

## Improved 1D and 2D barcode detection with morphological operations<sup>\*</sup>

Melinda Katona and László G. Nyúl

Department of Image Processing and Computer Graphics  
University of Szeged  
Árpád tér 2, H-6720 Szeged, Hungary  
katona.melinda@stud.u-szeged.hu, nyul@inf.u-szeged.hu

**Abstract.** Barcode technology is the pillar of automatic identification, that is used in a wide range of real-time applications with various types of codes. The different types of codes and applications impose special problems, so there is a continuous need for solutions with improved effectiveness. There are several methods for barcode localization, that are well characterized by accuracy and speed. Particularly, high-speed processing places need automatic barcode localization, e.g. conveyor belts, automated production, where missed detections cause loss of profit. In this paper, we mainly deal with segmentation of images with 1D barcode, but also analyze the operation of different methods for 2D barcode images as well. Our goal is to detect automatically, rapidly and accurately the barcode location by the help of extracted features. We compare some published method from the literature, which basically rely on the contrast between the background and the shape that represent the code. We also propose a novel algorithm, that outperforms the others in both accuracy and efficiency, and also improves on the detection performance of our previous algorithm in detecting 2D codes.

### 1 Introduction

Barcode detection is required in a wide range of real-life applications. Computer vision algorithms vary considerably and each application has its own requirements for accuracy and detection speed. Many barcode localization methods have been developed for automatically segmenting barcode patterns from images.

The term barcode can be used for various types of visual codes. In this paper, we deal with classical 1D barcodes and stacked 2D barcodes segmentation. Barcodes are not human-readable and traditional devices have been widely adopted for personal use. The traditional barcode structure is simple: the variation of different thickness of parallel light and dark bars represent information. Such

---

<sup>\*</sup> Part of this work has been published under Katona, M., Nyúl, L.G.: A novel method for accurate and efficient barcode detection with morphological operations. In the proceedings of The 8th International Conference on Signal Image Technology (SITIS 2012).



**Fig. 1.** Barcode patterns (from left to right). Top row (1D codes): I2of5, Code39, Codabar, Code128; Middle row (1D codes): UPC-A, UPC-E, EAN-13, EAN-8; Bottom row (2D codes): Codablock, PDF417, Data Matrix, QR.

codes can be read optically by a machine. The code types vary from each other in what black and white bars correspond to a given character. The most frequent application of barcodes is the trade, e.g. in goods packing, where it permits fast identification of goods data such as manufacturer's country, manufacturer's identification number and the product's item number. The barcode identification is the most known element of the GS1 system, which is an inseparable part of trading procedures. The identification number visualized with symbols permit the use of electronic reading by machines which support and speed up the information streaming. Barcodes can have fix or variable length. In fix length codes, the standard specifies how many characters are represented in a code, while the other type may encode an arbitrary number of characters. The different standard code types have specific features that help their localization. Figure 1 shows a set of standard 1D barcode types and some widely used 2D barcode types. [1]

Barcode localization methods have two objectives, speed and accuracy. For industrial environment, accuracy is crucial since undetected (missed) codes may lead to loss of profit. Processing speed is a secondary desired property of the detectors. On smartphones, the accuracy is not so critical, since the device interacts with the user and re-shoting is easily possible, but a fast (and reasonably accurate) barcode detection is desirable. Various techniques are used to locate and decode barcodes from photographs, from the classical line scanning technique [2], through the widely studied morphological approaches [3–11], and recent studies using wavelets [12].

We propose a novel barcode detection algorithm based on bottom-hat filtering and other morphological operations, which ensure high efficiency. We compare the effectiveness of the new method with several approaches from the literature and can show that, in most cases, our algorithm outperforms the others.

It also improves on the detection accuracy of our previous algorithm [11] for 2D stacked barcodes.

## 2 Methods

In this section, we present five barcode detection algorithms. They use different approaches to determine the barcode location. In the pseudocode we use the notation of [13]. The methods presented in Sections 2.1, 2.2, and 2.3 were re-implemented according to the original papers. Where some details were not available in the papers, we made our best effort to fill in the gaps and this is noted in the text. The new proposed algorithm is presented in full detail in Section 2.4, and in Section 2.5 we outline the differences from the previously published version [11]. Throughout the paper we shall refer to the published algorithms by the lastname of the first author of the original publication: Tuinstra (Timothy R. Tuinstra’s method [3]), Telkin (Ender Telkin és James M. Coughlan’s method [4]), Juett (James Juett és Xiaojun’s method [5]), Katona (our previous method [11]), and as “Proposed” (or “Prop.”) for our new proposed algorithm. Figure 2 shows intermediate stages of the processing by the five algorithms.

