A Fast and Efficient Method for Reading Barcodes

Jeffery Opoku-Mensah, Meet Patel

March 14, 2017

Abstract

In this article, we present an algorithm for efficiently scanning barcodes, maintaining approximately 86% accuracy over 49 trials. The algorithm takes on average 1.46 seconds to execute from image load to storing the output. For a user capturing footage of the barcode in near optimal conditions(defined later), it is expected to take 1.703 seconds to read. We outline and discuss the specificities of the algorithm.

1 Requirements

The algorithm performs best when in optimal conditions, specifically, conditions satisfying the following properties:

1. The barcode is not taken at an extreme angle.
2. The barcode is not bent or torn.
3. The environment is sufficiently bright so that color values are distinguishable.
4. The barcode contrasts sufficiently with the environment so that they do not have similar hues.
5. Glares are not present such that the true color of the barcode can not be read.
6. The barcode is read right-side up.

2 Execution Outline

The execution of the algorithm follows these steps(refer to code for more detail):

1. The image of the barcode is loaded into memory.
2. The image is downsized to contain approximately 10^6 pixels.
3. The image is filtered to selectively include potential pixels belonging to the barcode.
   (a) The BGR(blue-green-red) values are converted to HSV(hue-saturation-value).
   (b) Pixel values that lie in a range of certain hue values are selected and set to white. Similar processes apply to saturation and value. The rest of the pixels are discarded/set to black.
4. Using the Canny Edge Detector, the edges of the filtered barcode are determined.

5. Using Hough Line Detection, lines are drawn over the edges, resulting in many points of intersection.

6. The corners of the barcode are determined from the intersection points.

   (a) The average of all the intersection points is captured.

   (b) All intersection points to the left and right of the average point are considered as separate clusters and also are averaged.

   (c) For each of the left and right averages, all intersection points above and below them are considered as separate clusters and are averaged to obtain the four corner points.

7. The four intersection points determine a quadrilateral, from that, a perspective transform is applied to map the quadrilateral to a rectangle aligned with the coordinate axes.

8. The filtered barcode is inverted(black turns white and vice versa) and warped with the transformation.

9. The barcode is shrunk to a set size and regions where dots can potentially be are surveyed, values being recorded.

10. The values are mapped to alphanumeric symbols including the underscore through a custom mapping.

Images given during processing are shown below. These images do not necessarily come from the same trial.

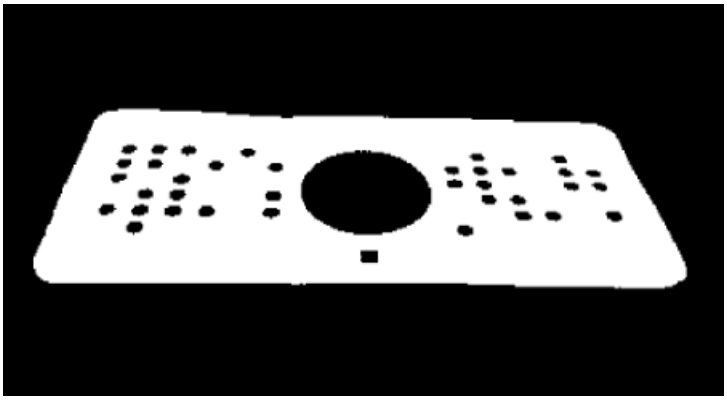

Figure 1: Steps 1-3: The barcode is downsized and selectively filtered.

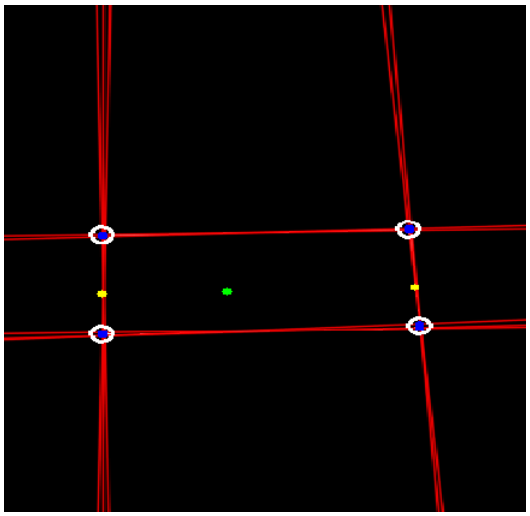

Figure 2: Steps 4-6: The green point is the total average, the yellow points are the averages of the left and right clusters, and the white circles are the averages of the four resulting clusters, i.e., the corners of the barcode.

