International Journal of Computer Applications (0975 – 8887)
Volume 129 – No.7, November2015

Barcode Localization and Decoding using a Modified ScanLine Approach for Industrial Nameplate Verification

Aakash Thakkar
Sardar Patel Institute Of Technology
Mumbai, India.

ABSTRACT

Increased Automation in the manufacturing sector with the introduction of low cost embedded systems and servo drives has resulted in a substantial increase in the precision and speed of packaged goods output. This increase in speed has rendered manual verification incapable to keep up. Recent developments in the area of computer vision and embedded computing has allowed for replacement of traditional human verification by machine vision systems.

This study aims to implement a robust solution for Barcode Localization and Decoding for an affordable, reliable and industrial solution for a general purpose automated product verification system.

General Terms

Algorithms, Pattern Recognition, Barcode Localization, Barcode Decoding, Nameplate.

Keywords

Barcode; Decoding; Localization; Scanline; Nameplate; Verification

1. INTRODUCTION

The biggest challenge for any packaging unit is to ensure no mismatch takes place when switching over between two different products while using the same packaging equipment. A huge list of safety procedures need to be followed, as it would be disastrous if a face cream is filled into a tube that belongs to toothpaste. In addition to this, another important issue that needs to be addressed is the wrong printing of the batch/expiry details on the product packaging.

Working to eliminate the age-old adage that no amount of verification is enough, Product Name Plate Verification System makes use of modern vision technologies in order to ensure wrong packaging is the thing of the past. With the use of an industrial camera accompanied by a manual focus lens, high quality input can be captured and processed with the use of an embedded ARM system that is capable of identifying the product nameplate and communicating with the machinery using trigger/output and MODBUS communication mechanisms.

A Nameplate of the product can be defined as the unique quality of the product or product packaging that distinguishes it from other products, This Includes:

1) Barcode on the Product
2) Text on the Product
3) Color and Shape of the Product

2. SYSTEM FLOW

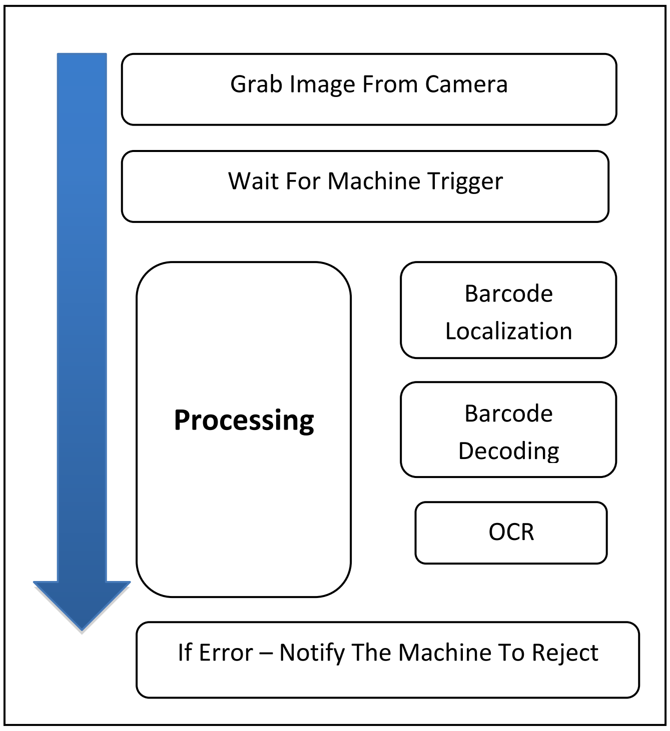


Fig 1. System Flow of Nameplate Verification System

The Fig. 1 maps the sequence of activities occurring in the process of nameplate verification. This process begins with the camera retrieving images at 30 Frames per Second (FPS). It also monitors the General Purpose Input Output (GPIO) pins for a trigger. Upon receiving the trigger it localizes & decodes the barcode, following which the OCR deciphers the text on the product being manufactured.

Any error signals the machine to reject the product, multiple errors signal the machine to halt packaging.

Let us understand in depth how the process mentioned in Figure 1. Are carried out.

