# Image Processing Final Report

For my research project, I decided to create a program that reads a Universal Product Code (UPC). Specifically a UPC-A code because there are four types of UPC codes.[[1]](#footnote-1) UPC was invented by George J. Laurer at IBM in 1973 in order to make grocery stores more efficient. [[2]](#footnote-2)

A UPC-A is a series of 30 black and white lines of different strokes that stores 12 numbers. These 12 numbers represent a 6 digit manufacturer ID, a 5 digit item ID and one check digit which is used to verify the UPC code reader accuracy.

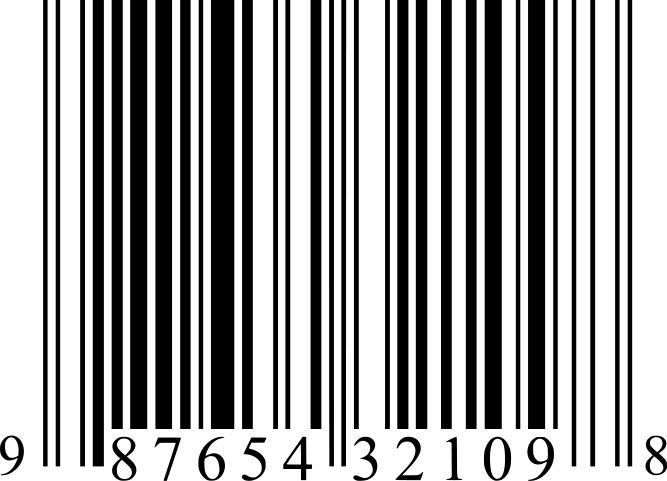


Fig 1[[3]](#footnote-3)

Every two lines and spaces in the UPC-A code represents a number or is a boundary. The first two lines, the last two lines and the middle two lines are boundaries. The first set of 12 lines and spaces represents the first six numbers; the last set of 12 lines and spaces, the last 6 numbers.

Each set of two lines and spaces represents one number. Each bar and space can vary between 1 unit and 4 units wide, yet, each combination of two bars and two spaces is always 7 units wide total. Figure 2 shows how each number both on the left and right of the boundary is represented.

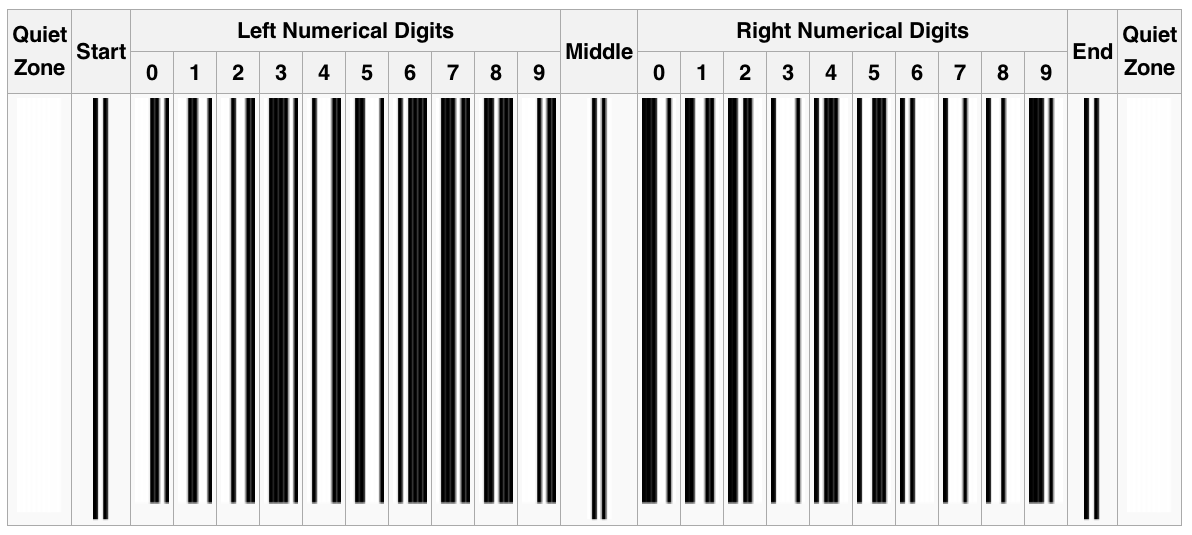


Fig 2[[4]](#footnote-4)

As you can see, the number 0 is represented by a white space of three units, one black bar of two units, one white space of one unit and one black space of one unit for a total of seven units.

The numbers left of the middle are the optical inverse of the numbers right of the middle. That means that every white space becomes a black line and vice versa. So, the number zero right of the middle is becomes a black bar of three units, a white space of two units, a black space one unit long and a white space one unit long. That way a system can read the barcode in halves and not even on the same pass if necessary. 5

The numbers on the right all have an even number of black bars while the numbers on the left have an odd number of black bars. Using this information, a scanner can further determine which half a barcode it is reading. 5

The middle segment is only 5 units wide compared to all the other segments which are 7 units wide. It has one white space, one black line, a white space, black line and a white space. That way the second bar is always four units away from the first white space no matter which direction you read the UPC in. 5 If you look at the position of the second bar in each number, you’ll see that the second bar is always greater than 4 units away from each black line. (Except for the start and end bars.)