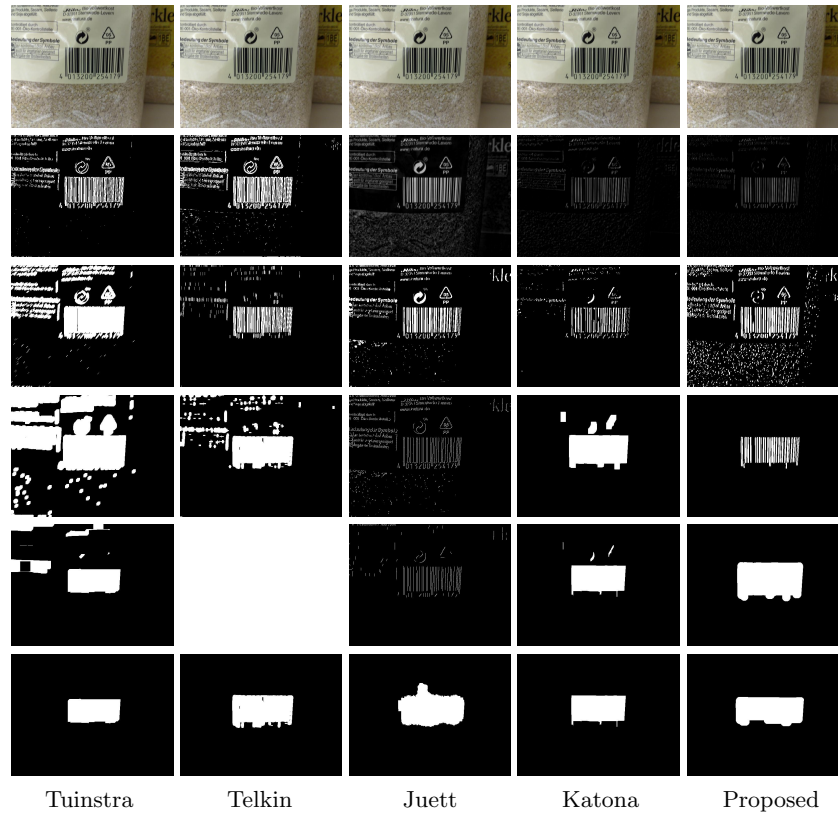
Nowadays, the quality of digital images are usually very good, although low-quality recordings are also made. The reason for lesser quality may be e.g. the capture device, but the environment may also cause troubles. Subsequently, often there is a need for correcting (rather improving) the image quality before starting the particular processing.

### 2.1 Method based on basic morphological operations

First, we present Tuinstra’s [3] algorithm. This method relies on basic morphological operations. Most algorithms use simple operations in preprocessing, such as quantization, wherein the input image is converted into another range of intensity values (e.g. binary). In this case, the author relies on that in the barcode region, the intensity difference between the stripes is high, so the gradient would highlight the bars. To estimate the gradient in the  $x$  and  $y$  directions, Sobel kernels are used. Next, the gradient image is thresholded and pixels having a high gradient value are selected.

The rest of the procedure works on the binary image. The localization is based on morphological operations. First step is a hit-or-miss transformation with a line structuring element. Unfortunately, the structuring element is not specified in the article [3]. In our implementation, the structuring element was chosen considering the length of the longest bar in the images. We used a  $10 \times 10$  block SE which is sufficiently large for the objective and smaller than the expected bar length in the codes. In the course of the procedure all objects are removed that likely do not contain barcode.

After the hit-or-miss transformation, morphological dilation is performed in order to merge nearby but not necessarily connected objects to be able to compose a region. The structuring element is square shaped, but in the article its



**Fig. 2.** Intermediate stages of the processing by the five algorithms. First row: original image, last row: final output.

exact size was not fixed. We used a  $10 \times 10$  block SE matching the size used for the previous step. In practice, the image size determines this parameter. Certainly, dilation fuses some regions that do not contain barcode, but in a subsequent phase these false results are eliminated.

The next step is to perform morphological erosion to discard thin objects from the image. The structuring element size is greater than that used for dilation. We used a  $20 \times 20$  block SE. This step removes undesired segments which were fused by the dilation.

The final operation is a solidity test which compares the number of pixels turned on in a region to the convex hull of region. So, in this step, probably false positive objects are removed and only barcode regions remain.

## 2.2 Method based on image scanning

The procedure of Telkin and Coughlan [4] was designed for visually impaired or blind people to facilitate their everyday lives. The article gives a complete description from the quality improvement of the input image to reading the barcode. Here we just recall the part of barcode localization.

The algorithm is most suitable for the UPC-A barcode type, but is also suitable for the localization of other 1D codes like UPC-E, and the EAN standard family of codes. The procedure is based on scanning the image at different orientations.

The procedure is presented in two main phases. The quality of the discrete images obtained from the input device is not ideal, so a preprocessing phase is necessary before the particular barcode localization. The first step is Gaussian smoothing to reduce the noise level. Since the original paper did not specify the  $\sigma$ , we used the same value of  $\sigma = 0.3$  as in our proposed method. Then pixel gradient values are calculated for the entire image using the Sobel operator, thus yielding a kind of edge enhancement in the image.

In the next step the algorithm works with only those pixels where the gradient value is above a threshold, so a binarization is done so that pixels having a gradient above the threshold become white and the others black. We used a threshold value set to 95 % of the maximal gradient value.

The detection phase is next. Firstly, the image is scanned in four directions (horizontally, vertically, and in direction of the diagonals ( $\pm 45^\circ$ )). The horizontal scan comes first, wherein those edge pixels are found whose orientation is vertical. The method looks in the vicinity of each edge pixel for opposite polarity pixels. If there is a sufficient number of such pixels in an area, as if they were part of a line segment, this area will not be removed. Subsequently, vertical scan follows, when those segments remain which have almost the same beginning and end so they most likely belong to barcode areas. If there appears to be more than one such region, then the next section of operations will filter out the false positive regions.

