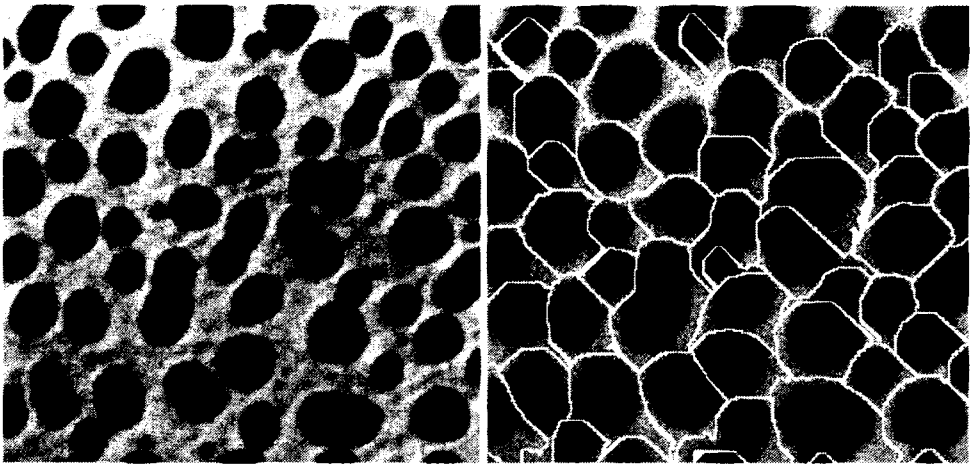


Figure 2.5: Morphological Watershed Method for Segmentation.

ing dams to separate them. The image on the left shows a series a round dark objects to be segmented and the image on the right show the resulting regions separated by a thin white line or "dam". The dams produced appear to be equidistant boundaries between each region.

In the barcode problem, the image could be inverted, and the original white elements would tend to fill up and the dam boundaries occur at white/black edges. The result would be similar to an image contour and the contour lines could be useful in determining the element layout.

Chapter 3

Methodology

3.1 Algorithm Development

3.1.1 Image Acquisition and Decoder Integration

In attempts to create an initial controlled environment for image acquisition, a machined imaging station was constructed, as shown in figure 3.1. Acquisition took place after writing a C++ imaging application in Microsoft Visual Studio 2010 while harnessing the capabilities of the OpenCV 2.3.0 function libraries. The next step required the integration of black box Data Matrix decoders into a single application that would attempt to decode a given image. After significant research into available decoders, the decision was made to try three: 2DTG, AYPSYS, and ClearImage. All are closed source decoders that offered free trials and were able to be integrated into a C++ executable application. The source code for this can be found in Appendix B.

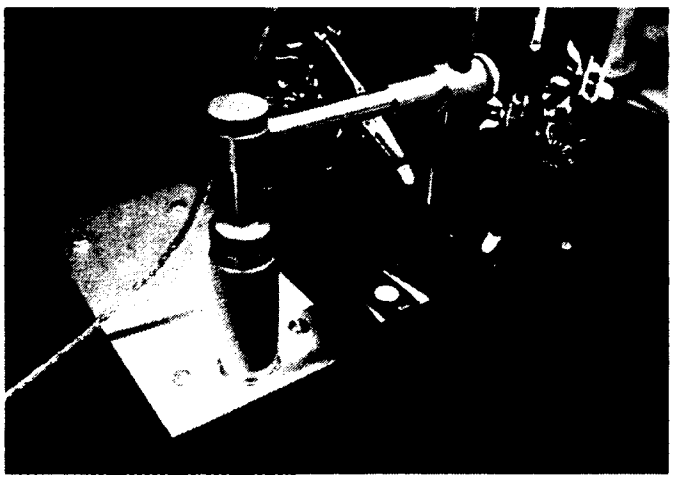


Figure 3.1: Image Acquisition Setup

Using over 1000 slides, two separate databases of raw images were acquired, 510 images each. These images are included in the CD database attached to the thesis. One database was used for algorithm development and the second database was reserved specifically for final testing. This allows for a large sample set of images but also prevents the algorithms from being tuned to a specific image type, thus compromising results. All further figures and coding using the acquired image databases were completed in MATLAB 2011a.

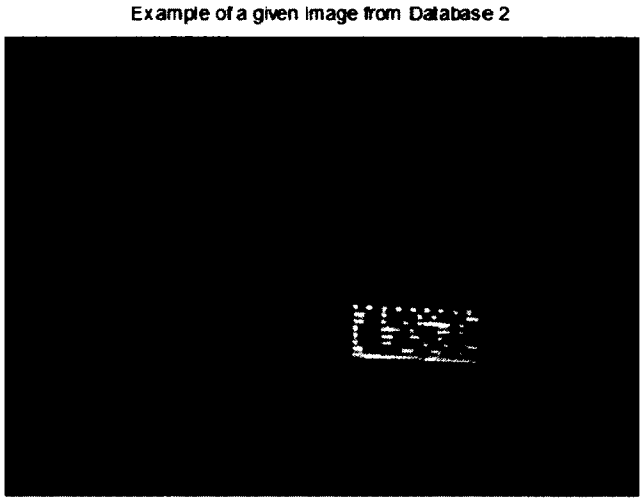


Figure 3.2: Example Image after Acquisition and Conversion from Color to Grayscale

Figure 3.2 shows a typical image that would be acquired from the imaging station in figure 3.1. This image type provides sufficient contrast and resolution while maintaining minimal background clutter. Certain variables are even able to be adjusted such as the working distance and angle of the lighting while maintaining a high level of image quality. The rest of the following theory and methods depends on acquiring a similar image type and quality, but barcode localization in cluttered backgrounds can be made much more robust by implementing methods described by Liu (13) or Pârvu (17). The assumption is that the conditions of the imaging system can be controlled enough to neglect

other acquisition challenges, enabling the project focus to be centered on higher-level image pre-processing classification algorithms that assume some of the preliminary problems have been overcome, as they have been in (13) and (17).

3.1.2 Pre-Processing

This section describes the rudimentary pre-processing algorithms that transform the raw acquired images into clean, cropped, and rotated versions of each barcode with white elements on a virtually black background. Refer to figure 3.3 for the function highlights.

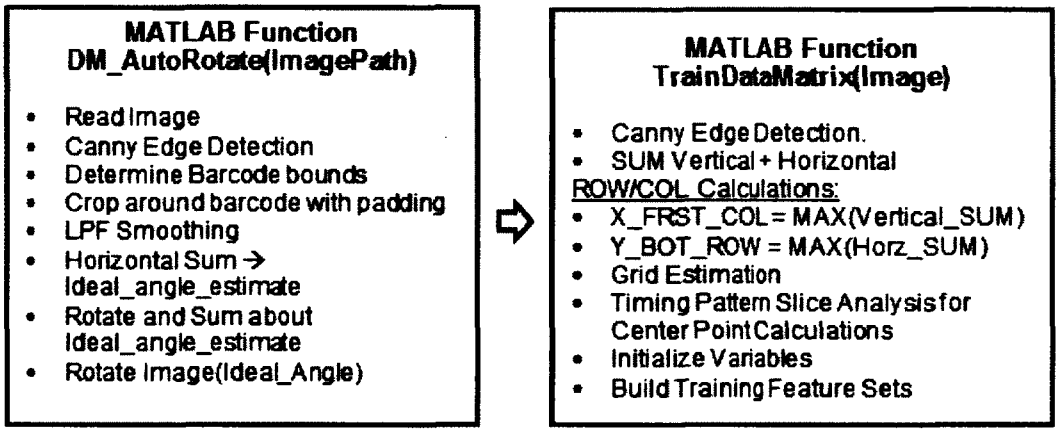


Figure 3.3: Developed Image Pre-Processing Functions

The first step is localization of the barcode. Using a Canny edge detection algorithm, as seen in figure 3.4, the boundaries of the barcode are discovered given that the background clutter is sufficiently small. Edge detection is a fundamental tool in image processing that aims to identify points in an image at which the image brightness changes sharply. In this application, the bright white printed dots of the elements are generally very bright or have high pixel intensity values compared to the black background, and will therefore cause distinct edges. The Canny algorithm finds the local maxima of the gradient of the image by using the derivative of a Gaussian filter and classifies strong and weak edges.

The output includes only strong edges and weak edges connected to strong edges, making the algorithm more robust at handling noise artifacts as compared to other edge detector operations such as the Sobel or Prewitt methods.

Figure 3.4: Image after a canny edge detection and bounding box placed

Once the bounds of the barcode are known, it is important to crop out the barcode from the background because this drastically reduces the image size, and therefore the number of pixels that need to be processed. A simple step such as this is important as the amount of processing time that can be saved is significant, especially when algorithms need to be efficient. Figure 3.5 is a cropped version of the original image with a generous border around the outside of the barcode.

Figure 3.5: Cropped Image with Border Padding

Before element processing can occur, the barcode must be rotated into its final position where the corners are perpendicular to the base background. In this

application, horizontal pixel summing is used to determine barcode alignment. By taking advantage of the prior knowledge that the entire bottom row of elements are white, the horizontal profile peak is maximized at the bottom element center point, except in the unlikely and ambiguous case in which an entire row of information elements is white. Using a wide low-pass filter, roughly the size of an element, to smooth the image beforehand eliminates bright artifacts that can interfere with the sums. In this case, the average element diameter is about 7 pixels. In choosing a mask that is shown in figure 3.6 only pixels near element centers will be maximized and all others will be attenuated slightly. This creates nice maximum peaks aligned with the solid white border elements, and allows for rotation corrections as well as a starting point for element center calculations.