3. BARCODE LOCALIZATION

Barcode Localization is the process of locating the exact coordinates of barcode in an input image so the task of barcode decoding can be performed.

A barcode is simply a sequence of parallel lines having rectangular shape of varying thickness. We can use this unique quality of barcode to localize its location in an image.

Let's consider our "Block Localization Algorithm" inspired by the work done by Douglas Chai and Florian Hock [3], on the following sample image to understand how we have solved this problem with a sample image:

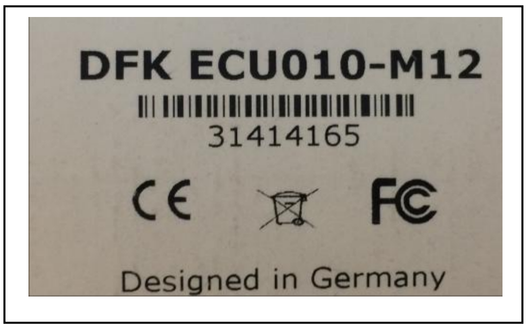


Fig 2. Sample Image

Step 1: Apply Adaptive Threshold on the image.

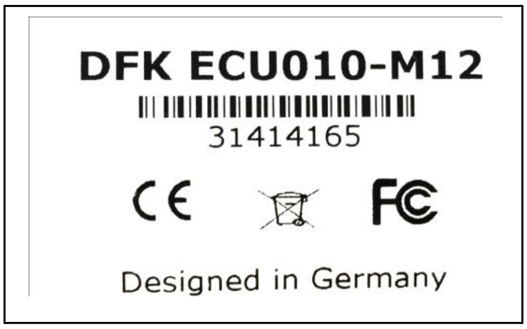


Fig 3. Sample Image After Performing Adaptive Threshold

Step 2: Divide the image into rectangular blocks of appropriate size depending on the image resolution.

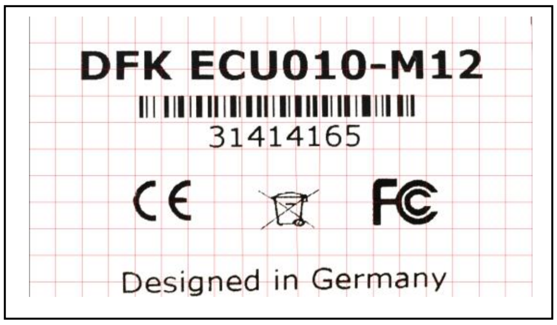


Fig 4. Division in rectangular block

Step 3: Check each block for following:

1. The block contains at least two contours/objects.
2. The objects can extend into another block but cannot extend beyond one neighboring block.
3. Anything below 10% of block size has to be ignored.

Step 4: Discard each block that doesn't satisfy the above. This gives us the following result:

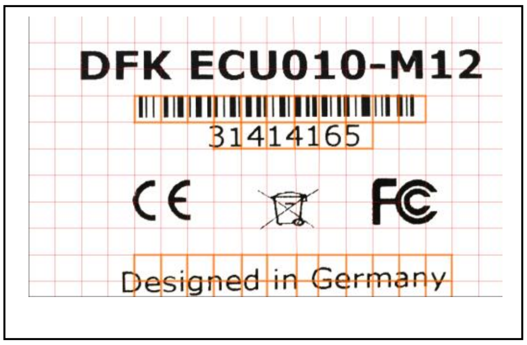


Fig 5. Discarding unwanted blocks

Blocks shown with an Orange Outline satisfy the above conditions.

Step 5: Use contour shape analysis on non-discarded blocks in Fig. 5 to ensure that each of the contours in the block is of rectangular shape, If not discard the block.

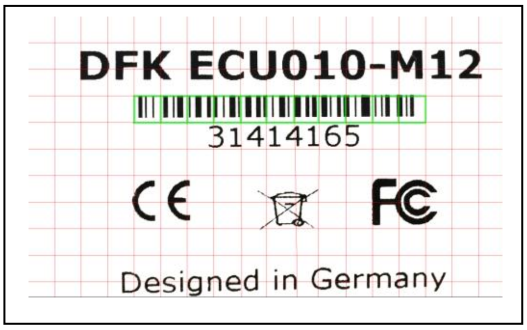


Fig 6. Discard blocks that fail Contour Shape Analysis

Step 6: The sequence of blocks shown with a green border in Fig 6 satisfying the above condition is the barcode. We can perform decoding on the same.

