Barcode Decoding in a Camera-Based Scanner: Analysis and Algorithms

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Madeline J Schrier

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Fadil Santosa

June, 2015

Acknowledgements

This thesis represents the culmination of my time in graduate school. It would not be possible without the support and love of so many to whom I am eternally grateful. Here I would like to express my gratitude to some of the big players:

My advisor Fadil Santosa, an always calming and reassuring presence. Thank you for seeing something in me when I first came to Minnesota. I could not envision a better advisor to listen to my wants, push me in the right directions and help me achieve my goals. My math-grad friends, particularly Alanna, David, Jeffrey and Joe. Thank you for serving as sounding boards, counselors, gym buddies, and happy hour regulars. It's been a good four years. My family - a constant source of love, prayers and emotional relief. You told me I could do it and I did. Thank you for always believing in me. And finally, Ryan. Thank you for sticking by my side from the very first day of orientation. This is an accomplishment for both of us, and I cannot imagine a better person to share it with.

Abstract

In this thesis we present the first rigorous study of resolution requirements in camera-based barcode scanners. The question we wish to address is "What is the resolution needed in the captured image to unambiguously decode a barcode?" For simplicity, we consider the UPC barcode, which is widely used in retail and commerce. A UPC barcode consists of black and white bars of different widths. The widths of these bars encode a 12-digit number according to a look-up table. We show that the camera model can be completely determined by a set of parameters defining the width of the bars and the shift in the image. These parameters can be determined by utilizing features of the UPC symbology, and the knowledge of these parameters allows us to decode exactly. We show that these two parameters can be recovered from the image data for narrowest bars larger than three-fourths a pixel and in some cases, only half a pixel. We examine two extreme cases and show that unique determination of the digit is possible in these "worst case scenarios," even under the presence of noise. Following this rigorous investigation, we illustrate the procedure from start to finish with a numerical example and also examine the performance of the algorithm under noise.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Barcodes are a very familiar technology to most people. We encounter them on numerous products, such as food packaging and shipping boxes, and see them used in a multitude of applications including retail and advertising. This work seeks to characterize the limits of a camera-based barcode decoding method. The question we wish to address is "What is the resolution needed and the noise level permitted in the captured image to unambiguously decode a barcode?" For simplicity, we consider the UPC barcode which is widely used in retail and commerce. The question amounts to unique and stable determination of the digits for a fixed ratio of the pixel size to narrowest bar width. Under some circumstances, we show that the digits are uniquely determined if the narrowest bar width is no smaller than half the pixel size.

1.1 Brief history

The story of the barcode dates back to the early 1930s with punch card sorting machines. The motivation for development of barcodes arose from a desire to automate the checkout process at the grocery store and thus decrease the amount of time and effort spent by both customers and employees. By the 1960s, several different symbologies had been proposed by various companies. These symbologies included the bull's eye, circular codes and codes designed to be recognizable to the human eye as the characters they were representing. It was not until the early 1970s that the Universal Product Code (better known as UPC) was selected as an industry standard. As technology improved,

barcode use spread beyond the grocers to be used by the military, health sector and various manufacturing industries. By the late 1980s, the United States Postal Service (USPS) had developed its own symbology of bars with varying heights still seen today on mail delivered by USPS. [1]

1.2 Barcode usage and symbologies

Barcodes may be one-dimensional (a line of black and white bars) or two-dimensional (concentric rings or a square of modules) and exist in various shapes and sizes with various capacities and error-corrections. We define a *symbology* as the representation of any ASCII character by a sequence of 0s and 1s. To make this into a barcode image, "0"s and "1"s are replaced by white and black modules respectively of some fixed size.

As some symbologies lend themselves to certain applications better than others, there are many different uses for barcodes. Barcodes are used for inventory, health care applications, sales, manufacturing, postal applications, warehousing, shipping, virtual tickets, and many other applications. Although there new technologies being developed that perform similar functions, barcodes remain in high use today due to their inexpensive cost, flexibility and reliability [1].

As a particular example, we can look at Universal Product Code (UPC). Most people are familiar with UPC barcodes from their trips the grocery store. The UPC symbology is a common symbology for retail. Most UPC barcodes consist of 12 digits. Each of these digits is represented by a sequence of seven black and white bars so that each digit is "overcoded" and allows for error correction. For this one-dimensional symbology, a module is a single black or white bar of some fixed width. Every UPC barcode starts and ends with the same sequence of three bars: black-white-black. Similarly, in the middle of every UPC barcode, between the sixth and seventh digits is the sequence: white-black-white-black-white. Thus each UPC barcode is represented by a sequence of 95 black and white bars [1][2]. Figure 1.1 shows this structure on the code '814723686393'. The digits have complementary representations on either side of the middle sequence as can be seen in Table 1.1.

One interesting structure of the UPC code is the check-digit. In any true UPC barcode the twelfth digit is a check-digit, which allows for verification of correct decoding

Figure 1.1: '814723686393' encoded using UPC symbology. Each sequence for start, left digit, middle, right digit, and end is shown with divisions for identification ease.

Table 1.1: Left and Right Patterns for UPC Symbology

Character	Left	Symbol (Left)	Right	Symbol (Right)
0	0001101		1110010	
1	0011001		1100110	
2	0010011		1101100	
3	0111101		1000010	
4	0100011		1011100	
5	0110001		1001110	
6	0101111		1010000	
7	0111011		1000100	
8	0110111		1001000	
9	0001011		1110100	

or a request to re-read the barcode. The check-digit is calculated as three times the sum of all digits in odd-numbered positions, plus the sum of the digits in even-numbered positions, taking that result modulo ten, and finally ten minus the modulus [1]. For example, we consider the UPC barcode for a particular hand cream: '019045182749'. We now verify that 9 is the check-digit as outlined above.

$$3(0+9+4+1+2+4)+(1+0+5+8+7)=81$$

$$10-(81 \mod 10)=9$$

On the other hand, the barcode shown in Figure (1.1) is just a randomly generated 12-digit number and not an actual UPC barcode. Thus we see that although the last

digit is 3, we calculate a check digit of 9. For our work, we disregard the check-digit and thus consider all possible 12-digit combinations of digits zero through nine.

Barcodes are not restricted to one dimension. There are many two-dimensional barcode symbologies that serve various applications. In these 2D barcodes, the white and black bars from the 1D barcodes are now white and black modules, most often squares. The increase in dimension allows for more data to be encoded, but the added dimension can also require more complex decoding algorithms and increase the effect of noise on the image.

Figure 1.2: A sample Version 3 (29 by 29) QR Code encoding the link to the University of Minnesota SIAM Student Chapter homepage. This image was generated using www.qrstuff.com

Recently, there has been a surge in popularity of the QR Code. QR stand for "Quick Response." This symbology was developed in 1994 by Denso Wave in Japan in response to a desire to encode Japanese characters as well as alphanumeric characters. An international standard was developed for it in the year 2000, and QR Code is now in the public domain. QR codes consist of black and white modules. There are forty different sizes of the code, varying from Version 1 at 21×21 modules up to Version 40 at 177×177 modules. Version 40 can encode up to $7,089$ numeric or $4,296$ alphanumeric data [1][3][4]. With two dimensions, barcode orientation becomes increasingly important. QR Code has a distinct finder pattern of three fixed orientation targets. These targets are black squares with a white and black ring located in the both upper and lower left-hand-side corners as well as the upper right-hand-side corner. In addition to the finder pattern, as the size of the QR Code increases, additional alignment patterns are added. Reed-Solomon error correction is used for this symbology and ranges from

level L, where 7% of characters can be corrected, to level H, where 30% of characters can be corrected [1]. QR Code is used in many applications including logistics and manufacturing. This symbology can encode symbols, binary data, control codes and multimedia data [3][4]. One popular application is advertising. Companies will put a QR code on a billboard, bench advertisement or magazine page hoping the consumer will capture the code with his or her camera phone and be taken to a webpage with more information about the company or product. These mobile versions only use Versions 1-10 because of the limitations of camera phones [3][4].

Figure 1.3: A sample 27×27 Aztec barcode encoding the data "An Aztec Bar Code Example using www.barcode-generator.org"

Another prominent two-dimensional symbology is Aztec. Aztec Code was developed in 1995 by Andy Longacre at Welch Allyn. Like UPC and QR it is now in the public domain. As with QR, an Aztec barcode is a square comprised of black and white square modules. An Aztec code features a central bullseye with either two (compact) or three (full-range) rings. From this center pattern, data is encoded in layers that spiral out in a clockwise direction. Each of these layers is two modules thick, and there can be a maximum of thirty-two data layers. The smallest Aztec codes are 15×15 modules and the largest are 151×151 modules. Unlike QR and UPC, which require quiet-zones not included in the size, Aztec Code does not require any quiet-zone for proper decoding. The error-correction level of Aztec is adjustable, but the recommendation is approximately 23% plus three codewords. With this recommended level, the largest Aztec codes can encode up to $3,047$ alphanumeric characters or $1,914$ bytes of data [1][5]. The Aztec symbology lends itself particularly useful to online ticket applications. For example Aztec barcodes may be found of printed tickets that were purchased online or on electronic boarding passes for many airlines [5].

1.3 Challenges in barcode decoding

Although barcode technology has been around since the 1960s, camera based decoders add a new set of challenges to the decoding process. For example, we now must concern ourselves with the resolution of the image. Although modern cameras have many pixels, the barcode in the captured image usually is contained in a small subregion of the image domain. As such, resolution may be limited unless one is deliberate in maximizing resolution, which is not usually practical. Long-range scanning is another challenge area where resolution is relevant. This has practical applications in warehouse inventory and the like. In this work, we study the level of resolution needed to decode.

Additionally, we are concerned with noise from distortion of the barcode image. Such distortions can result from a variety of environmental conditions including non-uniform lighting, the angle of the camera at the time of image capture, and the physical surface e.g. curved or reflective.

Many complications arise in finding the barcode in the image, a problem known as *localization* and discussed in further detail in §2.2. In this work, we additionally address the challenges of identifying the start and end sequences of the barcode in order to have reference points with which to decode the barcode.

Finally, a practical algorithm must be used. Even in the best image conditions, there is great demand for fast and efficient algorithms. Practicality and functionality demand that one must be able to scan and decode within a few milliseconds.

1.4 Outline of this work

Having explained the several symbologies and highlighted some of the challenges of this problem, we will fine tune the problem and embark on a more detailed examination of the problem. In Chapter 2, we present a model of barcode image capture and restrict the focus of this study to one-dimensional UPC barcodes. Chapter 3 outlines the inverse problem of decoding a barcode from image data. Key to this process is the recovery of the camera model parameters, the subject of Chapter 4. We explain the parameter recovery process for both clean and noisy data. In a related study, Chapter 5 examines the resolution requirements for the unique determination of a barcode element in both well resolved and poorly resolved images. In order to better understand the process,

we provide a numerical example in Chapter 6. This chapter demonstrates the process from image capture to barcode recovery and illustrates the performance of our greedy algorithm under various noise levels. Finally, Chapter 7 presents a final discussion of the investigations and key results.

Chapter 2

Camera-Based Barcode Decoding

In retail and non-retail alike, there is a strong movement away from the traditional laser based scanning and decoding toward camera-based decoding. Instead of running a laser over the barcode, the barcode is captured by a camera, allowing for potentially more robust decoding.

2.1 A model of image capture

Figure 2.1: g_{ij}: intensity at the ij^{th} pixel (shown in green) is the average intensity over the pixel.

Figure 2.1 represents the image capture of one particular pixel of a barcode. Let $\square_{ij}$ denote the exposed region for pixel (ij), meaning the region whose intensity will be measured by the ijth pixel. The intensity is modeled simply as the "total photon count" over the pixel region,

$$g_{ij} = \int_{\square_{ij}} w(x, y) dx dy, \tag{2.1}$$

where $w(x, y)$ is the intensity level of the image at (x, y). Thus for a camera with $M \times N$ pixels, the captured image is a matrix with entries g_{ij}. In practice, we capture black-and-white images, so g_{ij} is an integer taking values between 0 and 255. Finally we note that when a barcode is scanned using a camera, the image captured is not only that of the barcode, but also the surrounding environment (or clutter). Nevertheless, the camera model proposed above applies to cluttered images as well.

2.2 Localization problem

The localization problem consists of identifying the barcode in a cluttered image. This is typically done by identifying a bounding box that contains the barcode. Traditionally the localization problem is solved by one of two methods: spatial domain based or frequency domain based method. For one-dimensional barcodes, some algorithms use the spatial domain based Hough Transform, which utilizes the linear structure of the barcode [6][7]. Some of the frequency domain based methods, such as Gabor filtering, depend greatly on training images sets [6][8]. These methods utilize the high gray change frequency of the barcode. Other strategies aim for efficiency in finding the orientation of the barcode on a low resolution image before reading in the entire image data [9][10].

For the following work, we assume that the bounding box has been found and the orientation of the barcodes in question are in alignment with the standard x and y axes. The latter can be achieved by resampling the barcode image in the xy-oriented pixel layout. We are keenly aware that by so doing, some information is lost.

2.3 Inverse problem – decoding a *word* from the image

Of particular interest is the problem of decoding a word from the image. A *word* is the information encoded by the black and white bars of the barcode. For example, a UPC barcode encodes the a 12-digit number. This 12-digit number is the word. The relationship between the word and the image is given by

$$g = \alpha T z, \tag{2.2}$$

where g is the image and α is a scaling factor to reassign pixel values between 0 and 255. The forward operator T incorporates the structure of the symbology and the

information of what each pixel sees. z is a binary vector picking out the corresponding characters for each piece of the barcode. A more detailed characterization of T will be provided below.

2.3.1 From *word* to barcode

For the moment, we will only consider UPC codes. The *word* is a 12-digit number represented as a concatenation of 12 10-vectors interspersed with three entries of "1". Each 10-vector has only one nonzero entry, namely a "1" corresponding to the digit it represents. For example, $[0,1,0,0,0,0,0,0,0,0]^T$ represents the digit 1, etc.

Let $z^{(i)}$ be the i^{th} digit. Then the *word* vector is

$$z = \begin{bmatrix} 1 \\ z^{(1)} \\ z^{(2)} \\ \vdots \\ z^{(6)} \\ 1 \\ z^{(7)} \\ \vdots \\ z^{(12)} \\ 1 \end{bmatrix}, \tag{2.3}$$

where the "1" entries correspond to the start, middle, and end sequences which are always present on UPC codes.

2.3.2 Dictionary

The dictionary maps between the word and the barcode. For UPC, it is a 95 by 123 matrix consisting of only of zeros and ones. Let A represent the UPC dictionary. A has

the following structure:

$$A = \begin{bmatrix}
S & 0 & \cdots & & & & & & & & \cdots & 0 \\
0 & L & & & & & & & & & \ddots & \vdots \\
\vdots & & L & & & & & & & & & \\
 & & & L & & & & & & & & \\
 & & & & L & & & & & & & \\
 & & & & & L & & & & & & \\
 & & & & & & L & & & & & \\
 & & & & & & & M & & & & \\
 & & & & & & & & R & & & \\
 & & & & & & & & & R & & \\
 & & & & & & & & & & R & \\
 & & & & & & & & & & & R & \\
 & & & & & & & & & & & & R & \vdots \\
\vdots & \ddots & & & & & & & & & & R & 0 \\
0 & \cdots & & & & & & & & & \cdots & 0 & E
\end{bmatrix}$$

where S and E are the 3-vector $[1,0,1]^T$, and M is the 5-vector $[0,1,0,1,0]^T$. L and R are $\mathbb{R}^{7\times 10}$ binary matrices representing the "look-up" table for the digits in a UPC. The columns of L and R are given in Table 1.1, and can also be found in [2]. Letting $c \in \mathbb{R}^{95}$ be a binary vector representing the barcode, we have $c = Az$.