At the final stage of the detection the entropy value is calculated for each pixel in the result image describing the disorder of intensities within a given neighbourhood around each pixel. It is recalled that the barcode lines are parallel with each other, so region entropy would be likely low, and the entropy of false positive segments are expected to be high.

## 2.3 Method based on bottom-hat filtering

Next, we describe an algorithm by Juett and Qui [5], that is based on bottom-hat filtering.

If the input is not a grayscale image, the algorithm will convert it to that by quantization. In preprocessing, the method corrects the non-ideal image with simple contrast stretching in order to highlight differences between light and dark areas. When we converted the color image to grayscale, the image intensity range scales into a given interval. The next step is bottom-hat filtering, when the

closing of the stretched image is subtracted from the original stretched image. [14] The structuring element size depends on the widest bar in the barcode to be detected. The article specified  $25 \times 25$  block SE for images of size  $720 \times 480$  px. In less complex images, after bottom-hat filtering the false areas are less than after gradient calculation. The rest of this algorithm works with binary images, so the next step is the binary conversion.

After this, the contour is defined. The binary image is eroded using a  $5 \times 5$  structuring element and subtracted from the original binary image.

This is followed by the step to determine the orientation, which is performed by directional image openings using a relatively large linear structuring element. These openings are performed at 16 different orientations, with a step of  $11.25^\circ$ . Selecting one suitable orientation is difficult, because the barcodes could be in any orientation between  $0^\circ$  and  $180^\circ$ . Thus several orientation is probed which significantly increases the execution time of the algorithm. The directional opening images are summarized and a low resolution density image is calculated. Then, this image is converted back into binary and each region represents a potentially barcode region.

In the last phase, objects whose area is smaller than a threshold are eliminated. Using the centroids of the remaining objects lines that are next to each other are found and the potential corner points of the object are computed.

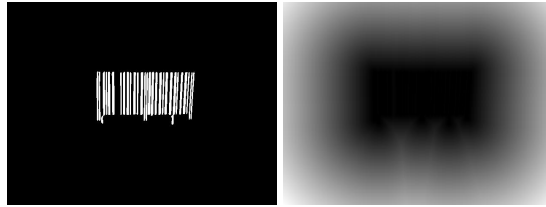
## 2.4 Proposed method using bottom-hat filtering

Our aim was to devise an algorithm with sufficiently high speed and accuracy. This is a modified version of our previously published algorithm [11].

Similarly to the above described methods [3–5], our algorithm consists of two main phases. In the preprocessing phase, the input image is converted to grayscale, because although barcodes may be printed in various colors, the pattern of dark bars on a light background is equivalent.

To reduce the image noise, we use smoothing with a Gaussian kernel where  $\sigma = 0.3$ . In next step, edge enhancement follows, which in most barcode localization method is done by calculating gradient values. Instead, we use bottom-hat filtering that is also based on intensity differences. Although bottom-hat filtering is less attractive regarding operation time than other non-directional edge enhancement operations, its accuracy is higher. Using the histogram of the edge-enhanced image, the proposed algorithm finds the most frequent intensity. Later we refer to this frequency as *MaxFreq*. We define a linear structuring element and its rotated version at 90 degrees, perform bottom-hat filtering using each line structuring element and later use the one that produces better matches.

For the further steps of the procedure the grayscale edge-enhanced images are converted to binary. We suppose that if *MaxFreq* is relatively high (above 20 %, using a 256 bin histogram), the image is noisy or blurred or both. In this case, we set the threshold to 95 % of the maximal value to keep only the strongest edges (without risking picking up noise), otherwise the threshold is set to 80 % of the maximal value to allow picking up more edges. These ‘magic numbers’ were selected based on our prior experiments with varying the values and visually



**Fig. 3.** A thresholded binary image (left) and the distance map computed from it (right).

checking the binary images. To our experience the method is not very sensitive to the choice of these particular values and has almost identical performance with thresholds selected around the above given values. Thus, we did not employ more sophisticated (automatic) threshold selection techniques, e.g. by analyzing the outcome using multiple threshold choices. Since all other discussed methods used this kind of simple early binarization, we have not explored in this work the possibilities of using binarization-less grayscale processing further on in the subsequent phases.

After preprocessing, the digital image is suited to finding segments which contain barcode. The resulting image so far contains many false positive regions. There are many small connected components which satisfy the criteria but do not belong to barcode regions. These false regions are eliminated using an area threshold. Selecting a too high threshold value could remove small barcodes from high-resolution images. If the image size is higher than  $800 \times 800$  px, we set the threshold to half the size of the largest component, otherwise the threshold is set to  $1/4$  of the size of the largest component. These fractions were also set based on prior experiments, they are also not highly sensitive values and are used only to filter out some small false positives.

We take advantage of the structure of the barcode, the fact that it consists of approximately regularly spaced parallel stripes. So, for each pixel, we calculate the Euclidean distance of the pixel from the nearest nonzero pixel. Figure 3 shows a thresholded binary image and its distance map. Using the distance map, objects that are far from other objects can be easily dropped, and only nearby objects (sort of a cluster of bar segments) will be kept. The distance threshold is selected as the minimum of the row averages of the distance map.