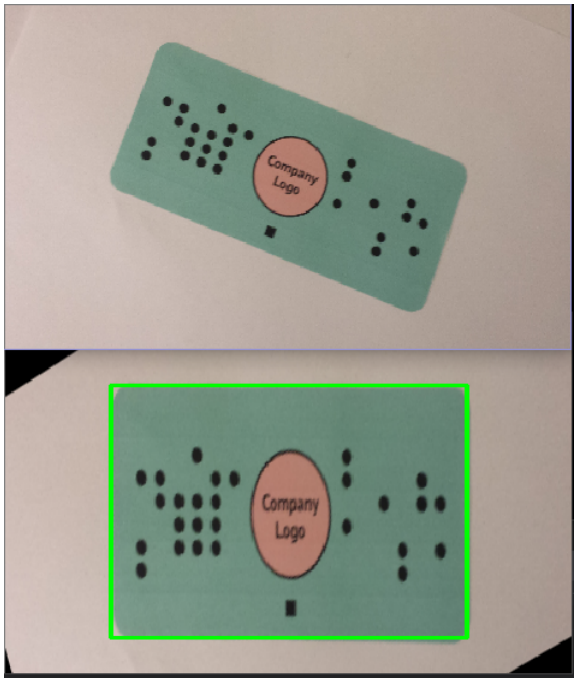

Figure 3: Steps 7-8: The barcode is perspective warped into a rectangle aligned with the coordinate axes.

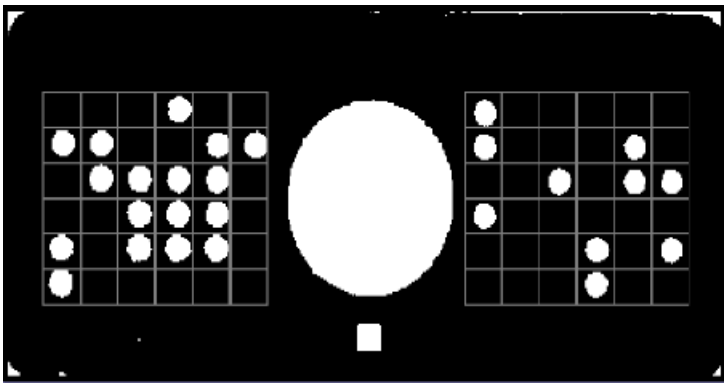

Figure 4: Steps 9-10: The values of the barcode are read through surveying the image for bright values(the dots).

3 Execution Analysis

The testing of the algorithm was used on 49 images separated into different groups based on the angle the barcode was taken at. The average time taken in a trial of 49 successive barcodes is approximately 1.46 seconds, with the accuracy being 85.7%. We calculate the expected time to successfully read a barcode as 1.703 seconds.

Common sources of failure stem from reading the barcode in suboptimal conditions. Most commonly, these sources are a lack of bright lighting, an undesirable camera angle, and a deformed barcode. Respectively, these cause the following problems: an inability to differentiate the barcode from the environment(darkness); an inability to preserve the shape of the dots through a perspective warp(camera angle); an inability to properly detect edges and intersections(bent barcode). A less likely but serious problem is a blurred image, which can easily be prevented by first maintaining image focus. A blurred image effectively reduces the darkness of the dots.

These aspects of a suboptimal image can be found in the 7 images that did not successfully pass through the algorithm.

Below is a table with the data from the successive trials of 49 barcodes. The barcodes are attached in a .zip file. Each barcode has the format TB_[N][M].png, where N ranges from 0 to 6, and describes which camera angle was used, and M describes the number of the photo taken(save for one exception, TB_26 which is at camera angle 6).

Barcode Trials			
Image Name	Actual Value	Output	Accuracy
TB_00	1TLtkT1TETlF	1XLtkT1TETlF	Incorrect
TB_01	TmgRBltnwYVS	TmgRBltvgstu	Incorrect
TB_02	2qGMTsNJpFbs	2qGMTsNJpFbs	Correct
TB_03	joeKugQ083oa	joeKugQ083oa	Correct
TB_04	WTKiwmpYm2Uq	WTKiwmpYm2Uq	Correct
TB_05	WTKiwmpYm2Uq	WTKiwmpYm2Uq	Correct
TB_06	m4I9cY8GsPTn	m4I9cY8GsPTn	Correct
TB_10	2qGMTsNJpFbs	2qGMTsNJpFbs	Correct
TB_11	joeKugQ083oa	joeKugQ083oa	Correct