2.3.3 Camera Model

For the moment, we restrict our analysis to the case where the barcode is one-dimensional and horizontal with respect to the camera aperture. Here, x_0 is the shift or offset from the left edge of the image to the very first bar on the lefthand side; d is the size of the pixel; and x_i are the righthand edges of the bars, which are assumed to have widths h as illustrated in Figure 2.2. One code element corresponds to a single digit and requires seven x_i. We can reduce the number of parameters in our problem to two by normalizing the pixel size, d, to be one. Thus the only parameters we need be concerned with are the horizontal offset or shift of the barcode, x_0, and the narrowest bar width, h.

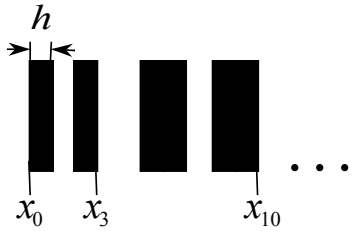


Figure 2.2: Front end of a barcode illustrating camera parameters (x_0, h)

We represent the barcode as a binary function $w(x) \in \{0, 1\}$. To represent this binary function we introduce $\chi_j(x)$, a characteristic function defined as:

$$\chi_j(x) = \begin{cases} 1 & x_{j-1} < x < x_j \quad (j\text{th bar}) \\ 0 & \text{otherwise} \end{cases}. \tag{2.4}$$

This function determines whether or not a particular bar is seen by the pixel k. Once again, c is a binary vector representing the barcode, where $c \in \mathbb{R}^{95}$ and $c_j \in \{0, 1\}$. With this construction, a barcode, represented as a binary function, takes the form

$$w(x) = \sum_{j=1}^{95} c_j \chi_j(x). \tag{2.5}$$

Letting N be the number of pixels, our camera matrix will be denoted $D \in \mathbb{R}^{N \times 95}$. The kj^{th} element of D is

$$D_{kj} = \int_{y_{k-1}}^{y_k} \chi_j(x) dx. \tag{2.6}$$

Without loss of generality, we can set $y_k = k$, $k = 0, 1, 2, \ldots, N$, and $x_j = x_0 + jh$. We also observe that

$$D_{kj} = 0 \text{ for } (i)\ y_{k-1} > x_j$$
$$\text{and for } (ii)\ y_k < x_{j-1}.$$

This is obvious from Figure 2.3. Pixels that do not overlap with $[x_0, x_{95}]$ are assumed to have zero intensity, such as the first pixel in Figure 2.3. Let $\lfloor x \rfloor$ denote the integer part of x. $g_k = 0$ for $k < k_-$, where $k_- = \lfloor x_0 \rfloor$, and $g_k = 0$ for $k > k_+$, where $k_+ = \lfloor x_{95} \rfloor + 1$. Necessarily, $D_{kj} = 0$ for $k \leq k_-$ and for $k \geq k_+$. With this understanding, the observed

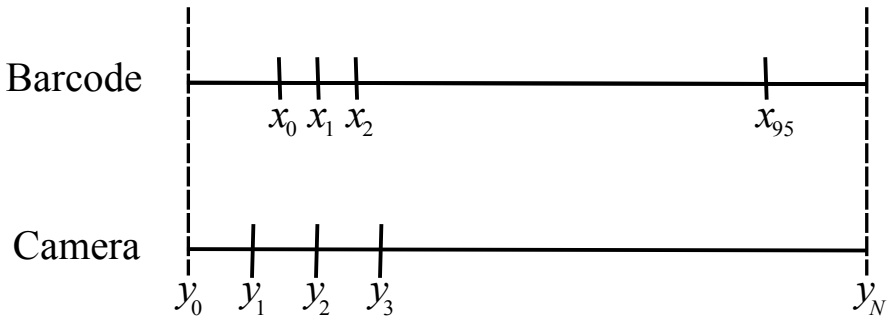


Figure 2.3: Notation illustration for barcode captured by N pixels

image is

$$\begin{aligned} g &= \alpha Dc \\ &= \alpha DAz, \end{aligned}$$

where $g \in \mathbb{R}^N$ are the pixel values, viewed as a vector. For now, we will assume α is known and study the properties of the forward operator $T = DA$.

Remark 2.3.1. *We note that in practice, zero intensity refers to black and intensity at 255 refers to white. We have reversed the scale here at no loss of generality.*

Chapter 3

Decoding Process

The process of decoding a barcode using the camera model begins with the capture of an image containing the barcode. As discussed in §2.2, the barcode is localized and the edges are detected. Features specific to the symbology are identified and used to determine the camera parameters. Once the camera matrix is known, the code is recovered by running a greedy algorithm from either end of the localized image.

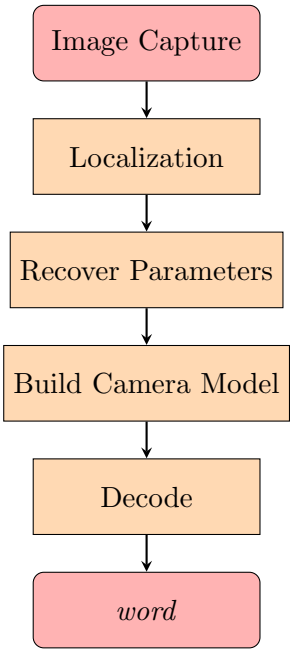


Figure 3.1: Decoding Process

Given our assumptions of symbology, alignment, and barcode dimension, the input image can be further simplified to a mere strip of N pixels reading a sequence of values representing image intensity at the pixels.

3.1 Edge detection

The first step in our problem, regardless of noise, is to detect the edges of the barcode in our image. At this stage we assume that the barcode has been localized. Let us define our edges in the following way. The *start* pixel, with intensity g_{start}, will be such that $x_0 \in [start - 1, start)$, i.e. $start := \lfloor x_0 \rfloor + 1$, where $\lfloor \cdot \rfloor := floor(\cdot)$. Once we have determined *start*, we modify x_0 such that

$$x_0 \leftarrow x_0 - \lfloor x_0 \rfloor.$$

Thus the new $x_0 \in (0, 1)$. Similarly we relabel the pixels, ignoring everything before the *start* pixel. In this way $start \equiv 1$, so we call $g_{start} = g_1$. The last pixel, with intensity g_m will be such that $x_0 + 95h \in (m - 1, m]$, i.e. $m = \lceil x_0 + 95h \rceil$, where $\lceil \cdot \rceil := ceil(\cdot)$. Thus the barcode is contained in m pixels.

In the noiseless case, edge detection is as simple as finding the first and last nonzero entries of the signal, i.e. identifying the pixels containing the first (last) bar of the start (stop) sequence. When we introduce a noisy signal, more sophisticated techniques are required to locate the edge pixels. For noisy signals, we utilize Canny edge detection, first presented in [11]. This method of edge detection convolves the signal with the first derivative of a Gaussian. The edges are thinned using non-maximum suppression in which each pixel is compared to its neighbors. Finally, some thresholding is applied to determine the beginning and end of an edge. Since we are considering only 1D barcodes the edge detection problem is simplified significantly.

3.2 Barcode Parameters

For the purpose of simplifying the discussion, we relabel the origin of the x-axis so that the left edge of the barcode, x_0 lies on the first pixel. Moreover, we truncate the image to m pixels as described earlier so that the right edge of the barcode, x_{95}, falls on the m^{th} pixel.

In order to decode correctly, we must retrieve the parameters that locate the barcode within the camera window from the data. As is previously mentioned, we assume that the pixel width d has been normalized to 1. Referring to Figure 3.2, we see the barcode as it is read from left to right and are able to more clearly define the relationship between the parameters. The front-end shift x_0 and back-end shift x_0' are related via the narrowest bar width h and the number of pixels m needed to see the barcode:

$$m - x_0' = x_0 + 95h \tag{3.1}$$

In Chapter 4 we will discuss the process of obtaining these barcode parameters. For

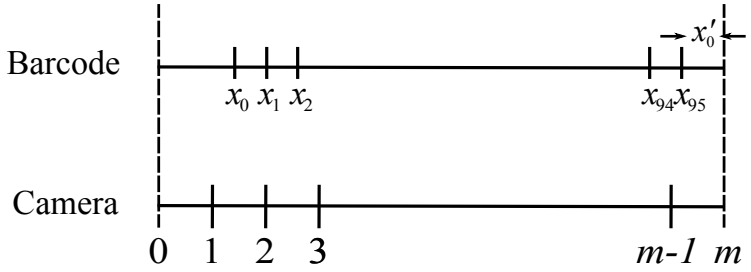


Figure 3.2: Reading the barcode from left to right

the time being, let us assume the data has been processed to return (x_0, h, x_0'). Then we may begin the decoding process as these parameters completely determine the camera. Furthermore, we will show that these parameters can be recovered from the image.

3.3 Decoding Algorithm

After detecting the edges and determining the camera parameters (discussed in greater detail in §4) we have enough information to decode the barcode provided that the narrowest bar in the barcode is sufficiently large in comparison to the pixel size. The first step of the algorithm is to use the newly generated camera matrix to subtract the set start, middle and end sequence information from the input data. To determine the first digit we calculate the minimum energy for each of the ten choices $\{0, \ldots, 9\}$ and return the minimum, i.e. the correct digit. The intensity corresponding to the first digit is removed from the data and the algorithm moves to determine the next digit. The

algorithm ends when all the digits have been determined. Such an algorithm is known as a greedy algorithm.

3.4 Focus of this study

In this study, we will show that the camera model can be completely determined by a set of parameters defining the width of the bars and the shift in the image. The parameters will be determined by utilizing features of the symbology. The knowledge of these parameters will allow us to decode exactly. Then, we will examine two extreme cases and show that unique determination of the code element is possible in these "worst case scenarios," even under the presence of noise. Following this rigorous investigation, we illustrate the procedure from start to finish with a numerical example and also examine the performance of the algorithm under noise.

Chapter 4

Determining the Barcode Parameters from Edge Information

In this chapter we explain in detail the process of determining the barcode parameters. This process is extremely important as these parameters completely determines the camera matrix and thus knowledge of the parameters allows us to solve the inverse problem of decoding the barcode from the data. The parameters in question are the narrowest bar width h, the shift x_0 from the edge of the image to the start of the barcode, and the corresponding shift x_0' from the opposite edge of the image to the end of the barcode, as seen in Figure 3.2.

4.1 Exploiting start and end sequences

As noted in §1.2, UPC barcodes have a set start and end sequence. In fact, as we can see from Table 1.1, once the barcode begins, we know the first and last four bars for every code: (black, white, black, white) and (white, black, white, black) respectively. Considering the low resolution case, where $h < d$, we normalize the pixel size $d = 1$ and examine the region of parameter space where narrowest bar width $h \in (\frac{1}{2}, 1)$ and shift $x_0 \in (0, 1)$. If the parameters (x_0, h) are known, the barcode can be uniquely

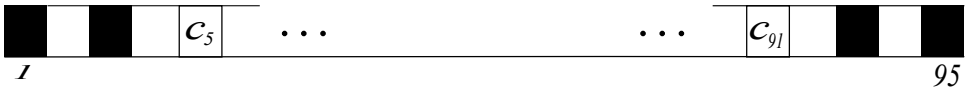


Figure 4.1: Known bars at either end of UPC barcode

determined, as we will explain later. By using the set start (end) sequence and the intensity information from the first (last) three information containing pixels, we can recover the parameters (x_0, h), allowing information from the barcode to be isolated. We are interested in the inverse problem of determining these parameters given only the first (last) three pixel intensities. We narrow the problem down to only three pixels because of the restrictions imposed by the parameter space under consideration. With $\frac{1}{2} < h < 1$ and $x_0 \in (0, 1)$, the set three-bar sequence will always appear in three pixels.

4.1.1 Data G and G'

Assuming no noise in the system, we let g_i, $i = [3]$, be the intensity seen in pixels one through three, respectively. That is

$$g_i = D_{(i,:)}(x_0, h) * A_{(:,1)}.$$

Furthermore, let us define G and G' as follows

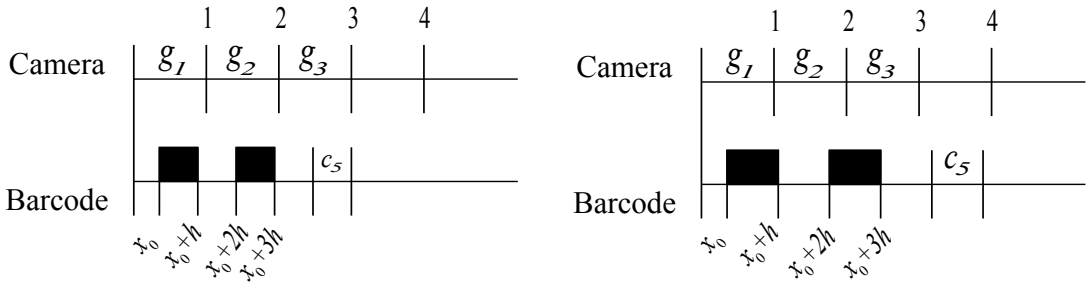
$$G = (g_1, g_2, g_3)^T \qquad G' = (g_{1'}, g_{2'}, g_{3'})^T,$$

where G is as above and G' is the intensity seen in the last three pixels such that $g_{1'}$ is the intensity of the last pixel to carry any information from the right end of the barcode.

4.1.2 Classifying different cases based on G and G'

In the region of parameter space under consideration, we find that there are unique identifying equations corresponding to the various subregions. These subregions are determined by inequalities relating x_0 and h.

Recall that the intensity at a pixel is equal to the length of the black line segment(s) that fall on the interval defined by the pixel. Thus, in Figure 4.2a, we see if $x_0 + h < 1$, then $g_1 = h$. On the other hand in Figure 4.2b, if $x_0 + h > 1$, we see that $x_0 = 1 - g_1$. Either side of the line $x_0 + h = 1$ is further divided by more characteristic inequalities.



(a) Start Sequence with $x_0 + h < 1$

(b) Start Sequence with $x_0 + h > 1$

Figure 4.2: Two start sequence examples

By solving for each parameter successively from g_1 to g_3, we are able to fully describe this region of parameter space.

