Decoding Book Barcode Images

Yizhou Tao
Claremont McKenna College

Recommended Citation
Tao, Yizhou, "Decoding Book Barcode Images" (2018). CMC Senior Theses. 1777.
http://scholarship.claremont.edu/cmc_theses/1777

This Open Access Senior Thesis is brought to you by Scholarship@Claremont. It has been accepted for inclusion in this collection by an authorized administrator. For more information, please contact scholarship@cuc.claremont.edu.
# Contents

Abstract 4

Chapter 1. Introduction to Barcode Images 5
  1.1. One-Dimensional UPC Barcodes 5
  1.2. ISBN Barcodes 6
  1.3. Reading One-Dimensional Barcode Images 6
  1.4. Image Data Formats on MATLAB 6

Chapter 2. Mathematical Modeling of One-Dimensional Barcode and Recovering Algorithm 8
  2.1. A Scanning Model with Gaussian Convolution 8
  2.2. UPC Barcode Symbology 8
  2.3. Block-Diagonality of the Forward Map 11
  2.4. Column Incoherence 12
  2.5. Decoding Procedure 13
  2.6. Recovery of Unknown Barcode 13

Chapter 3. Data Acquisition and Numerical Results 16
  3.1. Methodology 16
  3.2. Numerical Results 17

Bibliography 20

Appendix 21
Abstract

This thesis investigated a method of barcode reconstruction to address the recovery of a blurred and convoluted one-dimensional barcode. There are a lot of types of barcodes used today, such as Code 39, Code 93, Code 128, etc. Our algorithm applies to the universal barcode, EAN 13. We extend the methodologies proposed by Iwen et al. (2013) in the journal article "A Symbol-Based Algorithm for Decoding barcodes." The algorithm proposed in the paper requires a signal measured by a laser scanner as an input. The observed signal is modeled as a true signal corrupted by a Gaussian convolution, additional noises, and an unknown multiplier. The known barcode dictionaries were incorporated into the forward map between the true barcode and the observed barcode. Unlike the one proposed by Iwen et al., we take dictionaries of different patterns into account, specifically for decoding book barcodes from images which are captured with smartphones. We also presented numerical experiments that examined the performance of the proposed algorithm and illustrated that the unique determination of barcode digits is possible even in the presence of noise.
CHAPTER 1

Introduction to Barcode Images

1.1. One-Dimensional UPC Barcodes

To better develop our algorithm, we want to study how a barcode is constructed. In this way, we can get a sense of how to best approach the problem. A barcode is a form of representation of data. For example, for the Universal Product Code (UPC) one-dimensional (1D) barcodes, the information is stored within its varying widths between black and white parallel lines. Although usually barcodes are designed to be read by machines, humans are still able to identify the information within the barcode, which can be translated into a 12-digit number if the number is not printed under the barcode initially. An UPC barcode is 95 modules wide, namely with 84 modulus for the encoded information and 11 modulus for the guard patterns. Each real number that the barcode represents corresponds to 7 modulus since the length of every neighboring 4 barcode numbers have the fixed width of 7. All 1D barcodes has three pairs of longer black and white bars, with one on the left, one in the middle and another one on the right. For humans to read a 1D barcode, one has to identify the widths of each individual bar, except the three pairs of longer black and white bars. The value of the width ranges from 1 to 4, where 1 is the thinnest bar and 4 is the thickest bar. After all the widths are identified, we start with the first white bar on the left and record the 4 neighboring bars together. Then we will move to the first black bar after the pair of longer bars in the middle till the end. The last step would require us to decode the 4-digit number into actual numbers. Table 1 summarizes all the corresponding numbers of each 4-digit number.

<table>
<thead>
<tr>
<th>Corresponding Barcode Digits</th>
<th>Actual Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>3211</td>
<td>0</td>
</tr>
<tr>
<td>2221</td>
<td>1</td>
</tr>
<tr>
<td>2122</td>
<td>2</td>
</tr>
<tr>
<td>1411</td>
<td>3</td>
</tr>
<tr>
<td>1132</td>
<td>4</td>
</tr>
<tr>
<td>1231</td>
<td>5</td>
</tr>
<tr>
<td>1114</td>
<td>6</td>
</tr>
<tr>
<td>1312</td>
<td>7</td>
</tr>
<tr>
<td>1213</td>
<td>8</td>
</tr>
<tr>
<td>3112</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 1. UPC-A barcode number conversion table

