Journal of ELECTRICAL ENGINEERING, VOL 69 (2018), NO4, 286–292

sciendo

Comparing the impact of different cameras and image resolution to recognize the data matrix codes

Ladislav Karrach*, **Elena Pivarčiová***, **Yury Rafailovich Nikitin****

Data matrix codes are two-dimensional (2D) matrix bar codes, which are the descendants of the well known 1D bar codes. However, compared to 1D bar codes, they allow to store much more information in the same area. Comparing data matrix codes with QR codes, for example, we find them much more effective in marking small objects or in the case that you have only a very small area for placing a code in. Their capacity and ability of decoding also a code that is partly damaged make them an appropriate solution for industrial applications. In the following paper we compare the impact of various cameras on the detection and decoding of data matrix codes in real scene images. The location of the code is based on the fact that typical bordering of a data matrix code forms a region of connected points which create "L", the so-called finder pattern, and the parallel dotting, the so-called timing pattern. In the first step, we try to locate the finder pattern using adaptive thresholding and connecting neighbouring points to continuous regions. Then we search for the regions where 3 outer boundary points form a isosceles right triangle that could represent the finder pattern. In the second step, we have to verify the timing pattern. We look for an even number of crossings between the background and foreground. Experimental results show that the algorithm we have proposed provides better results than competitive solutions.

K e y w o r d s: data matrix code location, adaptive thresholding, connected components labelling

1 Introduction

Traditional linear or 1D barcodes can only contain a limited amount of information, roughly the equivalent of about 20–25 characters. 2D barcodes can store much more information, up to thousands of characters. There are a wide range of barcode readers on the market, including mobile devices and phones equipped with cameras that can scan both 2D and 1D barcodes. Creating and using 2D barcodes is inexpensive.

While 1D barcodes store information only in one direction (in one row), usually as parallel black lines of various widths, altered with various width spaces, 2D barcodes contain many different rows of data.

Data matrix belongs to a group of 2D bar codes (like the QR code) and consists of black and white squares (the so-called modules) organized into a 2D matrix. They represent the binary 1 (black) or 0 (white). Data matrix could be square or rectangular shaped whereby the adjacent sides are bordered by black modules creating an "L" (the so-called finder pattern) which is there for location of the code. On the facing sides, there are black and white squares placed alternatively (so-called timing pattern) and they serve for determining the number of rows and columns. The data themselves are coded inside. See Fig. 1 – sample of a data matrix codes 10×10 and 12×12.

Data matrix is usually used for marking of small objects such as electronic components as it allows effectively to code up to 50 characters into a small area from 2 to $3\,\mathrm{mm}^2$.

There is a capacity of 3 alphanumeric (alt. 6 numeric) characters in the data matrix of minimal size 10×10 and capacity up to 2335 alphanumeric (alt. 3116 numeric) characters in the data matrix of maximal size 144×144. Data matrix in specification ECC 200 allows to decode even a partly damaged code (up to 30 % damage if the code can be located).

In this article we are focusing on location and decoding of the data matrix (DMX) code that is laser marked in a metal tool (Fig. 2). Here DMX code specifies a serial number of the tool, which serves for tracing the tool in the production process and using the tool on a machine and following storing. We pursue a price effective solution that enables us to scan DMX in real time after every operation with the tool in the production process.

1.1 Related work

In our previous articles [1, 2] we presented and compared several methods for DMX code location. They all try to locate DMX by first locating Finder Pattern "L" which consists (in our case) of two mutually vertical lines of the same length.