Our observations showed, that for the final detection steps, mostly dense text regions remain along with the supposedly barcode areas. Since a barcode consists of a sequence of parallel bars that are located at varying distance from each other, they do not compose a connected component. Therefore, to merge these patterns, we use dilation with a square structuring element whose size is defined as  $S = \max(40, \text{width of the widest bar} * 3)$ . Nevertheless, this dilation may also thicken and merge unwanted, non-barcode locations as well. To cope with this problem, we also use the dual operation, i.e. erosion. Here, the structuring element is

---

**INITIALIZATION:**

|                           |                            |                             |
|---------------------------|----------------------------|-----------------------------|
| $f$ = grayscale image     | $\bullet$ = morph. closing | $SE_1$ = bottom-hat str.el. |
| $min_a$ = minimum area    | $\oplus$ = morph. dilation | $SE_2$ = dilation str.el.   |
| $i_t$ = minimum threshold | $\ominus$ = morph. erosion | $SE_3$ = erosion str.el.    |

---

**ALGORITHM:**

```

 $f := (f \bullet SE_1) - f$ 
for all pixels do
    if  $f_{i,j} > i_t$  then  $f_{i,j} \leftarrow 1$ 
    else  $f_{i,j} \leftarrow 0$ 
Calculate Euclidean distance map
Compute the minimum  $X$  of row averages in the distance map
Drop those pixels, whose distance value is greater than  $X$ 
 $f := f \oplus SE_2$ 
 $f := f \ominus SE_3$ 
for all components do
    if component area  $> min_a$  then Record this component as a barcode segment
    else Discard this component
    
```

---

**Fig. 4.** Pseudocode of the Proposed algorithm

linear, consistent with the size of the 1D barcodes. The structuring element size is less than (about 1/3 of) the one used for dilation, i.e., it matches the width of the widest bar. After erosion, those areas that likely contain barcodes can be found. Of course, there may still be false positive objects. These are removed in the last step on the basis of their size and proportions. Similarly to the previous step, if the image size is higher than 800×800 px, objects smaller than half the size of the largest object are removed, otherwise objects smaller than 1/4 of the size of the largest object are removed.

In the application area which we aim for with this rapid algorithms false positives are not critical since they can be quickly dropped by the subsequent decoding module. Missing a potential code region, on the other hand, is considered crucial. Therefore, detecting anything that resembles barcode-like visual features (be it a zebra in a picture taken at the zoo, or the keyboard of a piano) could be considered a valid output. Also, the area size, used at two points in our algorithm, is only a secondary filter since we also use visual features strongly depending of the alternating dark and bright stripes, the distinctive visual feature of the barcodes. Thus, if the analyzed image contains anything appearing like a barcode and having the size within a reasonable range (that can be also preset in a particular application setup) will be detected and if the images does not contain anything presenting barcode features, or none within the reasonable size range, the algorithm will not detect anything (although, as already indicated above, false positives are not that much of a problem as false negatives).

The proposed algorithm is summarized in Figure 4.



## 2.5 Our previous method using bottom-hat filtering

This method was previously published in [11]. The main steps are the same as in the above proposed version of the algorithm. Here we only describe the key differences.

- After noise reduction, no frequency analysis is done.
- A square-shaped structuring element is used for bottom-hat filtering.
- For binarization, a fixed threshold of 95 % of the maximal value is used.
- The distance map-based filtering is not performed at all. Instead, we analyze the entire binary image using a horizontal scan to check whether white pixels are located at appropriate distances at the given direction. The maximal distance parameter depends on the image resolution.
- After scanning, small, falsely detected regions are eliminated using an area threshold of half the size of the largest component, regardless of the image dimensions.
- A square shaped structuring element is used for the final dilation and erosion step.
- The final area thresholding step also uses a threshold of half the size of the largest component, regardless of the image dimensions.

## 3 Evaluation

In this section, we compare the discussed methods' effectiveness under specific characteristics.

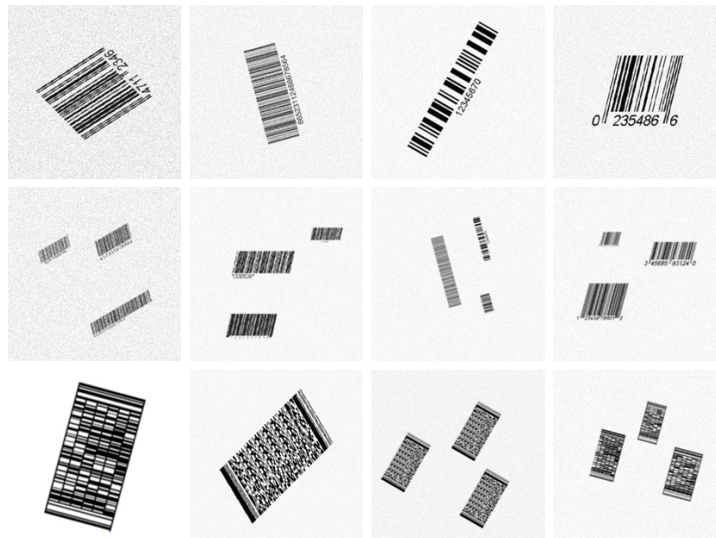
### 3.1 Test suite, test environment, and implementation

We generated barcodes digitally with the types shown in Figures 1. Only one base image was chosen for each code type. Test images contained one or three barcodes from each types, respectively, and images were affected by distortions. For each base image, we generated all combinations of the following properties: rotation in every  $15^\circ$  from  $0^\circ$  to  $180^\circ$ , Gaussian blur filter with  $3 \times 3$  kernel with 6 different  $\sigma$ , additive noise from 0 % to 50 % with a step of 10 %, and 15 % shear. In summary, the test set contained images containing either 1 or 3 codes of 10 different barcode types, with 12 orientations, 6 different blur filters, 6 different rates of additive noise, with or without shear, with a total of 17,280 images.