To begin the project, I started with a picture of a barcode I took on my cell phone. See Figure 3.



Figure 3

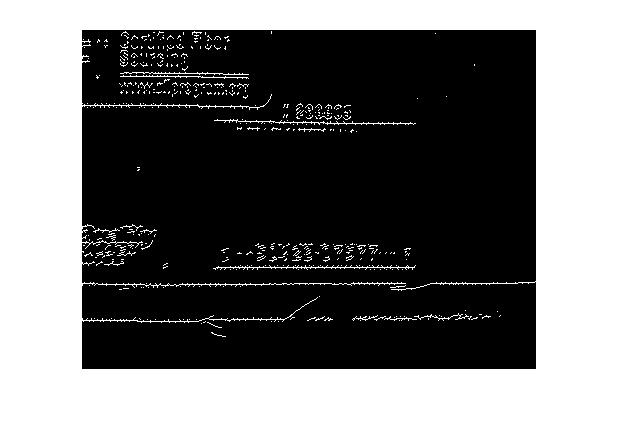
A barcode scanner needs a series of pure black and white pixels and this picture has too much blurriness and a background that needed to be removed.

I figured if I shrunk, equalized and grayscaled the image, I could get rid of a lot of unnecessary noise.

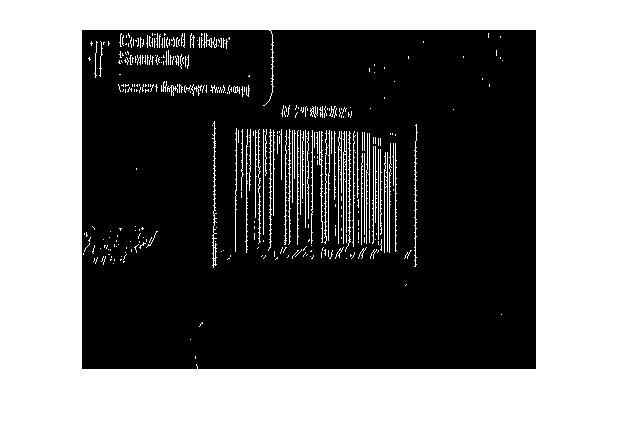
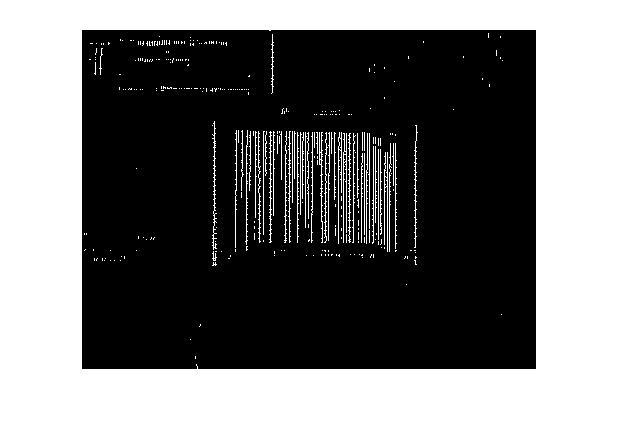


Then, knowing the sobel edge detector can detect vertical and horizontal lines, I figured that it would detect a region with vertical but no horizontal lines on the UPC code. In order to get a region that contained the entire UPC code, I had to dilate the results of the edge detectors manually.

First, I found the horizontal edges and dilated them to produce regions.



Then, I found the vertical edges and removed noise by ignoring edges that lie on a region containing a horizontal edge.



Then, I found the intersection of all points which have vertical but not horizontal lines and dilated that result to produce a large region that contained a barcode.



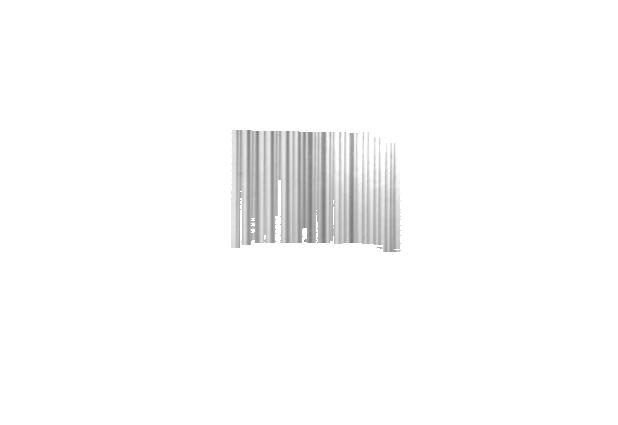
Then, I identified each region using a color.



