Acta Cybernetica **21** (2013) 21–35.

Barcode Detection Using Local Analysis, Mathematical Morphology, and Clustering*

Péter Bodnár† and László G. Nyúl†

Abstract

Barcode detection is required in a wide range of real-life applications. Imaging conditions and techniques vary considerably and each application has its own requirements for detection speed and accuracy. In our earlier works we built barcode detectors using morphological operations and uniform partitioning with several approaches and showed their behaviour on a set of test images. In this work, those ideas have been extended with clustering, contrast measuring, distance transformation and probabilistic Hough transformation. Using more than one feature for localization leads to better accuracy, which makes detectors based on simple features, a competitive solution for commercial softwares and helps to fulfill the requirements of industrial applications even more.

Keywords: barcode detection, computer vision, clustering, feature extraction, morphological filters, distance transformation, Hough transformation

1 Introduction

Barcodes are 1D codes that consist of a well-defined group of parallel lines aiming easy automatic identification of carried data with endpoint devices such as PoS terminals, smartphones, or computers. Barcode decoding is fast and most barcode standards provide redundant information for error correction purposes. 2D codes are also referred to as barcodes, but in this paper, only codes in Fig. 1 are discussed.

Barcode localization methods have two main objectives, speed and accuracy. On smartphones, fast detection of barcodes is desirable, but accuracy is not so critical since the user can easily reposition the camera and repeat the scan. Accuracy is critical for industrial environment (e.g. postal services), where false negatives cause loss of profit. Speed is also a secondary desired property in those applications.

*This work was supported by the European Union and co-financed by the European Regional Development Fund within the project TÁMOP-4.2.1/B-09/1/KONV-2010-0005.

†Department of Image Processing and Computer Graphics, University of Szeged, E-mail: {bodnaar,nyul}@inf.u-szeged.hu

DOI: 10.14232/actacyb.21.1.2013.3

(a) Code 128 (b) EAN-13 (c) UPC (d) UPC-A

Figure 1: Barcode patterns

This is an important problem because the localization step is probably the most difficult part of general barcode detection and once localization is completed, decoding the barcode is relatively straightforward.

For 1D barcodes, the basic approach for localization is scanning only one, or just a couple of lines of the whole image. This is common for hand-held PoS laser scanners or smartphone applications. Scanned lines form an 1D intensity profile, and barcode-detector algorithms [1, 11, 16] work on those profiles to find an ideal binary function that represents the original encoded data. The main idea is to find peak locations in blurry barcode models, then thresholding the intensity profile adaptively to produce binary values.

Valley tracing (or bar tracing) [16] is a method for finding barcodes in blurry, low resolution images, mostly on live smartphone camera frames. It consists of three steps. At first, starting points have to be found on the picture, then "valleys" are followed, and finally, ends of the valleys (bars) are reached.

Algorithms with morphology [3, 4, 9, 13, 14] use the combination of basic morphological operation like erosion and dilation. White blobs on those images (Fig. 3) show the possible barcode locations. Further processing, like segmentation and filtering of small blobs are required on those difference images. It can be used on both 1D and 2D barcodes. Our work also involves morphology for efficient barcode localization.

Methods based on wavelet transformation [15] look at images for barcode-like appearance by a cascaded set of weak classifiers. Each classifier working in the wavelet domain narrows down the possible set of barcodes, decreasing the number of false positives while trying to keep the highest possible accuracy.

Variants of Hough transformation [2] detect barcodes by working on the edge map of the image. The two most common methods are standard and probabilistic Hough transformation. Both transform edge points into Hough space first, and make decisions of line locations. We are also experimenting with the idea of probabilistic Hough transformation extended with decisions about the features by projections of the Hough-space.

In the following section, simple features are presented to track for barcode localization. They are based on Hough-transformation, morphological operations, and uniform partitioning with distance transformation, contrast measuring and clustering.

2 Proposed Barcode Localization Approaches

In this section several different approaches are introduced for barcode localization. In most cases, image is examined in small, disjoint tiles (see Fig. 5 for optimal tile size), and local measurements are made. In this paper distance transformation [8] and contrast variance is used for these measurements. Code parts, like other textures, have well-traceable features. One of them is neighbor similarity, which means code parts in close proximity share similar local statistics with a well-chosen tile size. Thanks to the repeating patterns in the codes codes can be localized by observing how many code-like image parts (tiles) can be found. Finally, compactness of code parts is also important, which influences the final decision about code localization. Contrast information of the tiles and the number of clusters at pixel level are also examined.