The values of g_1 to g_3 determine the region in the (x_0, h) space in which special relationships hold so that both parameters can be identified. Consider for example the situation where x_0 and h are such that

$$x_0 + h > 1, \quad x_0 + 2h < 2, \quad \text{and } x_0 + 4h > 3.$$

Under these circumstances, the first bar (black) falls on the boundary between pixels 1 and 2, and the fourth bar (white) falls on the boundary between pixels 3 and 4. These inequalities imply that the second bar (white) falls entirely in pixel 2. Therefore, recalling that the intensity value at each pixel is the amount of black that falls in the pixel, we have

$$g_1 = 1 - x_0 \tag{4.1}$$

$$g_2 = 2 - (x_0 + 2h) + x_0 + h - 1 \tag{4.2}$$

$$g_3 = x_0 + 3h - 2 \tag{4.3}$$

with $g_i > 0$. From (4.1) and (4.2), we conclude that $x_0 = 1 - g_1$, and $h = 1 - g_2$. The last equation constrains the g_i's

$$g_1 + 3g_2 + g_3 = 2$$

which is the identifying characteristic on the data for this particular region in parameter space. We have labeled it as region *(3)* in Figure 4.3.

The parameter space $\{x_0 \in (0,1), h \in (\frac{1}{2},1)\}$ can be divided into seven regions each with a unique identifying characteristic. Thus given noiseless data $[g_1, g_2, g_3]^T$, one can identify a region in parameter space and values for (x_0, h). Table 4.1 gives each identifying equation in terms of g_i, $i \in [3]$, for the corresponding region in Figure 4.3.

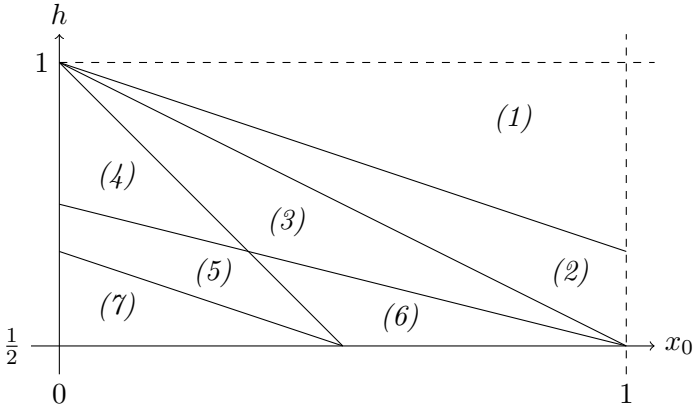


Figure 4.3: The seven regions in (x_0, h) parameter space for low resolution with $h > \frac{1}{2}$.

Table 4.1: Identifying Equations for Regions of Parameter Space

Region	Description	Identifying Equation
(1)	$x_0 + 3h > 3$	$\frac{1}{2}g_1 + g_2 + \frac{1}{2}g_3 = 1$
(2)	$x_0 + 3h < 3$, $x_0 + 2h > 2$	$g_1 + g_2 = g_3$
(3)	$x_0 + 2h < 2, x_0 + 4h > 3$, $x_0 + h > 1$	$g_1 + 3g_2 + g_3 = 2$
(4)	$x_0 + h < 1$, $x_0 + 4h > 3$	$u_1 = u_2 + u_3$
(5)	$x_0 + 4h < 3, x_0 + 3h > 2, x_0 + h < 1$, $c_5 = 0$	$g_1 = g_2 + g_3$
(5)	$x_0 + 4h < 3, x_0 + 3h > 2, x_0 + h < 1$, $c_5 = 1$	$g_3 = 1 - g_1$
(6)	$x_0 + 4h < 3, x_0 + h > 1$, $c_5 = 0$	$g_1 + 3g_2 + g_3 = 2$
(6)	$x_0 + 4h < 3, x_0 + h > 1$, $c_5 = 1$	$g_2 = g_3$
(7)	$x_0 + 3h < 2$	$g_1 = g_2$
(7)	$x_0 + 3h < 2$, $c_5 = 0$	$g_3 = 0$

When one of these identifying equations is satisfied, we may then determine the parameters (x_0, h) and uniquely decode the signal, with one exception. Notice in Table 4.1 that regions *(1)-(4)* all have a single identifying equations, where as regions *(5)-(7)* have two such equations. This ambiguity arrises whenever $x_0 + 4h < 3$ as in such a case g_3 includes information from the fifth bar c_5, which could be black or white depending on

the specific digit the barcode is encoding. A further discussion of this case is presented in §4.1.4. When $x_0 + 4h > 3$, as in regions *(1)-(4)*, we can immediately determine one of the parameters from g_1 and then solve for the other. For example, regions *(1)-(3)* (where $x_0 + h > 1$), $x_0 = 1 - g_1$. Narrowest bar width h can then be deduced by solving with g_2, g_3 and the recently determined x_0. When $x_0 + h < 1$, as in region (4), $h = g_1$ and x_0 follows. Specific formulas for all regions are found in Table 4.2.

Table 4.2: Parameter Solutions for Identifying Equation Regions

Region	Identifying Equation	Parameters		
		1^{st}	2^{nd}	c_5
(1)	$\frac{1}{2}g_1 + g_2 + \frac{1}{2}g_3 = 1$	$x_0 = 1 - g_1$	$h = g_3$	-
(2)	$g_1 + g_2 = g_3$	$x_0 = 1 - g_1$	$h = 1 - g_2$	-
(3)	$g_1 + 3g_2 + g_3 = 2$	$x_0 = 1 - g_1$	$h = 1 - g_2$	-
(4)	$g_1 = g_2 + g_3$	$h = g_1$	$x_0 = 2 - 2g_1 - g_2$	-
(5)	$g_1 = g_2 + g_3$	$h = g_1$	$x_0 = 2 - 3g_1 + g_3$	0
(5)	$g_3 = 1 - g_1$	$h = g_1$	$x_0 = 2 - 2g_1 - g_2$	1
(6)	$g_1 + 3g_2 + g_3 = 2$	$x_0 = 1 - g_1$	$h = 1 - g_2$	0
(6)	$g_2 = g_3$	$x_0 = 1 - g_1$	$h = 1 - g_2$	1
(7)	$g_1 = g_2$ & $g_3 = 0$	$h = g_1$	$x_0 < 2 - 3h$	0
(7)	$g_1 = g_2$ & $g_3 \neq 0$	$h = g_1$	$x_0 = 3 - 4g_1 - g_3$	1

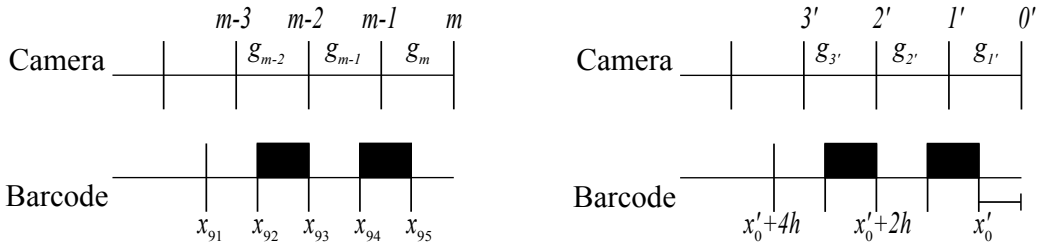
Theorem 4.1.1. *The pixel intensities in the first three pixels* $(g_1, g_2, g_3)^T$ *uniquely determine* (x_0, h) *in regions (1) – (4) in Figure 4.3 and in the remainder of the parameter space* if *the fifth bar in the barcode is black.*

Remark 4.1.2. *From the way the parameter space is separated into the seven regions, we can claim that the parameters* (x_0, h) *are uniquely determined if* $0 < x_0 < 1$ *and* $\frac{3}{4} < h < 1$. *This result is somewhat unsatisfactory because as we shall see later, the code itself is uniquely determined if* $h > \frac{1}{2}$ *in some circumstances. However, as we shall see in the next section, there is an obstruction to uniqueness in the determination of* (x_0, h).

4.1.3 Determination of parameters using stop sequence

Now let us pay specific attention to the stop sequence. As discussed in the beginning of §3.2, we know from the symbology that the end of the barcode is a reflection of

the beginning. Introducing parameter $x_0' = m - (x_0 + 95h)$, the distance between the end of the bars and the end of the last pixel, and relabeling the pixels such that $\{m, m-1, m-2, \ldots\} = \{0', 1', 2', \ldots\}$, we find ourselves in an identical situation as the start sequence case (See Figure 4.5).



(a) Old Notation (b) New Notation

Figure 4.4: Two notations for stop sequence

In Figure 4.4, we see the stop sequence illustrated with the new notation. Using this representation, all of the defining characteristic equations are the same as for the start sequence. Previously we used the line $x_0 + h = 1$ to determine whether or not the first bar fell entirely into the first pixel. Now, we use the line $x_0' + h = 1$ to determine whether or not the last bar is entirely contained in the last pixel. For example, if $g_m = g_{1'} = h$, then we know $x_0' + h < 1$.

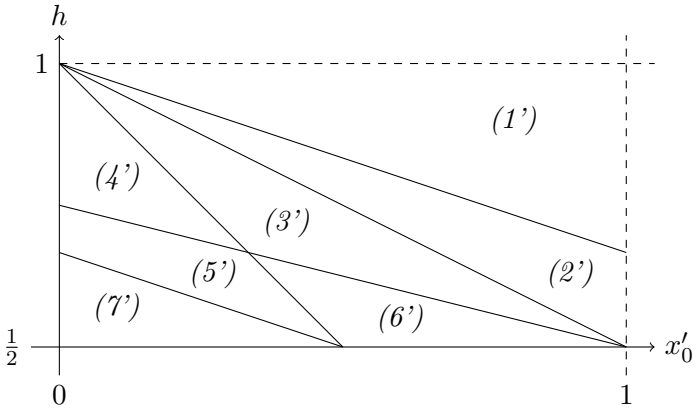


Figure 4.5: The seven regions in (x_0', h) parameter space for low resolution with $h > \frac{1}{2}$. Observe that these regions perfectly correspond to regions *(1)-(7)* in Figure 4.3.

Along with matching regions, we obtain matching identifying equations under the

following substitutions in Table 4.1:

$$x_0 \to x_0'$$

$$g_1 \to g_{1'} \quad (\equiv g_m)$$

$$g_2 \to g_{2'} \quad (\equiv g_{m-1})$$

$$g_3 \to g_{3'} \quad (\equiv g_{m-2})$$

The mapping between the x_0 and x_0' is found by equating the two systems. With respect to the start sequence the end of the barcode is represented as $x_0 + 95h$. With respect to the end sequence, the end is marked by $m - x_0'$. Thus $x_0' = m - (x_0 + 95h)$. As m is known immediately from the edge detection process, we can treat m as a fixed number, rather than as the ceiling function $m = \lceil x_0 + 95h \rceil$.

Not only does this additional information from the end of the barcode verify parameters x_0 and h, but it also reduces the number of cases of ambiguity in region *(7)*. Unfortunately the regions *(7)* and *(7')* have a nonempty intersection, so there are still some instances where both parameters cannot be determined from the start and end intensities; however outside this region we are guaranteed to uncover the parameters necessary for decoding.

4.1.4 Non-uniqueness

Ambiguity in parameter space

Regions *(1)* through *(4)* can be well described by their identifying equations and the inverse problem presents straightforward solutions when we find (x_0, h) in these regions. In regions *(5)* through *(7)*, g_3 includes information from the fifth bar c_5. In these regions, when c_5 is black ($c_5 = 1$), we have unique identifiers (as can be seen in Tables 4.1 and 4.2). When c_5 is white our identifying equations merge regions in parameter space, although the parameter calculations remain the same. In this way, although each of these ambiguous regions splits into two cases dependent on c_5, when c_5 is white, regions *(5)* and *(6)* are identical to *(4)* and *(3)*, respectively.

The knowledge of c_5 may be valuable in some applications. Should we wish to separate the ambiguity between regions when c_5 is white, we need simply determine on

which side of the line $x_0 + 4h = 3$ the parameters (x_0, h) fall. If they fall below this line, then c_5 is white. Otherwise, g_3 does not include c_5.

Region *(7)* is the most troublesome. The identifying equation for this region is $g_1 = g_2$. When c_5 is black, we can determine both parameters and thus the barcode. The case when c_5 is white comes with the additional identifier $g_3 = 0$. An example can be seen in Figure 4.2a. Although h can still be determined, we are limited in our information of x_0, only knowing an upper bound for this parameter: $x_0 < 2 - 3h$. In region *(7)*, $h \in (\frac{1}{2}, \frac{2}{3})$. Thus as $h \to \frac{2}{3}$ the upper bound on x_0 tightens.

Middle sequence counter example

One may hope to gain further information to recover parameters (x_0, h, x_0') by also exploiting the set middle sequence, using the same techniques as in the start and end sequences. The apparent advantages of such exploitation would seem to be profound as the middle sequence contains a set sequence of five bars: white-black-white-black-white. Furthermore, since every digit on the left ends in black and every digit on the right starts with black we end up with an invariant sequence of seven bars alternating black-white. In order to utilize this information, we examine the intensities of the middle two or three pixels for m even or odd, respectively. From our constraint $2h > d$ we know $4h > 2d$ and $6h > 3d$, thus we will not see outside of our seven digit set sequence in the chosen middle pixels. We then employ the same techniques as we did with the start and end sequences, identifying relations between g_i, x_0, and h as well as $g_{i'}, x_0'$, and h, where $i = \frac{m}{2}, \frac{m}{2} + 1$ for m even and $i = \frac{m-1}{2}, \frac{m+1}{2}, \frac{m+3}{2}$ for m odd.

Once again however we have ambiguous cases with the middle sequence where the right and left sides are mirror images. In fact, we can construct a counter example for the usefulness of exploitation of the middle sequence. Consider the case $(x_0, h, x_0') = (.5, .6, .5)$, as illustrated in Figure 4.6. Here $m = 58$ so we examine u_{29} and u_{30}. In this case $g_{29} = g_{30} = 0.6$. Furthermore, $g_{28} = g_{31} = 0.5$ giving complete symmetry to the problem and failing to provide any additional information.

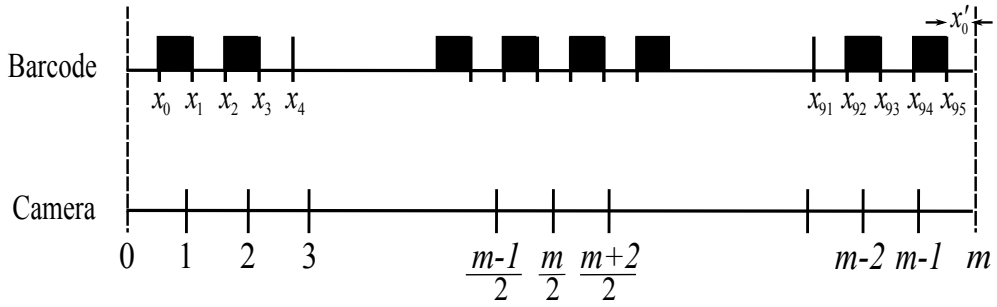


Figure 4.6: Counterexample for helpfulness of utilizing middle sequence in determining (x_0, h, x'_0)

4.2 Determining parameters from noisy data

When we add noise to the signal, the results from §4.1 no longer hold since a slight variation from the identifying equations causes the algorithm to fail. The solution to this problem is to look at the identifying equations themselves as planes in (g_1, g_2, g_3) space. Then when given noisy data for G, we project to the closest plane and use the projection point as our new intensity values.