Figure 5 shows a selection of generated test images with various code content and applied distortions.

Another 100 images containing barcodes were collected from real-life examples without any modifications. These images presented scratches, blur, minor light reflections and distortions also.

We implemented the methods in MATLAB, with the help of the Image Processing Toolbox. Evaluation was performed on a computer with Pentium(R) Dual-Core 2.30 GHz CPU.



**Fig. 5.** A selection of generated test images with various code content and applied distortions.

### 3.2 Results and discussion

In this section, we show how effective the implemented algorithms are on images with various characteristics, and compare their running time w.r.t each other. For calculating accuracy we used the Jaccard the coefficient of similarity, which measures the overlap of the bounding boxes of the real and the detected barcode region.  $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$ , where  $A$  denotes the real barcode and  $B$  is the bounding box of the detected code. This not only measures the success of detection, but also considers the delination accuracy of the methods.

We can conclude that all five algorithms are capable of effectively detecting barcodes. Telkin's method is fast, however, when there are several barcodes in the image and noise is also present, it loses efficiency. As already mentioned, we also applied stretching when generating the images. The efficiency of the algorithms do not seem to depend on the stretching and also seem to be insensitive to rotation as well.

Tuinstra's algorithm is slower than the others, which can most probably be attributed to the fact that here the image is scanned in four different orientations.

Juett's method based on bottom-hat filtering falls off to the discussed fast algorithms with respect to running time. The process is slower than the others because in the detection phase the image is scanned in 16 different orientations. This brings greater accuracy to the process but at the expense of more computation.

**Table 1.** Running time of the algorithms. in images with 1D (above), respectively 2D (below) barcode Mean values ( $\pm$  standard deviation) (expressed in percent) for all images with 1D barcodes (top), and for all images with 2D codes (bottom).

|         |    | Tuinstra        | Telkin          | Juett           | Katona          | Proposed        |
|---------|----|-----------------|-----------------|-----------------|-----------------|-----------------|
| 1 code  | 1D | 0.23 $\pm$ 0.13 | 0.67 $\pm$ 0.22 | 0.81 $\pm$ 0.43 | 0.13 $\pm$ 0.10 | 0.16 $\pm$ 0.08 |
|         | 2D | 0.21 $\pm$ 0.14 | 0.77 $\pm$ 0.47 | 0.73 $\pm$ 0.44 | 0.12 $\pm$ 0.12 | 0.14 $\pm$ 0.09 |
| 3 codes | 1D | 0.54 $\pm$ 0.35 | 1.73 $\pm$ 0.66 | 1.86 $\pm$ 0.78 | 0.18 $\pm$ 0.09 | 0.35 $\pm$ 0.12 |
|         | 2D | 0.46 $\pm$ 0.21 | 1.70 $\pm$ 0.54 | 1.68 $\pm$ 0.71 | 0.18 $\pm$ 0.10 | 0.28 $\pm$ 0.07 |

Our proposed algorithms are also very fast. In most cases, faster procedures locate barcodes less accurately or not at all, as if loss of accuracy were the price for the speed gain. However, in our case, speed is not coupled with a high cost in accuracy. From the evaluation output we can conclude that the proposed procedure has the best running efficiency for all test images, and in many cases it also shows best accuracy. The weakness of the method appears when the images is very noisy or when there are such image areas which are similar to a barcode.

The efficiency of the methods, i.e. processing time is also an important aspect. Table 1 presents the execution time of the detection methods for images with 1 or 3 barcodes. Here, one can easily appreciate the significant differences between the different approaches.

Subsequent tables show, how the algorithms behave on images which contain either one or three code pieces of various 1D and 2D barcodes types. The structure of barcode types varies which also has an effect on how well the algorithms can perform. Accuracy of the detection methods for images containing a single piece of code for various code types is presented in Table 2. Tuinstra's, Telkin's, and Juett's algorithm have very bad performance for Code128, however our two proposed algorithms handle this variable-length code as well as the fixed-length types. For the 2D codes, Telkin's method shows exceptionally good accuracy, but the other methods also perform well on these stacked barcodes. As to images with three barcodes, results are also shown. There is more variance in performance and some methods (Tuinstra, Katona, and Proposed) show a solid performance while the others very much depend on the type of code present in the image.

The accuracy of the algorithms degrades on distorted images. Next, we analyze the effect of distortions on efficiency. Table 3 shows the methods' behavior with respect to the level of blur applied to the images. Image smoothing does not change the performance considerably. Although there is a noticeable inverse relation between  $\sigma$  and accuracy, this is not significant. It is interesting to notice that in some cases the barcode detection accuracy is higher in images with three barcodes than for images with a single code present. This can be due to the fact that relative size (w.r.t. image size) of individual codes are smaller in the 3-code examples. Telkin's algorithm and our two proposed methods show outstanding accuracy in images which contain 2D barcodes and there are not considerable difference between the precision values.