With probabilistic Hough transformation, single lines of barcode can be detected for further processing. Clustering those lines helps to decide if a line is part of a barcode.

The way of clustering can also be applied at pixel level. A measure can be introduced with the help of the number and shape of pixel clusters.

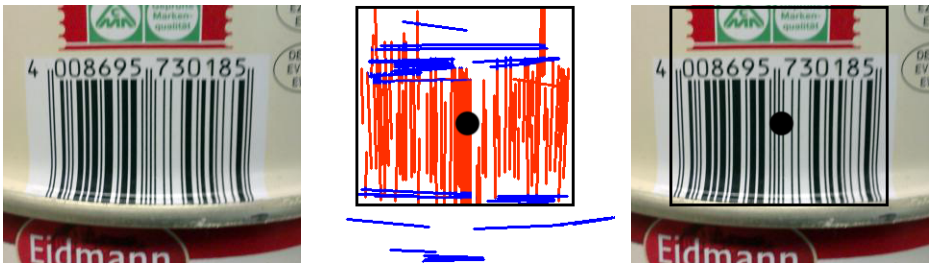
Lastly, with morphological operations, processing also leads to high accuracy. Features extracted with basic morphological operations, form a considerable base of more complex barcode localization algorithms.

2.1 Preprocessing

Digital images acquired from a camera often need preprocessing because of device flaws or environmental difficulties. In images having low contrast, intensity levels should be normalized. Unsharp masking is used in this paper, which is the weighted addition of the original image pixel intensities to the inverted pixel values of the gaussian-blurred version of the image. The blurring Gaussian filter is adjusted to not to destroy the narrowest line of the barcode, which parameter can be estimated by the specific endpoint application. Since some features need to be extracted from binary images, thresholding is necessary. A simple threshold is sufficient on images with uniform lighting, otherwise adaptive threshold [17] is required.

Image resolution does not have to be high, barcodes having the narrowest line of two pixels (px is used for pixels later in this work) is sufficient (3×3 px median filters can be applied to eliminate salt-and-pepper noise). Higher resolution yields better results, but also increases computation time. The least time-consuming solution for downsampling such images is the nearest neighbor interpolation, which is also a good choice because it preserves strong edges. However, at least 3 px minimum line width is desired for accurate code detection.

Color information could also be taken into account, however, most visual codes maximize the contrast by using only black (or dark blue) and white colors. Furthermore, industry hardware are often set to operate and record only in specific frequency ranges. According to these, only intensity values are processed and color information is dropped.



(a) original image (b) feature image (c) overlay

Figure 2: Canny edge detector with Probabilistic Hough transform. In (b), detected lines that are part of a barcode-like cluster are shown in red while the other detected lines are shown in blue

2.2 The Canny + Hough Method

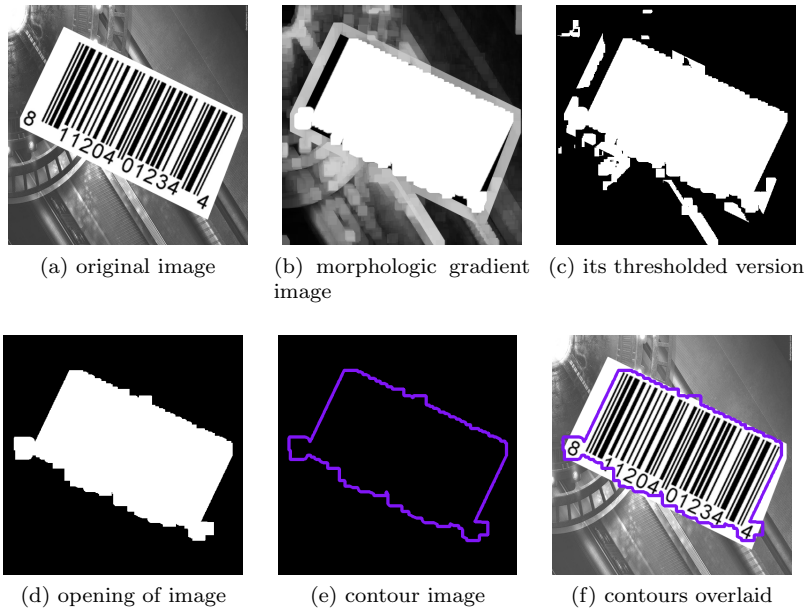
This method applies general image processing methods like Canny [6] edge detection and Probabilistic Hough transform [12], as barcodes consist of roughly equally long, parallel lines in a small area. With the help of the edge points, it gives a probabilistic estimation for line segments in the image, thus outperforming the standard Hough [18] transform, which only gives a set of lines as result. For preprocessing, a blur filter is recommended, since smooth images are desired for Canny edge detector. Since all barcodes in the test suite (see Subsection 3.1) have at least 50 px bar height, we set the minimum line length to 40 in the Hough transformation.