For example, the 1D UPC barcode in Figure 1.1.1 has widths of \{2221, 2112, 1411, 1132, 1231, 1114, 1312, 1213, 3112, 3211, 2221, 2112\}. Therefore, the barcode represents the number 123456789012.
1.2. ISBN Barcodes

In the publishing industry, the barcode number is referred to as the International Standard Book Number (ISBN), which is assigned to each version of a book. The 13-digit ISBN consists of 5 parts. The first 1–3 digits are European Article Numbers (EAN), which refer to the Bookland country code. Bookland is a fictitious country that only exists in the EAN coding system. Usually they are of the form “978” and “979”, where “979” is an extension Bookland “978”. The 4–5 digits are the group numbers, and two language-sharing countries have the same registration group numbers. The 6–9 digits represent the publisher of the publication, and each publisher has an unique registration number. The 10–12 digits represent the title and the last digit is the check digit of ISBN. Denote an ISBN code as $x_0x_1...x_{12}$. The check digit $x_{12}$ is the last digit in the barcode. It has to satisfies the following equation:

$$(3x_1 + x_2 + 3x_3 + x_4 + 3x_5 + x_6 + 3x_7 + x_8 + 3x_9 + x_{10} + 3x_{11} + x_{12}) \mod 10 = 0$$

As a result, in order to calculate the check digit, generally we can sum up all the odd digits and times the sum by three. Then we need to add the sum of the even digits to the sum that we calculated previously, and also find the result modulo 10 \[2\].

1.3. Reading One-Dimensional Barcode Images

In real life, barcodes are scanned at a constant speed by a laser beam across the black and white bars. Most barcode scanners consist of three major components: the illumination system, the sensor, and the decoder, where the illumination system sends out the red light across the barcode, the sensor detects the reflected light, and the decoder analyzes the signal and determines the corresponding number. Because of the presence of the black bars, the reflected energy is low since the color black absorbs most of the energy, while the reflected energy is high when the laser is on the white bars \[4\].

1.4. Image Data Formats on MATLAB

Since we will use MATLAB to implement our proposed algorithm and decode barcode images, here we briefly discuss how images are stored and processed on MATLAB. In MATLAB, the most basic image data format is the two-dimensional matrix, where each element in the matrix corresponds to one pixel of the image.
When displaying images, MATLAB uses three different classes: double-precision floating display (double), 8-bit unsigned integer (uint8), and 16-bit unsigned integer (uint16). More specifically, an uint8 can store $2^8 = 256$ distinct values, and an uint16 can store $2^{16} = 65,536$ distinct values. In terms of image types, there are three in total: indexed images, intensity images, and RGB images. An indexed image can be created by a data matrix, $m$, and an additional colormap matrix, map, which is a $m \times 3$ matrix with each row specifies the red, green, and blue components of a color in the data matrix of the image. The data matrix, $m$, can only consist of integer values, since each integer value, $n$, in the matrix corresponds to the color of $n$-th row in the colormap matrix. If the class of $m$ is of the type uint8 and uint16 instead of double, there is an offset effect, as now each integer value, $n$, in the matrix corresponds to the color of $(n + 1)$-th row in the colormap matrix. In MATLAB, one can display an indexed image with the following command: `image(m); colormap(map)` Utilizing indexed images can save a lot of memory and storage space, so they are widely used in early personal computers and hardware to cut costs. However, because of the small storage cost, indexed images have a small set of simultaneous colors in one single image. Another form of image is the intensity image, which is represented by a single matrix, $I$. To display an intensity image, one can use the following command in MATLAB: `imagesc(I, [0, 100])`. The command "imagesc" can be used to set the range of the intensity values and it scales the values to match the values in the entire colormap. For example, the function "imagesc" maps the lower limit of the range to the first colormap entry and the upper limit to the last colormap entry, with values in between linearly distributed [3]. RGB images can sometimes referred to as the truecolor images which are stored by $m \times n \times 3$ 3D arrays on MATLAB. Each $m \times n \times 1$ of an array defines red, green, and blue components of the image. In this way, a pixel located at $(5, 6)$ are stored by RGB(5, 6, 1), RGB(5, 6, 2), and RGB(5, 6, 3), which represents the red, green, and blue components of the pixel, respectively. There are a maximum potential of 16 million colors that can be shown by a RGB image, with each color component’s value between 0 and 1. It will be displayed as black when a pixel’s color components is $(0, 0, 0)$, white when $(1, 1, 1)$, and other color combinations with the values in between. The command “imshow(RGB)” displays the truecolor image when the input is an RGB image.
CHAPTER 2

Mathematical Modeling of One-Dimensional Barcode and Recovering Algorithm

2.1. A Scanning Model with Gaussian Convolution

In this section, we describe and discuss the mathematical modeling of barcodes and the recovering algorithm described in [1].

