**Project Report**

# ISBN Bar Code Detection using Image Acquisition

**Jain Chinmay, Kamat Deepali**

**Aim:**

To device an efficient and affordable technique to detect and scan ISBN bar codes using images taken from a camera.

**Challenges:**

Obtaining images under uniform light and detecting barcodes at different angles. Reading barcodes using computer vision algorithms.

**Related Work:**

Barcode reading is one of the most interesting problems of computer vision. It has been tried and to some extent successfully solved by few. The algorithms used for reading the barcode binarized the image, algorithms perform some sort of binarization, either by gray scale thresholding [1] or by finding the bar edges [2] [3]. Latest approaches involves use of stochastic methods that uses deformable barcode digit models in a maximum likelihood setting. Latest approaches involves use of extreme learning machine [5] algorithms and other computational intelligences model [6]. Another interesting approach used for barcode reading involved use of a scanline. A scanline in a line that is drawn across the center of the image in the hope that the barcode is at the center of the image. Later the scanline is decoded to get the barcode reading [2].

**Challenges Faced:**

There are many different types of one dimensional barcodes used in the industry. Each of these have a different technique to decipher the data. Hence we are limiting the scope of our project to ISBN barcodes. ISBN stands for International Standard Book Number. Earlier works on one dimensional barcode reading was done on UPC and EAN barcodes. Other challenges faced were the data. Data was presented in the form of images. Reading of barcodes from these images was difficult as the image was noisy. Noise in the image could be from device defect, resolution mismatch, focusing capabilities of the source, illumination and/or motion-blur. Although there are noise removal algorithms to tackle these challenges, barcode data being black and white in nature is also affected by these algorithms. Working with image data can challenging as the image can have different orientation. Images taken at different angles or from different length change the position of the barcode in the image. Rotating of the image, implies reading the barcode inversely, hence finding the correct location and orientation of the barcode in the image is important

**Applications:**

Our algorithms will be tested on pictures of the books in the RIT Wallace Library, taken using a USB plug-in camera, therefore it might be subject to ethical considerations. However, if need be, we will take permission to do so from the RIT Wallace Centre authorities.

Barcode readers are electronic devices that read and output printed barcodes to a computer or other devices that process them. Congruent to a flatbed scanner, barcode scanners consists of a light source, a lens and a light sensor translating optical impulses into electrical ones. Also, barcode readers are designed with decoder circuitry that analyzes the barcode's image data provided by the

sensor and sends the barcode's content to the scanner's output port[4]. This kind of design needs bulky packaging which makes it expensive and more subject to failures than incorporating the same task into a more preferred and commonplace device, the cell phone.

Though cellphone cameras without auto-focus are not the best choice for reading some common barcode formats, certain codes can be read quickly and accurately with or without auto-focus. Smartphones with third party scanning applications can be used to scan and read barcodes to obtain information relative to the product.

This would open up a number of applications for consumers:

- Groceries: nutrition information, regular update of shopping lists when the last of an item is used, etc.
- Book catalogs and devices
- Movies: DVD/VHS movie catalogs
- Music: CD catalogs, search and play MP3 upon scanning
- Personal Property inventory (for insurance records) code can be scanned and stored into personal software and then, receipts can be scanned and read in such a manner that the system be designed to automatically link the receipts to their appropriate entries.

**Dataset:**

Our dataset is designed with the project in mind. We captured images of 20 ISBN barcodes for books across a variety of subjects. The images were captured from the same source and under same conditions. By same condition we mean, we kept the illumination, the distance of the source to the floor and the floor background constant. However the floor background was not influential as we

took close capture images of the barcode, thus the book cover being the background that was captured in the image. The source used for capturing these images was a simple USB camera. We kept the settings of the camera constant for all images. Although the distance of the camera from the floor was constant, each barcode was printed in different style and size. Because of this, the few images had a smaller barcode image area as compared to the other. To avoid this problem we cropped the book cover background from the relevant images. This led us to images with different sizes. We wrote a small Matlab script that would resize all images to a standardized size. We choose the standard size as 1024X768 pixels. Although each image was taken under same condition and using same device, we expect the images to contain some noise and blurriness. For this purpose we wrote an image cleaning module in our code. We do aim to rotate the bar codes while taking the pictures that would make the entire software more robust and efficient. We took few images with rotated barcodes in them, so as to cover all test cases possible. So now we have a mix of straight and rotated images that are cleaned and filtered.

**Methodology:**

Before we begin our experiments, we need to convert our images to a black and white binary format. The reason we perform this step here and not while preparing our data is, at the end when we present the image to end user, we didn't want user to be confuse the image with the image he has provided and the image we used for the experiments. We first filter the image using the median filtering technique. Median Filter is suitable for our application as it straighten the crooked lines in the image. Next for shape edges and to clear the blur we improve the contrast of the image using image adjustment. We convert our image to a gray scale image for this operation. Now we take this adjusted image and perform the 'im2bw' operation on it with the default threshold value of

0.5. This gives a pure black and white binary image. We are using this format of the image as by intuition barcodes are black and white images made up of various lines of different width. So by using this format we are also able to ignore all other inconsequential details that might not have been filtered in the image cleaning step.