**Table 2.** Accuracy of the algorithms for various types of codes, for images containing a single code and for those containing 3 codes. Mean values (expressed in percent) for all distorted images with 1D barcodes (top), and for all images with 2D codes (bottom).

|        |           | 1 code   |        |       |        |       | 3 codes  |        |       |        |       |
|--------|-----------|----------|--------|-------|--------|-------|----------|--------|-------|--------|-------|
|        |           | Tuinstra | Telkin | Juett | Katona | Prop. | Tuinstra | Telkin | Juett | Katona | Prop. |
| 1D     | Codabar   | 99.0     | 98.9   | 99.2  | 99.9   | 98.6  | 83.8     | 56.2   | 52.7  | 92.8   | 87.5  |
|        | Code128   | 50.4     | 63.6   | 70.9  | 95.7   | 93.7  | 83.6     | 95.0   | 79.6  | 80.1   | 84.8  |
|        | Code39    | 98.7     | 94.8   | 86.3  | 99.4   | 100.0 | 83.3     | 90.5   | 67.3  | 94.7   | 90.5  |
|        | EAN-13    | 98.8     | 94.7   | 91.7  | 98.9   | 99.3  | 83.3     | 91.4   | 76.6  | 99.3   | 99.2  |
|        | EAN-8     | 98.7     | 99.6   | 84.6  | 99.6   | 98.2  | 83.3     | 97.2   | 81.8  | 99.4   | 95.9  |
|        | I2of5     | 98.6     | 84.6   | 84.4  | 93.1   | 92.5  | 78.8     | 35.5   | 93.8  | 99.2   | 84.9  |
|        | UPC-A     | 99.1     | 98.2   | 94.9  | 86.4   | 100.0 | 74.6     | 65.5   | 92.6  | 99.9   | 95.4  |
|        | UPC-E     | 82.7     | 92.6   | 76.0  | 99.6   | 100.0 | 98.7     | 77.5   | 96.7  | 99.3   | 92.5  |
| all 1D |           | 91.8     | 86.4   | 78.9  | 96.6   | 96.8  | 74.0     | 74.7   | 77.4  | 95.6   | 89.2  |
| 2D     | Codablock | 87.0     | 99.8   | 83.3  | 87.7   | 100.0 | 86.6     | 100.0  | 80.0  | 90.7   | 94.2  |
|        | PDF417    | 81.7     | 99.6   | 79.7  | 93.9   | 97.6  | 83.3     | 100.0  | 83.9  | 91.7   | 90.8  |
|        | all 2D    | 85.0     | 99.7   | 82.5  | 90.8   | 99.1  | 85.0     | 100.0  | 82.4  | 91.2   | 92.5  |

**Table 3.** Accuracy of the algorithms for different blur levels, for images containing a single code and for those containing 3 codes. Mean values (expressed in percent) for all distorted images with 1D barcodes (top), and for all images with 2D codes (bottom).

|    |        | 1 code   |        |       |        |       | 3 codes  |        |       |        |       |
|----|--------|----------|--------|-------|--------|-------|----------|--------|-------|--------|-------|
|    |        | Tuinstra | Telkin | Juett | Katona | Prop. | Tuinstra | Telkin | Juett | Katona | Prop. |
| 1D | 0.0    | 100.0    | 100.0  | 100.0 | 100.0  | 100.0 | 100.0    | 100.0  | 100.0 | 100.0  | 100.0 |
|    | 0.5    | 91.2     | 88.4   | 91.3  | 96.1   | 95.7  | 93.8     | 94.4   | 92.4  | 98.0   | 100.0 |
|    | 1.0    | 90.4     | 88.5   | 89.4  | 97.1   | 95.4  | 92.4     | 86.7   | 89.1  | 98.0   | 100.0 |
|    | 1.5    | 90.4     | 88.5   | 89.4  | 96.8   | 95.4  | 91.7     | 86.7   | 89.1  | 97.0   | 100.0 |
|    | 2.0    | 90.3     | 90.4   | 87.5  | 97.1   | 93.3  | 86.3     | 82.9   | 90.2  | 97.0   | 100.0 |
|    | 2.5    | 90.4     | 90.4   | 87.5  | 97.1   | 93.3  | 82.7     | 82.0   | 90.2  | 97.0   | 100.0 |
|    | all 1D | 90.5     | 89.1   | 89.6  | 96.7   | 94.3  | 82.7     | 77.9   | 90.6  | 97.5   | 100.0 |
| 2D | 0.0    | 100.0    | 100.0  | 100.0 | 100.0  | 100.0 | 100.0    | 100.0  | 100.0 | 100.0  | 100.0 |
|    | 0.5    | 88.8     | 100.0  | 83.1  | 95.8   | 100.0 | 89.1     | 100.0  | 83.2  | 100.0  | 100.0 |
|    | 1.0    | 88.4     | 100.0  | 82.6  | 95.7   | 100.0 | 87.7     | 100.0  | 82.5  | 100.0  | 100.0 |
|    | 1.5    | 87.7     | 100.0  | 82.6  | 95.7   | 100.0 | 87.0     | 100.0  | 82.5  | 100.0  | 100.0 |
|    | 2.0    | 83.6     | 100.0  | 82.6  | 95.7   | 100.0 | 85.9     | 100.0  | 74.5  | 100.0  | 100.0 |
|    | 2.5    | 83.6     | 100.0  | 82.6  | 95.7   | 100.0 | 85.0     | 100.0  | 74.5  | 100.0  | 100.0 |
|    | all 2D | 87.5     | 100.0  | 83.3  | 95.8   | 100.0 | 87.0     | 100.0  | 82.9  | 100.0  | 100.0 |