The reflected energy at a particular position can be modeled by a Gaussian distribution and it is the integral of the product of the intensity level. First, we can take a look at the Gaussian beam’s intensity with respect to time $t$:

$$g(t) = \alpha e^{-\frac{t^2}{2\sigma^2}}$$

where $\sigma^2$ is the variance and $\alpha$ is the constant multiplier.

Denote the barcode as $z(t)$ which represents the true black and white image. The sampled data is represented by the following function:

$$d_i = \int g(t_i - \tau)z(\tau)d\tau + h_i, i \in [m]$$

where $h_i$ represents the noise that the laser scanner adds to the image, $t_i \in [0, n]$ are equally spaced discretization points, and $[m] = \{1, 2, ..., m\}$.

Consider the UPC-A black and white image of barcodes, $z(t)$ can be written as a linear superposition of characteristics functions:

$$z(t) = \sum_{j=1}^{n} c_j \chi(t - (j - 1))$$

where

$$\chi(t) = \begin{cases} 
1 & \text{for } 0 \leq t \leq 1, \\
0 & \text{elsewhere}.
\end{cases}$$

2.2. UPC Barcode Symbology

The width of each barcode is fixed at $n = 95$. Usually a barcode consists of the start pattern, the codes for the first six digits, the middle pattern, the codes for the next six digits, and the end pattern. Following are the components of a barcode:

$$SL_1L_2L_3L_4L_5L_6M R_1R_2R_3R_4R_5R_6E$$

where $S$, $M$, $E$ represents the start, middle, and end pattern respectively. If we represent a unit of a white bar by 0 and a black bar by 1, then $S = E = [101]^T$ and $M = [01010]^T$.

In EAN-13 encoding, although the first digit does not have a barcode pattern that corresponds to that digit, it does have an effect on the barcode pattern of
the next 6 digits on the left-hand side. If the first digit is 0, then it is a UPC-A barcode and all of the first 6 digits have odd parity patterns. Usually, an ISBN textbook starts with the number 9. See Table 1 for all the patterns with the first digit starting from 0 to 9 to determine the specific conversion types.

<table>
<thead>
<tr>
<th>First Number System Digit</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 (UPC-A)</td>
<td>odd</td>
<td>odd</td>
<td>odd</td>
<td>odd</td>
<td>odd</td>
<td>odd</td>
</tr>
<tr>
<td>1</td>
<td>odd</td>
<td>odd</td>
<td>even</td>
<td>odd</td>
<td>even</td>
<td>even</td>
</tr>
<tr>
<td>2</td>
<td>odd</td>
<td>odd</td>
<td>even</td>
<td>odd</td>
<td>even</td>
<td>even</td>
</tr>
<tr>
<td>3</td>
<td>odd</td>
<td>odd</td>
<td>even</td>
<td>even</td>
<td>even</td>
<td>odd</td>
</tr>
<tr>
<td>4</td>
<td>odd</td>
<td>even</td>
<td>odd</td>
<td>even</td>
<td>odd</td>
<td>even</td>
</tr>
<tr>
<td>5</td>
<td>odd</td>
<td>even</td>
<td>even</td>
<td>odd</td>
<td>even</td>
<td>even</td>
</tr>
<tr>
<td>6</td>
<td>odd</td>
<td>even</td>
<td>even</td>
<td>odd</td>
<td>even</td>
<td>odd</td>
</tr>
<tr>
<td>7</td>
<td>odd</td>
<td>even</td>
<td>odd</td>
<td>even</td>
<td>even</td>
<td>odd</td>
</tr>
<tr>
<td>8</td>
<td>odd</td>
<td>even</td>
<td>odd</td>
<td>even</td>
<td>even</td>
<td>odd</td>
</tr>
<tr>
<td>9</td>
<td>odd</td>
<td>even</td>
<td>odd</td>
<td>even</td>
<td>even</td>
<td>odd</td>
</tr>
</tbody>
</table>