Here we summarize in brief these methods:

* Department of Manufacturing and Automation Technology, Faculty of Environmental and Manufacturing Technology, Technical University in Zvolen, Masarykova 24, 960 01 Zvolen. karrach@zoznam.sk, pivarciova@tuzvo.sk, ** Department of Mechatronic Systems, Institute of Modern Technologies in Mechanical and Automotive Engineering and Metallurgy, Kalashnikov Izhevsk State Technical University, Studencheskaya street, 7, 426069, Izhevsk, Russia, doc_nikitin@mail.ru

DOI: 10.2478/jee-2018–0040, Print (till 2015) ISSN 1335-3632, On-line ISSN 1339-309X

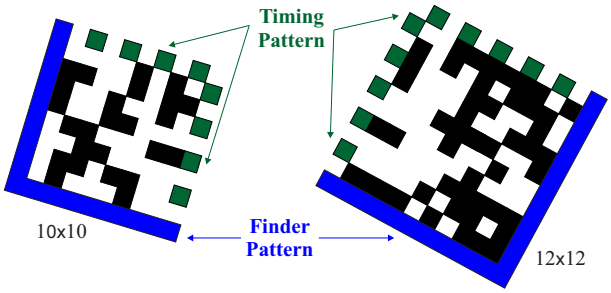


Fig. 1. Structure of a sample data matrix code

Fig. 2. Samples of DMX code marked with laser on metal surface

Fig. 3. (a) – original image, (b) – edge image, (c) – continuous regions in edge image

Fig. 4. (a) – original image, (b) – continuous region with 3 vertices

- Lines that form the finder pattern can be found in places, where edges in an image are present. So we convolve the image with a 3×3 Sobel kernel and select points with "strong" magnitude – edge points, Fig. 3(b). Then we connect these edge points that have "similar" gradient angles into continuous regions (such region will contain edge points a "one-direction" edge – linear segment). Consequently we look for perpendicular regions, Fig. 3(c). A similar technique is also used by Qiang Huang *et al* in [3] who make use of the algorithm for detection of linear segments (line segment detector) [4, 5].

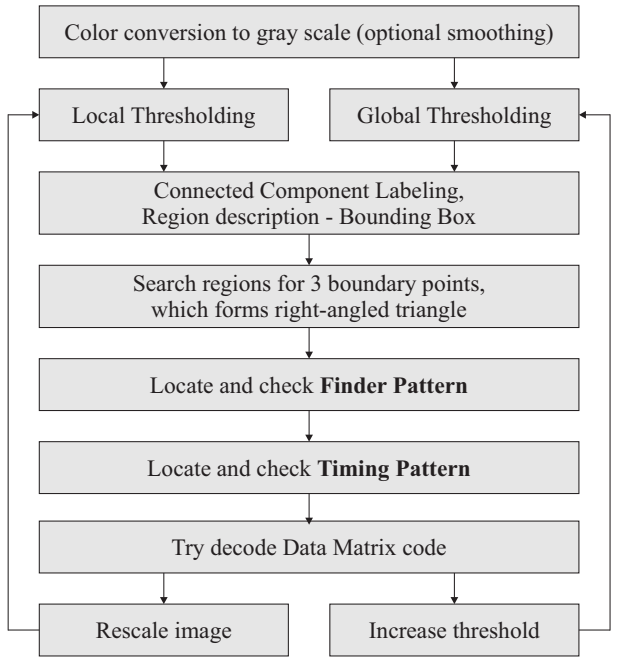


Fig. 5. Processing scheme

- Lines that form the finder pattern represent a continuous region. In this region we look for 3 outer boundary points forming the 3 vertices of a right-angled triangle (with two sides of equal length, see Fig. 4, with vertices V1, V2, V3).
- Beside this approaches, there were other techniques published which use the Radon transform [6] or the Hough transform [7] and many approaches for detecting QR codes [8–10] including deep learning methods [11].

2 Theory

2.1 Location of the data matrix code

The presented method goes step by step and in each step we come closer to the exact position or dimensions of DMX code. In this process, we reject a candidate region if it does not meet classification criteria. The simplified image processing scheme is shown in Fig. 5.

Before starting location, we transfer an original image (possibly colour) into a gray scaled image. Each point of such an image is coded in 8 bits (0 – black colour, 255 – white colour).

Next we convert the gray scaled image into a binary image (0 background, 1 foreground) using adaptive thresholding (with window size 31 and with delta 10; the window size we choose to be at least 5 times the size of the expected size of DMX module) [12]. We expect that black points which belong to DMX code will become foreground points.