4. BARCODE DECODING

Different Barcode Symbologies have a different arrangement of spaces and bars and achieves a different business objective. No one type of symbology is perfect for all kinds of business applications.

The most common barcode symbologies are:

1) **Universal Product Code (UPC): Used in North America Retail.**

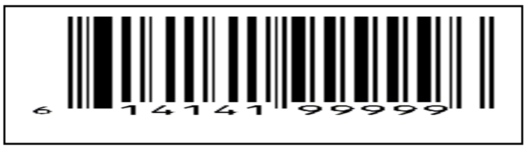


Fig 7. Universal Product Code(UPC) [7]

2) **International Article Number(EAN): Used in World Wide Retail.**

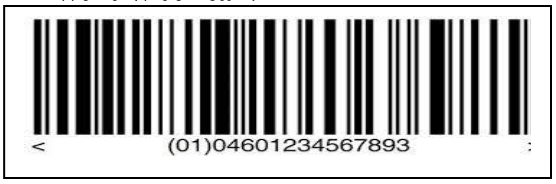


Fig 8. International Article Number(EAN) [8]

3) CodaBar: Used in Libraries, Blood Banks.

Fig 9.CodaBar [9]

4) PharmaCode: Pharmaceutical Packaging.

Fig 10. PharmaCode [10]

We will focus of Decoding PharmaCode as our product is aimed towards an industrial environment.

The specification for PharmaCode [11] specifies that the code must be of minimum three bars and a maximum of 8bars. It also recommends that each code must have at least one thick and fine bar.

Depending on the position of the thick or fine bar, weight is assigned to each bar. The summation of weights would give decoded value of PharmaCode. Example:

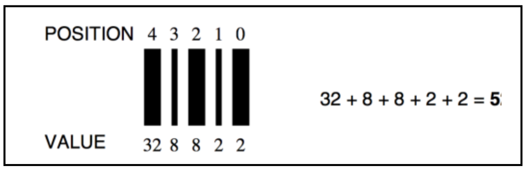
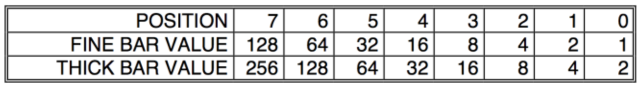


Fig 11. PharmaCode decoded [11]

Weights are assigned to each of the bar as following:



POSITION	7	6	5	4	3	2	1	0
FINE BAR VALUE	128	64	32	16	8	4	2	1
THICK BAR VALUE	256	128	64	32	16	8	4	2

Fig 12. Weights for each bar [11]

Now the main problem that arises in Barcode Decoding is how to identify the number and thickness of bars.

Let’s consider an example to understand this better:

One of the challenges with barcode decoding involves identifying the number and thickness of bars present in the barcode.

The decoding logic used by Orazio Gallo and Robert Manduchi [2] involves thresholding the values that comprise the ScanLine. However, this approach performs weakly in low lighting conditions as demonstrated in Fig. 14.a. and 14.b.

Fig 13 shows the presence of a reflection, which significantly distorts the barcode image hindering the decoding process. The pixel values from the ScanLine are plotted in Fig 14.a. The traditionally implemented decoding algorithm thresholds the pixel values and counts the data points that correspond to 0 value. Thresholding to zero is performed primarily to identify the presence of a bar and secondly distinguish between a thick and thin bar. As seen by the graph plotted, Fig 14.b we realize that the algorithm only detects 2bars while actually there are 6 bars, Also the amount of data points in each bar is very low hence thick bars are considered to be fine bars.

In order to overcome this issue, this paper proposes a “Baseline Decoding Algorithm”. This approach is based upon the hill-climbing algorithm commonly used in artificial intelligence. It involves tagging the local minima’s and maxima’s for a particular ScanLine. The average of all maxima’s excluding the first and the last maxima are calculated. From the second and second last minima, two points closest to the average are identified to serve as the local maxima on both end points.