**Table 4.** Accuracy of the algorithms for different noise levels, for images containing a single code and for those containing 3 codes. Mean values (expressed in percent) for all distorted images with 1D barcodes (top), and for all images with 2D codes (bottom).

|    |        | 1 code   |        |       |        |       | 3 codes  |        |       |        |       |
|----|--------|----------|--------|-------|--------|-------|----------|--------|-------|--------|-------|
|    |        | Tuinstra | Telkin | Juett | Katona | Prop. | Tuinstra | Telkin | Juett | Katona | Prop. |
| 1D | 0      | 100.0    | 100.0  | 100.0 | 100.0  | 100.0 | 100.0    | 100.0  | 100.0 | 100.0  | 100.0 |
|    | 10     | 90.2     | 89.5   | 93.9  | 97.4   | 97.1  | 94.8     | 97.5   | 92.6  | 96.0   | 95.8  |
|    | 20     | 89.7     | 86.9   | 87.5  | 97.4   | 97.1  | 89.4     | 87.6   | 89.5  | 96.3   | 91.6  |
|    | 30     | 89.4     | 86.2   | 79.2  | 97.4   | 92.5  | 84.6     | 86.3   | 82.7  | 96.7   | 95.8  |
|    | 40     | 90.1     | 90.1   | 78.5  | 97.8   | 92.5  | 87.2     | 84.9   | 79.9  | 96.7   | 95.8  |
|    | 50     | 93.3     | 84.2   | 69.9  | 94.9   | 91.6  | 73.8     | 82.7   | 64.4  | 90.6   | 91.6  |
|    | all 1D | 90.5     | 88.6   | 83.5  | 96.8   | 95.4  | 87.6     | 87.3   | 84.7  | 95.6   | 95.1  |
| 2D | 0      | 100.0    | 100.0  | 100.0 | 100.0  | 100.0 | 100.0    | 100.0  | 100.0 | 100.0  | 100.0 |
|    | 10     | 87.7     | 100.0  | 84.7  | 94.4   | 100.0 | 86.3     | 100.0  | 84.0  | 100.0  | 100.0 |
|    | 20     | 87.7     | 98.6   | 81.9  | 94.2   | 100.0 | 85.8     | 100.0  | 78.9  | 100.0  | 100.0 |
|    | 30     | 87.7     | 100.0  | 79.3  | 94.2   | 100.0 | 90.4     | 100.0  | 73.0  | 100.0  | 100.0 |
|    | 40     | 85.7     | 100.0  | 76.7  | 94.2   | 100.0 | 86.3     | 100.0  | 72.6  | 100.0  | 100.0 |
|    | 50     | 77.8     | 98.6   | 69.6  | 92.3   | 100.0 | 84.9     | 100.0  | 71.4  | 85.7   | 100.0 |
|    | all 2D | 85.8     | 99.5   | 77.3  | 94.8   | 100.0 | 86.4     | 100.0  | 73.3  | 95.6   | 100.0 |

**Table 5.** Accuracy of the algorithms for real-life images containing a single 1D barcode. Mean values (expressed in percent).

| Tuinstra | Telkin | Juett | Katona | Proposed |
|----------|--------|-------|--------|----------|
| 92.0     | 91.0   | 92.0  | 94.0   | 96.0     |

In a similar manner, we analyzed how the algorithms perform on images with different levels of noise. We can see from Table 4 that each algorithm is somewhat sensitive to noise. The studied literature methods lose much of their accuracy as the noise level increases, however, our proposed methods degrade to a lesser extent.

We also tested the methods on a set of 100 real-life images, too. These images contain 1D barcodes each. Table 5 demonstrates the accuracy of the algorithms.

In Figure 6 we show a visual comparison. Using a single image example, the intermediate and final output of the four tested algorithms are shown at their key phases. The chosen real image (column (a)) is noisy and non-ideal. Column (b) shows the result after preprocessing. In column (c), the segmentation output before the last step is displayed. Here, the result of Telkin's, Tuinstra's, and our new proposed method contain the barcode location and also some other false regions. Juett's and our previous algorithm only show the barcode segment at this stage. From the images in column (d), one can have an impression of how accurately each method locates the barcodes.

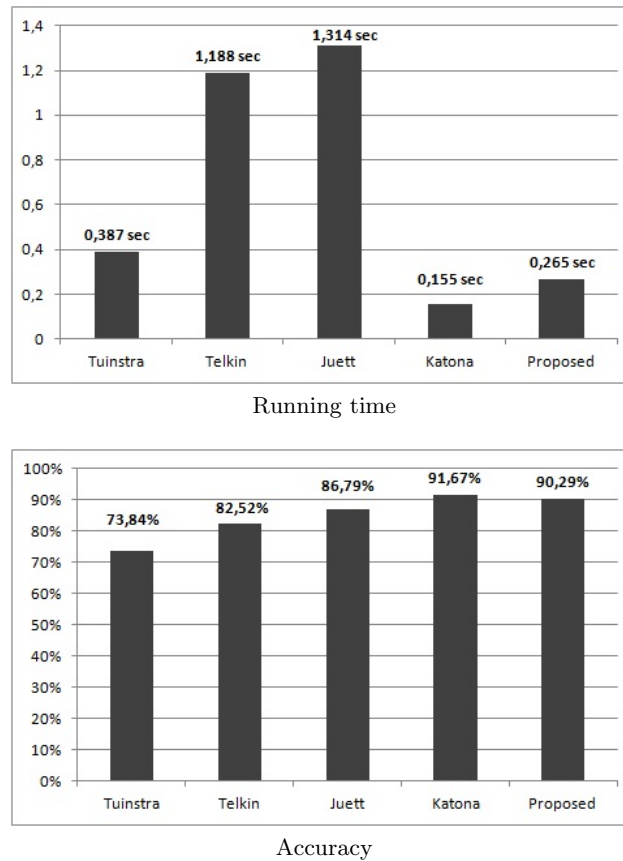
For the digitally generated images maximal size was  $800 \times 800$  px. At this resolution Tuinstra's and Juett's methods cannot compete with the others in speed. Our two proposed methods are not far better on  $800 \times 800$  px images than