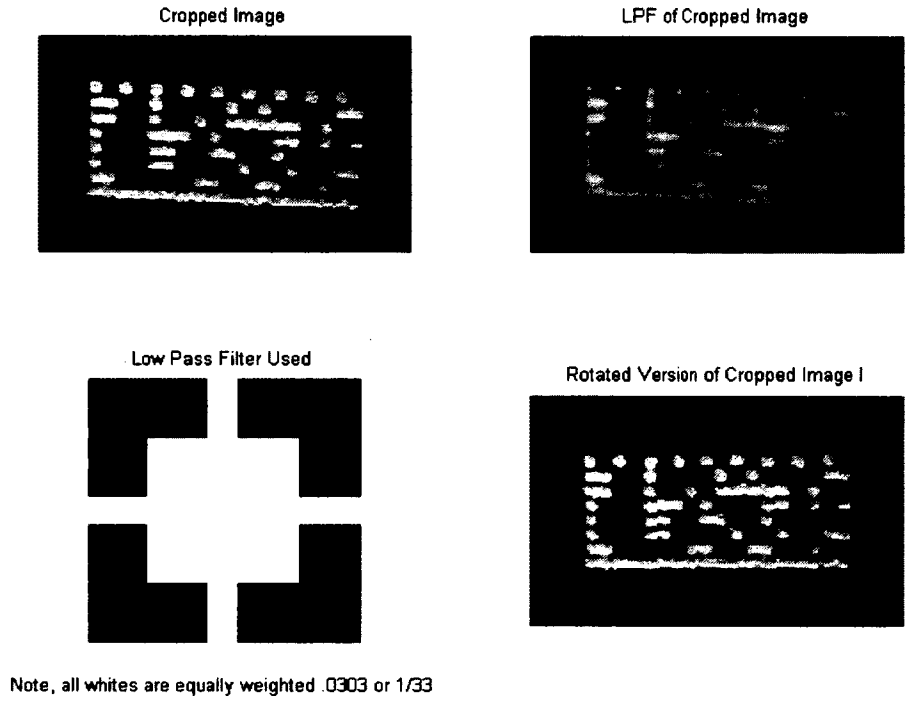


Figure 3.6: Cropped Image Smoothed and Rotated

The horizontal profile, shown in figure 3.7, is analyzed to estimate the

optimal rotation. Then, by performing a series of rotations and summing profiles about the estimated ideal angle, the profile with the maximized peak identifies the actual ideal angle to rotate the image. Ensuring that the barcode orientation is perpendicular allows simplicity and further assumptions to be made in future algorithms. Figure 3.6 shows the typical results from an automatic rotation which is sufficient for this application. The method is admittedly inefficient, but implementation of more efficient methods like the Hough Transform or the Radon Transform (Liu (13)) were dropped in pursuit of the main research objectives.

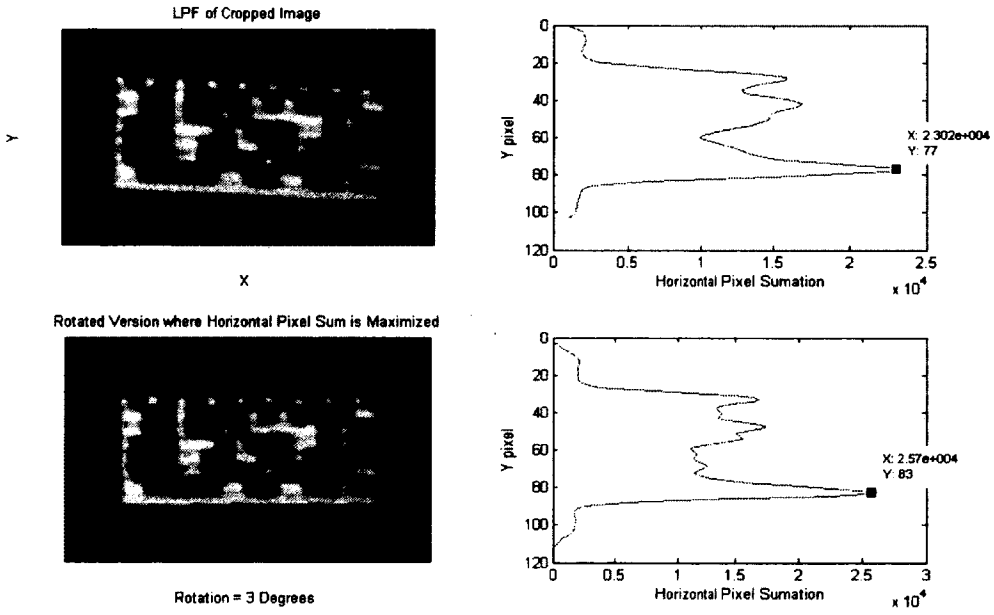


Figure 3.7: Image with Vertical and Horizontal Summing Profiles

3.1.3 Bayes Likelihood Estimation

Thus far, the assumption is that any barcode image acquired can quickly be pre-processed into a cropped and rotated version of itself. The research investigation begins at likelihood probability density functions as an attempt to utilize Bayes probabilistic

approach to element classification.

Initially, it was very convenient to overlay an evenly spaced grid onto the barcode so that each cell would contain pixels exclusively belonging to a single element of the barcode which allows for logical and straightforward indexing. It was effective to visually describe where every element location ideally should be, but this also made it clear that, due to the barcode printing process, the uniformly distributed grid lines did not always perfectly enclose a single element. Figure 3.8 shows that even a well aligned grid will display slight misalignments due to the circular nature of the elements; therefore, the cell's contents are not perfectly restricted to one element. Regardless, the investigation begins by looking at looking at histograms of black and white elements as shown in figure 3.8. It can be seen that any cell will inevitably contain some interference from neighboring elements, and this becomes an additional constraint that has to be accounted for. Our investigation then was to determine defining characteristics of the cells unique to black and white elements.

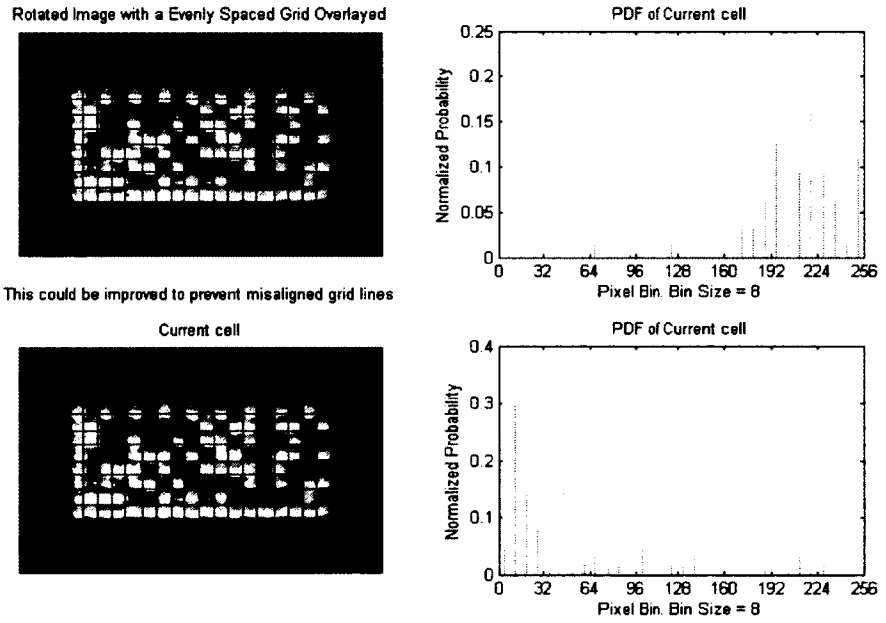


Figure 3.8: Typical Probability Density Functions white and black cells

The cell's histogram can be interpreted as the cell area pixel intensity density distribution. A histogram algorithm was created to quickly loop through all cells, take each cell of information, and create a normalized histogram for the given cell that describes the proportion of pixels that fall into given intensity ranges or bins. In particular, the function takes a matrix of pixel information, reshapes the data into an array, and sorts the data into 32 different bins with a bin size of 8. The function then returns the normalized probability density function (PDF) of pixel distribution as well as the associated cumulative distribution function (CDF) and axis information for plotting. A PDF is a distribution that describes the probability or percent of pixels that fall within the bin region, and a CDF is the cumulative probability of all pixels at or below the current bin. In the case of a single cell, the PDF is simply the cell histogram, but when referring to the combination of pixels from multiple cells, it begins to have meaning as an intensity probability distribution. The collective distribution quantitatively describes the cumulative makeup of certain element state pixel intensities. This is an important function because there is a wealth of knowledge that can be extracted from this information; for example, it tells the viewer where the bulk of the intensity values are. Figure 3.8 shows that a typical white element displays a PDF where a majority of the pixels are well above the bin 160, and a typical black cell displays mostly low values, with a second smaller high valued peak caused by some white overlap.

Each cell will have a given distribution but there are certain trends that develop in each state type and by combining all of the similar state element's pixel distributions, many of these trends become apparent. For a specific barcode image, the combination of all similar state cells represents the true posterior probability distribution of pixels. These results are the actual intensity probability density functions that describe the exact expected distribution for each cell type for the given barcode image. Figure 3.9 shows accumulated probability density functions for a particular barcode example. As expected, all accumulated zero cells contain most of the probability mass below bin 96, while the white cells are more spread out but the majority of probability still lies in the upper half of the distribution

range.

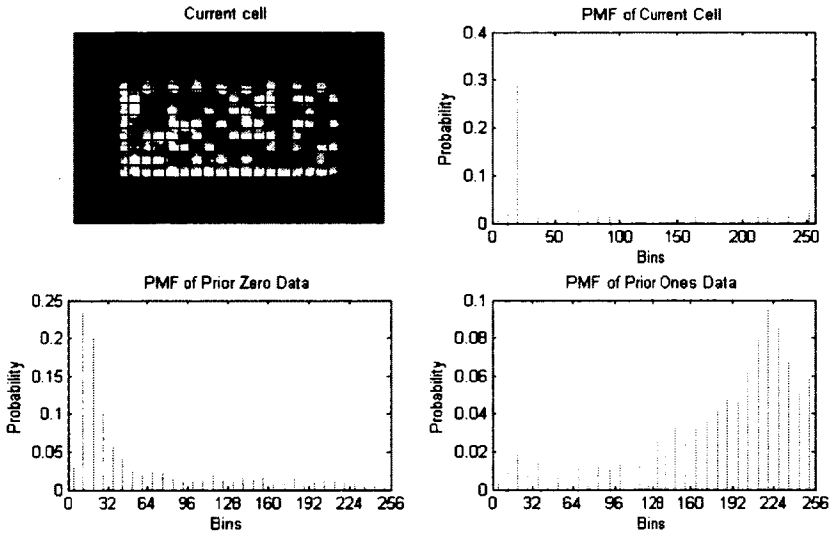


Figure 3.9: Posterior Probability Distributions

Bayes theory states that the posterior probability (the true state, given feature x has been observed) is highly dependent on the likelihood function (the state-conditional probability distribution for feature x). Based on the knowledge of the Data Matrix scheme, the states of all border elements are known. The training theory is such that, by looping through and accumulating density functions for each of the binary states, we are effectively estimating the likelihood function for each state. This estimation gives a power tool for classification, and is more accurate with more sample cells.