We use Canny edge detection discussed in §3.1 to identify the first and last pixels to capture intensity from the barcode. Recall our definition of these pixels. The *start* pixel, with intensity g_{start}, will be such that $x_0 \in [start - 1, start)$, i.e. $start = \lfloor x_0 \rfloor + 1$. The last pixel, with intensity g_m will be such that $x_0 + 95h \in (m-1, m]$, i.e. $m = \lceil x_0 + 95h \rceil$. Thus the barcode is contained in m pixels.

Under our assumptions, each g_i can see at most h. Since $d = 1$ and we are constrained to the low-resolution case where $h < d$, thus our identifying equation planes are restricted to the unit cube. A visualization of these planes is show in Figure 4.7.

With a noisy signal, the input intensities likely do not lie on one of these planes. Letting (g_1, g_2, g_3) be the intensity inputs, we calculate the distance to the nearest plane. Using standard plane notation, each plane can be written as

$$ag_1 + bg_2 + cg_3 = d.$$

or equivalently,

$$\mathbf{n}^T G = d,$$

where $\mathbf{n} = [a, b, c]^T$ is a normal vector to the plane and $G = [g_1, g_2, g_3]^T$. In this way,

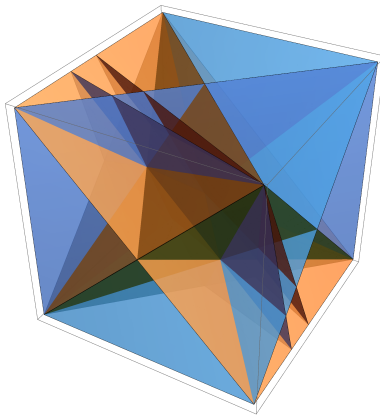


Figure 4.7: Planes in (g_1, g_2, g_3)-coordinates corresponding to the identifying equations of regions *(1)-(7)* of (x_0, h)-parameter space, restricted to the unit cube $[0, 1]^3$

we calculate the distance to each plane as

$$D(G) = \frac{|ag_1 + bg_2 + cg_3 - d|}{\sqrt{a^2 + b^2 + c^2}} = \frac{|\mathbf{n}^T G - d|}{\sqrt{\mathbf{n}^T \mathbf{n}}} \tag{4.4}$$

Calculating $D(G)$ for each of the seven planes and taking the minimum distance returns the closest plane to the noisy data point onto which we project in order to run the algorithm. Thus let $r = \arg\min_i\{D_i(G)\}$. Once we detect which plane is closest, we project the noisy data onto said plane and recover the parameters (x_0, h) using the projected data point

$$\tilde{G} = G + \mathbf{n}_r \left(\frac{d_r - \mathbf{n}_r^T G}{\mathbf{n}_r^T \mathbf{n}_r} \right). \tag{4.5}$$

Once we have the projected point, we test for satisfaction of an identifying equation and solve for (x_0, h, x_0') as in §4.1. It may be the case that there are multiple planes minimally distant from the noisy data point. In this case, a random choice of plane is made for the projection.

With the results of the projection for (x_0, h, x_0') we then perform a constrained optimization on either set of parameters (x_0, h) and (x_0', h') subject to the constraints $h = h'$ and $m - x_0' = x_0 + 95h$. Letting F be the function that maps parameters (x_0, h) to the corresponding intensity data of the first three pixels (and similarly for the last three), our optimization problem is as follows:

$$(x_0, h) = \arg\min_{x_0, h} \|G - F(x_0, h)\|^2$$

$$(x_0', h') = \arg\min_{x_0', h'} \|G' - F(x_0', h')\|^2,$$

where in each step we update the parameters and test for satisfaction of the constraints under some tolerance level. In this way, we leverage each end of the barcode to accurately determine the camera parameters.

Chapter 5

Unique Determination of a Barcode Element

We now investigate the issue of decoding a digit in a barcode image. Referring to Table 1.1, we see that each digit is associated with a binary sequence which is translated into a sequence of black-and-white bars of variable widths. The data given to us are pixel intensities where the pixels may be larger than the smallest bar width. Moreover, there is a potential for "cross-talk" between adjacent pixels in the sense that a bar is "seen" by two pixels.

Our approach will be to consider a set of contiguous pixels whose extent cover the part of the barcode corresponding to a single digit. We focus on this subproblem and attempt to give a rigorous characterization of the limiting conditions at which a digit can still be determined from the data. The analysis will be separated into two cases: (i) small pixels, (ii) large pixels. Case (i) is the easy case as the narrowest bars are sufficiently resolved. Case (ii) tests the limits of decoding and requires the use of the symbology of the UPC.

5.1 Unique determination in the case of well resolved images

We consider the case of "small" pixels. This is a desirable condition for higher resolutions as a single bar of the barcode will span multiple pixels. We can see a representation of this scenario in Figure (5.1). Algebraically, we represent this case as $h \geq 1$. In this section, we show that for small pixels, the transformation matrix $T = DA$ is block-diagonal so long as any row $k \in [N]$ of the camera matrix D has no more than two adjacent non-zero entries, where $[N] = \{1, \ldots, N\}$. Since D only has entries in a neighborhood of a diagonal transformation, we can restate the restriction to be "so long as any row $k \in [N]$ of matrix D has no more than two non-zero entries."

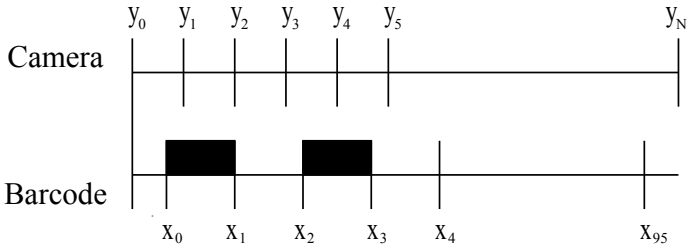


Figure 5.1: A high resolution case in which the pixels are smaller than the narrowest bar width.

Lemma 5.1.1. *Any row $k \in [N]$ of the camera matrix D has no more than two adjacent non-zero entries for "small pixels," i.e. when $h \geq 1$.*

Proof. Let us first recall the construction of D in (2.6). D possesses the following properties (regardless of the relationship between d and h):

$$\sum_{k=1}^{N} D_{kj} = h \text{ for every column } j \in [95] \tag{5.1}$$

and

$$\sum_{j=1}^{95} D_{kj} = \begin{cases} 1 & \text{for every row } k \in \{\lceil x_0 + 1 \rceil, \ldots, \lceil x_0 + 95 \rceil\} \\ 0 & \text{elsewhere.} \end{cases} \tag{5.2}$$

Now recall from our model that the pixel width is $d = y_k - y_{k-1} = 1$, $k \in [N]$ and the narrowest bar width is $h = x_{j+1} - x_j$, $j \in [95]$. An arbitrary pixel starts at y_{k-1} and

ends at y_k. We consider $h > 1$. We are interested in rows of D with non-zero entries, i.e. non-empty pixels. Thus we consider only the cases where the k^{th} pixel sees at least part of the j^{th} bar. There are three such cases.

$$D_{k,j} = \begin{cases} x_j - y_{k-1} & \text{for } x_{j-1} \leq y_{k-1}, x_j \in [y_{k-1}, y_k] \\ 1 & \text{for } x_{j-1} \leq y_{k-1}, y_k \leq x_j \\ y_k - x_{j-1} & \text{for } [x_{j-1}, x_j] \not\subseteq [y_{k-1}, y_k] \end{cases} \tag{5.3}$$

Then the k^{th} pixel sees this much of bar $j+1$:

$$D_{k,j+1} = \begin{cases} y_k - x_j & \text{for } x_{j-1} \leq y_{k-1}, x_j \in [y_{k-1}, y_k] \\ 0 & \text{for } x_{j-1} \leq y_{k-1}, y_k \leq x_j \\ 0 & \text{for } [x_{j-1}, x_j] \not\subseteq [y_{k-1}, y_k] \end{cases} \tag{5.4}$$

Now summing these together we have

$$D_{kj} + D_{k,j+1} = \begin{cases} x_j - y_{k-1} + y_k - x_j = 1 & \text{for } x_{j-1} \leq y_{k-1}, x_j \in [y_{k-1}, y_k] \\ 1 + 0 = 1 & \text{for } x_{j-1} \leq y_{k-1}, y_k \leq x_j \\ y_k - x_{j-1} & \text{for } [x_{j-1}, x_j] \not\subseteq [y_{k-1}, y_k] \end{cases} \tag{5.5}$$

From (5.2), it immediately follows that for the first two condition sets, $D_{kl} = 0 \; \forall l \neq j$, $j+1, l \in [95]$, and hence the claim holds. For the third condition set we observe the k^{th} pixel reads only one bar of $\{j, j+1\}$; however we see in this case $D_{k,j-1} = x_{j-1} - y_{k-1}$. Thus

$$D_{k,j-1} + D_{kj} = x_{j-1} - y_{k-1} + y_k - x_{j-1} = 1,$$

and hence the same logic holds. Therefore, for $h > 1$, each pixel sees at most two bars. □

Now we want to show that $T = DA$ is block-diagonal, $A \in \{0,1\}^{95 \times 123}$ and $D \in \mathbb{R}^{N \times 95}$. So by matrix multiplication, the ijth element of M is

$$T_{ij} = \sum_{k=1}^{95} D_{ik} A_{kj}.$$

For example, $T_{i,1} = \sum_{k=1}^{95} D_{ik} A_{k1}$. Since the only nonzero entries in the first column of A are in rows 1 and 3, we have $T_{i,1} = D_{i,1} + D_{i,3}$.

Theorem 5.1.2. *For small pixels, i.e. $h > 1$, $T = DA$ is block-diagonal, where A is the sparse UPC dictionary matrix and D is the camera matrix.*

Proof. Since A is completely block-diagonal by construction, T inherits the column structure from A and as such, T has no horizontal overlap of blocks. We are concerned with the transition points of the matrix blocks. Define j' to be the columns of transition for A, and hence T, meaning the furthest right column of any one of the blocks of the left half of A. So $j' \in \{1, 11, 21, 31, 41, 51, 61, 62\}$. Note $j' = 62$ corresponds to the transition to the right half of A.

Define i' to be the row of T containing the last nonzero entry of column j'. Therefore by definition, $T_{i',j'} \neq 0$, $T_{i>i',j'} = 0$ and $T_{i>i',j<j'} = 0$. We want to show

$$T_{i',j>j'} = 0. \tag{5.6}$$

To do so, we first need to make an observation about the structure of the integer blocks from matrix A. Call the left integer block L and right integer block R. We observe that for any left integer block, $L(i, 1) = 0\ \forall i$, and correspondingly, $R(i, 10) = 0\ \forall i$. (See Table 1.1.)

Let us begin by proving (5.6) for $j' = 1$. Since the last nonzero entries of column $j' = 1$ correspond to the last digit of the start sequence which lies in row 3 of A, $D_{i',k<3} = 0$. Also, from Lemma 5.1.1, for $D_{i,k} \neq 0$, at most one of $\{D_{i',k-1}, D_{i',k+1}\} \neq 0$. Thus for $k = 3$, we have $D_{i',3} \neq 0$ and $D_{i',4} \neq 0$. $D_{i',k>k+1} = 0$, so

$$D_{i',k} = [0, 0, D_{i',3}, D_{i',4}, 0, 0, \ldots, 0] \in \mathbb{R}^{1\times 95}.$$

Hence $D_{i',k}$ only picks up $A_{3,j}$ and $A_{4,j}$. We know that $A_{3,j} = [1, 0, \ldots, 0]$ and $A_{4,j} = 0$ for all j. Therefore (5.6) holds since the only nonzero term in $\sum_{k=1}^{95} D_{i',k}A_{k,j}$ is $T_{i',1}$.

Now let us show the separation between integer blocks. For the first two blocks we have the transition column $j' = 11$ and hence i' is row of the last non-zero entry in column $j' = 11$. Thus $T_{i',j'} \neq 0$, $T_{i>i',j<j'} = 0$ and $T_{i>i',j'} = 0$. We want to show (5.6) holds for $j' = 11$. We observe that $T_{i',j'}$ corresponds to the last entry of the first integer block which lives in row 10 of A. Thus $D_{i',k<10} = 0$. So by our claim at most $D_{i',k=10} \neq 0$ and $D_{i',k=11} \neq 0$. Now from the structure of A and L, we know $A_{11,j} = 0$ for all j and

$$A_{10,j} = [0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, \ldots, 0]$$

and so it follows that (5.6) holds for $j' = 11$. Therefore, (5.6) holds for all left integer blocks.

We now consider the transition from the last left integer block to the middle sequence. Let $j' = 61$. Thus i' is the row of the last non-zero entry in column $j' = 61$, $T_{i',j'} \neq 0$, and $T_{i>i',j'} = 0$. Observe that $T_{i',j'}$ corresponds to the last entry of the last left integer block which lives in row 45 of A. Thus $D_{i',k<45} = 0$. So by the claim, at most $D_{i',k=45} \neq 0$ and $D_{i',k=46} \neq 0$. Now from the structure of A, we know $A_{45,j} \neq 0$ for $j \in [52, 61]$ and $A_{46,j} = 0\ \forall j$. Both of these row vectors have zero entries for $j > 61$, hence (5.6) is satisfied. Note $j' = 62$ corresponds to the transition from the middle sequence to the right half of A and can be calculated in the same way despite $R(1, j) = 1$ $\forall j$, since the last digit of M is 0.

The proof for the right half of A mirrors the one for the left half since the codes for the right integers are the opposite (exchange "0"s and "1"s) of the left integer codes. Thus the first row of any right integer block is ten ones and the last row consists of ten zeros. Hence it follows that T is block-diagonal.[1] □

Since we now know that matrix T is block-diagonal, we can recover the correct z as in (2.3) for our image by examining each $z^{(i)}$ block of T separately. Since $g = \alpha T z$, and

$$z(1) = z(62) = z(123) = 1$$

are always constant we need only look individually at each $z^{(i)}$, the portion of z corresponding to one of the 12 integers. From the construction of z we know that each of these $z^{(i)}$s corresponds to one column of the 10×10 identity matrix I_{10}.

We want to partition T in such a way that

$$g|_{p^{(i)}} = T|_{p^{(i)}} z^{(i)},$$

where $p^{(i)}$ is the block of T corresponding to integer i. This is possible because of the block-diagonal structure of T. We need to explicitly determine the indices of the rows at which we partition T. Recall our definitions:

$$k_- = \lfloor x_0 \rfloor$$

[1] The block-diagonality of T comes from the structure of the dictionary A. Not only is A block-diagonal, but one end of each block transition consists of a row of zeros.

and

$$k_+ = \lfloor x_{95} \rfloor + 1.$$

Thus y_{k_-+1} is the first pixel containing information from the bar x_1. Correspondingly, x_3 marks the end of the start sequence code, and so we define $k_0 = \lfloor x_3 \rfloor + 1$such that pixel y_{k_0} is the last pixel to carry information regarding the start sequence. In a similar matter, we continue defining k_i:

$$k_1 = \lfloor x_{10} \rfloor + 1, \quad k_2 = \lfloor x_{17} \rfloor + 1, \quad \dots \quad k_M = \lfloor x_{50} \rfloor + 1, \quad k_7 = \lfloor x_{57} \rfloor + 1,$$

and so on, ending at k_+. Now we define $p^{(i)} = \{\dot{k} : k_{i-1} < \dot{k} < k_i\}$.[2] So for example, $p^{(1)} = \{\dot{k} : k_0 < \dot{k} < k_1\}$.