After a list of lines with their center point is obtained, length, and orientation, we can group them to decide whether they constitue a barcode or not. The minimum number of lines, the proximity needed for the lines to be in the same group, and the tolerance for length and orientation from the means inside the group vary for the final application. Since our barcodes consist of at least 25 parallel lines, we defined the minimum number of lines as 20. In the final step, group centers are returned, and the image can be cropped for decoding with known barcode decoding implementations (Fig. 2).

2.3 MIN–MAX

There are several works in the state of the art that use binary or grayscale morphology, see e. g. [10]. The two basic operations, erosion and dilation, and more complex operators such as bottom-hat are sufficiently robust and show good accuracy for barcode-like feature extraction.

The approach labeled as MIN–MAX treats the image as a whole, and therefore requires a fair amount of RAM and computation time. Supposed that intensity levels have already been normalized, no other preprocessing operations are required since this method manages well noisy, blurry or distorted images. Knowing the maximum bar width of a barcode, the morphologic gradient (`dilate`() − `erode`())



(a) original image (b) morphologic gradient image (c) its thresholded version

(d) opening of image (e) contour image (f) contours overlaid

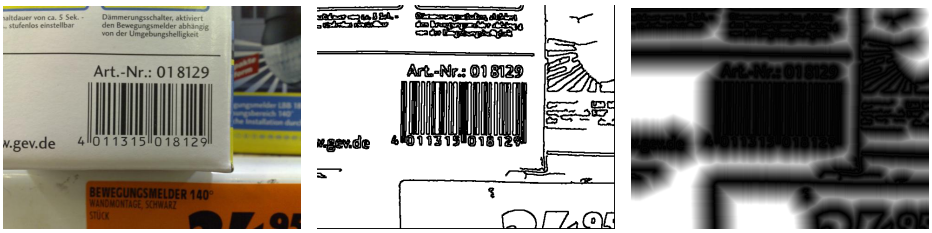
Figure 3: Stages of MIN–MAX method

operator is applied on the image with a box kernel of size $2 \times \lfloor \texttt{max\_barwidth}/2 \rfloor + 1$ (Fig. 3b). The next step is removing components that consist pixels with low intensity or being small. For component removal by intensity, the simplest way is thresholding (Fig. 3c). A good threshold can be at 75 % of the full intensity scale (e.g. 192 for 8-bit grayscale images), since barcodes produce areas close to the maximum intensity. After that, morphologic opening operation `(dilate(erode()))` is performed on the feature image (Fig. 3d), with the previously defined kernel. This ensures filtering out components that are too small or thin. Contours are extracted to the final step, which is to give a bounding box for image parts that might contain barcode (Fig. 3e).

Filtering of small areas can also be performed by connected component analysis. Each component size is measured and ones having smaller area than the defined minimum, are dropped. Experiments showed that setting the minimum component area to $0.75wh$ or smaller is satisfactory (where w and h are barcode width and height respectively).

2.4 Uniform Partitioning with Distance Transformation

Most barcodes, like regular textures, can be easily identified by observing only small parts of them. Those barcode parts together form the desired barcode region with known height and width. The first part of the method is partitioning the image to square tiles and look at each tile for barcode-like appearance. Each tile is



(a) original image (b) edge map (c) distance map

Figure 4: Canny edge map (b) and distance map (c) of a real-life example image (a). Barcodes have compact dark areas. Note: the values in the distance map are scaled for visualization.

assigned a value that indicates the grade of the presence of this feature. Globally, a matrix is formed from those values. Texture parts have similar local statistics in their neighborhood, so searching this matrix for compact areas defines image ROIs representing a barcode with high likelihood.