Taking advantage of these known priors to build up known PDF distributions of white and black cells before attempting to decode the barcode is called "barcode training". The concept of training is useful because it will adapt the distributions to each and every barcode image, which can be vastly differing in element intensities and quality parameters. By "training" an image by accumulating the prior distributions based on the border cells, an unknown cell's PDF can be classified from a probabilistic standpoint. Because of the direct relationship between $P(w_j|x)$ and $p(x|w_j)$, correlation techniques seemed

to hold high potential. Once the likelihood black and white functions are estimated, it is possible to loop through each cell's distribution and compare that particular histogram to each prior likelihood. This is useful since correlation coefficient is a powerful quantifiable method in determining the similarity between two vectors of data. The coefficient result is a scalar value between one and zero, and experiments preformed here show that typical correlation between a cell's histogram and the corresponding prior PDF is around 0.6-0.9, while the correlation with the opposite state is between 0.1-0.4. Now by looping through each cell and performing two quick correlations at each stop, it is quickly possible to estimate the most probable state of each cell.

This method in itself is fairly effective and is successful at classifying over 90% of elements in any barcode, but the necessary success rate needs to be over 99% so this technique alone is insufficient. The problem becomes apparent when viewing particularly misaligned cells, such as the one in figure 3.10. A misaligned cell can introduce enough differing pixels at the edges to distort a cell's PDF and therefore lower the correlation result. Additionally, the printing process can be so poor that many elements will be severely misaligned and significant bleeding of white elements into black cells complicates things further. Realizing that this particular barcode printing process is one of many, there are likely worse printing procedures that further reduce the barcode quality. This demands a high level of cognition in the algorithms to be able to handle poor quality and misaligned barcodes. The grid is a good idealized concept, but alone it is not practical because of the possibility of severe misalignments. Further work needs to be done to extend the robustness of this promising concept.

3.1.4 Feature Extraction

Feature extraction is an information reduction process. By taking objects in images or groups of pixels and boiling it down to several feature values, it essentially replaces the description of the element from a matrix of many pixels to a set of much fewer,

but descriptive features. Thus, we attempt to condense hundreds or thousands of pixels down to a much smaller feature set that can still be used to correctly classify the results.

A cell's pixel distribution is a useful descriptor, but more quantitative analysis is necessary to enable the processor to make the state decision. Additional feature types are investigated in an attempt to describe elements as a coherent feature set and ultimately use the element features as a means of classifying each element and successfully binarizing the barcodes.

The first compelling feature to be extracted was the region intensity mean, which describes the average pixel intensity for a given area of pixels. This is simple yet powerful way to describe an element, since the state of the element is most directly determined by region intensity. The mean was met with success except in misaligned cell cases, meaning the cell had other pixels that were not part of the element object, yet were brought into the mean calculation. Grid misalignments cause a cell centered on a white element to have a significant amount of black due to the inaccurate placement of the grid lines. This effectively brings down the mean to a mid-level value that could be identical to a black cell that has a significant amount of white bleed. Despite some ambiguity, it is still the most useful feature at this point because it directly describes local intensity values.

$$x = \begin{bmatrix} \text{Region Mean} \\ \text{Peak Bin} \\ \text{Peak Weight} \\ \text{\% Upper Quartile} \\ \text{\% Lower Quartile} \\ \text{Bin(CDF} = 0.5\text{)} \\ \text{Correlation Coefficent} \end{bmatrix}$$

Many other features were additionally extracted and tested. The preced-

ing feature vector x summarizes the most successful features evaluated. For instance, the weighted peak probability describes where the highest pixel count occurs (peak bin), shifted by the peak "weight" of probability in nearby pixel bins. Percentages of pixels that fall in the upper and lower quartiles can be illuminating, as well as the bin where the CDF reaches 0.5. This means at that bin, there is still 50% of pixels to be accounted for. If the value is 150, then 50% of pixels are greater than 150, and therefore the element is likely white. All these features, and more, can be extracted from each particular cell and stored in the feature vector, x. Using these methods we have then simplified a cell containing hundreds of pixels down to a half dozen scalar numbers that describe each cell.

Any given cell will have a vector x that corresponds to its particular features. In this way, can begin to accumulate features sets and expand our options for describing and classifying a cell. Any cell can be described in terms of feature space, which simply describes the features of the cell. It helps to be able to visualize the feature space, but we are often limited to only two or three dimensions for visualization. The human mind typically has difficulties visualizing a six to eight variable feature space.

We can also attempt to estimate the likelihood functions by looking at the distribution of feature values for each state. Another classification method that makes use of the feature vector approach is called Fishers Linear Discriminant (FLD).

3.1.5 Fishers Linear Discriminant

Features of particular states can be ambiguous in determining the actual state, but the collection of features is far more powerful. A black element may contain some ambiguous features like a mean halfway between each range and no significant probabilities in either quartile, but across an entire feature set there will likely be enough evidence to classify it as black. The job of interpreting features seems trivial to a human user, but the ultimate goal is automization. Therefore, there needs to be a distinct set of rules that describe exactly how the classifier will operate. With a set of many features, the

solution becomes difficult to interpret; this is why it was decided to experiment with the Fisher Linear Discriminant. It is a method from linear algebra that allows for many feature variables and creates a boundary in feature space. In this way, given a set of features for a cell, as discussed in section 2.1.2.2, the result of the discriminant will determine the state of the cell.

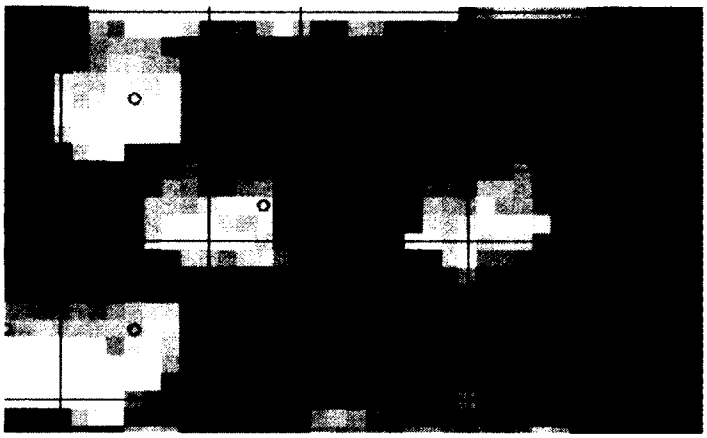


Figure 3.10: Example of Misaligned Elements

The elements are neatly organized into distinct rows and columns but there were often misalignments, usually created by white elements smearing into black element locations, so final algorithms must be robust to handle these types of situations. Figure 3.10 shows how severe element misalignments can be even with a grid aligned with the timing pattern. Initial results were very promising, but the recurring problem was misalignments with the grid overlay; a white cell with a misaligned grid can have a nearly identical distribution to a black cell containing severe white bleed. Thus, any amount of features extracted from the cell's PDF will not be sufficient for classification. It became clear then that ignoring the pixel layout is throwing away too much information. We surmise that it is logical to trust the pixels towards the middle of a cell much more than the pixels at the fringes, which are often influenced by adjacent elements; hence the reasoning for developing region growing algorithms.

3.1.6 Region Growing

The proposed solution is a method of region growing, as discussed in section 2.3.4. In the grid concept, the borders were often misaligned, but the cell was generally centered over the element. Our theory as described in the previous paragraph is that the center of the cell provides more accurate and unambiguous data than the border. The idea is to use the center point of each grid as a seed point, then implement a region growing algorithm to find the area of the cell that contains only the desired element. The algorithm starts from the seed point and checks all of its four-neighborhood pixels; the ones that are within a threshold of intensity level are then added to the region. The region mean is then recalculated and all the newly adjacent neighbors are tested to see whether they will be added to the region as well. The result is a region of adjacently connected pixels that have similar intensity values to the seed point. Figure 3.11 shows how severely convoluted regions will transform into only pixels similar to that of the center point. By utilizing the histogram algorithms of only the grown "element region," it drastically reduces any bimodal tendencies, and thus increases the success rate of the correlation algorithm to about 95%, which is unfortunately still below acceptable levels. Figure 3.12 is another example of how the region growing algorithm works to eliminate the white bleed problem.

Results were even more promising when extracting the feature vector x and using Fischers method for the discriminant classification. Almost 99% of barcodes binarized by this method could be decoded correctly by most of the decoding software. The only element problems remaining were ones caused by severe element misalignment, such as in figure 3.10. The grid lines were placed using the border elements edges assuming that all the elements would line up with the border. This is the standard two-dimensional model, but a universal assumption that all elements will be perfectly aligned is not always correct and it can be seen that the center points of the cells may not always fall over the intended element.

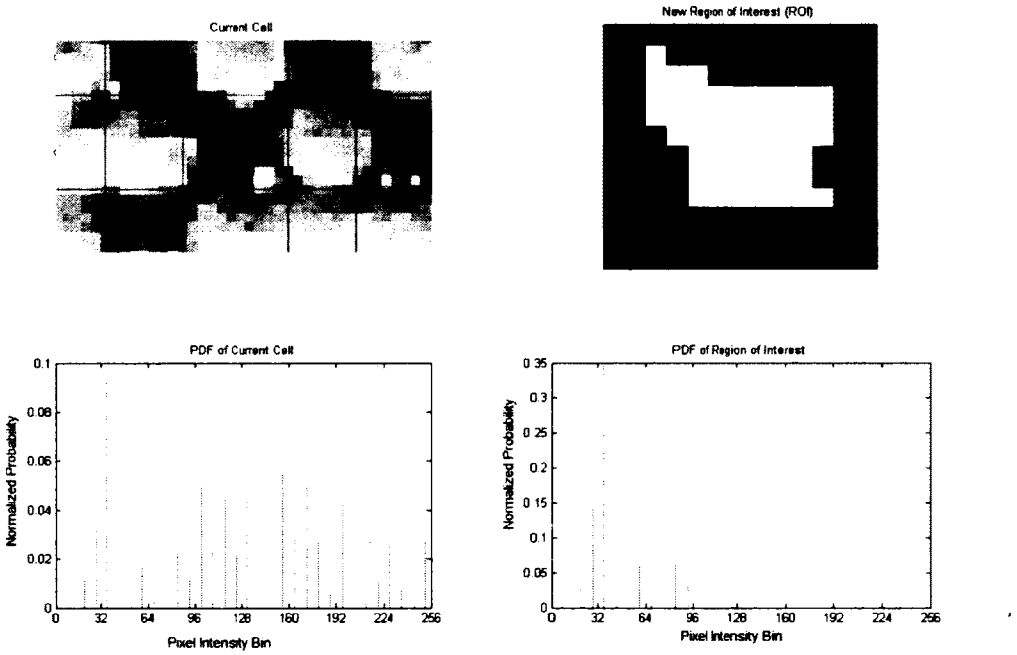


Figure 3.11: Results of Cell Region Grow

Figure 3.12: Example of Region Grow Algorithm (Region: White Cells)

While the region growing algorithm is showing great promise in image segmentation, it is dependent on having correct seeding points. Simply lining up the geometry with the border elements is not always a feasible option, so we must look to other global processing methods to find seed points for this type of algorithm.