Fig. 6. Sample of an outer boundary of a continuous region

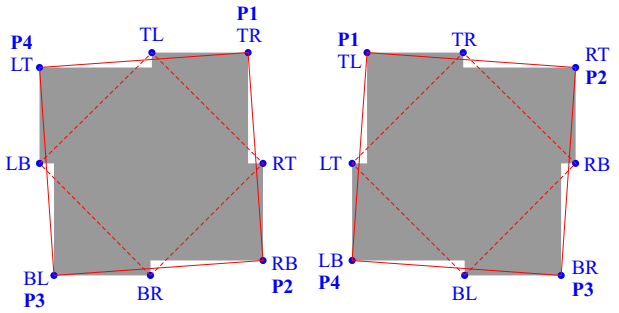


Fig. 7. Outer boundaries of region

We use the well known adaptive thresholding technique that calculates individual thresholds for every point in the image. This threshold is calculated as the average intensity of points under the sliding window. To speed up the thresholding we pre-calculate the integral sum image and we also use the global threshold value (points with intensity above 150 we always consider as background points)

$$B(x,y) = \begin{cases} 0 \leftarrow I(x,y) > 150\,, \\ 0 \leftarrow I(x,y) \geq T(x,y)\,, \\ 1 \leftarrow I(x,y) < T(x,y)\,, \end{cases}$$
$$T(x,y) = \frac{1}{961} \sum_{i=-15}^{15} \sum_{j=-15}^{15} I(x+i, y+j) - 10\,. \qquad (1)$$

Consequently we apply the 2-pass connected-component labelling algorithm [15], whereby we connect the points into continuous regions. A set of descriptors is maintained for every region. When adding a new point in the region, we have to update its descriptors as follows:

– Area: M00←M00 + 1

– Border points: top-left, top-right, right-top, right-bottom, bottom-right, bottom-left, left-bottom, left-top

Once we have segmented the image into continuous regions, the location of DMX runs in two basic steps. First, the finder pattern is located. This means that the two mutually perpendicular sides of the square are detected (*ie* two perpendicular line segments of the same length). In the following step, we check the timing pattern, verifying if the opposite sides to the finder pattern have the same number of crossings between black (foreground) and white (background). At the same time the number of crossings determines the number of rows and columns (dimensions) of DMX.

We scan region descriptors looking for regions that have the area bigger than 80 points (we ignore small regions which cannot set-up the finder pattern), aspect ratio (region width to height) between 0.5 and 2.0 and the region where it is possible to make up a right-angled isosceles triangle from border points.

In Fig. 6, one can see an example of the marked continuous region (in magenta colour) enclosed by a green octagon defined by 8 boundary points. There are at least 3 such boundary points which make up vertices of a right-angled isosceles triangle.

We take 8 edge points and make up two boundary squares, which are defined by the following points:

- top-left (TL), right-top (RT), bottom-right (BR) and left-bottom (LB)
- top-right (TR), right-bottom (RB), bottom-left (BL) and left-top (LT)

From these squares we select that one whose perimeter is bigger (*ie* it represents outer boundary of a continuous region). In this way we get 4 border points labelled as P1, P2, P3, P4. In Fig. 7 there are outer boundaries marked with a solid red line (unlike the red dashed lines, which also connect boundary points but do not form an outer bounding box).

We test these 4 points subsequently taking 3 of them in one testing step (*ie* P1-P2-P3, P2-P3-P4, P3-P4-P1, P4-P1-P2). We check whether we can make up a right-angled isosceles triangle. For example, the first three points P1-P2-P3 must match the following conditions

$|P1,P3|^2 \cong |P1,P2|^2 + |P2,P3|^2$ and $|P1,P2| \cong |P2,P3|$ whereby the maximal deviation allowed is 4. If we do not find 3 such points, we shrink the region by 1 point and repeat the step once again with new boundary points. If we find 3 such points, we suppose that they are the vertices of the "L" which defines the finder pattern. Their location, however, is not completely exact due to inaccuracy of thresholding. This is why we have to get their location more exactly in the following steps to get the possible best outlines defining the finder pattern.