The assigned value showing barcode-like appearance is based on distance transformation of the edge map (Fig. 4). Canny edge detector is applied for providing the point set to the transformation. For each tile of the distance map, means and standard deviations are calculated. For 1D codes, distance values spread between half of the minimum and half of the maximum line width.

After evaluating all tiles locally, clusters are looked up in the feature matrix. Experiments show that the mean of distance values spread around half of the average line width. Thresholding is performed for the values to classify whether or not an area contains a barcode segment. Letting 25 % tolerance for these idealistic values, detection accuracy becomes satisfactory. For end-user applications we have to take noise, scratches and reflections into consideration. For images containing heavy noise, distance means drop. Barcodes suffering from scratches, dust, handwriting or reflections, change the distance means significantly according to the dark or bright intensity values of the flaw. Tolerance should be set according to amounts of these distracting properties and exact values can also be measured via trial and error. Tolerance value is a compromise between accuracy and the rate of false positive detections, so it should be set with respect to the final application.

The resulting binary matrix can be further analized via connected component labelling [7]. Finally, small components are dropped, and momentums of the remaining components are returned. A component is considered small if it contains less than N tiles (Eq. (1)), where h is barcode height, w is barcode width and s is the tile size, respectively.

$$N = \max\left(4, \frac{|h-s| \times |w-s|}{s^2}\right) \tag{1}$$

Since the smallest barcode in the test suite (see Subsection 3.1) has a 60 px

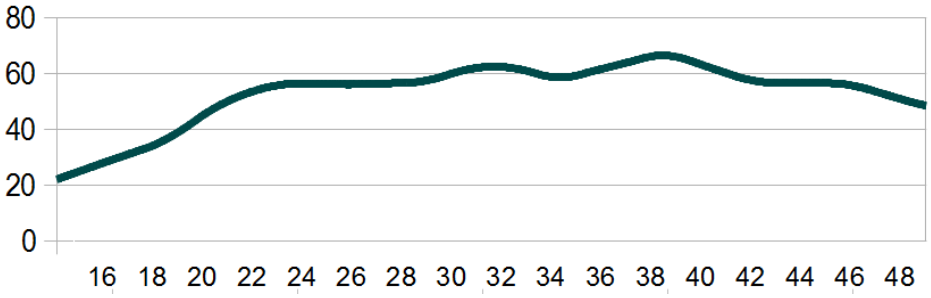


Figure 5: Detection accuracy with respect to the tile size. X-axis: proportion of tile size and barcode height; Y-axis: detection accuracy (both expressed in percent).

height, 30×30 px or greater tile sizes have poor recognition capability. However, very small tile sizes also lead to larger error for computing the center of the codes, because of the characters appearing below the code with code pieces nearby also have a barcode-like property (plain text is not affected). Also, choosing the tile size below two times the width of the widest barcode line leads to poor accuracy, since only two clusters can be detected on the tile, and that does not characterize a barcode part well. The best tiling size appears to be about 1/3 of the barcode height (Fig. 5). Since all examined codes consist of the same pattern (parallel lines), we looked in this paper for the optimal tile size for all types of codes together.

It is possible to run this method with disjunct or overlapping tiles, but overlapping does not improve the method's accuracy, only increases computation time. Offsetting the tiles has no significant effect, since it just produces some blocks to be more and others to be less "barcode-like" at the barcode's opposite sides.

This method gives a moderate rate of false positives (Fig. 6). Text with a comparable font size to barcode line thickness reacts similar to distance transformation, and distance values have similar mean and standard deviance in both cases. However, distance transformation can be used as a weak "classifier" of image areas, and its output is a good starting point for more accurate methods. The protocol to test optimal parameters is highly experimental, however, in end-user applications, expected element size or bar width of the barcode can be expected within a certain range. In case if the approximate distance between the camera and its observed plane of interest, object type (letter, box having dimensions in a limited range), and code dimensions are known, code size can be estimated in the acquired image. The intensity and characteristics of illumination is also limited in most cases, especially in industrial environment.