3.1.7 Morphology

As stated earlier, morphology is built around two fundamental operations: erosion and dilation. Since the white regions are the most distinct in that they have high intensity values compared to the background, it would be prudent to attempt to identify all those regions that are quantifiably white. By using erosion principles, it is possible to erode each continuous white region down to a single center pixel. This will not necessarily find each element center, but knowledge of the region center is also a useful tool. Using these center points as seeds for the region growing algorithm, we can effectively find all the pixels in a given barcode image that are of similar intensity to white regions. This is so important because it can tell so much about the barcode. Knowing all, or nearly all, of the white elements, it aids in the black element classification. Overall, it is a significant step forward in the binarization process.

3.2 Final Algorithmic Design

3.2.1 Theory

The goal behind the theory for the final design is to get a correct barcode read as fast as possible. If the decoding software can read the barcode without any pre-processing, it is pointless to waste time with unnecessary algorithms. Furthermore, often only some global intermediate pre-processing is required without creating a synthetic image. When the decoder still fails, then it is time to get into the more involved methods to binarize the barcode. Here some image analysis is required to determine the best algorithm track for high confidence reading. The confidence score is a good way to ensure a successful read by utilizing a bit flipping procedure cycling through the combinations of the least confident elements (as described later this Chapter). Refer to figure 3.13 for a visual description of the algorithm flow.

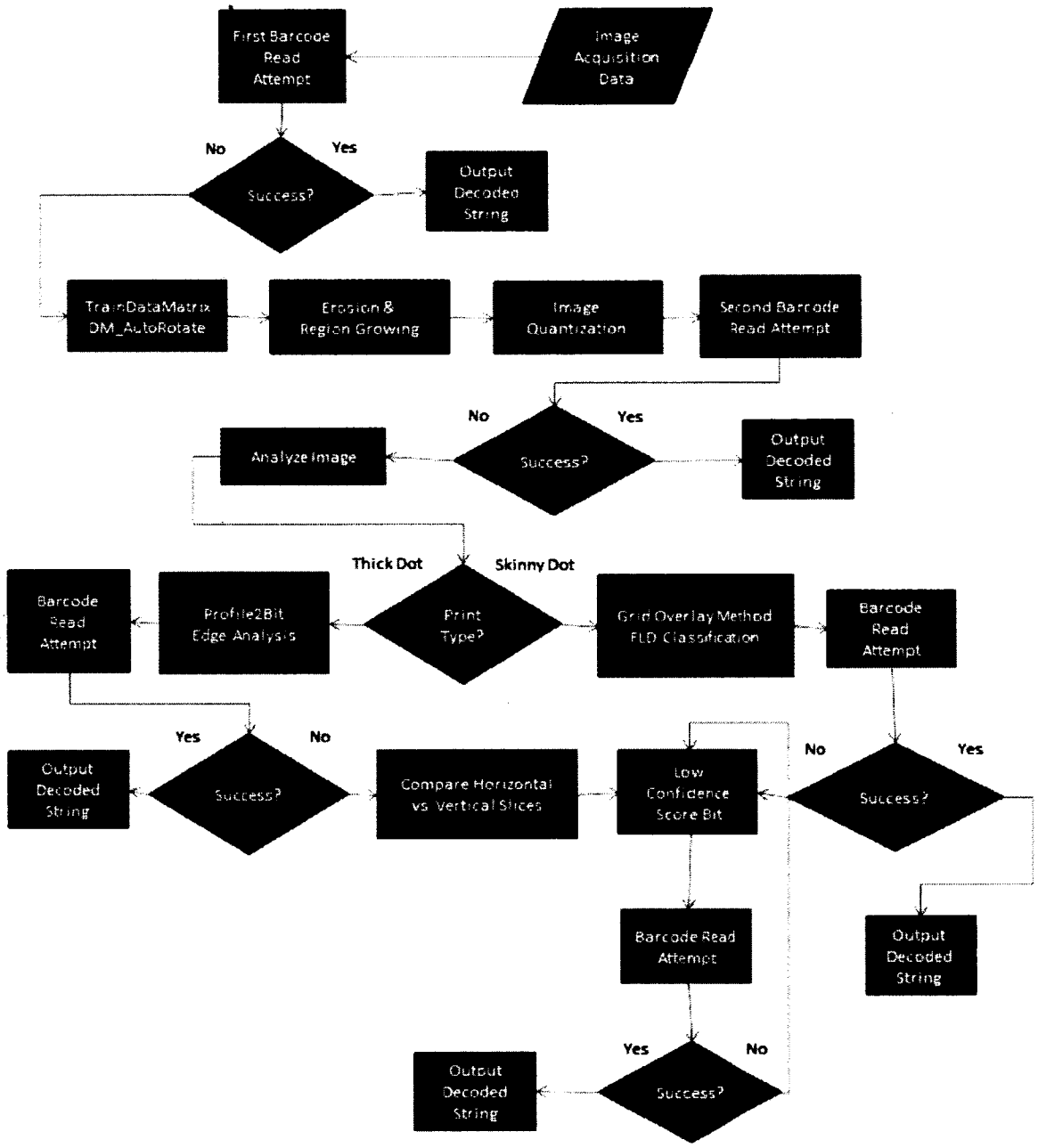


Figure 3.13: Algorithm Flowchart

3.2.2 Implementation - Two Track Approach

It was discovered that there are two distinct types of printing processes used for identical Data Matrix barcode schemes. The primary characteristic that sets the two types of barcodes apart is the printing process for the white dots. One type, the white elements typically fill and often overflow the space allotted for a single element. This creates chains of connected white pixels for adjacent white elements, but also leaves a smaller pixel area for black elements. In the other type of barcode, the white elements are merely a small dot aligned precisely at the center of the intended element location. This type has much more black space in between elements and even adjacent white elements have black space in between. This is an important distinction because algorithms based on edge detection will detect many additional edges even over adjacent white elements. Typical images of each printing process are shown in figure 3.14. We shall denote the two processes as thick dot printing and skinny dot printing.

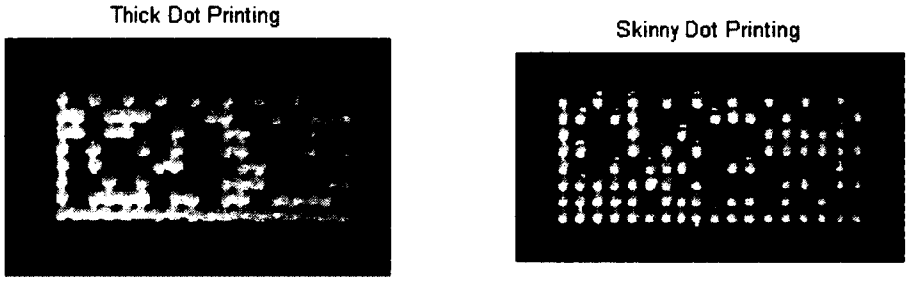


Figure 3.14: Example of Two Data Matrix Printing Types

After preforming various testing on both types of barcodes printing processes, it was discovered that different algorithmic approaches have varying results for the different dot printing barcode types. For example, the slice edge detection algorithm was designed with the thick dot printing in mind and expects a single edge transition between unlike elements, and no edge between identical elements. On the other hand, the skinny

dot barcodes are almost all successfully binarized by an older set of algorithms based on the grid location and an adaptive thresholding of the mean of pixels around the center point. It seems that the skinny barcodes are generally better aligned with the border timing pattern.

3.2.2.1 Train Data Matrix

After a first read failure, it is necessary to pre-process the image with some preliminary processing to locate, crop, and rotate the barcode as well as analyzing the timing pattern for element bounds and center points. After reading in the image, a wide low-pass filter is applied to smooth out irregularities for summation. The image is cropped with a generous border around the barcode after an edge detection to determine bounds. Then the pixels in the image are summed horizontally. This takes advantage of the bottom row of elements which are all white and will produce a maximum intensity sum. By performing a quick estimation, we create a rotation estimate and rotate the barcode with several iterations above and below the estimate. By summing each rotation, the angle that produces the maximum horizontal sum is the ideal angle for rotation.

At this point, the assumption is the barcode is aligned perpendicularly to the horizontal and vertical edges, provided there is minimal skew. Profile analysis looks at slices through the horizontal and vertical timing patterns results in approximate sinusoidal waveforms in which the critical points correspond to the center points of the border elements. Observe figure 3.15 for an example of a raw profile cut through taken from the timing profile. These values are compared to an equally spaced vector of theoretical points to verify and eliminate false peaks. If the barcode is straight and aligned, these center points can be extrapolated to also correspond to element centers throughout the barcode.

The training portion of the function then loops through every cell, selects the center point as a seed, grows a region of interest (ROI), and sends the new ROI to collect a feature set for each cell (BuildFeatureSet A.8). The result of the TrainDataMatrix() function produces a cropped and rotated barcode with grid and center point information

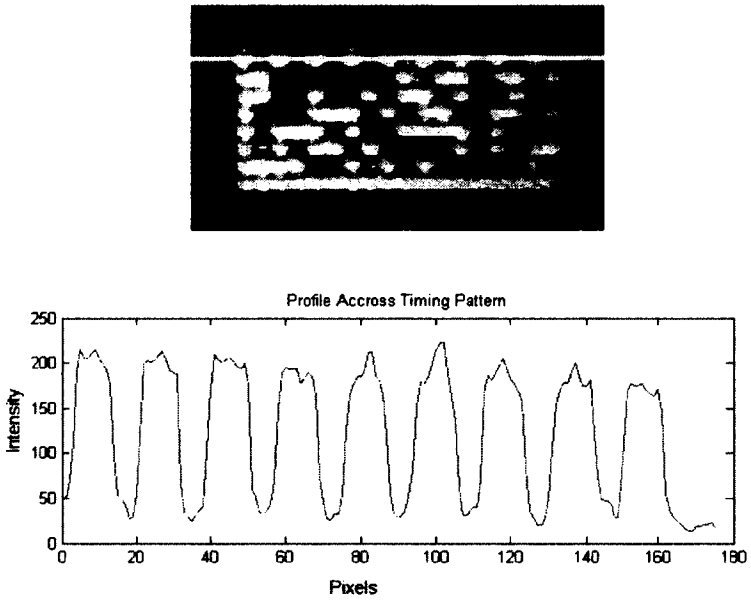


Figure 3.15: Horizontal Timing Profile

for future processing. Additionally, the training information is made available for later classification depending on the algorithm track.