Now with this partition, we examine the following algorithm:

$$\arg\min_k \left\| g|_{p^{(i)}} - \alpha T|_{p^{(i)}} z_k^{(i)} \right\| \tag{5.7}$$

for $i \in [12]$, $k \in [10]$, where $z_k^{(i)}$ is one column of I_{10}. This algorithm iteratively determines one barcode digit at a time. There are 12 calculations of k_{min} and each calculation of k_{min} requires 10 calculations of the l_1 norm of a vector of length 10. Thus the runtime complexity is $O(10)$ [2].

Theorem 5.1.3. *If g is noiseless, and x_0, h, α are given with $h > 1$, then the algorithm (5.7) recovers the correct z.*

Proof. Let $\bar{k}$ be the correct column of I_{10}, corresponding the correct ith integer. Observe

$$\left\| g|_{p^{(i)}} - \alpha T|_{p^{(i)}} z_{\bar{k}}^{(i)} \right\| = 0.$$

Assume to the contrary that for iteration i the arg min of the algorithm returns a $\hat{k} \neq \bar{k}$. So

$$arg\min_k \left\| g|_{p^{(i)}} - \alpha T|_{p^{(i)}} z_k^{(i)} \right\| = \hat{k} \neq \bar{k}.$$

So

$$\left\| g|_{p^{(i)}} - \alpha T|_{p^{(i)}} z_{\hat{k}}^{(i)} \right\| \leq \left\| g|_{p^{(i)}} - \alpha T|_{p^{(i)}} z_{\bar{k}}^{(i)} \right\|.$$

[2] Note the slight abuse of notation around the middle sequence: $p^{(M)} = \{\dot{k} : k_6 < \dot{k} < k_M\}$ and $p^{(7)} = \{\dot{k} : k_M < \dot{k} < k_7\}$.

Now, observe

$$\begin{aligned} \left\| g|_{p^{(i)}} - \alpha T|_{p^{(i)}} z_{\hat{k}}^{(i)} \right\| &= \left\| \alpha T z|_{p^{(i)}} - \alpha T|_{p^{(i)}} z_{\hat{k}}^{(i)} \right\| \\ \text{since } \alpha T z|_{p^{(i)}} = \alpha T|_{p^{(i)}} z_{\bar{k}}^{(i)}, \text{ we have} \quad &= \left\| \alpha T|_{p^{(i)}} z_{\bar{k}}^{(i)} - \alpha T|_{p^{(i)}} z_{\hat{k}}^{(i)} \right\| \\ &= \left\| \alpha T|_{p^{(i)}} (z_{\bar{k}}^{(i)} - z_{\hat{k}}^{(i)}) \right\| \end{aligned}$$

We have $z_{\bar{k}}^{(i)} - z_{\hat{k}}^{(i)} \neq 0$ by assumption and $\alpha T|_{p^{(i)}}$ is nonzero, so since $\|f\| \geq 0$ and $\|f\| = 0$ iff $f = 0$, thus

$$\left\| g|_{p^{(i)}} - \alpha T|_{p^{(i)}} z_{\hat{k}}^{(i)} \right\| > 0 = \left\| g|_{p^{(i)}} - \alpha T|_{p^{(i)}} z_{\bar{k}}^{(i)} \right\|$$

and we arrive at our contradiction. Therefore the algorithm retrieves the correct z. □

Remark 5.1.4. *We expect the algorithm to be stable to small noise and small errors in α and x_0. Further examination into this stability will be a future endeavor.*

5.2 Unique determination in the case of poorly resolved images

Now that we have shown successful decoding for "small" pixels, we consider the more interesting case where we have "large" pixels, meaning that the pixel width is greater than the narrowest bar width ($h < 1$). Figure (5.2) shows this low resolution situation.

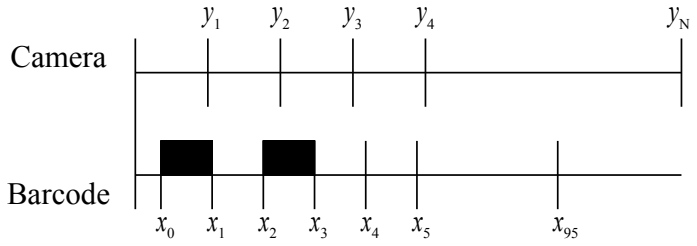


Figure 5.2: A low resolution case in which the pixels are larger than the narrowest bar width.

Although we no longer have block diagonality of the forward operator T in this case, in this section we will show given certain constraints, the product of the camera matrix and code element uniquely determines the element, and thus the decoding algorithm will properly uncover the correct code word.

The analysis in this section will be restricted to the "worst" cases where the segment of the barcode corresponding to a digit is seen only by a minimal number of pixels. We conjecture that beyond the two extreme cases we study, unique determination holds because in those regions of the parameter space, more information is available since the barcode segment in question appears in many more pixels.

The first set of constraints we consider is a low resolution scenario where four pixels cover an entire code element, including the shift, x_0. In this scenario, the last bar of one element and the first bar of the next element will be seen in the same pixel, but no three bars will be seen in the same pixel. For preciseness, let us consider $\frac{1}{2} < h < 1$. Since we assume that x_0 and h are known, after a shift of the origin and with a slight abuse of notation, we let x_0 be the left edge of the element in question. Since the element is from the left half of the barcode, the last bar to the left of x_0 is necessarily black.

Lemma 5.2.1. *For horizontal shift $x_0 \in (0,1)$ and narrowest bar width $h \in (1/2,1)$ s.t. $4 > x_0 + 7h$, the camera matrix block D is a 4×7 matrix. It acts on u, a length seven binary vector corresponding to an element in the UPC barcode. More specifically, each pixel detects parts of a two- or three-bar sequence.*

Proof. Given our assumptions for parameters x_0 and h, let $Dv(i)$ denote the i^{th} row of the product Dv. We will show that Du has the form:

$$Du = \begin{bmatrix} x_0 + [1 - (x_0 + h)]u_2 \\ (x_0 + 2h - 1)u_2 + hu_3 + [2 - (x_0 + 3h)]u_4 \\ (x_0 + 4h - 2)u_4 + hu_5 + [3 - (x_0 + 5h)]u_6 \\ (x_0 + 6h - 3)u_6 + h \end{bmatrix}, \tag{5.8}$$

corresponding to the alignment in Figure 5.3, where $[u_1, u_2, \ldots u_7]^T$ represents a UPC digit. It is worth observing that $[Du]_1$ includes the first element $u_1 = 0$. $[Du]_4$ includes the knowledge that for any u, $u_7 = 1$ and u_8 (i.e.) the first element of the adjacent word equals zero. Given our initial set of relations between x_0 and h we know that the region of interest is $x_0 \in [0,1), h \in (1/2,1)$ and $x_0 < 4 - 7h$. For all (x_0, h) in this region, we maintain the same structure as given in (5.8), provided the coefficients multiplying u_i

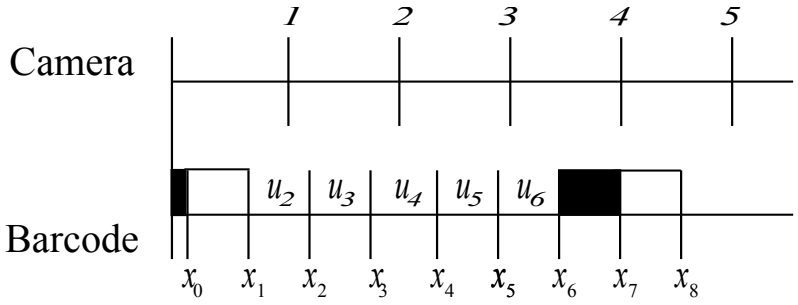


Figure 5.3: One element of a UPC barcode as in Lemma 5.2.1

are greater than zero, i.e.

$$\begin{aligned} 1-(x_0+h) &> 0 \\ x_0+2h-1 &> 0 \\ 2-(x_0+3h) &> 0 \\ x_0+4h-2 &> 0 \\ 3-(x_0+5h) &> 0 \\ x_0+6h-3 &> 0 \end{aligned}$$

Clearly, the second, fourth, and final inequalities are satisfied by $1/2 < h < 1$ and x_0 positive. To show our target region fits within the remaining inequalities, we refer to Figure 5.4 where we see our target region represented in the darkest shade. Therefore, having satisfied all of our constraints, we may conclude that Du has the same structure as in (5.8). □

We will use the structure of Du given in the preceding lemma to prove the following theorem.

Theorem 5.2.2. *Let u, v be two elements in the left half of a UPC barcode. For a camera matrix D with parameters $x_0 < 1$ and $1/2 < h < 1$, such that $4 > x_0 + 7h$, then $Du = Dv \implies u = v$.*

Proof. The codewords u, v are 7-vectors where the first and last elements are 0 and 1

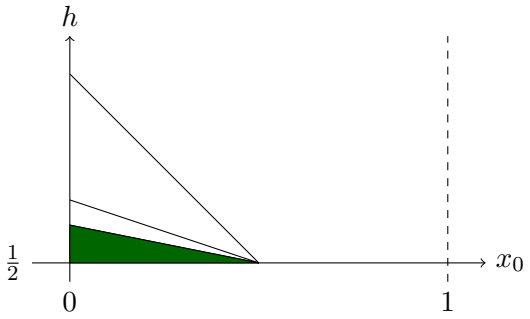


Figure 5.4: The region in the parameter space in which uniqueness is guaranteed for Lemma 5.2.1

respectively by the structure of UPC. Thus

$$u = \begin{bmatrix} 0 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ 1 \end{bmatrix}, \quad \text{and } v = \begin{bmatrix} 0 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ 1 \end{bmatrix}, \tag{5.9}$$

where u_i and $v_i \in \{0, 1\}$.

We obtain the structure of Du from Lemma 5.2.1, which holds for any u from the left-half of UPC, so in particular, it holds for u. Thus given $Du = Dv$ we have:

$$x_0 + [1 - (x_0 + h)]u_2 = x_0 + [1 - (x_0 + h)]v_2 \tag{5.10}$$

$$(x_0 + 2h - 1)u_2 + hu_3 + [2 - (x_0 + 3h)]u_4 = (x_0 + 2h - 1)v_2 + hv_3 + [2 - (x_0 + 3h)]v_4 \tag{5.11}$$

$$(x_0 + 4h - 2)u_4 + hu_5 + [3 - (x_0 + 5h)]u_6 = (x_0 + 4h - 2)v_4 + hv_5 + [3 - (x_0 + 5h)]v_6 \tag{5.12}$$

$$(x_0 + 6h - 3)u_6 + h = (x_0 + 6h - 3)v_6 + h \tag{5.13}$$

Since we are using the same parameters (x_0, h) for each, it follows immediately from (5.10) and (5.13) that $u_2 = v_2$ and $u_6 = v_6$. Substituting this into equations (5.11) and (5.12) we have:

$$hv_3 + (2 - (x_0 + 3h))u_4 = hu_3 + (2 - (x_0 + 3h))v_4,$$
$$(x_0 + 4h - 2)u_4 + hu_5 = (x_0 + 4h - 2)v_4 + hv_5,$$

which is equivalent to

$$\begin{bmatrix} h & 2 - (x_0 + 3h) & 0 \\ 0 & x_0 + 4h - 2 & h \end{bmatrix} \begin{bmatrix} u_3 - v_3 \\ u_4 - v_4 \\ u_5 - v_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{5.14}$$

We want to show the only solution to (5.14) is $u_i = v_i \ \forall i$. Since $u_i, v_i \in \{0, 1\} \forall i$, we know for $u_i \neq v_i$, $u_i - v_i = \pm 1$. Setting

$$\Gamma = \begin{bmatrix} u_3 - v_3 \\ u_4 - v_4 \\ u_5 - v_5, \end{bmatrix}$$

we thus examine all possibilities for Γ, desiring to show none satisfy (5.14), save $\Gamma = 0$.

Case I: Let us first consider the possibility that $v_i \neq u_i$ for only one $i \in \{3, 4, 5\}$. i.e.

$$\Gamma = \begin{bmatrix} \pm 1 \\ 0 \\ 0 \end{bmatrix}, \Gamma = \begin{bmatrix} 0 \\ \pm 1 \\ 0 \end{bmatrix}, \text{ or } \Gamma = \begin{bmatrix} 0 \\ 0 \\ \pm 1 \end{bmatrix}.$$

For the first or the last ($i = 3$ or $i = 5$), the contradiction is obvious. For $i = 4$, we have

$$\begin{bmatrix} \pm(2 - (x_0 + 3h)) \\ \pm(x_0 + 4h - 2) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix};$$

however, in the proof of the lemma we showed that $2 - (x_0 + 3h) > 0$ and $x_0 + 4h - 2 > 0$, so equality can never be achieved.

Case II: Let $u_i = v_i$ for only one $i \in \{3, 4, 5\}$. We can first eliminate the case for $i = 4$, since from above we know $\Gamma = [\pm 1, 0, \pm 1]^T$ will not satisfy (5.8). Our remaining two cases result in the following:

$$\begin{bmatrix} \pm 1 \pm (2 - (x_0 + 3h)) \\ \pm(x_0 + 4h - 2) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{5.15}$$

$$\begin{bmatrix} \pm(2-(x_0+3h)) \\ \pm(x_0+4h-2)\pm 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{5.16}$$

Using the contradictions from Case I, it remains to show $\pm 1 \neq \pm(2-(x_0+3h))$ for (5.15) and $\pm 1 \neq \pm(x_0+4h-2)$ for (5.16). Since both arguments have been shown to be positive, we need only show $h \neq 2-(x_0+3h)$ and $h \neq x_0+4h-2$.

Assume to the contrary that $2-(x_0+3h) = h \equiv 2-x_0 = 4h$. Since $h \in (1/2,1)$ we know $2-x_0 = 4h \in (2,4)$, but $2-x_0 \in (1,2)$. So we have a contradiction.

Assume again to the contrary that $x_0+4h-2 = h \equiv x_0+3h = 2$, but in the Lemma, we have shown $2 > x_0+3h$. Thus another contradiction.

Case III: Worst case scenario: Consider $u_i \neq v_i \; \forall i \in \{3,4,5\}$. Thus $\Gamma = [\pm 1, \pm 1, \pm 1]^T$, which gives a combination of (5.15) and (5.16), namely:

$$\begin{bmatrix} \pm h \pm (2-(x_0+3h)) \\ \pm(x_0+4h-2)\pm h \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Thus it follows from Case II that $\Gamma = [\pm 1, \pm 1, \pm 1]^T$ does not satisfy (5.14). Having eliminated all the alternative configurations of Γ, we conclude $u_i = v_i \; \forall i$. □

We now consider a new set of constraints on the parameters x_0 and h. In this second low resolution scenario, five pixels entirely cover a code element as well as the first two bars of the adjacent element, including the shift, x_0. As such, the fifth pixel sees bars u_7, u_8, and u_9. This case is more challenging since the fifth pixel may include non-zero information from the next element as well.