2.5 Contrast Measuring with Uniform Partitioning

Using the approach of image tiling, contrast information is examined for each tile. One-dimensional barcodes usually have high variance in contrast at one specific direction, while having low variance perpencidularly to that. The discussed rules

(a) original image (b) distance map (c) detected rectangular ROIs

Figure 6: Real-life example of a bar-coded product case with non-uniform illumination. Original image (a), distance transformation (b), and the detected rectangular ROIs based on the distance map (c)

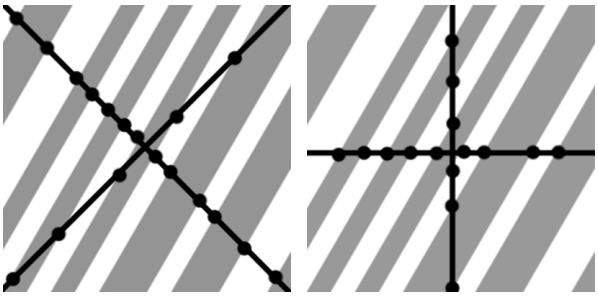
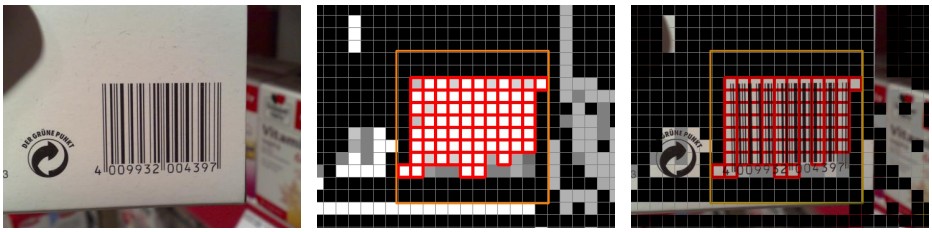


Figure 7: Two pairs of scanlines sweep through the image. One gives significant difference in contrast variance between perpendicular directions. The first pair of scanlines shows well barcode-like property of image tile, while the other one does not.

of tiling and forming the final decision also apply to the approach of contrast measurement, only the value of each tile is assigned differently.

Each tile is checked locally for barcode-like appearance with a modified version of the scan-line analisys. Two pairs of perpendicular scanlines are used, one pair at 0° and 90° and the other at 45° and 135° (Fig. 7). A measure is derived from contrast variance along the scanlines (Eq. (2)), i.e., a horizontally aligned barcode has a lot of contrast changes when scanned with a horizontal scanline, but has few or none in case of a vertical scan (Fig. 7). With the 2 pairs of scanlines, barcode pieces of any orientation can be safely recognized. The final measure assigned to a tile is the maximum of the two values. This measure gives 1 if parallel lines are present on an image tile, and 0 if a tile contains homogeneous area or noise.

$$C_i = \frac{|V_{i1} - V_{i2}|}{\max(V_{i1}, V_{i2})} \tag{2}$$



(a) Original image (b) feature image with bounding box (c) overlay

Figure 8: Contrast measuring on a real-life example.

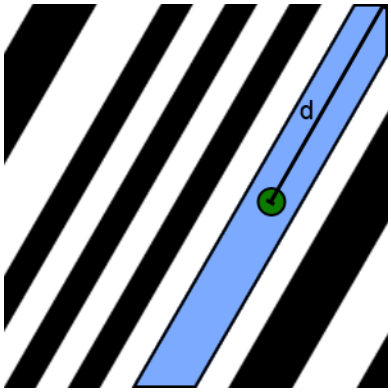


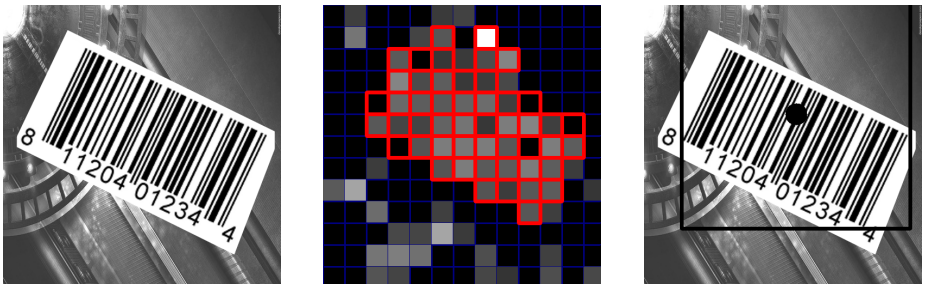
Figure 9: The idea of Local clustering. Here d is the maximum distance from the cluster center, i.e. the half of the cluster diameter

where V_{ij} is the contrast variance along a scanline j in a scanline pair i.