Table 1. EAN-13 decoding table based on the first digit

<table>
<thead>
<tr>
<th>Barcode Digit</th>
<th>LHS Odd Pattern</th>
<th>LHS Even Pattern</th>
<th>RHS Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0001101</td>
<td>010011</td>
<td>110010</td>
</tr>
<tr>
<td>1</td>
<td>0011001</td>
<td>011001</td>
<td>110011</td>
</tr>
<tr>
<td>2</td>
<td>0010011</td>
<td>001101</td>
<td>101100</td>
</tr>
<tr>
<td>3</td>
<td>0111101</td>
<td>0100001</td>
<td>1000010</td>
</tr>
<tr>
<td>4</td>
<td>0100011</td>
<td>0011101</td>
<td>1011100</td>
</tr>
<tr>
<td>5</td>
<td>0110001</td>
<td>011001</td>
<td>1001110</td>
</tr>
<tr>
<td>6</td>
<td>0101111</td>
<td>0000101</td>
<td>1010000</td>
</tr>
<tr>
<td>7</td>
<td>0111011</td>
<td>0010001</td>
<td>1000100</td>
</tr>
<tr>
<td>8</td>
<td>0110111</td>
<td>0001001</td>
<td>1001000</td>
</tr>
<tr>
<td>9</td>
<td>0001011</td>
<td>001011</td>
<td>1110100</td>
</tr>
</tbody>
</table>

Table 2. EAN-13 left and right hand side number to pattern conversion table

Furthermore, the digits on the left-hand side of the odd parity pattern and the digits on the right-hand side can be represented by a 7-by-10 matrix,

\[
L = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

and
2.2. UPC BARCODE SYMBOLOGY

Each column of the above two matrices represents a digit code of one number from 0 to 9. Combining all the matrix representations of the start pattern, left digits of odd parity, middle pattern, right-hand side digits, and the end pattern, we can form a 95-by-123 block matrix representation of the barcode dictionary.

\[
D = \begin{bmatrix}
S & 0 & \cdots & \cdots & 0 \\
0 & L & & \cdots & \cdots \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
& & & L & \vdots \\
& & & & L \\
0 & \cdots & \cdots & \cdots & 0 & E
\end{bmatrix}.
\]

Essentially, after the barcode dictionary multiplication, the barcode will have the following form:

\[
c = Dx
\]

where \( x \in \{0,1\}^{123} \) and the 1st, 62nd, and 123rd entries of \( x \) is 1 because each of them corresponds to the start pattern, the middle pattern, and the end pattern respectively. In addition, the entries from the 2nd to the 11th of \( x \), only one of them will take the value of 1, which corresponds to the number the first barcode digit represents, while the rest will be 0. The same applies from the 12th entry to the 22nd entry, and stops at the 61st entry. This pattern for the right-hand side digits starts again after the 62nd entry of \( x \). Overall, the column vector \( x \) has:

\[1 \times 3 + 1 \times 12 = 15\]

non-zero entries, and it will take on the form of the following:

\[
x^T = [1, v_1^T, \ldots, v_6^T, 1, v_7^T, \ldots, v_{12}^T, 1]
\]

where for each \( v_j, j = 1, 2, \ldots, 12 \), the entries of \( \{0,1\}^{10} \) have only one nonzero one. In this linear algebraic representation, we can form the following equation to address the barcode reconstruction problem:

\[
d = \alpha G(\sigma)Dx + h.
\]
We defined the barcode data represented by the matrix \( P = \alpha G(\sigma)D \) as the forward map which is a function of the parameters \( \sigma \) and \( \alpha \). If \( \sigma \to 0 \), then the mapping will become an identity map.

Figure 2.2.1. An example of the forward map of the barcode representation \( P = \alpha G(\sigma)D \) with the following parameters \( r = 5 \), amplitude \( \alpha = 2 \), and Gaussian standard deviation \( \sigma = 0.75 \). The long column vectors at the start, middle, and end correspond to the start, middle, and end patterns in the original barcode.