The following method is based on these 3 starting points which are defined by line segments P1-P2 and P2-P3 and have one common vertex P2. These points lie in the same continuous region "O". We process independently both line segments and after getting their location more exact, we finish the process by calculating their mutual intersection point.

The line segment P1-P2 is moved in a vertical way away from region O at the beginning and then the ends of the line are alternately approached to region O until the overlapping of the line and region O achieves at least 90 %.

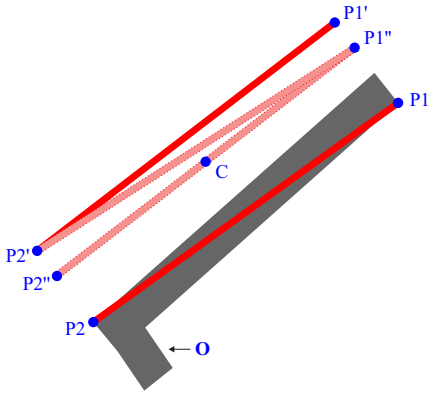


Fig. 8. Line is approaching region O

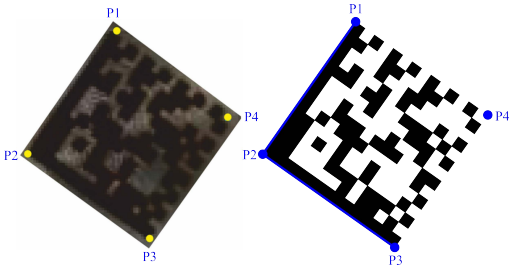


Fig. 9. DMX code candidate

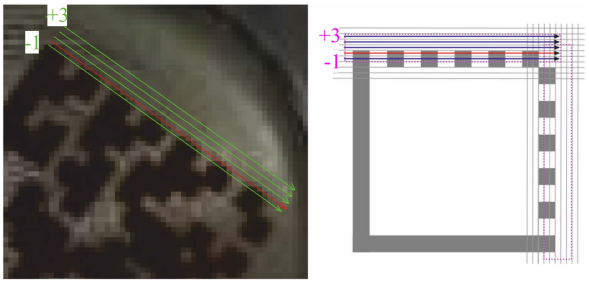


Fig. 10. Projection along the timing pattern

We can describe this procedure as follows:

1. We shift the starting location of line segment P1-P2 by 5 points away from region O in the direction perpendicular to this line. In this way we get a new position of the line segment P1-P2, now P1'-P2', see Fig. 8.

2. Then we move line segment P1'-P2' closer to the region O, approaching alternatively by one point firstly point P1' (so we get the new position P1") and then we make the same with point P2' (we get a new position P2") until the line touches region O at least in one point.

3. In every step of approaching we count the number of points which are common to the first half of the line segment (P1-C) and region O. We make the same for the other half of the line segment (C-P2). In next approaching we move only that end of the line segment which has smaller overlapping with region O.

4. We stop approaching when we have achieved at least 90% overlapping.

2.2 Verification of the timing pattern

We have a candidate for DMX defined by 4 vertices P1, P2, P3, P4, where P1-P2-P3 form the Finder Pattern. In the next step we verify whether the other two sides P1-P4 and P3-P4 correspond with the outlines, where "black" and "white" modules alternate (Fig. 9).

For the separation of the foreground and background points we have to set the threshold for region defined by P1, P2, P3, P4:

$$T_{\text{reg}} = \frac{1}{N} \sum_{i \in \text{ region}} I_i \,, \tag{2}$$

where N is the number of points in the region. Points with brightness over the threshold will be considered as "white" the others as "black" (we assume uniform lighting conditions in the region).

The bounding box does not define DMX code exactly, so we have to look for a sequence of black and white modules (the Timing Pattern). We have to search in wider surroundings of the initial bounding box (Fig. 10).

For every shift we count the number of black to white crossings (*ie* modules) and the sum of gradients. We look for the maximum sum of gradients (for non-uniform lightning, it is better to count the number of local extremes that differ by more than 10).

If the number of modules is at least 10 (minimum DMX size) and the same in both directions (both vertical and horizontal), then the DMX candidate will be marked as a valid DMX code.