The rest of the method is the same as it is presented at Section 2.4.

2.6 Local Clustering

The main idea of local clustering is that an image region that contains a barcode segment has many similar stretched pixel clusters (Fig. 9). The minimum count of expected clusters can be derived from the widest bar of the barcode. Degree of stretch can be measured with the diameter of the cluster (defined as twice the distance of the furthest cluster point from the cluster center). For exactly horizontal or vertical lines, the largest cluster diameter is the tile size, in oblique situations, the largest cluster diameter is expected to be longer than that. Furthermore, stretched separate clusters need to be aligned approximately in the same direction, otherwise one cluster would touch another, thereby decreasing the number of separate clusters in a tile below our threshold. On real-life images that have low contrast at barcode areas, adaptive thresholding is necessary.



(a) original image

(b) feature matrix visualized as gray values. Red squares are above threshold.

(c) code center and bounding box

Figure 10: Stages of Local clustering

Another important property is the minimum cluster size measured in pixels. This can be easily computed from `min_barwidth` × `tile_size`.

After assigning the value to each tile, the same rule as in Section 2.4, applies for making the final decision, like dropping small groups of tiles and analyzing connected components.

Bounding boxes in our examples do not enclose the whole barcode in every case. This is because only the lower and upper bounds for the clusters in the feature matrix are calculated, and barcode corner pieces are too weak to signify a feature. The bounding boxes can be simply improved by finding aligned rectangles instead.

Running the method on the same scenario with different offsets yield different detection accuracy, which shows that this approach is sensitive to the choice of tiling. Further investigations are currently in progress on how to select the best tiling offset, as well as possible setups with overlapping tiles (which obviously increases computational requirements). This 2-phase approach works as follows. In the 1st phase, Local clustering is performed with zero-offset tiling, and in a 2nd phase the same step is done using an offset of half the tile size in both directions. The code centers detected by the 2 phases are pooled together with an extra filtering wherein those code centers that are detected in both phases and are close to each other are merged into one (the larger cluster is kept), because they are likely to correspond to the same code.

3 Evaluation

The discussed methods were tested on 100 images and accuracy has been compared to known barcode detection techniques, like the one based on bottom-hat filter [9], and another using morphological gradient [16].

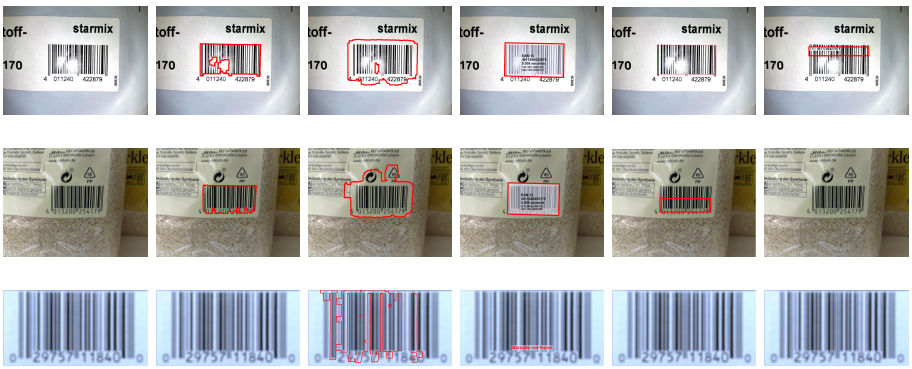


Figure 11: A qualitative test of barcode detectors from the state of the art and the market. From left to right: original image, Tuinstra et al. [16], Juett et al. [9], DataSymbol software, DTK barcode reader, BC tester. Each row represents a different test image. Results show that Datasymbol has the most robust localization approach, while BCTester had very poor accuracy even with clean and sharp images. The Tuinstra et al. and Juett et al. algorithms worked reasonably well, both has performed at least on the level of commercial softwares.

3.1 Test Suite

Since we have not found many official barcode detection test image databases, about 100 images of grocery product barcodes are made, with a Nokia N95 smartphone camera. Images has been downsampled to 640×480 px with bilinear interpolation. Minor reflections, blur, scratches and distortions were present in these images. We also found one barcode image database for comparative assessment, which was created by Tekin and Coughlan[1]. Ground truth to those images had been made manually, without marking the "quiet" zones and the digits that belong to the code.