2.3. Block-Diagonality of the Forward Map

As we can see in Figure 3, the forward map \( P \) almost creates a perfect block-diagonal matrix structure from \( D \). If we decrease the Gaussian standard deviation \( \sigma \), which represents the amount of blur the forward map will add to the barcode figure, the forward map will become closer to the perfect black-diagonal barcode matrix. More specifically, we can partition the forward map \( P \) into the following representation by the block-diagonal structure in the barcode dictionary \( D \):

\[
P = [P^{(1)} P^{(2)} ... P^{(15)}].
\]

For the partition, there are two cases. The first case involves the 1st, 8th, and 15th submatrices of \( P \) as they correspond to the start, middle, and end patterns of the barcode. As a result, these submatrices are just column vectors with length \( m \),

\[
P^{(1)} = p_{1}^{(1)}, P^{(8)} = p_{1}^{(8)}, P^{(15)} = p_{1}^{(15)}.
\]

The second case involves the remaining submatrices that are blurred version in \( P \) of the original barcode representations of \( L \) and \( R \), which are represented by \( m \times 10 \) real matrices. The components of each are as the following:

\[
P^{(j)} = [p_{1}^{(j)} p_{2}^{(j)} ... p_{10}^{(j)}], j \neq 1, 8, 15,
\]

where for each \( p_{k}^{(j)} \), \( k = 1, 2, ..., 10 \) is a column vector of length \( m \). For example, \( p_{8}^{10} \) represents the 8th column in the 10th block matrix in the forward map \( P \).

The reason that each column vector \( p_{k}^{(j)} \) is not of length 95 is that we have an oversampling rate \( r = m/n \), which indicates the number of times each unit barcode
width is sampled during the scanning process. Once we have \( r \), we can further break down the rows of \( P \) into 15 blocks with index set \( I_j \), and we will use \( |I_j| \) to represents the size of the index set. Since we know that the submatrices \( P^{(1)}, P^{(8)}, P^{(15)} \) corresponds to the start, middle, and end patterns, then \( |I_1| = |I_{15}| = 3r \) and \( |I_8| = 5r \). For the rest of submatrices, each corresponds to samples from a digit of length that is of 7 unit widths. As a result, \( |I_j| = 7r \) for \( j \neq 1, 8, 15 \). Therefore, the range of indices of \( I_1 \) would be from 1 to 3r, \( I_2 \) from 3r + 1 to 10r, \( I_3 \) from 10r + 1 to 17r, and so on.

In order to develop a quantitative measure to show the block-diagonality property of the forward map, we set \( \epsilon \) to be the infimum of all positive numbers that are larger than 0 satisfying both

\[
(2.3.1) \quad \| p_k^{(j)} |_{[m] \setminus I_j} \|_1 < \epsilon 
\]

for all \( j \in [15], \ k \in [10] \) and

\[
(2.3.2) \quad \| (\sum_{j'=j+1}^{15} p_{k,j'}^{(j')}) |_{I_j} \|_1 < \epsilon 
\]

for all \( j \in [15] \) and for all \( k_{j+1}, ..., k_{15} \in [10] \). Essentially, the first inequality (2.3.1) indicates that, if we add up all the entries of any column in the forward map except for the indices in the set \( I_j \), the \( l_1 \)-norm of each of these subcolumns will be smaller than \( \epsilon \). The second inequality (2.3.2) indicates that the addition of the \((j+1)\)-th to 15-th row for the indices in the set \( I_j \) will all add up a value smaller than \( \epsilon \), too. For example, if \( j = 5 \), then \( \|(p_{k_6}^5 + p_{k_7}^5 + p_{k_8}^5 + ... + p_{k_{15}}^5) |_{I_5} \|_1 < \epsilon \). If there is no blur effects in the forward map, then there would be no overlap between different blocks, resulting in \( \epsilon = 0 \).

2.4. Column Incoherence

Another property that the forward map \( P \) has is column incoherence. When there is no blur effect in the forward map \( P \), which means that \( \sigma = 0 \), the \( l_1 \)-distance between any two different columns in the forward map is greater than or equal to 2. Let \( D_k \) represents the columns of the barcode dictionary \( D \), then we can conclude that \( \min_{k_1 \neq k_2} \| D_{k_1} - D_{k_2} \|_1 = 2 \). Taking the oversampling rate \( r \) and the constant multiplier \( \alpha \) into account, then

\[
\mu := \min_{j,k_1 \neq k_2} \| p_{k_1}^{(j)} - p_{k_2}^{(j)} \|_1 = \min_{j,k_1 \neq k_2} \| p_{k_1}^{(j)} |_{I_j} - p_{k_2}^{(j)} |_{I_j} \|_1 = 2\alpha r. 
\]

As we can see in the graph, as the blurring factor \( \sigma \) increases, the minimal column separation \( \mu \) decreases smoothly in a convex curve.
2.6. Recovery of Unknown Barcode

In this section, we will explore the justification that the aforementioned algorithms will correctly reconstruct the barcode from the noise-added data $d = Px + h$, if the forward map $P$ is known. In other words, both $\sigma$ and $\alpha$ are known. For this algorithm to work, we need the forward map $P$ to have block diagonality (2.3.1) and column incoherence (2.3.2) properties as we discussed in the previous sections.