3.2.2.2 Morphological Erosion and Region Growing

The training algorithm is assuming certain alignments exist in the barcode. As shown back in figure 1.2, there can be angles of distortion. The erosion and region growing are invariant to this parameter distortion because they act globally on the image. First, a high threshold is applied to take only the highest intensity white elements into a binary image, which all clusters are eroded down to a single point. These correspond to white elements or groups of white elements. This array of points is then used for seeding points for the region-grow. Growing regions in the original image produce areas that are similar to intensity and spatial location of the white elements. This works to eliminate random spikes that often occur. One admitted area for improvement is the static threshold,

which should be adjusted for each image. Currently it throttles all values above 80% of the maximum value, but in images with non-uniform contrast like that of figure 3.16 adaptive approaches may be more successful. It is suggested to attempt morphological dam construction or contour analysis to create blobs, and then calculate mass centroid points for each blob as seeding points. This would be an area of improvement for future work.

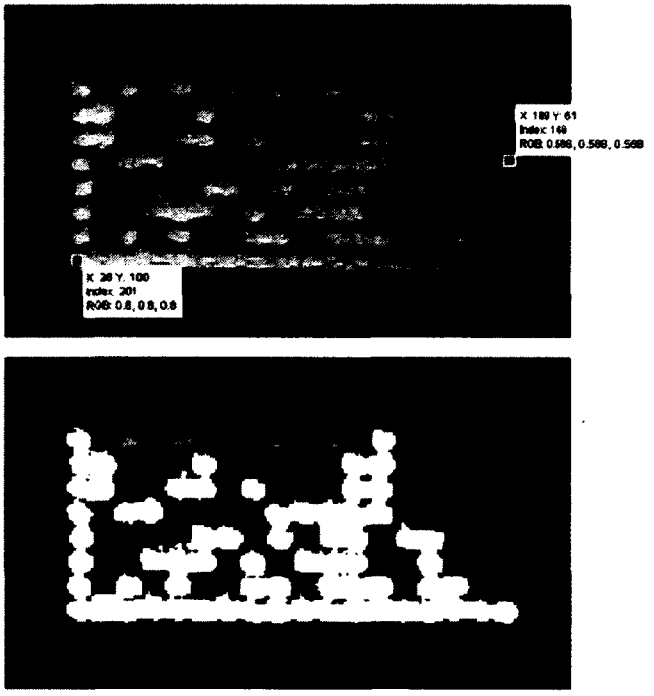


Figure 3.16: Non-Uniform Intensity is a Problem for Static Threshold

The resultant regions are throttled white and then an image quantization is performed. Three unequal levels of quantization were chosen to separate regions into a bottom 40%, a middle 20%, and a top 40%. The reasoning is that middle values are unsure, but medium high or medium low values most likely correspond to the closest extreme. This creates a nicely quantized result that is often enough for the decoding software to read the barcode. This is when a second read is attempted and if successful, ends the algorithm chain. If another failure occurs, the flowchart in figure 3.13 provides the path for a decision

on which path to take, based on image analysis.

To determine the printing process type, there are a variety of methods available in the arsenal. First and foremost would be a profile analysis through the solid white border elements. Since each element value is white, we know we will see a solid white profile, or an alternating white/dark profile if there are gaps between the white elements. Thus we will be able to determine whether the current barcode is skinny or thick dot printed. Since the thick dot printing is more successful with the profile edge detection analysis and the skinny dot barcodes works better with feature extraction and a linear discriminant classifier, the algorithms will determine the most likely printing type and continue the algorithm flow on the appropriate track. Other methods could include pixel length width and area and uniformity of pixels around the center point. Often the thick dot printing had assumed center points that were near the actually element edges. By looking at an 8-neighborhood or a 25-neighborhood, the similarity of pixel outliers to the region mean would be a convincing metric to whether the center point was actually over an element or near the element edge.

3.2.2.3 Horizontal and Vertical Profile Analysis

The horizontal and vertical profile analysis (Appendix A.4) works by taking an average value of pixels around the element center line and performing edge analysis to determine the intended bit layout. After smoothing and differentiation, the most likely critical values which indicate steep edges, are extracted and matched up to where the transition likely occurs. The function analyzes pixel data to eliminate likely false peaks or otherwise unwanted transitions and creates a list of strong and weak transitions. The result is an estimated stream of binary bit data that corresponds to the cut through. While one direction may not always be totally accurate due to random pixels causing false peaks, the two direction profiles can double check each other. The added redundancy helps to figure out where the problem elements occur. When differences from a vertical slice and a

horizontal slice occur, that element is added to a list of low confidence elements for later bit flipping. Figure 3.17 shows a small white isthmus, which creates a false peak in the horizontal line, but the vertical line would be unaffected.

Figure 3.17: Example of Random Spike Creating False Peaks

3.2.2.4 FLD Analysis

For the skinny dot printing and any printing process that ensures the alignment of barcode elements, Fisher's Linear Discriminant is a good binarization method. The algorithm loops through each element and grows a region using the center point as a seed. As long as the center point lies on the intended element, a region of similar pixel values is grown for each element and features are extracted. Features like region mean, pixel area, length, and width parameters make good extractable features as well as others like the percent of pixels in the upper and lower quartiles of a histogram and the bin in which the associated CDF reaches 0.5. Feature vector $\acute{x}$ can be any length and by applying FLD equation from section 2.1.2.2, each set of accumulated features will be on one side of the discriminant boundary. That will be the classification decision boundary, but also, the elements feature that lie closest to the boundary, will be flagged as elements with low confidence. If x is a feature vector with two features and m_0 and m_1 are mean vectors for each state.

$$x = \begin{bmatrix} RegionMean \\ Bin(CDF = .5) \end{bmatrix}$$

The decisions functions are $d_1(x) = x^T m_1 - \frac{1}{2} m_1^T m_1$ and $d_2(x) = x^T m_2 - \frac{1}{2} m_2^T m_2$. The resulting equation for the decision boundary is $d_{12} = d_1(x) - d_2(x) = 0$. The sign of the result will signify which side of the boundary the sample x lies in feature space. This is the means for classification and the distance from the boundary is also saved as a confidence score for future bit flipping. A more in depth analysis on decision methods can be found in chapter 12 of Gonzalez (7).

3.2.2.5 Confidence Score and Bit Flipping

Theoretically a bit flipping procedure will result in a successful read every time, given enough processing time. The obvious downside is the large number of combination potentials in a 8×18 matrix of 92 information elements. The protocol would be to attempt a barcode read and then starting with the element with the lowest confidence, begin inverting elements one at a time and attempting barcode reads. The previous algorithms are anticipating the problem elements and make attempts to flag low confidence elements so the bit flipping can be successful after just a few iterations.

As previously discussed, when performing FLD analysis, an element in feature space that lies very close to the classifier boundary signifies a lack of confidence because the features are not significantly spread out. The lowest confidence scores should be the elements that have the smallest absolute difference to the boundary $d_{12}(x)$. In edge detection process, any differences between the horizontal and vertical cuts should immediately be flagged as low confidence. Also, there are certain conditions that the programming needs to be suspicious off. Critical transition points that are too close to each other or not close to the timing pattern are often indicative of false peaks. These need to be flagged, and the

confidence score needs to reflect the magnitude of the distance from where the peak should be.

The bit flipping procedure will attempt a barcode read, and if the binary version fails, the element with the lowest confidence score will be inverted, and a new read will be attempted. This procedure will continue to flip and invert low confidence elements until a successful read occurs. In general, only a few element errors will occur, and with a highly tuned confidence scoring system, the barcode should read with just a few attempts.

Chapter 4

Results and Conclusions

4.1 Results

After development of the previously mentioned algorithms, the final algorithm flowchart became the basis for testing (refer to figure 4.1 for the processing stages). Each barcode read attempt means that the current image is sent to all three third party decoders and the results of each output are recorded. The first step then is to send the un-processed image to each decoder in an attempt to read the code without pre-processing the data. If there is a failure to read the barcode, the algorithms applied will follow the Process 1 chain indicated. Process 1 consists of morphological erosion, region growing, and a five-to-three level quantization method that was described in Chapter 3. This means that the image is quantized into five equal levels and then the upper two and lower two levels are merged, resulting in a three unequal level quantization represented by a 40%/20%/40% split. A second read is attempted and the results are recorded. If a second read failure occurs, the images are analyzed and then either analyzed by Processes 2 or 3 depending on whether the image analysis determined that a "thick" or "skinny" dot printing process was apparent. If, at the completion of Process 2 or 3 being applied, the barcodes cannot be read the next step would be Process 4. This process consists of bit flipping for low confidence elements. Process 4 has not been fully implemented, but it is discussed in Chapter 5: Future Work.

Two data bases were acquired, as described in Chapter 3, each consisting of 510 thick dot barcode images. The first data base was used for algorithm development and initial testing and iterative research phases, while the second data base was used exclusively

for testing. Additionally, the skinny dot barcodes were compiled into a third data base of images. If a correct read was possible anywhere in the flowchart (4.1), the barcode read is a success. All that is necessary then is to output the correct message string to the system. Based on the implementation as described, and according to the algorithm track, 100% of all barcodes from both data bases can be successfully decoded.

These results are very promising, as they should be for clearly printed barcodes with minimal parameter distortions besides the element misalignments. Table 4.1 shows the breakdown of the success at each stage along the thick dot progression. We attempted a decode at each of the stages with all three of the decoding software packages (i.e. 2DTG, AYPSYS, and ClearImage) and the read results are summarized in the table below.