**Fig. 6.** Intermediate and final outputs of the algorithms at key phases. From top to bottom: original image, after preprocessing, before the last step, final output.

Telkin's, but when executing them on  $2500 \times 1722$  px images, Telkin's runs for 3.2 sec, while our algorithms finish under 1.5 sec. The new proposed algorithm is somewhat slower than our previous version due to the extra computation of the Euclidean distance map. We can state, that for larger images there are considerable execution time differences between the algorithms. The chart in Figure 7 well illustrates that our algorithms outperform the three algorithms from the literature both in terms of accuracy and speed. Figure 7 displays each algorithm's average running time, and their accuracy for all digitally generated test images. Note, that methods which use bottom-hat filtering in the preprocessing phase, have higher accuracy than the other two, but their running times are quite different from each other. Juett's loses its accuracy proportional to image degradation, while our proposed algorithms maintain higher accuracy for those cases as well. Although the diagrams do not show, the tests also demonstrated that in all cases the methods' running time significantly grows with increasing image size. Among the two methods of ours, the newly proposed has slightly worse accuracy. However, the presented figure is based on the entire pool of our synthetic test images, including many 1D as well as 2D code examples. Regarding images with 2D stacked codes only, the newly proposed algorithm clearly outperforms the previous version.

In the test images the barcodes are in different orientations, but tests showed that the algorithms are not sensitive to code orientation.



**Fig. 7.** Running time and accuracy of the algorithms. Mean values for all generated and distorted images.

## 4 Conclusion

We have presented a new version of our barcode detection algorithm and compared its performance (in terms of accuracy and speed) with our previous version and three methods from the literature. We concluded that these algorithms do not specialize for individual barcode types, they can efficiently detect various types of 1D and stacked 2D barcodes, but there are subtle difference in performance. For the evaluation, we created a test database containing 17,280 synthetic images representing various degradations (blur, noise, shear) as well as 100 real images. We demonstrated that the algorithms selected from the literature are rather sensitive to noise, while the proposed new methods are less sensitive. Our previous and proposed methods outperform the other three in detecting 1D

codes both in terms of accuracy and speed, and the new proposed version has better accuracy for 2D stacked barcode examples.

## Acknowledgements

The work of the second author is supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

## References

1. Palmer, R.C.: The Bar Code Book: Reading, Printing, and Specification of Bar Code Symbols. Helmers Pub (1995)
2. Adelman, R.: 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)
3. Tuinstra, T.R.: Reading Barcodes from Digital Imagery. PhD thesis, Cedarville University (2006)
4. Tekin, E., Coughlan, J.M.: An algorithm enabling blind users to find and read barcodes. In: Applications of Computer Vision (WACV), 2009 Workshop on. (2009) 1–8
5. James Juett, X.Q.: Barcode localization using bottom-hat filter. NSF Research Experience for Undergraduates (2005)
6. Bodnár, P., Nyúl, L.G.: Efficient barcode detection with texture analysis. In: Signal Processing, Pattern Recognition, and Applications, Proceedings of the Ninth IASTED International Conference on. (2012) 51–57
7. Bodnár, P., Nyúl, L.G.: Improving barcode detection with combination of simple detectors. In: The 8th International Conference on Signal Image Technology (SITIS 2012). (2012) Accepted for publication.
8. Chai, D.: Locating and decoding ean-13 barcodes from images captured by digital cameras. In: Information, Communications and Signal Processing, 2005 Fifth International Conference on. (2005) 1595–1599
9. Lin, D.T., Lin, M.C., Huang, K.Y.: Real-time automatic recognition of omnidirectional multiple barcodes and dsp implementation. Machine Vision and Applications **22** (2011) 409–419
10. Katona, M.: Vonalkódok hatékony detektálása morfológiai módszerekkel. TDK dolgozat (2012)
11. Katona, M., Nyúl, L.G.: A novel method for accurate and efficient barcode detection with morphological operations. In: The 8th International Conference on Signal Image Technology (SITIS 2012). (2012) Accepted for publication.
12. Wachenfeld, S., Terlunen, S., Jiang, X.: Robust recognition of 1-d barcodes using camera phones. In: Pattern Recognition, 2008. ICPR 2008. 19th International Conference on. (2008) 1–4
13. Shih, F.Y.: Image Processing and Mathematical Morphology: Fundamentals and Applications. CRC Press (2009)
14. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Prentice Hall (2007)