Two subsequent maxima’s form a bar and the number of data points between the two maxima’s helps suggest the thickness of the bar. Clustering of these data points helps in classifying a bar as a thick or thin bar.

Figure 14.c proves that while the traditional algorithm [10] fails to correctly recognizing a distorted barcode, The Baseline Decoding algorithm not only correctly detects the number of bars but also distinguishes between the thick and fine bar accurately.

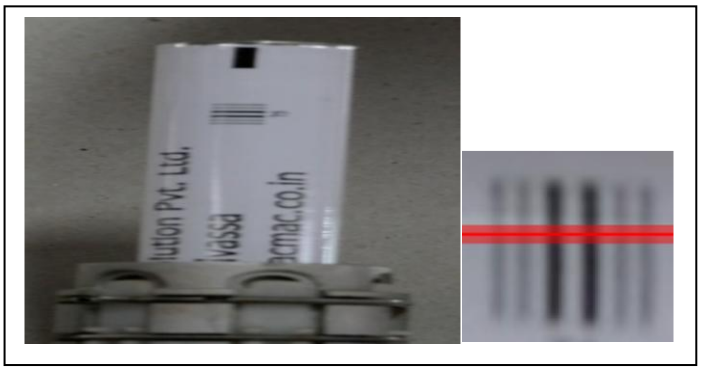


Fig 13. Real World Deformed Sample PharmaCode with corresponding ScanLine

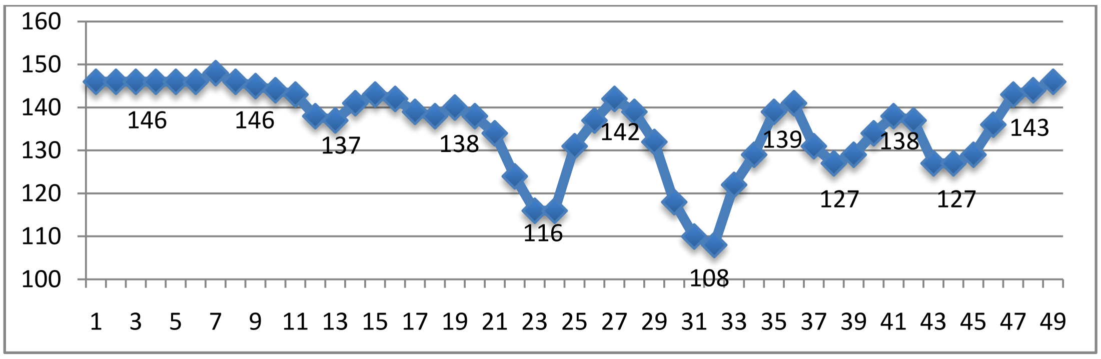


Fig. 14.a.

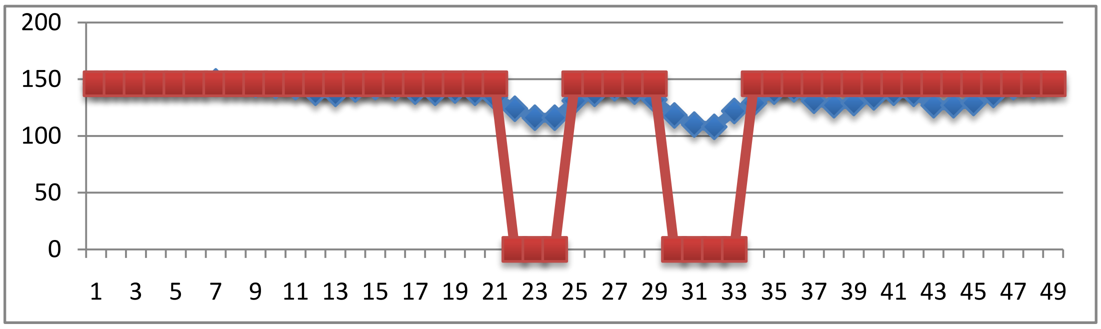


Fig. 14.b.