Now we want to reorient the barcode, so that the barcode lines are parallel to edge of the scan box. Since the lines are supposed to be parallel to the edges, we perform rotation of the scan box by a calculated angle. For calculating the angle we follow the following steps:

1. Get the position of the end points of the first barcode line in the scan box. Store them in a matrix X. Let these points be (x1, y1) and (x2, y2). Also add a row of homogenous coordinates to account for the translation in the point coordinates.
2. Now construct the matrix Y with points (x1, y1) and (x1, y1+30). Update matrix Y with a row of homogenous coordinates. We add 30 to y1 as the height of the scan box is 30 pixels.
3. Now we get the Rotation-Scaling-Translation matrix by diving matrix Y by matrix X, this is written as

$$RST = \frac{Y}{X}$$

4. Now we need to compute the rotation angle $\theta$ from the RST matrix. The mathematical definition of the RST matrix is given as

$$\begin{bmatrix} S_X.cos\theta & -sin\theta & T_X \\ sin\theta & S_Y.cos\theta & T_Y \\ 0 & 0 & 1 \end{bmatrix}$$

To the get the $\theta$ we take the mean of values at position 1,2 and 2,1 from the RST matrix and then take the sin inverse of that value. In short,

$$\theta = \sin^{-1}\left(\frac{abs\big(RST(1,2)\big) + abs\big(RST(2,1)\big)}{2}\right)$$

5. Now we rotate our scan box by the angle $\theta$ to get the straight bar code lines.

Next we pick a scan box across the image. For capturing the scan box, we go to the center of the image and draw a rectangle around that point. The rectangle is drawn by taking a total of 15 pixels in the upper and lower direction of the center pixel. Now the vertical edges of the scan box are expanded till the end of the image is reached. This gives us a scan box, which we then resize to a horizontal line to get scan line across the barcode. This scan line gets further resized so as to cover only the 92 slots of the bar code. Significance of each of these slots in decoding the barcode is explained below.

To decode the barcodes, we capture the pattern of the bars in the scan box. The bars are read in a binary format. 0 represents a single black bar and 1 represents a single white bar. The white and black bars have varying widths. Each binary digit creates a module representing a bar in the barcode pattern. Thicker black and white bars are represented using multiple black and white bars. The binary codes are 7 bits in length and are arranged together such that each code has groups of 0s and 1s of varying widths.

ISBN barcodes have a basic framework as shown below:

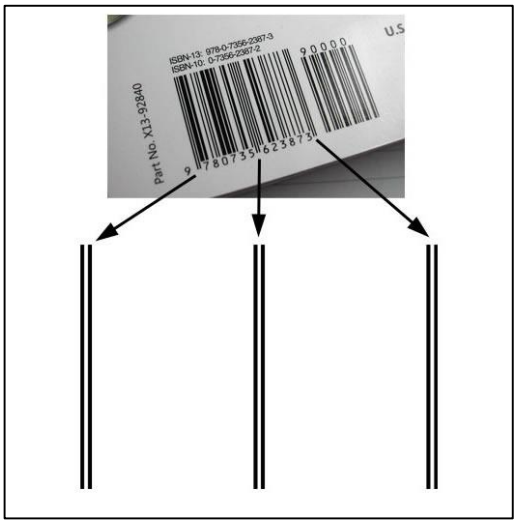

Figure 1: ISBN barcode framework

The three bars displayed in the picture above form the basic framework of the ISBN barcode. These bars do not contribute to the ISBN numbering pattern but the ISBN fundamental outline would be incomplete without them. Therefore, when reading the barcodes in binary format, we omit the values of the 3 modules on left, five modules in the middle and three modules on the right end of the pattern respectively. Each ISBN block has 6 digits, each having a 7-bit binary encoding. This sums up to a total of 84 modules in the end; 42 in the first block and 42 in the second.

When ISBN barcodes are generated, the following encoding pattern is followed. The first block or the first six digits are encoded using the following technique:


Figure 2: First block of barcode pattern

Two different encoding schemes are used to encode the characters in the initial section of the bar code. They are called as Scheme A and Scheme B. They are shown in table 1.

| ISBN Character | Binary Code - Scheme A | Binary Code - Scheme B |
|---|---|---|
| 0 | 1110010 | 1011000 |
| 1 | 1100110 | 1001100 |
| 2 | 1101100 | 1100100 |
| 3 | 1000010 | 1011110 |
| 4 | 1011100 | 1100010 |
| 5 | 1001110 | 1000110 |
| 6 | 1010000 | 1111010 |
| 7 | 1000100 | 1101110 |
| 8 | 1001000 | 1110110 |
| 9 | 1110100 | 1101000 |

Table 1: First Block Encoding Scheme

Each character in the ISBN barcode pattern, apart from the first digit (as the first digit is the same for each batch), is encoded from either of the two schemes, A or B. The choice between Scheme A and Scheme B is decided by another table. This decision is made according to the character position of the digits in the first block of code. The first character of an ISBN barcode pattern is always a 9. The encoding for rest of the following characters 2 to 7 follows the scheme as displayed in the table below.

| **Scheme for character at 2nd Position** | **Scheme for character at 3rd Position** | **Scheme for character at 4th Position** | **Scheme for character at 5th Position** | **Scheme for character at 6th Position** | **Scheme for character at 7th Position** |
|---|---|---|---|---|---|
| A | B | B | A | B | A |

Table 2: Scheme Encoding Decision Table

The second block or the final six characters of the ISBN barcode pattern are encoded taking into consideration the space between the middle bars and the final bars.