Lemma 5.2.3. *For narrowest bar width $h \in \left(\frac{1}{2}, 1\right)$ s.t. $5 > x_0 + 8h$ and $x_0 + h > 1$, the camera matrix block D is a 5×7 matrix. It acts on u, a length seven binary vector corresponding to an element in the UPC barcode.*

Proof. Given our assumptions for parameters x_0 and h, let $[Du]_i$ denote the i^{th} row of the product Du. We want to show that Du has the form:

$$Du = \begin{bmatrix} x_0 \\ hu_2 + (2-(x_0+2h))u_3 \\ (x_0+3h-2)u_3 + hu_4 + [3-(x_0+4h)]u_5 \\ (x_0+5h-3)u_5 + hu_6 + (4-(x_0+6h)) \\ (x_0+7h-4) + (5-(x_0+8h))u_9 \end{bmatrix}, \tag{5.17}$$

corresponding to the alignment in Figure 5.5, where $[u_1, \ldots, u_7]^T$ represents a UPC digit. It is worth observing that $[Du]_1$ and $[Du]_2$ come from the fact that for any u,

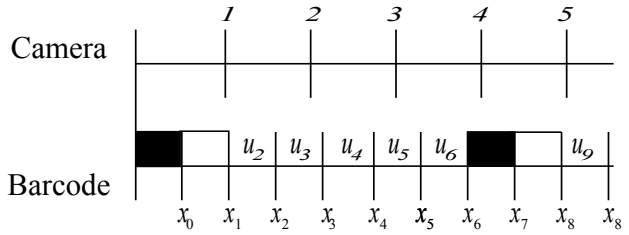


Figure 5.5: One element of a UPC barcode as in Lemma 5.2.3

$u_1 = 0$. Similarly $[Du]_4$ comes from $u_7 = 1$ and $[Du]_5$ from $u_7 = 1$ and $u_8[$ (i.e. the first bar of the adjacent digit) equals zero. We draw attention to the presence of u_9 in $[Du]_5$. u_9 is the second bar of the next digit in the barcode. It is included here because the pixel is now also picking up part of the next digit. Given our initial set of relations between x_0 and h we know that the max of h is 1, and the region of interest is further bounded by $x_0 + h > 1$ and $x_0 < 5 - 8h$. For all (x_0, h) in this region, we maintain the same structure as given in (5.17), provided the coefficients of v_i are greater than zero, i.e.

$$\begin{aligned} 2 - (x_0 + 2h) &> 0 \\ x_0 + 3h - 2 &> 0 \\ 3 - (x_0 + 4h) &> 0 \\ x_0 + 5h - 3 &> 0 \end{aligned}$$

These inequalities hold under the stated conditions. To further show our target region fits within these inequalities, we refer to Figure 5.6 where we see our target region shaded. Therefore, having satisfied all of our constraints, we may conclude that Du has the same structure as in (5.17). □

We will use the structure of Du given in Lemma 5.2.3 to prove the following theorem.

Theorem 5.2.4. *Let u, v be two elements in the left half of a UPC barcode. For a camera matrix D with parameters x_0, h such that $x_0 + h > 1$, $h \in \left(\frac{1}{2}, 1\right)$, and $5 > x_0 + 8h$, then $Du = Dv \implies u = v$.*

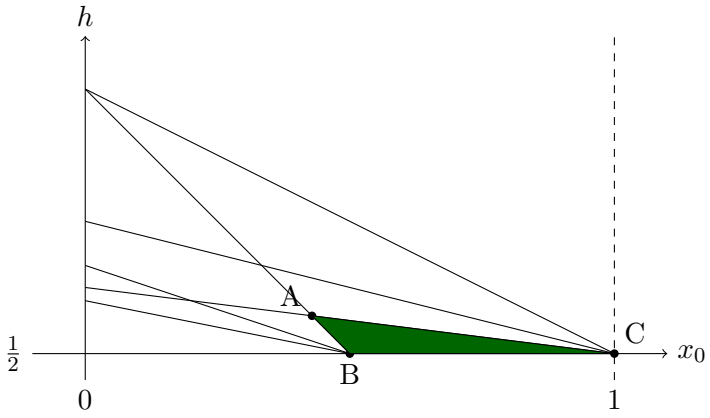


Figure 5.6: The region in parameter space in which uniqueness is guaranteed for Lemma 5.2.3; A= $(\frac{3}{7}, \frac{4}{7})$, B= $(\frac{1}{2}, \frac{1}{2})$, and C= $(1, \frac{1}{2})$.

Proof. As before, assume $Du = Dv$. The code elements u, v are 7-vectors where the first and last elements are 0 and 1 respectively by the structure of UPC, as in (5.9). Since Lemma 5.2.1 holds for any u from the left-half of UPC, in particular, it holds for v. Thus given $Du = Dv$ we have

$$x_0 = x_0 \tag{5.18}$$

$$hu_2 + (2 - (x_0 + 2h))u_3 = hv_2 + (2 - (x_0 + 2h))v_3 \tag{5.19}$$

$$(x_0 + 3h - 2)u_3 + hu_4 + (3 - (x_0 + 4h))u_5 = (x_0 + 3h - 2)v_3 + hv_4 + (3 - (x_0 + 4h))v_5 \tag{5.20}$$

$$(x_0 + 5h - 3)u_5 + hu_6 + (4 - (x_0 + 6h)) = (x_0 + 5h - 3)v_5 + hv_6 + (4 - (x_0 + 6h)) \tag{5.21}$$

$$(x_0 + 7h - 4) + (5 - (x_0 + 8h))u_9 = (x_0 + 7h - 4) + (5 - (x_0 + 8h))v_9 \tag{5.22}$$

Since Du and Dv use the same parameters (x_0, h), (5.18) is trivially satisfied and (5.22) implies $u_9 = v_9$. Subtracting $4 - (x_0 + 6h)$ from both sides of (5.21), we can rewrite the

relevant equations in matrix form:

$$\begin{bmatrix} h & 2-(x_0+2h) & 0 & 0 & 0 \\ 0 & x_0+3h-2 & h & 3-(x_0+4h) & 0 \\ 0 & 0 & 0 & x_0+5h-3 & h \end{bmatrix} \begin{bmatrix} u_2-v_2 \\ u_3-v_3 \\ u_4-v_4 \\ u_5-v_5 \\ u_6-v_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.23}$$

We want to show the only solution to (5.23) is $u_i = v_i \ \forall i$. This proof is more involved than the proof in Theorem 5.2.2 and requires some knowledge about the structure of our code elements. We know for UPC barcodes, two distinct code elements from the same side of the barcode (left or right) have a minimum Hamming distance of 2. In other words, $u_i - v_i \neq 0$ for two i's in $1, \ldots, 7$. We already know that structure mandates $u_i = v_i$ for $i = 1, 7$. Thus for $u \neq v$, $u_i - v_i \neq 0$ for *at least* two and *at most* four $i \in \{2, 3, 4, 5, 6\}$.

So, assuming $u \neq v$ we have $\min |u - v| = 2$ and $\max |u - v| = 4$. Thus we have three cases.

Case I:

$$|u - v| = 4 \quad \Longrightarrow \quad \exists j \in \{2, 3, 4, 5, 6\} \text{ s.t. } u_j = v_j \ , \ u_i \neq v_i \ \forall i \neq j$$

Case II:

$$|u - v| = 3 \quad \Longrightarrow \quad \exists j, k \in \{2, 3, 4, 5, 6\} \text{ s.t. } u_j = v_j, u_k = v_k, j \neq k \ , \ u_i \neq v_i \ \forall i \neq j, k$$

Case III:

$$|u - v| = 2 \quad \Longrightarrow \quad \exists j, k, l \in \{2, 3, 4, 5, 6\} \text{ s.t. } u_j = v_j, u_k = v_k, u_l = v_l, j \neq k \neq l \ , \ u_i \neq v_i \ \forall i \neq j, k, l$$

which can equivalently be written

$$|u - v| = 2 \quad \Longrightarrow \quad \exists j, k \in \{2, 3, 4, 5, 6\} \text{ s.t. } u_j \neq v_j, u_k \neq v_k, j \neq k \ , \ u_i = v_i \ \forall i \neq j, k$$

Let $w_i = u_i - v_i$. For Case I, $w_j = 0$ for a particular j. Letting $\Gamma = u - v$, then Γ is

one of the following:

$$\Gamma = \begin{bmatrix} 0 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix}, \begin{bmatrix} w_2 \\ 0 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix}, \begin{bmatrix} w_2 \\ w_3 \\ 0 \\ w_5 \\ w_6 \end{bmatrix}, \begin{bmatrix} w_2 \\ w_3 \\ w_4 \\ 0 \\ w_6 \end{bmatrix}, \text{ or } \begin{bmatrix} w_2 \\ w_3 \\ w_4 \\ w_5 \\ 0 \end{bmatrix}$$

where, for any nonzero , $w_i = \pm 1$.

As we saw in the proof for Theorem 5.2.2, using the resulting target region of Lemma 5.2.3, we need to show all choices of w_j do not result in zero equalities. For example, let us begin in Case I with $w_3 = 0$. Now, let us assume to the contrary that $Dw = 0$, for $w_3 = 0$. This looks like

$$\begin{bmatrix} h & 2-(x_0+2h) & 0 & 0 & 0 \\ 0 & x_0+3h-2 & h & 3-(x_0+4h) & 0 \\ 0 & 0 & 0 & x_0+5h-3 & h \end{bmatrix} \begin{bmatrix} w_2 \\ 0 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix} = \begin{bmatrix} hw_2 \\ hw_4 + (3-(x_0+4h))w_5 \\ (x_0+5h-3)w_5 + hw_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.24}$$

So from (5.24), recalling $w_{j\neq 3} = \pm 1$, we need to show the following inequalities to achieve contradiction:

$$\pm h \neq 0$$
$$\pm h \pm (3-(x_0+4h) \neq 0$$
$$\pm(x_0+5h-3) \pm h \neq 0$$

We pause for a moment to realize that following the same process for all other choices of j in Case I and all possible choices in Cases II and III, we can prove Theorem 5.2.4 by proving the following inequalities hold:

$$\pm h \pm (2-(x_0+2h)) \neq 0$$

$$\pm(x_0+5h-3) \pm h \neq 0$$

$$\pm h \pm (3-(x_0+4h)) \neq 0$$

$$\pm(x_0 + 3h - 2) \pm h \neq 0$$

$$\pm(x_0 + 3h - 2) \pm (3 - (x_0 + 4h)) \neq 0$$

$$\pm(x_0 + 3h - 2) \pm h \pm (3 - (x_0 + 4h)) \neq 0$$

From Lemma 5.2.3, all same sign versions of the inequalities are obvious. We need only be concerned with the opposite signs chosen in the first five inequalities and various signs in the last. Let us begin assuming to the contrary that $h = 2 - (x_0 + 2h)$ which is equivalent to $x_0 + 3h = 2$. Now using the constraints $x_0 + h > 1$ and $h \in (1/2, 1)$ we see

$$\begin{aligned} 2 = x_0 + 3h &= x_0 + h + 2h \\ &> 1 + 2h \\ &> 1 + 2 \end{aligned}$$

which leads to a contradiction. Therefore, $\pm h \pm (2 - (x_0 + 2h)) \neq 0$.

Using the same constraints we can prove $\pm h \pm (3 - (x_0 + 4h)) \neq 0$. Assume to the contrary, $h = 3 - (x_0 + 4h)$, or equivalently $x_0 + 5h = 3$.

$$\begin{aligned} 3 = x_0 + 5h &= x_0 + h + 4h \\ &> 1 + 4h \\ &> 1 + 4 \end{aligned}$$

which leads to a contradiction. Therefore, $\pm(x_0 + 5h - 3) \pm h \neq 0$. Using the same argument we can prove $\pm(x_0 + 3h - 2) \pm h \neq 0$.

To show $\pm(x_0 + 3h - 2) \pm (3 - (x_0 + 4h)) \neq 0$ holds for alternating signs, we assume $2x_0 + 7h = 5$. Then as before,

$$\begin{aligned} 5 = 2x_0 + 7h &= 2(x_0 + h) + 5h \\ &> 2 + 5h \end{aligned}$$

which leads to a contradiction.

Finally we want to show $\pm(x_0 + 3h - 2) \pm h \pm (3 - (x_0 + 4h)) \neq 0$. First,

$$x_0 + 3h - 2 - h + 3 - (x_0 + 4h) = 1 - 2h < 0.$$

Secondly,

$$x_0 + 3h - 2 + h - (3 - (x_0 + 4h)) = 2x_0 + 8h - 5 > 2 + 6h - 5 > 3.$$

And finally,

$$\begin{aligned} -(x_0 + 3h - 2) + h + 3 - (x_0 + 4h) &= -2x_0 - 6h + 5 \\ &> -2x_0 - 6h + x_0 + 8h \\ &= 2h - x_0 \\ &> 2 - x_0 > 0. \end{aligned}$$

By the proof of the inequalities, we have eliminated all alternative configurations of Γ, and hence we conclude $v_i = u_i \ \forall i$. Therefore we can uniquely determine the code element in the case of low resolution, i.e. "small pixels." □

Although the forward operator is no longer block-diagonal in the case of low resolution images, Theorems 5.2.2 and 5.2.4 reveal there is sufficient information in the data to uniquely determine the barcode.

5.3 Error Analysis

As the real world is full of noise and corruption, it is of extreme interest to know how much noise ε we can add to the data for the algorithm in (5.7) to still return the correct barcode. In this study we consider the reduced systems as we did in the uniqueness results in Theorems 5.2.2 and 5.2.4 for low resolution images.

For scenario I (as in Theorem 5.2.2), we are concerned with determining the third, fourth and fifth bars of a UPC element (u_3, u_4, u_5), each of which is either 0 or 1. The data consists of image intensities (g_2, g_3). We can view the map from (u_3, u_4, u_5) to (g_2, g_3) as a mapping from $[0, 1]^3$ to $\mathbb{R}^2$. Specifically, we are interested in to where the corners of the unit cube map. To determine this, we relabel the reduced system in terms of ξ_1, ξ_2, where $\xi_1 = 2 - (x_0 + 3h)$ and $\xi_2 = x_0 + 4h - 2$. Thus,

$$\begin{bmatrix} g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} h & \xi_1 & 0 \\ 0 & \xi_2 & h \end{bmatrix} \begin{bmatrix} u_3 \\ u_4 \\ u_5 \end{bmatrix}. \tag{5.25}$$

Table 5.1: Map from UPC dictionary to pixel intensity

(u_3, u_4, u_5)	(g_2, g_3)	Corresponding digit(s)
$(0,0,0)$	$(0,0)$	4
$(1,0,0)$	$(h,0)$	2,5
$(0,1,0)$	(ξ_1,ξ_2)	9
$(0,0,1)$	$(0,h)$	N/A
$(1,1,0)$	(ξ_1+h,ξ_2)	1,7
$(0,1,1)$	$(\xi_1,h+\xi_2)$	0,6
$(1,0,1)$	(h,h)	8
$(1,1,1)$	$(\xi_1+h,h+\xi_2)$	3

Furthermore, as all columns of $D(x_0,h)$ must sum to h, we utilize the substitution $\xi_2 = h-\xi_1$, such that for fixed h, we can represent the system with only one parameter (ξ_1).