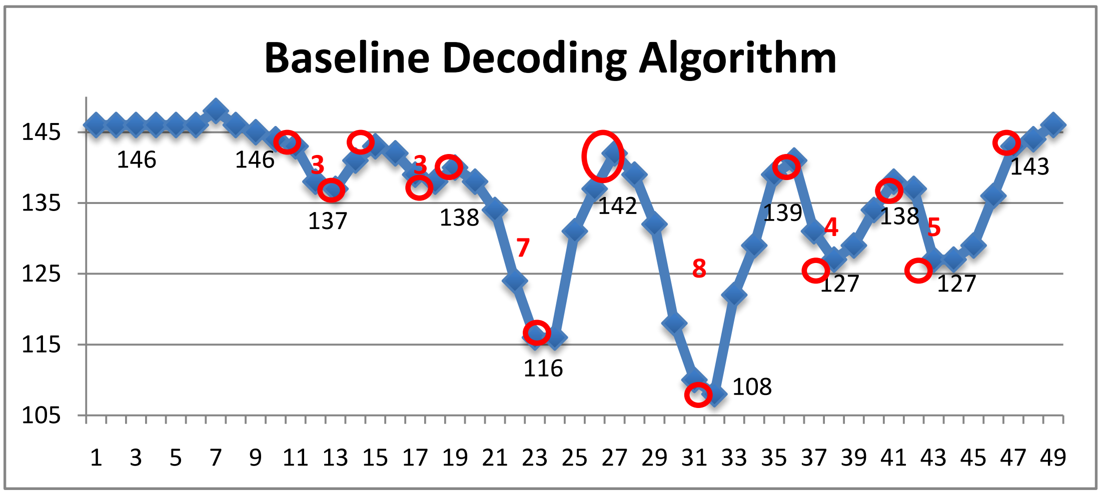


Fig. 14.c.

Fig 14. Plotting of Various PharmaCode Values, Fig 14.a. Plotting of Pixel Values, Fig 14.b. Plotting of Pixel Values along with Threshold Values, Fig 14.c Working of Baseline Decoding Algorithm.

5. CONCLUSION

The primary goal of Computing and Information Technology is to reduce human effort in repetitive work while enhancing the robustness and efficiency of the process.This paper attempts to accomplish the above by eliminating faults such as wrong and damaged packaging and misprinting with introduction of a low cost embedded solution and optimized image processing algorithms working reliably in real time.

The deployment of the above solution into an industrial environment would allow an operator to quickly teach the item being packaged, once the item is taught the product would accept triggers from the machine, and verify whether the parameters calculated meet the ones that are taught.

Hence this paper proposes a full proof solution to solve a common industrial problem and pushes upon the boundaries of image processing by increasing robustness and efficiency of current algorithms.

6. REFERENCES

[1] Pavlidis, T.; Swartz, J.; Wang, Y.P., "Fundamentals of bar code information theory," IEEE Computer, vol.23, no.4, pp.74, 86, April 1990

[2] Orazio Gallo and Robert Manduchi, "Reading 1D Barcodes with Mobile Phones Using Deformable Templates", IEEE transactions on pattern analysis and machine intelligence, Vol. 33, 2011

[3] Douglas Chai and Florian Hock, "Locating and Decoding EAN-13 Barcodes from Images Captured by Digital Cameras", IEEE, pp. 1556-1560, 2005

[4] http://www.barcoding.com/information/barcode-symbology-definition.shtml, accessed on 24th February 2015

[5] http://docs.opencv.org/trunk/doc/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html, accessed on 4th February 2015

[6] http://docs.opencv.org/trunk/doc/py_tutorials/py_imgproc/py_contours/py_contours_begin/py_contours_begin.html, accessed on 24th February 2015

[7] http://www.gs1us.org/Portals/0/images/barcodes/UPC-A.gif, accessed on 2nd March 2015

[8] https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcQWqT075ZAWtiOV-xmAmH34rPL6z3SCTOqrUJYc_Cdpxahn2Hea, accessed on 2nd March 2015

[9] http://www.barcodeisland.com/codabar.gif, accessed on 2nd March 2015

[10] http://wikiimages.qwika.com/images/en/d/da/Pharm.gif, accessed on 2nd March 2015

[11] http://www.oberhofer.co/how-barcode-localization-works-in-quaggajs/, accessed on 22nd February 2015