The following theorem is proved in [1].

Theorem 1: Suppose that $I_1, \ldots, I_{15} \subset [m]$ and $\epsilon \in \mathbb{R}$ satisfy block diagonality (2.3.1) and column incoherence (2.3.2) properties. Algorithm 1 will be able to correctly reconstruct the barcode signal $x$ from the noise-added data $d = Px + h$ where $h$ represents the random noise if

$$||p_{k_1}^{(j)}|_{I_j} - p_{k_2}^{(j)}|_{I_j}||_1 > 2(2\epsilon + ||h|_{I_j}||_1)$$

for all $j \in [15]$, with $k_1, k_2 \in [10]$ and $k_1 \neq k_2$.
Algorithm 1 A pseudo code to extract $x$ from $\delta$

initialize:
$x_l = \text{zeros}(123, 1); \quad x_l([1 \ 62 \ 123]) = 1$

$\delta \leftarrow d$

for $j = [2 : 7, 9 : 14]$
  for $j \leq 7$,
    $p_j \leftarrow P(:, 2 + 10 \times (j - 2) : 1 + 10 \times (j - 1))$
  else
    $p_j \leftarrow P(:, 3 + 10 \times (j - 3) : 2 + 10 \times (j - 2))$
  end
  for $k = 1 : 10$
    $p_k = p_j(:, k)$
    $K_{\text{min}} = \arg\min_k ||\delta - p_k||_1$
  end
  if $j \leq 7$,
    $l \leftarrow 1 + 10(j - 2) + K_{\text{min}}$
  else
    $l \leftarrow 62 + 10(j - 9) + K_{\text{min}}$
  end
  $x_l \leftarrow 1$
  $r \leftarrow \delta - p_{K_{\text{min}}}^{(j)}$
end

Algorithm 2 A pseudo code to extract barcode digits from $x$

initialize:
$\text{code} = \text{zeros}(12, 1)$
$x_1 = \text{find}(x = 1)$
$x_{12} = x_1([2 : 7, 9 : 14])$

for $i = 1 : 6$
  $\text{code}(i) = x_{12}(i) - (i - 1) \times 10 - 1 - 1$
end

for $i = 7 : 12$
  $\text{code}(i) = x_{12}(i) - (i - 1) \times 10 - 1 - 1 - 1$
end

Proof: Let us suppose that

$$d = Px + h = \sum_{j=1}^{15} p_{k_j}^{(j)} + h = p_{k_1}^{(1)} + p_{k_2}^{(2)} + p_{k_3}^{(3)} + ... + p_{k_{15}}^{(15)}.$$ 

Based on the for-loop in Algorithm 1, suppose that the numbers in columns $k_2, ..., k_{j'-1}$ have already been successfully recovered, then we will denote the residual data to be $\delta$, which represents the remaining data to be recovered. Then if we are at the stage of the algorithm where $k_{\text{min}} = k_j$, then $\delta$ will be

$$\delta = p_{k_j'}^{(j')} + \delta_{j'} + h$$

where $\delta_{j'} = \sum_{j = j' + 1}^{15} p_{k_j}^{(j)}$. 
Now we use show this algorithm will correctly recover the desired digits by induction. To break down the problem, we need to use prove by contradiction to prove that the $j^{th}$ loop of the for-loop in the algorithm will correctly recover $p^{(j')}_{k_{j'}}$.