2.3 Decoding of the data matrix

DMX code is made up by 2D matrix which consists of black and white modules representing bits 1 and 0. In a real image every such module corresponds to a group of neighbouring points. If we want to decode data in DMX code, we have to compose this 2D matrix that will have elements of 1 or 0 value. The size of the matrix is determined by the number of modules detected in the previous step.

Value 1 represents those points of image, where the brightness is lower than the threshold and value 0 represents the other points. We consider the decisive point will be the central point of the module and the brightness is determined by bilinear interpolation (if the calculated location of the central point does not correspond to integer coordinates in the image).

We used the open-source library libdmtx [16] for the final decoding of the binary matrix to receive the original text.

3 Experiments

The method mentioned above was tested on the test set consisting of 60 samples of DMX codes marked with laser on metal tools. Each sample was captured by 3 various cameras and we compared the impact of the cameras

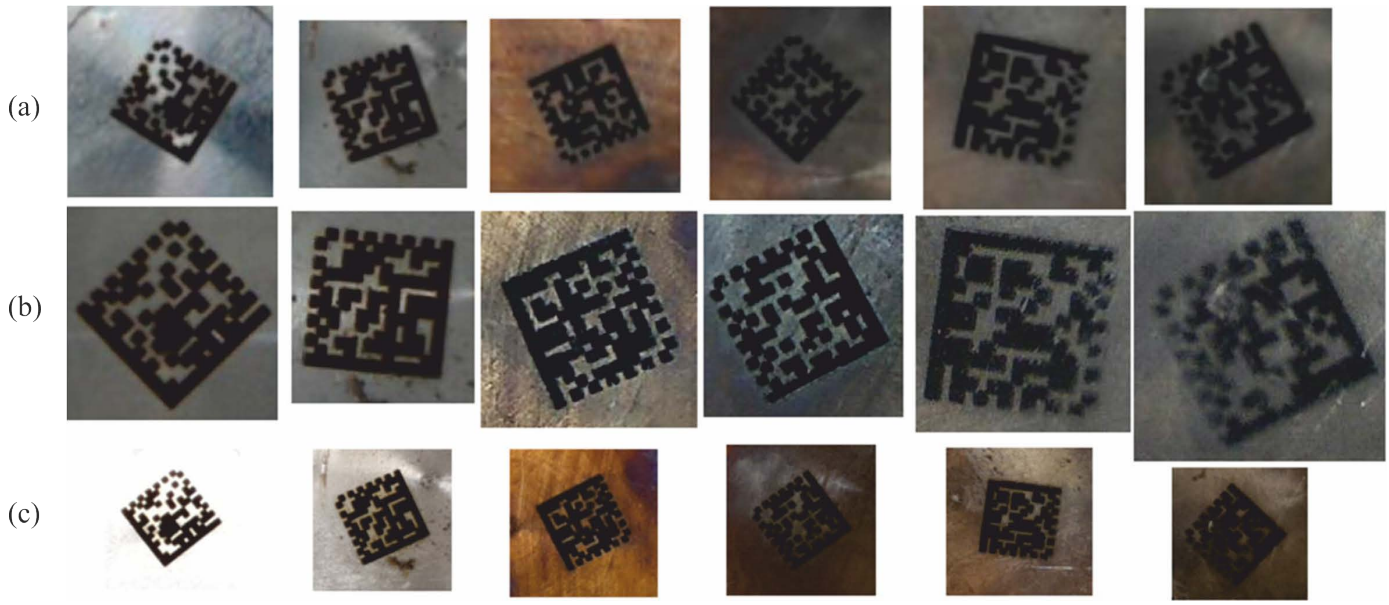


Fig. 11. Testing samples captured by various cameras

and of the image size on correct decoding of the DMX codes. The samples of the test codes are in Fig. 11, where a) mid-size samples were captured by Logitech USB Web Cam C920, b) big-size samples by Lindner digital USB microscope V6 and c) small-size samples by Raspberry Pi Camera Module V2.