		2DTG	AYPSYS	ClearImage
Data Base 1 (Development)	Unprocessed	481 (94.3%)	509 (99.8%)	444 (87.1%)
	Process 1	474 (92.9%)	505 (99.0%)	507 (99.7%)
	Process 2	456 (89.4%)	510 (100%)	510 (100%)
	Process 3	470 (92.1%)	501 (98.2%)	501 (98.2%)
Data Base 2 (Testing)	Unprocessed	489 (95.8%)	507 (99.4%)	462 (90.6%)
	Process 1	485 (95.1%)	509 (99.8%)	500 (98.0%)
	Process 2	472 (92.5%)	507 (99.4%)	507 (99.4%)
	Process 3	462 (90.6%)	496 (97.3%)	496 (97.3%)

Table 4.1: Thick Dot Barcode Decoding Results

Note that the unprocessed images themselves had fairly high read rates, especially AYPSYS. It appears as though these decoders are sufficient for most applications. AYPSYS read nearly 99% of the unprocessed images, while 2DTG and ClearImage read between 87-95%. The table 4.1 argues the value of at least minimal pre-processing to improve lower read rates.

The quantized images from Process 1 significantly improve the ClearImage decoding ability, but lead to a slight decrease in performance for 2DTG and AYPSYS. In Process 1, the images are eroded down for seeding points and the white regions are

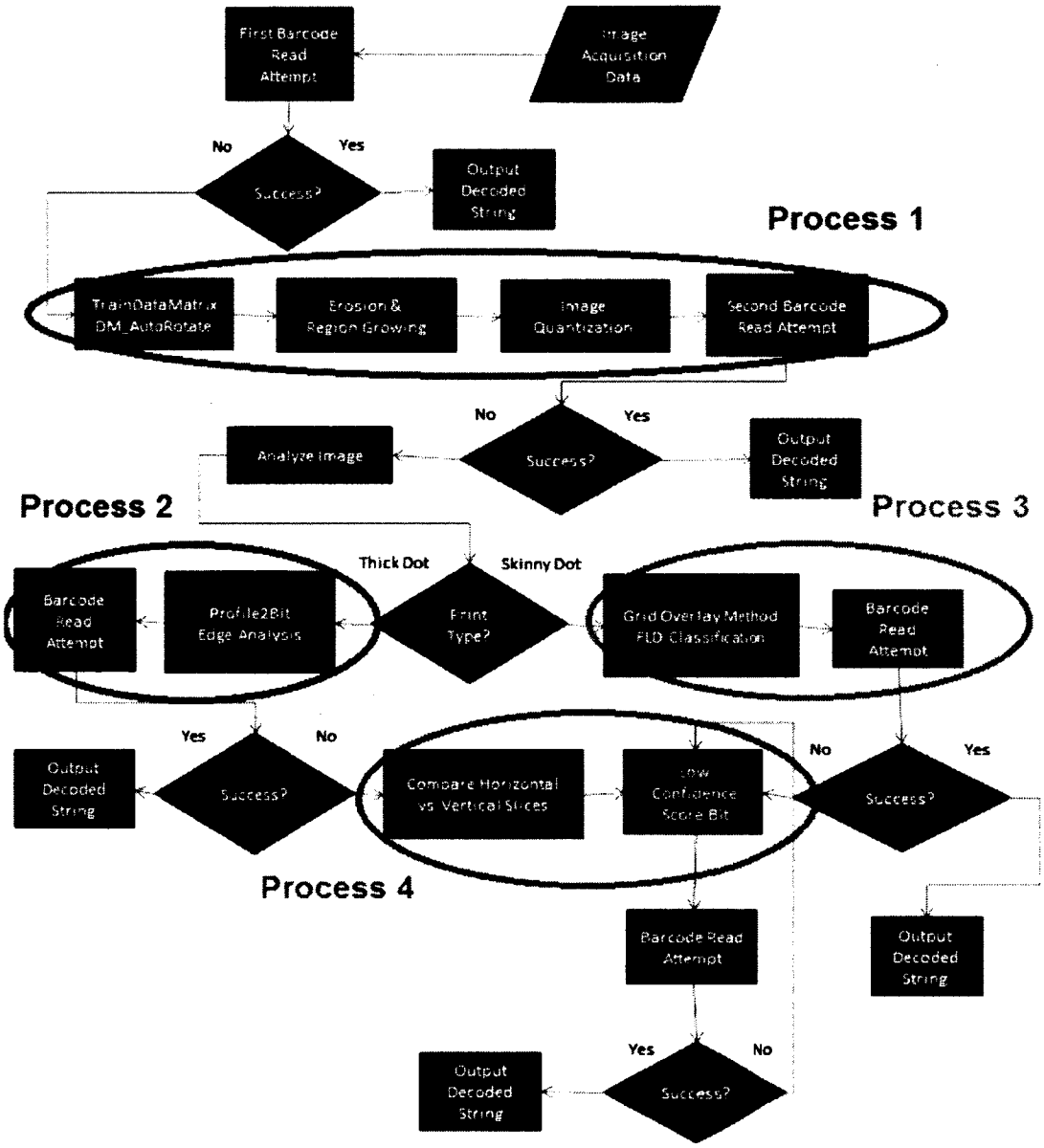


Figure 4.1: Algorithm Process Flow Chart

grown out; the resultant segmented regions are then throttled white and the remainder of untouched pixels are quantized into one of three levels. This supports the effectiveness of the global nature of the region growing operation.Without having to classify specific elements,

there is no chance to mis-classify and further distort the image. With a supposedly clear barcode, the quantization algorithms should be effective and should never significantly decrease read performance; as table 4.1 shows, read rates are greatly increased or relatively preserved. Conditions in which performance is adversely effected could include barcodes with lots of bleed or random white spikes that would be throttled white in quantization process. The region growing algorithm could include pixels in the black element regions because of the white bleed, and therefore decrease the amount of dark pixels for that element. A possible algorithm adjustment to handle bright isolated peaks could include low-pass filtering the pixels that were not segmented out in the region growing algorithm. This would tend to smooth the isolated bright artifacts, and possibly prevent poor element segmentation.

The binarized images from Process 2 present mixed results. The Process 2 images presented high read rates from AYPSYS and ClearImage but the 2DTG reader was not as successful in reading the binarized images. Since each of the decoders were issued the same binarized version of the barcode, and AYPSYS and ClearImage recorded high read rates, it is likely that the two decoders utilized the error correction effectively while the 2DTG correction methods may be insufficient. One plausible explanation is that 2DTG offers several pre-processing options that were not used processing results. Further research into 2DTG internals would be required to coherently explain which pre-processing options are valid for any particular case. Unfortunately, since the third party software is not open source, it is difficult to determine the exact reason for lower read rates. It is too early to write off the 2DTG decoder, but it is acceptable to claim that the pre-processing algorithms are more tuned for the AYPSYS and ClearImage decoders.

Each image took an average or 9-10 seconds to process and decode (which is unacceptable in real world applications) but the MATLAB code profiler explains the hangups. The decoding calls, which make a system command call to a Microsoft Visual Studio executable that we developed, take up over 60% of the total processing time and the

region growing algorithm accounts for slightly over 20% of processing time. The solution for system command calls is to integrate the decoder and the processing programs into a unified and streamlined wrapper program. Upon further analysis, the reason for the region grow processing time is because the code performs many redundant calculations of the segmented regions for every seeding point. A more streamlined approach would be to use a vector of seed points and continue processing until all neighbors are added before checking the next seed point, while skipping pixels already checked or in the region. By only eliminating external system commands and streamlining the region growing, it is possible to bring down the average pre-process and decode time to about 1-2 seconds. This is an acceptable performance level for some real world applications. Further smaller code speedups are possible throughout the algorithms as well. It is anticipated that with dedicated C++ coding that the total processing and decode time could be brought down to under 500 ms.

A third data base was created by collecting all skinny dot barcodes encountered. The data base is smaller in number (totaling 182), but it is important to be able to read them regardless to ensure total robustness and the results are still revealing. Table 4.2 summarizes the results of read attempts at each stage of the flowchart following the skinny dot FLD analysis in Process 3.

		2DTG	AYPSYS	ClearImage
Data Base 3	Unprocessed	180 (99.0%)	0 (0%)	0 (0%)
	Process 1	181 (99.6%)	0 (0%)	0 (0%)
	Process 3	171 (94.4%)	153 (84.5%)	158 (87.3%)

Table 4.2: Skinny Dot Barcode Decoding Results

Table 4.2 immediately illuminates a striking disadvantage of the AYPSYS and ClearImage decoders. In particular, the two decoders showed complete failure when attempting to read the unprocessed skinny dot barcodes. The table argues that the AYPSYS and ClearImage decoders are not designed to handle this type of barcode printing process. Even after Process 1, the quantized results still record zero success. The 2DTG software,

on the other hand, marks near perfect performance, which suggests a certain level of robust barcode handling in the 2DTG capabilities.

Process 3 posts significant improvements to the read rates of AYPSYS and ClearImage, while resulting in a slight decline in performance for the 2DTG decoder. This proves that the reading issues are because of the skinny dot printing process, and not an issue with the skinny barcode style element layout. Process 3 appears to have successfully binarized a majority of the barcodes, while admitting several binarization failures. We believe a majority of the binarization errors occur because of the preliminary processing, which was created and tuned for the thick dot barcodes. In addition, improper grid alignment results in erroneous center point calculations. These errors in turn snowball by providing non-ideal seeding points for region growing, and the result is poor pixel segmentation for an element region of interest that is not actually containing the element. As one would imagine, features extracted from an inappropriate region will not likely be indicative of the true element state. In summary, the binarizing Process 3 appears to be flawed in that it depends on successful pre-processing. These types of problems would likely be corrected with the robust localization algorithms developed by Liu (13) or Pârvu (17). When properly aligned however, the regions of different states display very clearly separable features, as shown in figure 4.2.

Figure 4.2 shows an example barcode described in 2D feature space with the linear discriminant boundary as the classifier. The green plus signs are the black elements and the blue circles correspond to the white elements while the large stars show the mean of each feature. This figure shows a well separated class of states, and any barcode with a similar feature spread will likely result in a perfect classification. A typical element region of interest should only include mostly high intensity pixels (for white elements) or low intensity pixels (for black elements), and the net result should therefore display a bimodal feature distribution (as figure 4.2 does). For read failures, bit flipping should be initiated with the elements that fall closest to the linear boundary, regardless of which side they are on. These

are the least confident elements, and would be the most likely misclassified elements.