There are several barcode detection software and frameworks on the market, like the DTK Barcode Reader SDK[2], BC Tester[3] and Barcode Recognition SDK of DataSymbol[4]. They do not indicate the applied algorithm behind their detection mechanism, however, a brief qualitative comparison has been made and results are shown in Fig. 11, and our results are shown in Fig. 12.

3.2 Implementation and Test Environment

The core of the proposed method has been implemented in `C++`, with the help of the `OpenCV` library. `C++` provides convenient object oriented approach and fast code execution, while `OpenCV` has all the functions needed for image preprocessing

[1] http://www.ski.org/Rehab/Coughlan_lab/Barcode
[2] http://www.dtksoft.com/
[3] http://www.bctester.de/
[4] http://www.datasymbol.com/

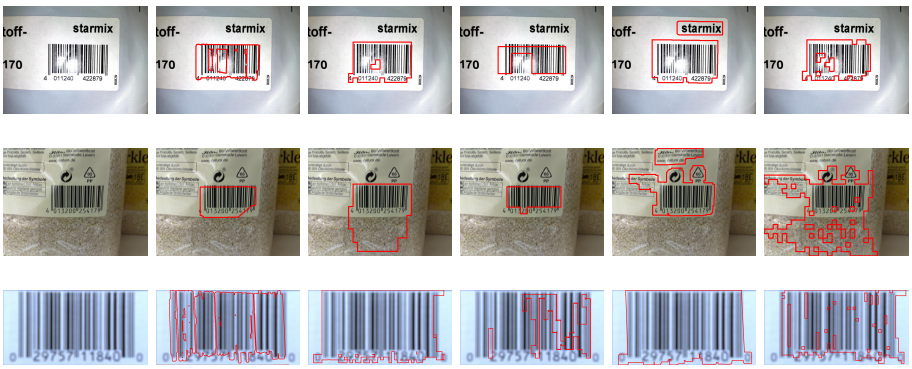


Figure 12: A qualitative test of barcode detectors based on our features. Contours of the possible barcodes are shown in red. From left to right: original image, Hough-transformation, distance transformation, local clustering, MIN–MAX, contrast measuring. Each row represents a different test image. Hough transformation is more tolerant to blur than noise, while MIN–MAX seemed to be the most robust approach. Distance transformation and local clustering also performed well, however, they showed a moderate amount of false-positives.

and manipulation. Evaluation is performed on a computer with Intel Core 2 Duo 3.00GHz CPU.

3.3 Accuracy and Detection Speed

For comparing the effectiveness of the proposed method, the most common measures like precision, recall, accuracy and F-measure (Eq. (3)) are used. The values are based on the Jaccard index (Eq. (4))

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{3}$$

$$J(G, D) = \frac{\sum_{x,y}(G(x, y) \wedge (D(x, y))}{\sum_{x,y}(G(x, y) \vee (D(x, y))} \tag{4}$$

where G and D give binary 0–1 values based on the pixel intensity of the ground truth and the detector output images (both are binary).

The average performance indicators of the detectors are shown in Table 1. Ensemble efficiency of these features are presented in [5].

Distance transformation is better to be used as a weak "classifier" instead of on its own. It produces the highest amount of false positives, however, it comes with high recall. It is more like an exclusion filter for image parts than a detector.

The Probabilistic Hough method is a robust choice to localize barcodes, because it can be parametrized to minimum line length and maximum gap between line

Table 1: Average detection performance of the proposed method

Algorithm	Precision	Recall	Accuracy	F-measure	Runtime
Tuinstra et al. [16]	57.08 %	85.29 %	84.19 %	48.39 %	160 ms
Juett et al. [9]	34.26 %	94.08 %	72.76 %	36.13 %	230 ms
Hough trans.	64.83 %	85.07 %	84.22 %	58.76 %	230 ms
Distance trans.	20.00 %	95.85 %	54.52 %	23.54 %	190 ms
Local clustering	81.68 %	72.34 %	89.22 %	62.12 %	120 ms
MIN–MAX	43.36%	85.38 %	77.47 %	36.82 %	360 ms
Contrast measuring	51.17 %	88.02 %	82.58 %	49.07 %	140 ms

segments. It also handles noise well to a certain level, but it is quite sensitive to distortions.