The digits for a given (x_0,h) are mapped to points on $\mathbb{R}^2$. When the data is noisy, the points $(\tilde{g}_1,\tilde{g}_2)$ do not necessarily coincide with one of the eight candidate points shown in Table 5.1. Our approach is to find a candidate point closest to $(\tilde{g}_1,\tilde{g}_2)$. As such we are concerned with the minimum distance between any pair of the eight candidate points.

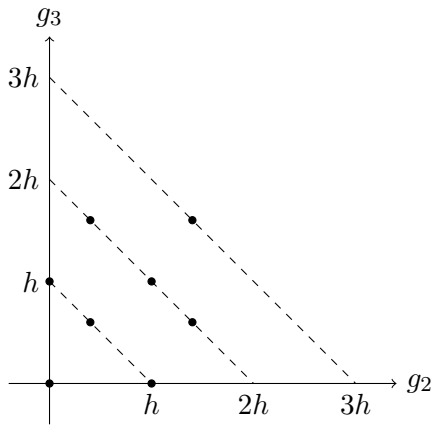


Figure 5.7: Range of mapping from (u_3,u_4,u_5) to (g_2,g_3).

For h fixed and $\xi_1 \leq h$, the four candidate points depending on ξ_1 move along parallel lines, which are separated by a distance of $h\sqrt{2}$. Thus, the smallest distance

between any two points for $h \in (\frac{1}{2}, 1)$ is

$$d_{min} = \min\{\xi_1\sqrt{2}, (h-\xi_1)\sqrt{2}\}.$$

These are distances from the fixed points $(0, h)$, $(h, 0)$ to (ξ_1, ξ_2) or the fixed point (h, h) to $(\xi_1 + h, \xi_2)$, $(\xi_1, h + \xi_2)$. Thus for noise level such that

$$\|\tilde{g} - g\|_2 < \|\min\{\xi_1\sqrt{2}, (h-\xi_1)\sqrt{2}\}\|_2,$$

the code element is uniquely determined in the scenario with no overlap.

Theorem 5.3.1. *Assuming a one-dimensional UPC barcode with narrowest bar width $h \in (0.5, 1)$ and $x_0 \in (0, 1)$ such that the assumptions of Theorem 5.2.2 are satisfied and $\xi_1 = 1 - (x_0 + h)$, let g be the image data and $\tilde{g}$ be the noisy data. If*

$$\|\tilde{g} - g\|_2 < \|\min\{\xi_1\sqrt{2}, (h-\xi_1)\sqrt{2}\}\|_2,$$

then the code is exact.

In scenario II when we do have overlap (as in Theorem 5.2.4), our map of the simplified systems maps from $[0, 1]^5$ to $\mathbb{R}^3$, takes bars (u_2, u_3, u_4, u_5) to intensities (g_2, g_3, g_4). The images of the digits appear as ten points lying on a line and two parallel planes. The line is given by $(\xi_1, 1, 3h - 1 - \xi_1)$, which contains the point corresponding to the digit 1. The second plane $g_1 + g_2 + g_3 = 2h$ contains points corresponding to the digits $0, 2, 4, 5$ and 9. Of particular importance in this plane is the fact that for h known, digits 4 and 9 are fixed points. The third plane $g_1 + g_2 + g_3 = 4h$ contains the corresponding images of the digits $3, 6, 7$ and 8. On this plane there are no fixed points; however, we observe with fixed h the trajectories of code elements 3 and 8 are parallel as we vary ξ_1. Increasing the dimension from the previous scenario produces parallel planes rather than lines; the two planes are a distance of $\frac{2h}{\sqrt{3}}$ apart. Now we examine the distance between any two points on the same plane. Since our image is in $\mathbb{R}^3$ with two fixed points, we can easily picture the image of the map in Figure 5.8.

We use the same approach as in scenario I, this time introducing two more parameters to represent our more complex system as shown below.

$$\begin{bmatrix} g_2 \\ g_3 \\ g_4 \end{bmatrix} = \begin{bmatrix} h & \xi_1 & 0 & 0 & 0 \\ 0 & \xi_2 & h & \eta_1 & 0 \\ 0 & 0 & 0 & \eta_2 & h \end{bmatrix} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{bmatrix} \tag{5.26}$$

Table 5.2: Map from UPC dictionary to pixel intensity

	0	1	2	3	4	5	6	7	8	9
g_1	0	ξ_1	ξ_1	$h+\xi_1$	h	$h+\xi_1$	h	$h+\xi_1$	$h+\xi_1$	0
g_2	$h+\eta_1$	1	ξ_2	1	0	ξ_2	$h+\eta_1$	ξ_2+h	$1-h$	h
g_3	η_2	η_2	h	η_2	h	0	η_2+h	h	η_2+h	h

Once again, we can represent everything in terms of h and ξ_1

$$\xi_2 = h - \xi_1$$

$$\eta_1 = 1 - 2h + \xi_1$$

$$\eta_2 = 3h - 1 - \xi_1$$

Now we are mapping from $5D$ to $3D$. The image is shown in Figure 5.8

We can calculate the distance d_{ij} between the image of any two digits and in doing so, we realize that for each fixed h, the minimum distance is

$$d_{\min} = \min\{\xi_1\sqrt{2}, |(1-2h+\xi_1)|\sqrt{2}, (h-\xi_1)\sqrt{2}\}. \tag{5.27}$$

Furthermore, for any choice of h and x_0 we can calculate the minimum distance as a function of ξ_1:

$$d(\xi_1) = \begin{cases} \xi_1\sqrt{2} & \text{for } \xi_1 \leq \frac{2h-1}{2} \\ |(1-2h+\xi_1)|\sqrt{2} & \text{for } \frac{2h-1}{2} < \xi_1 \leq \frac{3h-1}{2} \\ (h-\xi_1)\sqrt{2} & \text{for } \xi_1 > \frac{3h-1}{2} \end{cases}, \tag{5.28}$$

which can be seen along the bottom of Figure 5.9.

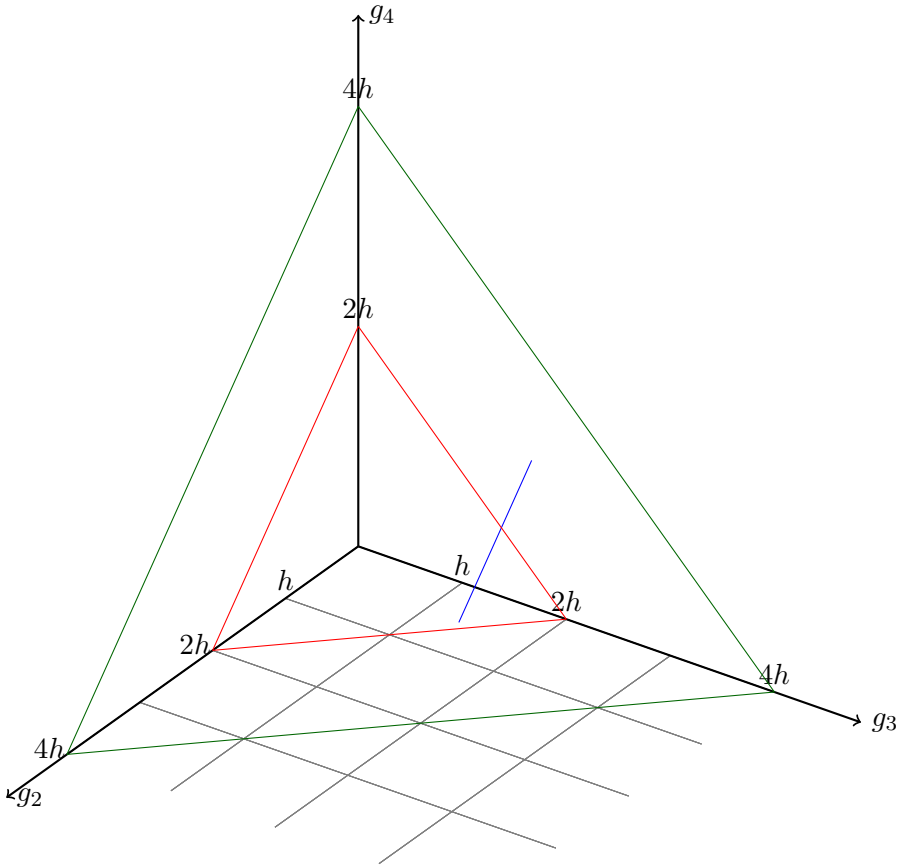


Figure 5.8: The line and two planes in $(g_2, g_3, g_4)^T$ onto which $(u_2, u_3, u_4, u_5)^T$ map

Theorem 5.3.2. *Assuming a one-dimensional UPC barcode with narrowest bar width* $h \in (0.5, 1)$ *and* $x_0 \in (0, 1)$ *such that the assumptions of Theorem 5.2.4 are satisfied and* $\xi_1 = 2 - x_0 + 2h$*, let* g *be the image data and* $\tilde{g}$ *be the noisy data. If*

$$\|\tilde{g} - g\|_2 < \|\min\{\xi_1\sqrt{2}, |(1 - 2h + \xi_1)|\sqrt{2}, (h - \xi_1)\sqrt{2}\}\|_2,$$

then the code is exact.

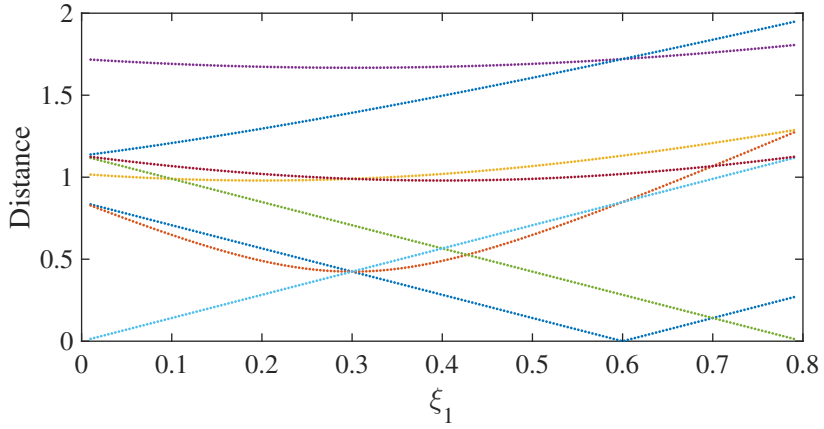


Figure 5.9: Distances between points for $h = 0.8$ fixed and $\xi_1 \in (0, h)$

Chapter 6

Numerical Simulations

6.1 Algorithm Implementation

Now that we have examined the process by which we decode and the rigorous results we can achieve using our process, we shall demonstrate via example the process from start to finish. In order to verify our results, we will begin by choosing an (x_0, h), which we will later estimate from data. For the purpose of this example, let

$$(x_0, h) = (25.3, 0.8).$$

Thus we have a shift of 25 pixels from the edge of the image to the barcode itself. Also, we know from §4.1.3, for an image of 120 pixels

$$x_0' = 18.7.$$

As this study pays no attention to the checksum requirement, we randomly generate a 12 digit sequence and add 5.0% Gaussian noise level. This particular experiment uses the 12 digit sequence

384815703554

The noisy data appears in Figure 6.1.

Before we can implement our algorithm, we must identify the edges of the barcode from the noisy data. As mentioned previously, we implement a Canny edge detection as in [11] to crop our data from all 120 pixels to pixels m_{start} to m_{stop}.

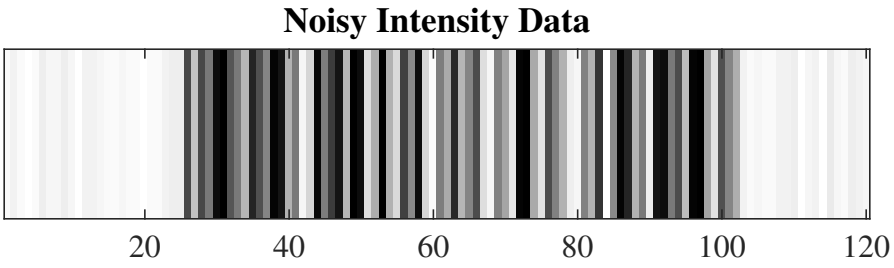


Figure 6.1: Simulated data with Gaussian noise level of 5.0%

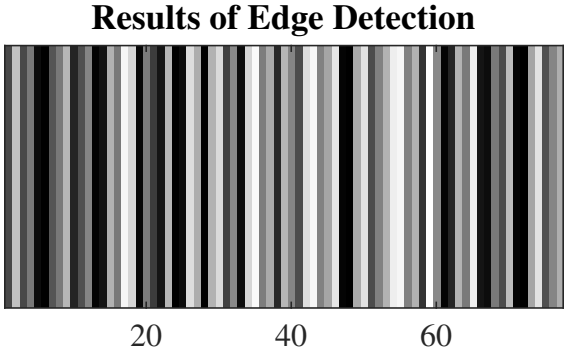


Figure 6.2: Results of Canny edge detection on our simulated noisy data

With the knowledge of the barcode edges, we now apply our method to the data, first identifying G and G':

$$G = \begin{bmatrix} 0.7292 \\ 0.2189 \\ 0.7152 \end{bmatrix} \qquad G' = \begin{bmatrix} 0.2976 \\ 0.4629 \\ 0.6871 \end{bmatrix}$$

Projecting these noisy data points onto the planes in Figure 4.7, we are then able to return estimates for the parameters (x_0, h) and (x_0', h'). With these estimates we optimize under the constraints

$$h = h'$$

and

$$m_{stop} - x_0' = x_0 + 95h.$$

The optimization returns our final estimators for the camera parameters,

$$(x_0, h, x_0') = (0.2893, 0.8001, .7031).$$

As seen in Table 6.1, the difference between the true parameters and the estimators is

Table 6.1: Accuracy of estimated parameters in numerical example

Parameter	True Value (x^*)	Estimated Value (x)	$\|(x^* - x)\|_2$
x_0	25.3000	25.2893	0.0107
h	0.8000	0.8001	7.9296E-5
x_0'	18.7000	18.7031	0.0031

small, particularly for h. These values are used in Algorithm 1 to obtain the digits in

Algorithm 1 Left to Right Decoding Algorithm

1: **procedure** LR DECODING
2: $data = g - \alpha D(x_0, h)A\zeta$
3: **for** each $j \in [12]$ **do**
4: $k_{min} = \arg\min_{k\in\{0,\ldots,9\}} \|data - \alpha D(x_0, h)Az_k^{(j)}\|$
5: $data = data - \alpha D(x_0, h)Az_{k_{\min}}^{(j)}$
6: $code(j) = k_{\min}$
7: **end for**
8: **end procedure**

the barcode, where ζ is the vector containing only information about the start, middle, and end sequences.