Table 1. Results in location and decoding

Camera	size	location	decoding
Logitex C920	100	56 (93%)	55 (92%)
	+75	60 (100%)	58 (96%)
Lindner microscope V6	100	36 (60%)	32 (53%)
	+75	51 (85%)	49 (82%)
	+50	58 (96%)	56 (93%)
Raspberry Pi Camera V2	100	58 (96%)	58 (96%)
	+75	59 (98%)	59 (98)

Table 1 presents the results for images as they were captured and also combined results, where the location of DMX code runs for original images (100 % of original size) and subsequently for 75 % and 50 % of original size.

The results obtained by Lindner microscope were bad. Using the 100 % size the defects in material and DMX code were amplified. When we smooth the image with a 3×3 Gaussian kernel, we got better results. When we decreased the size of the image, we obtained a smoother image and even better results. We can achieve the best results by combining the smoothing and appropriate adoption of the size. In this way we can get the same results independent of the used camera.

In Tab. 2 there are our results compared against competitive DMX decoding software (open-source and also commercial). In the table there are the numbers of successfully recognized/decoded DMX codes out of a total of 60 codes for each of the three cameras.

4 Discussion

Our solution outperformed the competitive solutions. We have tested competitive software using their default

Table 2. Our results compared to competitive solutions

Software	Logitech C920	Lindner mic. V6	Raspberry cam. V2
Google ZXing (open-source) [17]	–	–	–
OnBarcode .NET Barcode Reader [18]	1	2	2
Dynamsoft Barcode Reader SDK [19]	32	18	19
LEADTOOLS data matrix SDK [20]	25	26	26
libdmtx (open-source) [16]	47	45	52
Inlite Barcode Reader SDK [21]	52	53	34
Our solution	58	56	59

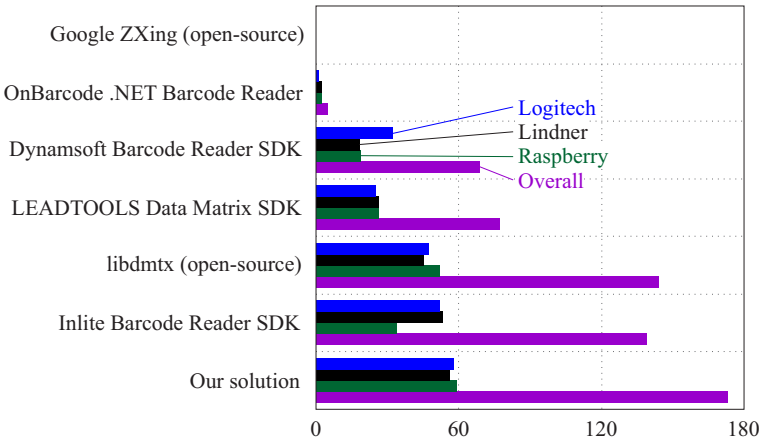
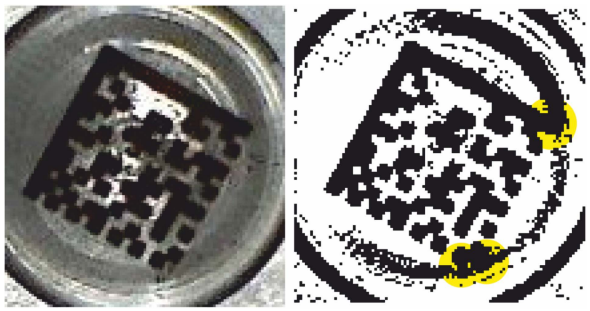
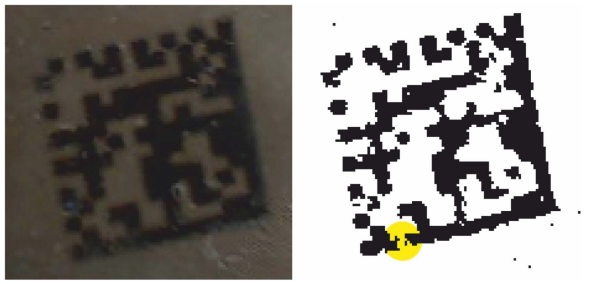


Fig. 12. Our results compared to competitive solutions



(a) (b)

Fig. 13. Imperfect local thresholding; (a) – original image, (b) – binarized image



(a) (b)

Fig. 14. Imperfect local thresholding; (a) – original image, (b) – binarized image

settings, mostly using an on-line decoding service, which does not allow us to set-up or tune-up recognition parameters. A surprise was the Google ZXing which failed to decode any of our testing DMX codes. Maybe the results of competitive software could be better if we had the possibility to adjust the recognition parameters. We have performed black-box testing, because most of competitive decoders are closed source solutions and therefore we cannot compare their and our detection algorithms in detail.