Thanks to the nature of the Distance Transformation and Local Clustering methods, they are reliable on images with minor distortions, unlike the Hough transformation, which detects barcodes based on line angles.

The least sensitive method is MIN–MAX. Because of the morphological approach, it handles well noise, blur and distortions up to a relatively high level. However, the convolutions used in the steps of the algorithm make it realtively slow.

Partitioning the image, assigning a measure to each partition, and looking for homogenous areas in the feature image is a general approach to detect patterns. With different features, like contrast variance, histogram information or distance information, it can be used well as a barcode locator method.

4 Concluding Remarks

Several features have been presented for barcode localization in raster images using various features. We studied their behavior on a set of images showing different barcode types.

In industrial setups, processing of the image tiles and parallel execution of different methods may also be possible for further improve detection speed. Furthermore, intermediate feature data, like edge map can be used as input for other, more accurate classifiers discussed in the first section.

An ensemble of detectors specially devised for certain code types can significantly improve the overall accuracy.

References

[1] Adelmann, Robert. Toolkit for bar code recognition and resolving on camera. In *Phones – Jump Starting the Internet of Things. In: Informatik 2006 workshop on Mobile and Embedded Interactive Systems*, 2006.

[2] Ballard, D.H. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[3] Bodnár, Péter and Nyúl, László G. Barcode detection with uniform partitioning and morphological operations. In *Conference of PhD Students in Computer Science, Proceedings of Conference*, pages 4–5, 2012.

[4] Bodnár, Péter and Nyúl, László G. Efficient barcode detection with texture analysis. In *Signal Processing, Pattern Recognition, and Applications, Proceedings of the Ninth IASTED International Conference on*, pages 51–57, 2012.

[5] Bodnár, Péter and Nyúl, László G. Improving barcode detection with combination of simple detectors. In *The 8th International Conference on Signal Image Technology (SITIS 2012)*, 2012.

[6] Canny, John. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, nov. 1986.

[7] Dillencourt, Michael B., Samet, Hannan, and Tamminen, Markku. A general approach to connected-component labeling for arbitrary image representations. *J. ACM*, 39:253–280, April 1992.

[8] Felzenszwalb, Pedro F. and Huttenlocher, Daniel P. Distance transforms of sampled functions. Technical report, Cornell Computing and Information Science, 2004.

[9] James Juett, Xiaojun Qi. Barcode localization using bottom-hat filter. *NSF Research Experience for Undergraduates*, 2005.

[10] Jean Paul Serra, Pierre Soille, editor. *Mathematical morphology and its applications to image processing.* Kluwer Academic, 1994.

[11] Joseph, Eugene and Pavlidis, Theo. Bar code waveform recognition using peak locations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):630–640, 1994.

[12] Kiryati, Nahum, Eldar, Yuval, and Bruckstein, Alfred M. A probabilistic hough transform. *Pattern Recognition*, 24(4):303–316, 1991.

[13] Lin, Daw-Tung and Lin, Chin-Lin. Multi-symbology and multiple 1d/2d barcodes extraction framework. In Lee, Kuo-Tien, Tsai, Wen-Hsiang, Liao, Hong-Yuan, Chen, Tsuhan, Hsieh, Jun-Wei, and Tseng, Chien-Cheng, editors, *Advances in Multimedia Modeling*, volume 6524 of *Lecture Notes in Computer Science*, pages 401–410. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-17829-0-38.

[14] Lin, Daw-Tung, Lin, Min-Chueh, and Huang, Kai-Yung. Real-time automatic recognition of omnidirectional multiple barcodes and dsp implementation. *Machine Vision and Applications*, 22:409–419, 2011. 10.1007/s00138-010-0299-3.

[15] Oktem, R. Bar code localization in wavelet domain by using binary morphology. In *Signal Processing and Communications Applications Conference, 2004. Proceedings of the IEEE 12th*, pages 499–501, april 2004.

[16] Tuinstra, Timothy R. *Reading Barcodes from Digital Imagery.* PhD thesis, Cedarville University, 2006.

[17] Wu, Sue and Amin, Adnan. Automatic thresholding of gray-level using multistage approach. In *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on*, pages 493–497 vol.1, 2003.

[18] Youssef, Sherin M. and Salem, Rana M. Automated barcode recognition for smart identification and inspection automation. *Expert Systems with Applications*, 33(4):968–977, 2007.