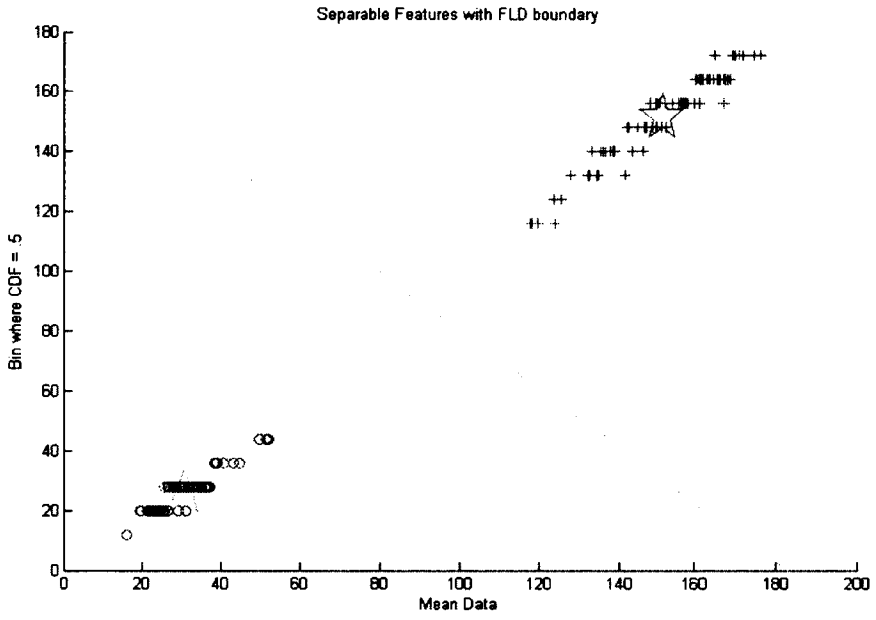


Figure 4.2: FLD Classifier Boundary

The ability to handle both types of barcodes necessitates robustness. It is important to note that the AYPSYS and ClearImage decoders overall had a difficult time in reading the skinny dot printed barcodes, while the successful read rates were much higher when reading the binarized version after processing. The skinny dot printing shows the importance of pre-processing due to the high rate of read failures when reading various types of barcodes. Additionally, even though the thick dot print reading presented mixed results, the processes presented here will form a solid basis as a test bed for future improvements.

4.2 Conclusions

This research lays solid foundation to build from. Significant theory crafting has been accomplished discussing many possible methods to pre-process barcode images and there are numerous approaches that will have varying degrees of effectiveness depending

on the situation. Since the barcode printing process is not as precise as desired, it becomes clear that powerful software needs to be created to be able to overcome these challenges. The developed platform provides the framework for study of other algorithm development and subsequent testing of data bases which contain contaminated or otherwise damaged barcodes. Since the barcode printing process is not as precise and uniform as desired, this platform becomes the vehicle to study just what pre-processing steps might be necessary to achieve the desired confidence level for barcode reading.

The results show that algorithms developed were successful at creating binarized versions of the barcodes, but with definite room for improvement. The key elements to the functionality of Process 1 are the erosion and region growing algorithms which takes pixel intensity and neighbors into account for image segmentation, while Process 2 uses a profile slice algorithm that extracts both horizontal and vertical edge data and decides the most likely element layout. The skinny dot printing algorithm (Process 3) uses a different approach and takes advantage of the orderly structure, it uses region growing at the center of each element to create best guess region of interest (ROI) for the element and applies Fishers Linear Discriminant to classify element features as white or black.

With the knowledge of the possibility of different printing processes, it leads us to explore options for designing a fully integrated system to handle all types of barcodes at high speed reads. Certain algorithms work better in certain instances, but it is unfavorable to run an untuned algorithm because it wastes time. The proposed solution attempts to decode the barcode with minimal processing and then subsequently implements the various algorithms in a logical manner to read the particular misaligned or damaged barcodes. By first attempting a quick barcode read we are able to read most clear barcodes on the first run and move on. When a read fail occurs, the system should decide whether there are thick or skinny elements present and which type of algorithm will be most effective, and then execute algorithms along that path until a successful read occurs.

The other method to ensure successful barcode reads is to include bit flip-

ping for the least confident elements. After processing, if the decoder still cannot read the barcode, we begin a process of bit flipping which involves starting with least confident element, inverting its value, and attempting reads. By cycling through all possible combinations we are likely to get a correct result provided we used a solid metric for the confidence score. Bit differences in directional profiles, weak transitions, abnormal pixel distances, and other artifacts that affect the smoothness of transitions can all indicate disturbances or distortions that could decrease our confidence about a particular element or set of elements. By flagging all of the potential element errors, we begin inverting those elements first in an attempt to get a successful read much earlier. This can also be applied to the older algorithms that rely on the grid method. In particular, if the region grown is unusually small or abnormally shaped, it could be evidence to support misalignments. When using the linear discriminant as a classifier, the least confident elements will simply be the ones closest to the equal probability boundary.

The primary downfall with using this method is the amount of processing time to run through all the permutations. Running every possible combination is clearly not feasible, but with prior knowledge of unsure elements we can limit the combinations to something computationally practicable. To wit, even just 10 unsure elements leads to over 1000 possible permutations, but the Reed-Solomon error correction coding enables successful reads even with some bit errors. This relaxes the computing amount necessary, and allows for a variety of element combinations to attain a successful read.

This work provides a solid test bed for industry developers. Finished product require maximized barcode reading capabilities and the system should be designed to adaptively apply algorithms based on read results, such as in figure 4.1. There is still significant future work to be performed (discussed in the final chapter) but the algorithms created here should a basis for future developments.

Chapter 5

Future Work

Creating a robust and efficient system tuned for speed is the goal for barcode readers in industry. It is optimal to create a decision tree that analyses each image for key features and then makes a decision about the best algorithm set to implement, if any, as the least amount of processing is essential for quick reads. We presented here a generalized system flow that should maximize barcode read rates while minimizing read times. Developing a test bed to determine which methods are most successful in certain circumstances would be very useful so developers should follow the suggested generalized development flow in figure 5.1. The flowchart contains three core algorithm types that should be implemented in order, after all options with the previous algorithm type have been exhausted.

There are many necessary improvements to be made to this contributing theory before being put into action. First and foremost, the methods developed here (included on the CD attached to this thesis) should be modularized and wrapped into a testing friendly program that includes access to the decoding software internals. This allows the system to access the decoder with the raw element information, and by eliminating the operating system executable calls, it immediately speeds up the entire decoding process. Fabricating synthetic images to send to the decoders then becomes unnecessary and intelligent system design can begin. This testing bed as described should provide the framework for creating a final system design.

Figure 5.1 builds from the following three ideals:

1. Minimal processing is optimal when possible.

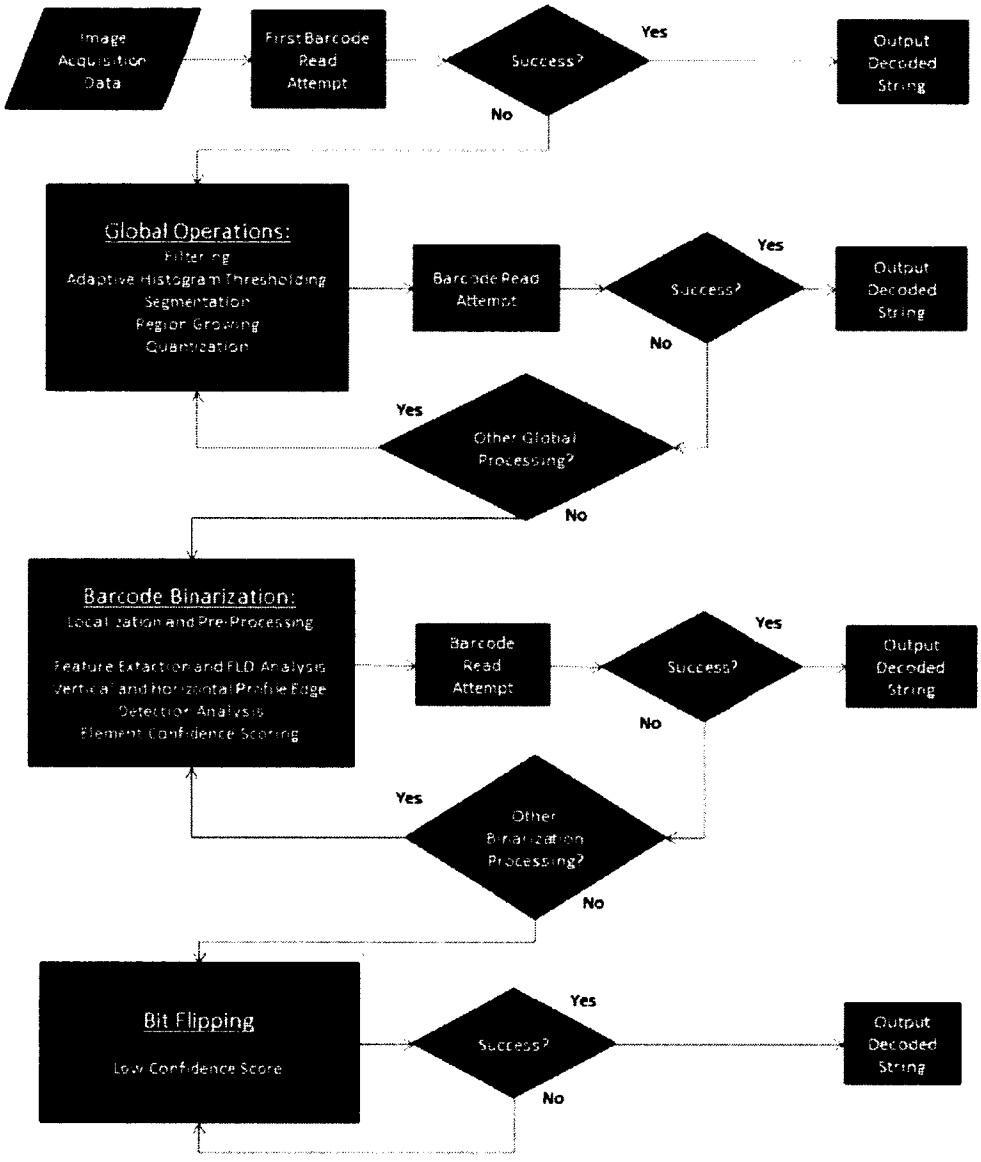


Figure 5.1: Suggested Algorithm Development Flow

2. Global operations that adjust pixel data without attempting to classify elements are superior because they eliminate the possibility of user misclassified elements.

3. Binarization and logical bit flipping should almost always be successful, but should only be attempted after exhausting all other options because of the processing time.