Suppose that the $j^{th}$ loop of the for-loop is not able to recover correctly. Since $k_{err} \neq k_{j'}$, incorrectly recovering $x$ for a specific column means that

$$||\delta - p^{(j')}_{k_{err}}||_1 \leq ||\delta - p^{(j')}_{k_{j'}}||_1.$$  

More specifically, when we are looking are the left hand side of the above inequality, we have

$$||\delta - p^{(j')}_{k_{err}}||_1 = ||\delta||_{I_{j'}} - p^{(j')}_{k_{err}}|_{I_{j'}}||_1 + ||\delta||_{I_{j'}} - p^{(j')}_{k_{err}}|_{I_{j'}}||_1.$$  

Since $\delta = p^{(j')}_{k_{j'}} + \delta_{j'} + h$, we can rewrite the above equation as

$$||\delta - p^{(j')}_{k_{err}}||_1 \geq (||p^{(j')}_{k_{j'}}|_{I_{j'}} - p^{(j')}_{k_{err}}|_{I_{j'}}||_1 - ||\delta_{j'}|_{I_{j'}}||_1 - ||h|_{I_{j'}}||_1)$$

$$+ (||\delta_{j'}|_{I_{j'}} + h|_{I_{j'}}||_1 - ||p^{(j')}_{k_{err}}|_{I_{j'}}||_1 - ||p^{(j')}_{k_{j'}}|_{I_{j'}}||_1)$$

because

$$|a + b + c - d| \geq |a - d| - |b| - |c|$$

and

$$|a + b + c - d| \geq |b + c| - |a| - |d|.$$  

After combining terms and applying our findings in section 2.3, we can get

$$||\delta - p^{(j')}_{k_{err}}||_1 \geq ||p^{(j')}_{k_{j'}}|_{I_{j'}} - p^{(j')}_{k_{err}}|_{I_{j'}}||_1 + ||\delta_{j'}|_{I_{j'}} + h|_{I_{j'}}||_1 - ||h|_{I_{j'}}||_1 - 3\epsilon.$$  

As for the last steps of the prove, we can add and subtract $||\delta_{j'}|_{I_{j'}} + h|_{I_{j'}}||_1$ from the above inequality.

$$||\delta - p^{(j')}_{k_{err}}||_1 \geq (||p^{(j')}_{k_{j'}}|_{I_{j'}} - p^{(j')}_{k_{err}}|_{I_{j'}}||_1 - ||h|_{I_{j'}}||_1 - 3\epsilon - ||\delta_{j'}|_{I_{j'}} + h|_{I_{j'}}||_1)$$

$$+ (||\delta_{j'}|_{I_{j'}} + h|_{I_{j'}}||_1 + ||\delta_{j'}|_{I_{j'}} + h|_{I_{j'}}||_1)$$

$$= (||p^{(j')}_{k_{j'}}|_{I_{j'}} - p^{(j')}_{k_{err}}|_{I_{j'}}||_1 - 2||h|_{I_{j'}}||_1 - 4\epsilon) + ||\delta_{j'} + h||_1$$

$$= (||p^{(j')}_{k_{j'}}|_{I_{j'}} - p^{(j')}_{k_{err}}|_{I_{j'}}||_1 - 2||h|_{I_{j'}}||_1 - 4\epsilon) + ||\delta - p^{(j')}_{k_{j'}}||_1$$

$$> ||\delta - p^{(j')}_{k_{j'}}||_1.$$  

The above equation shows that we arrive at a contradiction to the assumption that $k_{err} \neq k_{j'}$. As a result, the $j^{th}$ loop of the for-loop in the algorithm will correctly recover $p^{(j')}_{k_{j'}}$. 
CHAPTER 3

Data Acquisition and Numerical Results

3.1. Methodology

In order to test our recovery algorithm, we took 83 images of barcodes from various books in the Claremont Colleges Honnold Mudd Library as input dataset. The images which do not have the full barcode or were corrupted are excluded from our dataset. See Figure 3.1.1 for an example.

![Figure 3.1.1. An example of bad images.](image)

In the final dataset, we have 68 images in total. These images are shown in the Appendix. Once the images were obtained, we put all of the image files in a folder and used MATLAB to process the images. After renaming all the images to barcode numbers, we used the MATLAB subroutine “dir” to get the file names, so that the barcode numbers in the file names could be automatically extracted as the program read the images one by one. For each of the images, we converted them from RGB images to gray images for easy thresholding purposes. We also set $nx$ and $ny$ to be the length and width of the image. Since we are only interested the barcode, we only took and extracted the data at the location $0.75 \times nx$ of the image to obtain a single gray bar image.