Most frequent reasons of failure in the location were structural defects on the surface of the tools which caused light reflections and they consequently failures in local thresholding (Fig. 13). These defects caused also discontinuity of the regions and thereby an incomplete identification of data matrix in image (Fig. 14).

As a solution to overcome these problems we have tested these improvements to local thresholding.

When the intensity of the central point of a sliding window falls between the local intensity average and average minus delta, then we have shifted the local window in 4 directions (up, down, left, right) and we have compared the new local intensity average with the intensity of the central point. If any difference was greater than delta, then we have marked such a point as black, else it remains as white.

Decreasing of the local threshold for bright points (with intensity above 60) by $[I(x, y - 60]/5$, because the brighter the point, the lower the probability that such a point is part of "black" DMX code.

Decreasing of the local threshold for edge points (with magnitude above 20) by 10, because edge points are expected to form borders between foreground and background points.

However, all of the above mentioned improvements bring only slightly better results.

We have proposed and tested also a modified local thresholding technique which combines the local point intensity, the mean and the variance under sliding window and calculates the local threshold as follows

$$T(x, y) = m(x, y) - \frac{I(x, y)}{k_1} - \frac{s^2(x, y)}{k_2} \qquad (3)$$

where $m(x, y)$ and $s^2(x, y)$ are the local mean and variance of the pixels inside the local window and k_1, k_2 are constants. We have empirically set k_1 to 10 and k_2 to 120–130, where k_1 controls penalisation of bright points (DMX code in image is the darkest object) and k_2 controls decreasing of local threshold for points in which neighbourhood intensity significantly varies. This technique can be effectively implemented by using integral sum images of $I(x, y)$ and $I^2(x, y)$ and unlike the Niblack's and the Sauvola's techniques this one does not require computing the square root for every point to get standard deviation and we have observed that it is less sensitive to selection of k parameters.

As a viable solution we have used also iterative global thresholding with predefined fixed set of thresholds (30, 40, 50, 60, 70, 80, 90). At least one of these thresholds successfully differentiates the foreground and background points in images in our testing set.

5 Conclusion

We have proposed and tested a computationally efficient method for location of 2D data matrix codes in images. This method is suitable also for real time processing. The proposed method uses typical patterns of data matrix codes, the so-called finder pattern and timing pattern to identify data matrix codes in images. This method utilizes local thresholding, connected component labelling, outer region boundaries to localize finder pattern. We count local extremes to verify the opposite dotted timing pattern.

We have proposed and modified a computationally efficient local thresholding technique which uses the local mean and variance under sliding window. This technique has achieved significantly better results in the special cases which we had to resolve than the classic local thresholding that uses only the local mean and delta constant.

We have shown that image resolution has an impact on the recognition rate and we have also shown that iterative image rescaling and following recognition can further improve the results.

This method was validated on the testing set from a real industrial world and it was compared with competitive solutions. The experimental results show that our method has a better detection rate than those mentioned solutions.

The application of data matrix codes and their optical recognition has a wide use in the identification, tracing or monitoring of items in production, storing and distribution processes. It could be possible to use data matrix codes for enhancing the reliability of robot control processes [22, 23], or consider the use the water flow algorithm [24] or an algorithm based on the oriented anisotropic gaussian kernel [25].

Acknowledgements

This paper was prepared within the work on a research project KEGA MŠ SR 003TU Z-4/2016: Research and education laboratory for robotics.

References

[1] L. Karrach and E. Pivarčiová, "Data Matrix Code Location Marked with Laser on Surface of Metal Tools", *Acta facultatis technicae* vol. 22, no, 2, 2017, pp. 29–38.