Firstly, if a clean barcode is encountered, it should be able to be read without a problem on the first attempt. In the case of failure, global methods should be implemented to act on the image as a whole. There are several varieties of methods that could be attempted (together or recursively separate) for a successful barcode read. In general most clean barcodes should be read at this point and read failures here suggest significant parameter distortions as described in (14) and (8). By this point, image binarization/element classification is the goal as the barcode is too badly damaged to be read as is, and therefore the system needs to make decisions about every element in the hopes to get enough correct to decode the message. Binarization methods could include those developed here or further developed algorithms taking new or adjusted approaches. Here the element classification should be tied closely to the confidence scoring, in anticipation of misclassified elements, due to poor barcode quality. In this way, the bit flipping procedure is tuned to get a correct read as quickly as possible. The specifics of each stage will be left to the testing designers depending on what produces the highest read rates.

A single more generalized approach to binarization may also prove successful if properly designed. The choice of edge detection could be expanded into two dimensions essentially creating a contour map of the barcode at perceived edges. Our theory holds that there are distinct edges between white and black elements, and finding closed contours surrounding like elements coupled with the neighborhood element information could provide a powerful approach to improve on these algorithms. Post contour analysis can determine height, width, and pixel calculations to estimate the shape and layout of the element blob. The greatest challenge then is the development of a fitting algorithm to align all the blobs together to coincide with a valid matrix of binary information.

A glaring element of this work necessary for robustness is the analysis of barcodes in non-ideal conditions. In this research, the barcodes have been relatively clean with ideal lighting. To add significant robustness, experimentation must be done into many of the other non-ideal parameters such as non-uniform lighting, differing contrast,

bent or misshaped codes, and physical obstructions. This would really push the limit on what deformities a barcode can face while still being readable. If a test bed was developed in this way, it would possible to take real dirty environment slides and perform failure analysis to determine what other real world non-ideal conditions can and will exist. Refer to figure 5.2 for many examples of how real dirty environments can affect barcodes. Materials like liquids, dust, or physical obstructions can significantly interfere with the ability to distinguish elements. As can be seen, non-ideal conditions can cause bright white reflections, warping or distortions, significant decrease in contrast, or inability to see elements. In some cases, very robust software may be able to handle these situations, but it is likely that some will be impossible, such as dirty barcodes 5, 9, and 10. This is an area of research that is totally open due to the wide range of possible interference.

Regardless of the problem framework, it has been have proven that I developed good binarization methods for ideal condition barcodes and set ground work for a test bed for future work to analyze non-ideal images.

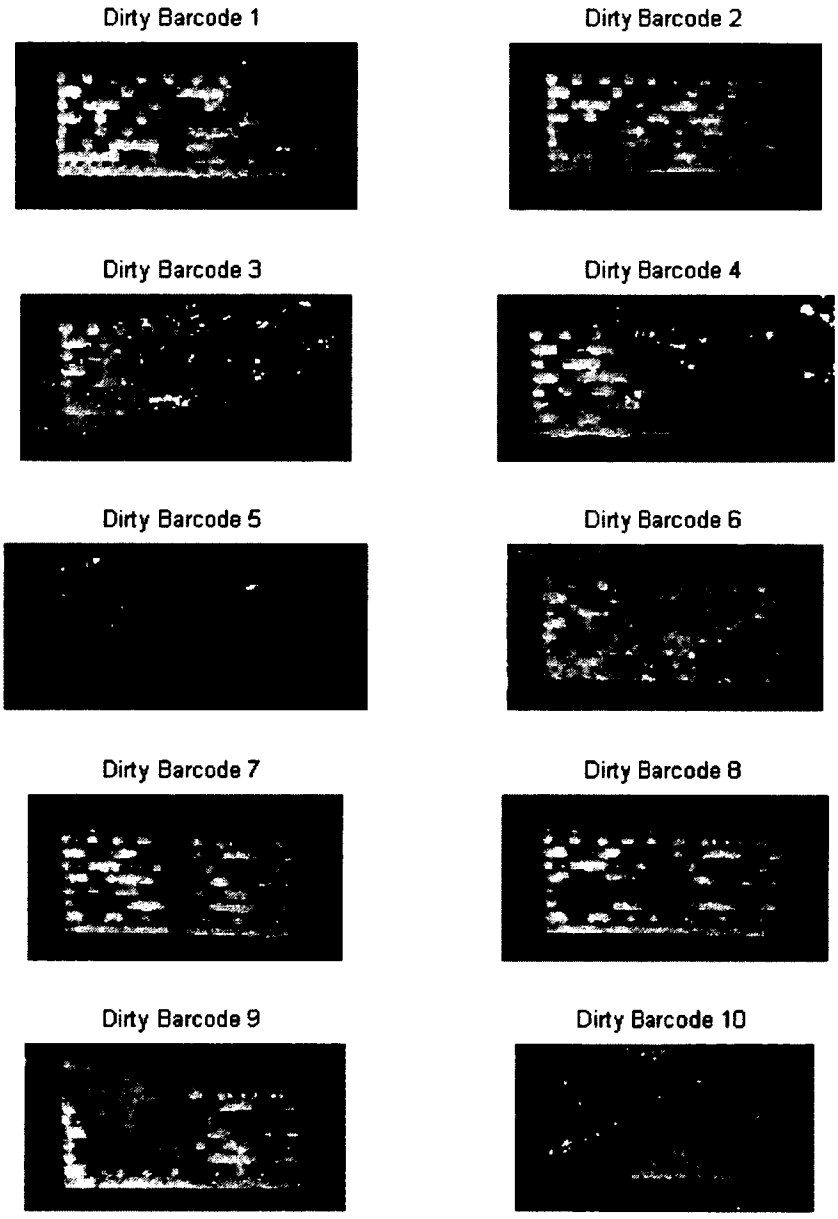


Figure 5.2: Dirty Barcode Examples

List of References

[1] R. DUDA, P. HART, AND D. STORK. *Pattern classification.* Wiley, 2nd edition, 2001. 13, 14, 15

[2] TASOS FALAS AND HOSSEIN KASHANI. Two-dimensional bar-code decoding with camera-equipped mobile phones. In *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*, pages 597–600, march 2007. 5

[3] R. A. FISHER. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **7**(2):179–188, 1936. 17

[4] A. FORMIGA, S. SIMSKE, AND M. THEILO. An assessment of data matrix barcode recognition under scaling, rotation, cylindrical warping. *ACM SYMPOSIUM ON APPLIED COMPUTING*, pages 266–267, 2011. 7

[5] KUNIHIKO FUKUSHIMA. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, **36**:193–202, 1980. 11, 13

[6] J. GAO, L. PRAKASH, AND R. JAGATESAN. Understanding 2d-barcode technology and applications in m-commerce-design and implementation of a 2d barcode processing solution. IEEE Computer Software and Applications Conference (COMPSAC), 2007.

[7] R. GONZALEZ AND R. WOODS. *Digital Image Processing.* Pearson Education, Inc., 3nd edition, 2008. 11, 12, 23, 24, 51

[8] GS1. *GS1 DataMatrix*, 2011. "An introduction to technical overview of the most advanced GS1 Application Identifiers compliant symbology". 3, 9, 64

[9] H. HAHN AND J. JOUNG. Implementation of algorithm to decode two-dimensional barcode pdf-417. In *Signal Processing, 2002 6th International Conference on*, **2**, pages 1791 - 1794, Aug. 2002. 5

[10] M. IWEN, F. SANTOSA, AND R. WARD. A symbol-based algorithm for decoding bar codes. *ArXiv e-prints*, October 2012.

[11] J. JUETT AND X. QI. Barcode locatization using a bottom hat filter. *NSF Research Experience for Undergraduates*, 2005. 19

[12] D. LIN, M. LIN, AND K. HUANG. Real-time automatic recognition of omnidirectional multiple barcodes and dsp implementation. *Machine Vision and Applications*, **22**:409–419, 2011. 8

[13] FENG LIU, ANAN LIU, MENG WANG, AND ZHAOXUAN YANG. Robust and fast localization algorithm for data matrix barcode. *Optoelectronics and Image Processing, International Conference on*, **2**:356–359, 2010. 7, 8, 29, 30, 33, 58

[14] MICROSCAN. *Understand 2D Verification*, 2011. "Comparing Quality Patameters for Data Matrix Symbol Verification". 3, 5, 64

[15] LAUGHTON MIKE. libdmtx: Open source data matrix software. `http://www.libdmtx.org/`, 2013. [Online; accessed 25-Feb-2013]. 4

[16] E. Ouaviani, A. Pavan, M. Bottazzi, E. Brunelli, F. Caselli, and M. Guerrero. A common image processing framework for 2d barcode reading. In *Image Processing and Its Applications, 1999. Seventh International Conference on (Conf. Publ. No. 465)*, **2**, pages 652–655, 1999. 1

[17] O. Pârvu and A. Bălan. A method for fast detection and decoding of specific 2d barcodes. *Telecommunications forum (TELFOR)*, pages 1137–1140, 2009. 6, 29, 30, 58

[18] T. Seideman. Barcodes sweep the world. *Wonders of Modern Technology*, October 2009. 3

[19] K. Voosen. It's all in the software [machine vision software]. *Manufacturing Engineer*, pages 43–45, 2007. 4

[20] A. Wilson. Vision system speeds barcode reading. *Vision Systems Design*, **36**:27–31, 2007. 5

Appendix A

MATLAB Functions

A.1 TrainDataMatrix (Process 1)

The source code for this function can be found on the CD appended to this thesis.

A.2 DM_AutoRotate (Process 1)

The source code for this function can be found on the CD appended to this thesis.

A.3 DM_erode (Process 1)

The source code for this function can be found on the CD appended to this thesis.

A.4 RegionGrowing (Process 1)

The source code for this function can be found on the CD appended to this thesis.

A.5 DM_Barcode (Process 2)

The source code for this function can be found on the CD appended to this thesis.

A.6 Profile2Bit (Process 2)

The source code for this function can be found on the CD appended to this thesis.

A.7 DM_Binarize (Process 3)

The source code for this function can be found on the CD appended to this thesis.

A.8 BuildFeatureSet (Process 3)

The source code for this function can be found on the CD appended to this thesis.

A.9 Histogram (Process 3)

The source code for this function can be found on the CD appended to this thesis.

Appendix B

Microsoft Visual Studio Code

B.1 DecodeDataMatrix

The source code for this function can be found on the CD appended to this thesis.