However, the gray bar image we obtained included parts of the image that are not barcode-relevant. Thus, we need to identify the start point and end point of the barcode on the gray bar, namely the transition points where the gray bar moves from either white to black or from black to white. We identified a point black if its intensity level is below $0.35 \times \text{(image intensity range + minimum intensity level)}$, and a point white otherwise. In this way, the irrelevant region can be easily removed. See Figure 3.1.2 for an example. The red curve on the left image indicates the location of 1D barcode signal and the intensity of the barcode signal is shown on the right figure.
Since the first digit of the barcode number is always 9, the second, third, and fifth digit of the first 6 numbers will represent the even pattern, while the first, fourth, and sixth will represent the odd pattern. In order to incorporate the even pattern into our existing algorithm, we decided to run through the algorithm twice, with each time corresponding to one barcode dictionary of either the odd pattern or the even pattern. More specifically, we replaced \( L \) with \( L_{\text{even}} \) in the barcode dictionary matrix \( D \) for the algorithm that deals with the even pattern, where \( L_{\text{even}} \) is as the following:

\[
L_{\text{even}} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{bmatrix}
\]

Once the algorithm was run twice, we got two barcodes corresponding to the odd and even patterns respectively. In order to get the final recovered barcode numbers, we replaced the second, third, and fifth digits of the odd pattern with the numbers from the even pattern.

### 3.2. Numerical Results

We also did stress test to see how lenient our algorithm is by changing different parameters in the model. As we can see in the table above, we create an identity map \( P \) as we set \( \sigma \) close to 0. As \( \sigma \) increases from 0.01 to 0.75, the recovery rate increased when the oversampling rate \( r \) is at 5 and 10, but not when it is 20. As \( r \) increases from 5 to 20, we saw the recovery rate increased from 32.35% to 44.1%. The increase is more dramatic from \( r = 5 \) to 10, but not so much from \( r = 10 \) to 20. We also investigated the reasons why our algorithm failed to recover some of the barcodes and the limitations of our algorithm. First, some of the books were tilted or the images were taken at an angle, so extracted 1D barcode signals have different trending. See Figure 3.2.1 for examples. The actual widths of the barcode patterns may get increased or decreased, which make it harder for our algorithm
Table 1. The percentage of recovery rate for different sampling rate $r$ and Gaussian blurring factor $\sigma$.

<table>
<thead>
<tr>
<th>Oversampling Rate ($r$)</th>
<th>Gaussian Blurring Factor ($\sigma$)</th>
<th>Recovery Accuracy Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.01</td>
<td>32.33%</td>
</tr>
<tr>
<td>5</td>
<td>0.75</td>
<td>33.82%</td>
</tr>
<tr>
<td>10</td>
<td>0.01</td>
<td>41.18%</td>
</tr>
<tr>
<td>10</td>
<td>0.75</td>
<td>42.65%</td>
</tr>
<tr>
<td>20</td>
<td>0.01</td>
<td>44.1%</td>
</tr>
<tr>
<td>20</td>
<td>0.75</td>
<td>42.65%</td>
</tr>
</tbody>
</table>

to recognize them. In reality, a laser scanner needs to scan a barcode in different angles, so that it can make sure that it captures the barcode at a horizontal angle [5].

Second, some of the images have inhomogeneity as they got corrupted when the pictures were taken. For example, some images have shadows on them because of the lighting. As a result, some parts of the images will be darker, while the others will be brighter. However, when processing these images, our algorithm does not take that in account, as the location where the red line will be extracted is fixed, which is $0.75 \times nx$. In other cases, some books have an additional layer on the barcode which would make them darker and blurer as well. See Figure 3.2.2 for examples.
Third, our algorithm did not include the check digit of the barcode into consideration. Usually the check digit is calculated based on the previous 12 digits. As a result, if some of the digits of the barcode get recovered incorrectly, the inclusion of the check digit can help recover and correct the previous digits.

As we recognize these limitations, we also realize the places for future improvement. First, we can acquire better images with higher quality. When we take pictures, we should make sure that there are no shadows and no inhomogeneity effect on the images, so that our algorithm can better process the barcode information. Second, we can also incorporate more barcode recovery information in the algorithm and improve the way that we extract the sample data. For example, we can include a checking process of the recovered barcode numbers by using the check digit. When selecting the sample data, we could also make the location of red line more flexible to get the best line, instead of extracting the sample barcode data from a fixed location.
Bibliography


Figure 3.2.3. The first 40 images of total 68 images.
Figure 3.2.4. The 41th-68th images of total 68 images.