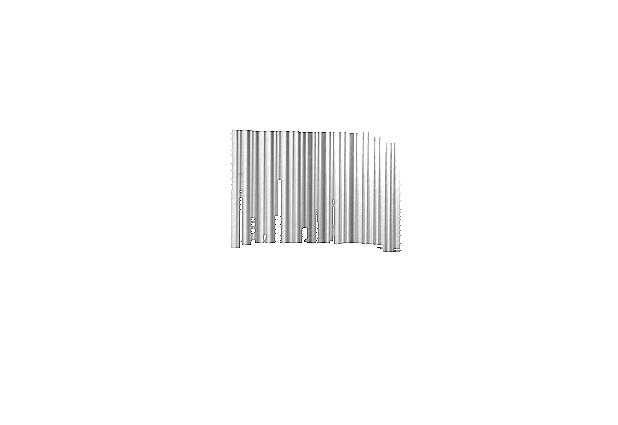
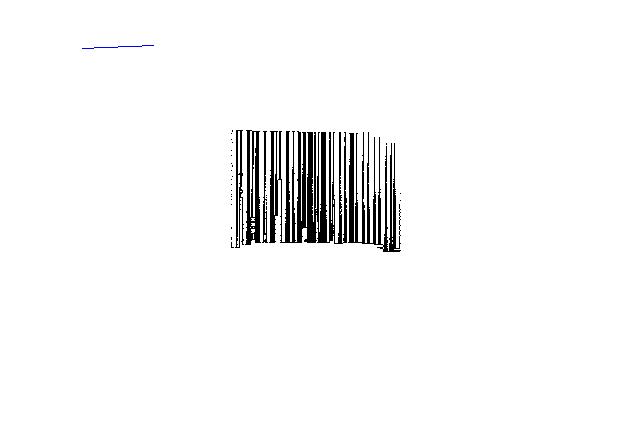
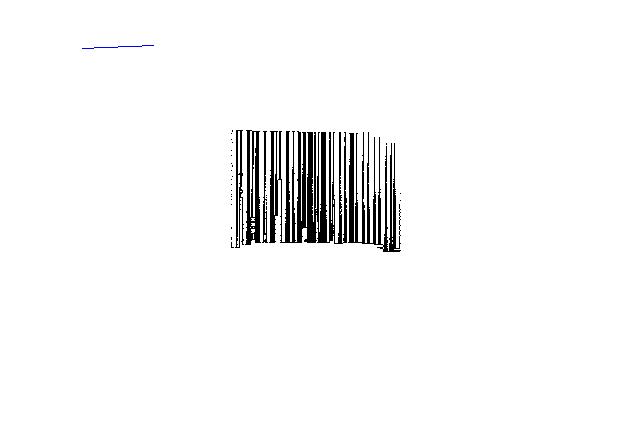
I found the biggest region by volume of pixels.



With this, I was able to get the barcode without the background.



I then sharpened the barcode and used histogram segmentation to convert the image into binary.



Even though the resulting image does not appear accurate, it was still readable around the middle.

Then, I used a program[[5]](#footnote-5) to record a line between two points and retrieve all the pixels that touch the line. It then cycles between counting consectutive white and black pixels. It groups each consectutive set of white and black pixels into 60 groups because there are 30 pairs of black lines and white spaces. Then, it divides the length of each consecutive group by the width of the first black pixel it it counters.

Then, for every group of four pixels starting from the 5th consecutive pixel group (after the 3 white spaces and 2 black spaces of the guard) until before 26th group (where the divider lines are), if the group matches one of the patterns for the numbers on lines 180 to 199, the program reads in that number. The patterns do not check if the groups of pixels are white or black, only their sizes because regardless of the orientation, there will always be 3 pixels, 2 pixels 1 pixel and then 1 pixel for a 0 and so on.

Other research:

<http://www.aimglobal.org/technologies/barcode/verify/x3182.pdf>

Savir, D.; Laurer, G. J.; , "The characteristics and decodability of the Universal

Product Code symbol," *IBM Systems Journal* , vol.14, no.1, pp.16-34, 1975  
 doi: 10.1147/sj.141.0016  
URL: <http://ieeexplore.ieee.org.ezproxy2.lib.depaul.edu/stamp/stamp.jsp?tp=&arnumber=5388144&isnumber=5388142>

<http://www.aimglobal.org/technologies/barcode/verify/Verif.Theory%20short.pdf>

<http://www.lvs-inc.com/ContentPDFS/ANSIOverview.PDF>

1. http://www.gs1us.org/standards/barcodes/ean\_upc [↑](#footnote-ref-1)
2. http://www.laurerupc.com/ [↑](#footnote-ref-2)
3. http://en.wikipedia.org/wiki/File:UPC\_A.svg [↑](#footnote-ref-3)
4. http://en.wikipedia.org/wiki/Universal\_Product\_Code [↑](#footnote-ref-4)
5. http://www.mathworks.com/matlabcentral/fileexchange/21899-recognize-barcode/content/runprogram.m [↑](#footnote-ref-5)