Figure 3: Second Block of ISBN barcode pattern

The table below displays the regime followed for decoding the second block.

| ISBN Character | Binary Code |
|---|---|
| 0 | 0001101 |
| 1 | 0011001 |
| 2 | 0010011 |
| 3 | 0111101 |
| 4 | 0100011 |
| 5 | 0110001 |
| 6 | 0101111 |
| 7 | 0111011 |
| 8 | 0110111 |
| 9 | 0001011 |

Table 3: ISBN Barcode Decoding Scheme for Second Block

**Results:**

We tested our algorithm against the 20 images we captured at the beginning of our project. We got the following results for the last 12 digits of the barcode. The first digit of the barcode always being 9, by design, we will not incorporate the first digit in our result analysis.

| Image Number | Actual Barcode Number | Decoded Barcode Number |
|---|---|---|
| 1 | 780262610377 | 7802626103nn |
| 2 | 780060934415 | 78n06nn34nnn |
| 3 | 781851684717 | 78n851684nnn |
| 4 | 781618770233 | 78161877nnnn |
| 5 | 780195334654 | 7801953346nn |
| 6 | 780306449024 | 780306n490nn |
| 7 | 780827359499 | 78082n3594nn |
| 8 | 780805746211 | 780805746nnn |
| 9 | 780231071833 | 780231n71nnn |
| 10 | 780750637756 | 780750n377nn |
| 11 | 781565842267 | 7815658422nn |
| 12 | 780131855861 | 78n13785nnnn |
| 13 | 781934356586 | 7819343565nn |
| 14 | 780674019270 | 7806nn019nnn |
| 15 | 780140102918 | 7801401029nn |
| 16 | 780387984575 | 78038n9845nn |
| 17 | 780670031443 | 78067n031nnn |
| 18 | 780060557546 | 78006n5575nn |
| 19 | 780374278649 | 78037n278nnn |
| 20 | 781570038396 | 7815700383nn |

Table 4: Results

As seen from the table 4, we were able to read 10 digits correctly, on an average, out of the 12 digits. The $13^{th}$ digit being the first digit is always 9. Another fact to notice is, a lot of digits that are at the end of the barcode are not being decoded correctly, which is because the images contain some blur towards the end of the image and that changes the encoding pattern found by the algorithm for that digit. Thereby resulting in wrong classification. While analyzing the results for the wrongly predicted digits we noticed that the bit pattern captured for those digits was very similar to the bit pattern expected for correct encoding. As mentioned before each digit is encoded using a combination of 7 bits, for the wrongly encoded digits, the bit pattern was off by 1 or 2 bits in most of the cases. This happens because of aliasing of the image while selecting the scan box for barcode decoding. While selecting this scan box, the narrower lines are lost while resizing the image.

**Future Work:**

We worked on few approaches to avoid the aliasing effect caused by selecting the scan box. One of the technique that we were eager to implement was to read each line vertically in the original cropped image and then squishing that line based upon the width of the bar in the original barcode. This might counter the aliasing effect we found while selecting the scan box. We can extend our algorithm to read different types of 1D barcodes like the EAN barcodes, UPC barcodes amongst others. We can ensemble our algorithm in a mobile application so as to have a handy application to read barcodes through cell phones. We can also integrate our algorithm with a B2C client and automate retail purchasing of products through the barcode.

**References:**

[1] *Reading 1D Barcodes with Mobile Phones Using Deformable Templates* - Orazio Gallo, Student Member, IEEE, and Roberto Manduchi, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 33, 2011

[2] *Robust Recognition of 1-D Barcodes Using Camera Phones* - Steffen Wachenfeld, Sebastian Terlunen, Xiaoyi Jiang (IEEE 978-1-4244-2175-6)

[3] *Reading Challenging Barcodes with Cameras* - Orazio Gallo, Roberto Manduchi - University of California, Santa Cruz.

[4] Barcode reader. (2015, September 16). In *Wikipedia, The Free Encyclopedia*. Retrieved 00:15, October 24, 2015, from https://en.wikipedia.org/w/index.php?title=Barcode_reader&oldid= 681325730

[5] *An ELM based Barcode Localization Algorithm* - Wany Chuncai, Li Yingtao and Yang Yudong, Applied Mathematics and Materials, Vol. 380-384(2013)

[6] *A Bayesian Algorithm for Reading 1D Barcodes* - Ender Tekin, James Coughlan, 2009 Canadian Conference on Computer and Robot Vision