Image Name	Actual Value	Output	Accuracy
TB_12	WTKiwmpYm2Uq	WTKiwmpYm2Uq	Correct
TB_13	1TLtkT1TETlF	00104k1nETlF	Incorrect
TB_14	TmgRBltnwYVS	TmgRBltnwYVS	Correct
TB_15	WTKiwmpYm2Uq	WTKiwmpYm2Uq	Correct
TB_16	m4I9cY8GsPTn	m4I9cY8GsPTn	Correct
TB_20	TmgRBltnwYVS	TmgRBltnwYVS	Correct
TB_21	2qGMTsNJpFbs	2qGMTsNJpFbs	Correct
TB_22	WTKiwmpYm2Uq	WTKiwmpYm2Uq	Correct
TB_23	WTKiwmpYm2Uq	WTKiwmpYm2Uq	Correct
TB_24	1TLtkT1TETlF	1TLtkT1TETlF	Correct
TB_25	m4I9cY8GsPTn	0000008asPnn	Incorrect
TB_26	m4I9cY8GsPTn	m4I9cY8GsPTn	Correct
TB_30	TmgRBltnwYVS	TmgRBltnwYVS	Correct
TB_31	2qGMTsNJpFbs	2qGMTsNJpFbs	Correct
TB_32	joeKugQ083oa	joeKugQ083oa	Correct
TB_33	WTKiwmpYm2Uq	WTKiwmpYm2Uq	Correct
TB_34	WTKiwmpYm2Uq	WTKiwmpYm2Uq	Correct
TB_35	m4I9cY8GsPTn	m4I9cY8GsPTn	Correct
TB_36	1TLtkT1TETlF	1TLtkT1TETlF	Correct
TB_40	TmgRBltnwYVS	TmgRBltnwYVS	Correct
TB_41	2qGMTsNJpFbs	2qGMTsNJpFbs	Correct
TB_42	joeKugQ083oa	joeKugQ083oa	Correct
TB_43	WTKiwmpYm2Uq	WTKiwmpYm2Uq	Correct
TB_44	WTKiwmpYm2Uq	WTKiwmpYm2Uq	Correct
TB_45	1TLtkT1TETlF	1TLtkT1TETlF	Correct
TB_46	m4I9cY8GsPTn	m4I9cY8GsPTn	Correct
TB_50	TmgRBltnwYVS	TmgRBltvMYVu	Incorrect
TB_51	2qGMTsNJpFbs	2qGMTsNJpFbs	Correct
TB_52	joeKugQ083oa	joeKugQ083oa	Correct
TB_53	WTKiwmpYm2Uq	WTKiwmpYm2Uq	Correct
TB_54	WTKiwmpYm2Uq	WTKiwmpYm2Uq	Correct
TB_55	m4I9cY8GsPTn	m4I9cY88sPTn	Incorrect

Image Name	Actual Value	Output	Accuracy
TB_56	1TLtkT1TETlF	1TLtkT1TETlF	Correct
TB_60	TmgRBltnwYVS	TmgRBltnwYVS	Correct
TB_61	2qGMTsNJpFbs	2qGMTsNJpFbs	Correct
TB_62	joeKugQ083oa	joeKugQ083oa	Correct
TB_63	joeKugQ083oa	joeKugQ083oa	Correct
TB_64	WTKiwmpYm2Uq	WTKiwmpYm2Uq	Correct
TB_65	WTKiwmpYm2Uq	WTKiwmpYm2Uq	Correct
TB_66	1TLtkT1TETlF	no value given	Incorrect

From this, it can be seen that the accuracy is $\frac{42}{49} \approx 85.7\%$. Thus, the probability of successfully reading a barcode in one try is $\frac{6}{7}$. This implies the expected amount of tries to successfully read the barcode is $\frac{7}{6} \approx 1.66$. This gives the expected time to successfully process a barcode to be

$$\frac{7}{6} \cdot 1.46 = 1.703 \text{ seconds}$$

4 Program

The program is given in the attached. Three files are included: the program without any code to assist in visualization; the program with the included visualization code; and a .zip file containing all the testing data.

5 Conclusion

In this article, we have established a fast and efficient method for reading a barcode and storing the value. This method maintains a high accuracy, allowing overall a short time to ensure success in reading.