In our experiments we found that Algorithm 1 can be improved by decoding from both ends, estimating two camera matrices each based on (x_0, h) and (x_0', h) respectively. The motivation for this updated algorithm, Algorithm 2, comes from a need to reduce blow-up of error in h. When decoding left to right, the far right end of the barcode lies at $x_0 + 95h$. By utilizing the stop sequence notation presented in §4.1.3, we avoid this blow-up by decoding from either end only up to the middle sequence. Thus, using Algorithm 2, the 12 digits in the barcode are correctly determined. It is informative to reconstruct the barcode from the recovered 12 digits which we do in Figure 6.3.

Algorithm 2 Decoding from Both Ends

procedure BOTH ENDS DECODING
  $data_L = g - \alpha D(x_0, h)A\zeta$
  $A_{flip} = flip(A, 1)$ %flip rows of A
  $data_R = flip(g) - \alpha D(x_0', h)A_{flip}\zeta$
  for each $j \in [6]$ **do**
    $l_{min} = \arg\min_{l\in\{0,\ldots,9\}} \|data_L - \alpha D(x_0, h)Az_l^{(j)}\|$
    $data_L = data_L - \alpha D(x_0, h)Az_{l_{\min}}^{(j)}$
    $code(j) = l_{\min}$
    $r_{min} = \arg\min_{r\in\{0,\ldots,9\}} \|data_R - \alpha D(x_0', h)A_{flip}z_r^{(13-j)}\|$
    $data_R = data_R - \alpha D(x_0', h)A_{flip}z_{r_{\min}}^{(13-j)}$
    $code(13 - j) = r_{\min}$
  end for
end procedure

Figure 6.3: The reconstruction of the barcode from Algorithm 2

6.2 Performance of algorithm under noise

In this section we present the results of the following experiment: Consider the parameter space of (x_0, h) where $x_0 \in (0, 1)$ and $h \in [3/4, 1)$. For random 12-digit sequences and random Gaussian noise, how well does Algorithm 2 perform? In this experiment we will sample points in the parameter space with a 60×15 mesh and generate 100 instances of random 12 digit numbers and their corresponding noisy data. We assume that the edges have been detected correctly. The experiment is repeated for noise levels of $2.5\%, 5.0\%, 10.0\%$. The figures below illustrate the average success of this experiment

for each point in the grid at the respective noise level, as well as the accuracy of our parameter estimates, $x = (x_0, h)$, compared to the true camera parameters $x^* = (x_0^*, h^*)$.

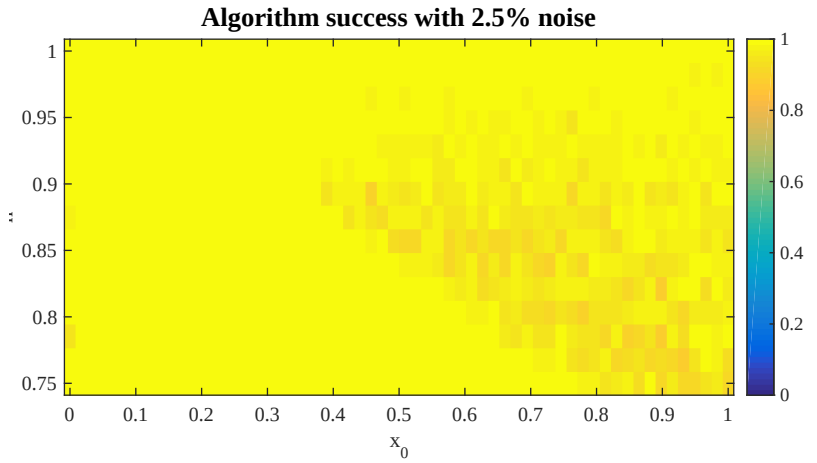


Figure 6.4: Average performance of Algorithm 2 post edge detection with 2.5% Gaussian noise level

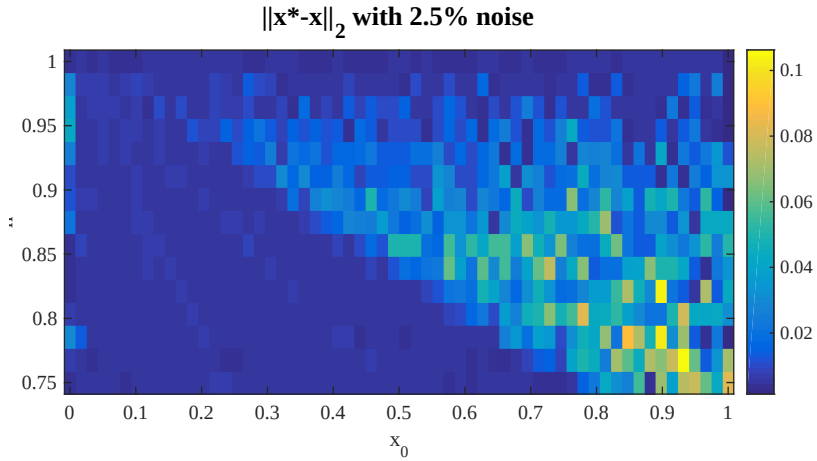


Figure 6.5: L_2 distance between estimated parameters $x = (x_0, h)$ and true parameters $x^* = (x_0^*, h^*)$ with 2.5% Gaussian noise level

What we see is that as the noise level is increased, the error in the recovered parameters increases accordingly. However, we also observed that the decoding of the barcode word is not heavily impacted by the inaccuracies in the parameters. Computing the total error rate, we find that the success rates are at 98.6%, 98.8%, and 99.3% for noise

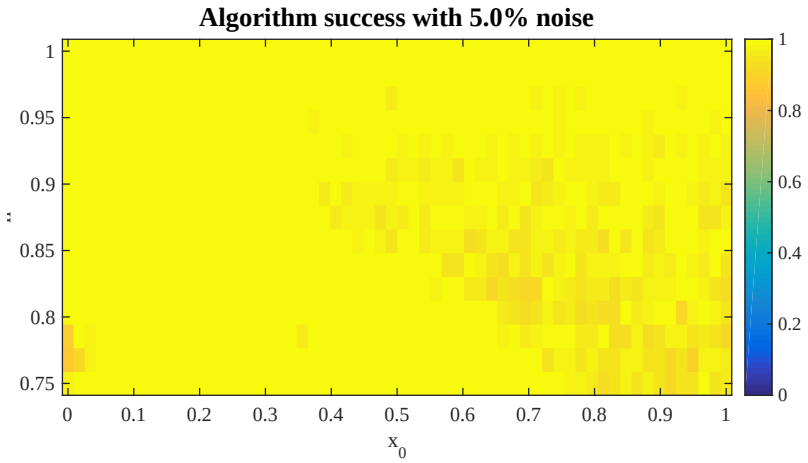


Figure 6.6: Average performance of Algorithm 2 post edge detection with 5.0% Gaussian noise level

levels of 2.5%, 5%, and 10%. Thus we are seeing near-perfect decoding of the barcodes.

To delve into this issue further, we ran two more experiments, at noise levels of 20% and 40%. We tabulate the success rates in the table below. The drop-off in the success

Noise Level	2.5%	5%	10%	20%	40%
Success Rate	98.6%	98.8%	99.3%	98.1%	66.8%

rate is now evident. Nevertheless, we note that the algorithm is extremely robust up to 20% noise, but we predict significant drop off beyond this noise level.

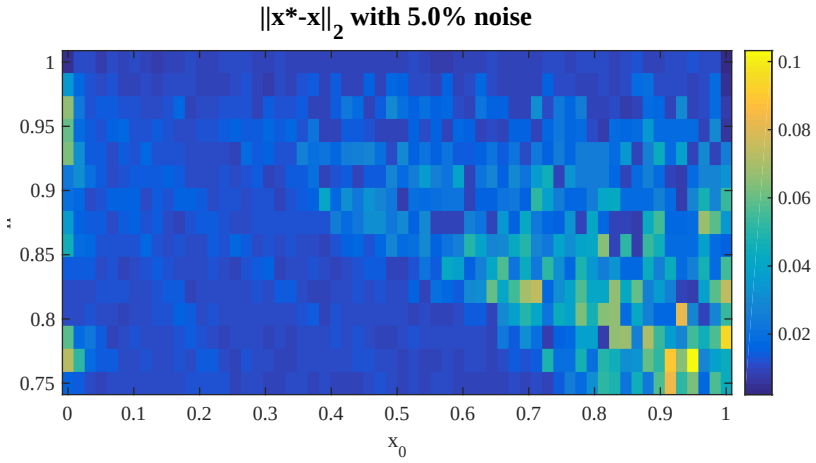


Figure 6.7: L_2 distance between estimated parameters $x = (x_0, h)$ and true parameters $x^* = (x_0^*, h^*)$ with 5.0% Gaussian noise level

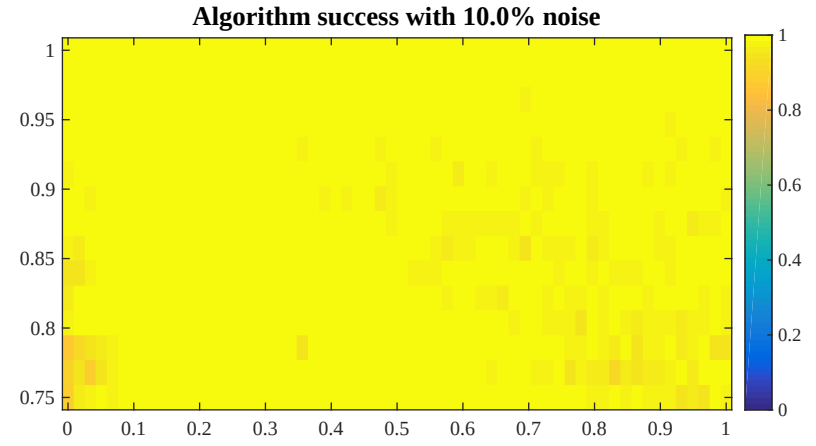


Figure 6.8: Average performance of Algorithm 2 post edge detection with 5.0% Gaussian noise level

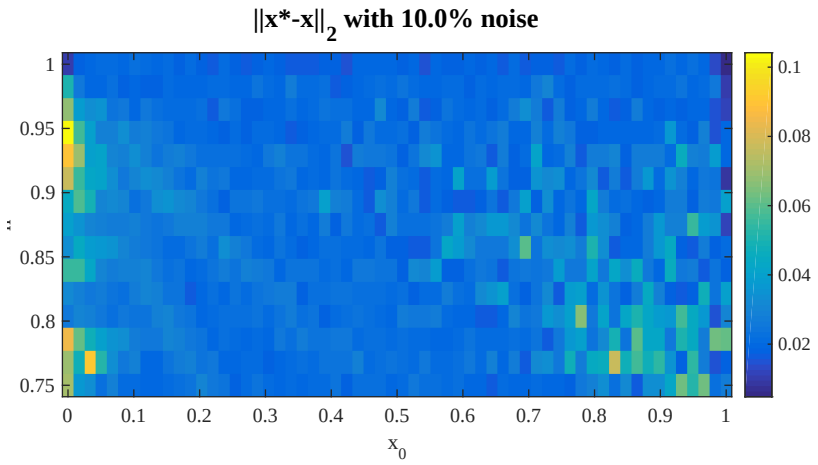


Figure 6.9: L_2 distance between estimated parameters $x = (x_0, h)$ and true parameters $x^* = (x_0^*, h^*)$ with 5.0% Gaussian noise level

Chapter 7

Conclusion

In camera-based barcode decoding, limitations are imposed on the problem by the resolution of the captured images. In this study, we found that in some circumstances recovery is possible even when the narrowest bar is half the pixel size. Restricting to the one-dimensional UPC symbology, we presented a model for camera-based imaging utilizing the dictionary method of [2] and investigated the inverse problem of recovering the 12 digit *word* from the image data. Key to the solution of this problem is the camera model, which can be completely determined by the shift from image edge to barcode and the narrowest bar width. We introduced a method for recovering these camera parameters (x_0, h, x_0'), with which we can estimate the camera matrix and use the greedy algorithm Alg. 2 to recover the *word*. In our analysis of parameter recovery, we found that for $x_0 \in (0, 1)$ and $h \in (3/4, 1)$ we will always be able to recover the camera parameters for noiseless data, relying only on the first three and last three pixel intensities. Additionally, this region of parameter space allows for unique determination of each digit.

Furthermore, during this investigation we studied two "worst case" examples in which the minimal number of pixels was used to cover a single code element, while still satisfying our basic resolution requirements. In these two cases, we showed that the element is still uniquely determined. In fact, we can even uniquely determine under some amount of noise.

Further improvements could be made in this problem by incorporating the checksum requirement, which has been neglected in this work. Using the checksum requirement

significantly reduces the search space of our problem as only eleven of the twelve digits are free to move in $\{0, \ldots, 9\}$.

It is interesting to note that it may be an easier problem to examine the barcode at an angle. That is, a barcode whose axes are *not* aligned with the pixels, as in Figure 2.1. Such conditions allow for multiple views of the same bar, providing opportunity to remove ambiguity otherwise present in the aligned case discussed previously.

Future research directions indicated by this work include

- Rigorous demonstration of unique code word recovery for all values of $0 < x_0 < 1$, $\frac{3}{4} < h < 1$
- Robustness of the recovery for the above range of parameters
- Exploitation of "multiple views" of the same barcode when the bars are not aligned with the pixel axes
- An analysis of two-dimensional barcodes

References

[1] Roger C. Palmer. *The Bar Code Book*. Trafford Publishing, 2007.

[2] Mark A Iwen, Fadil Santosa, and Rachel Ward. A symbol-based algorithm for decoding bar codes. *SIAM Journal on Imaging Sciences*, 6(1):56–77, 2013.

[3] Yue Liu, Bo Yang, and Ju Yang. Bar code recognition in complex scenes by camera phones. In *Forth International Conference on Natural Computation*, volume 5, pages 462–466. IEEE, 2008.

[4] Yue Liu, Ju Yang, and Mingjun Liu. Recognition of QR code with mobile phones. In *Control and Decision Conference*, pages 203–206. IEEE, 2008.

[5] Gaoyan Zhang and Haifeng Ke. An algorithm of distortion correction for Aztec code. In *2nd International Conference on Information Management and Engineering*, pages 173–176. IEEE, 2010.

[6] Aliasgar Kutiyanawala, Xiaojun Qi, and Jiandong Tian. A simple and efficient approach to barcode localization. In *7th International Conference on Information, Communications and Signal Processing*, pages 1–5. IEEE, 2009.

[7] Devi Parikh and Gavin Jancke. Localization and segmentation of a 2D high capacity color barcode. In *Workshop on Applications of Computer Vision*, pages 1–6. IEEE, 2008.

[8] Anil K Jain and Yao Chen. Bar code localization using texture analysis. In *Proceedings of the Second International Conference onDocument Analysis and Recognition*, pages 41–44. IEEE, 1993.

[9] Luping Fang and Chao Xie. 1-D barcode localization in complex background. In *International Conference on Computational Intelligence and Software Engineering*, pages 1–3. IEEE, 2010.

[10] Chunhui Zhang, Jian Wang, Shi Han, Mo Yi, and Zhengyou Zhang. Automatic real-time barcode localization in complex scenes. In *International Conference on Image Processing*, pages 497–500. IEEE, 2006.

[11] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 8(6):679–698, 1986.