[2] L. Karrach and E. Pivarčiová, "The Analyse of the Various Methods for Location of Data Matrix Codes Images", *Elektro 2018: 12th International Conference Mikulov*, 2018.

[3] Q. Huang. W-S. Chen, X-Y. Huang and Y-Y Zhu, "Data Matrix Code Location Based on Finder Pattern Detection and", *Mathematical Problems in Engineering*, 2012.

[4] J. B. Burns, R. Hansona and M. Risemane, "Extracting Straight Lines", *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1986.

[5] R. G. von Gioi, J. Jakubowicz, J. M. Morel and G. Randall, "LSD: a Fast Line Segment Detector with a False Detection Control", *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2010.

[6] H. Donghong, T. Hui and C. Xinmeng, "Radon Transformation Applied Two Dimensional Barcode Image Recognition", *Journal of Wuhan University* 2005.

[7] Z. Chenguang, Y. Na and H. Rukun, "Study of Two Dimensional Barcode Identification Technology based on Hough Transform", *Journal of Changchun Normal University* 2007.

[8] Jeng-An, Lin and Chiou-Shann Fuh, "2D Barcode Image Decoding", *Mathematical Problems in Engineering* 2013.

[9] P. Gaur and S. Tiwari, "Recognition of 2D Barcode Images Using Edge Detection and Morphological Operation", *International Journal of Computer Science and Mobile Computing* 2014.

[10] S. Li, J. Shang, Z. Duan and J. Huang, "Fast Detection Method of Quick Response Code based on Run-Length Coding", *IET Image Processing* vol. 12, no. 4, 2018, pp. 546–551.

[11] D. K. Hansen and K. Nasrollahi, "Real-Time Barcode Detection and Classification Using Deep Learning", *9th International Joint Conference on Computational Intelligence* 2017.

[12] D. Bradley and G. Roth, "Adaptive Thresholding Using the Integral Image", *Journal of Graphics Tools* 2007.

[13] W. Niblack, "An Introduction to Digital Image Processing", 1986.

[14] J. Sauvola and M. Pietikainen, "Adaptive Document Image Binarization", *Pattern Recognition* vol. 33, 2000, pp. 225–236.

[15] A. Rosenfeld and J. Pfaltz, "Sequential Operations Digital Image Processing", *J. ACM* 1966, pp. 471–494.

[16] M. Laughton, "Open Source Software for Reading and Writing Data Matrix Barcodes" 2011, <https://github.com/dmtx>.

[17] Google, "ZXing" ("Zebra Crossing") Barcode Scanning Library for Java Android", <https://github.com/zxing>, Accessed April 12, 2018.

[18] OnBarcode, "NET Barcode Reader Component" URL: http://www.onbarcode.com/products/net_barcode_reader, Accessed April 12, 2018.

[19] "Dynamsoft, Barcode Reader SDK", URL: https://www. dynamsoft. com/Products/Dynamic-Barcode-Reader.aspx, Accessed April 12, 2018.

[20] Leadtools, "Data Matrix SDK", URL: https://www. leadtools. com/sdk/barcode/2d-datamatrix, Accessed April 12, 2018.

[21] Inlite Research Inc. "Barcode Reader SDK", URL: https://online-barcode-reader.inliteresearch.com, Accessed April 12, 2018.

[22] Y. Turygin, P. Božek, Y. Nikitin, E. Sosnovich and A. Abramov, "Enhancing the Reliability of Mobile Robots Control Process via Reverse Validation", *International Journal of Advanced Robotic Systems* vol. 13, no. 6, 2016, pp. 1–8.

[23] P. Božek, "Robot Path Optimization for Spot Welding Applications Automotive Industry", *Tehnicki Vjesnik – Technical Gazette* vol. 20, no. 5, 2013, pp. 913–917.

[24] D. Brodić, "Text Line Segmentation with Water Flow Algorithm based on Power Function", *Journal of Electrical Engineering* vol. 66, no. 3, 2015, pp. 132–141.

[25] D. Brodić and Z. N. Milivojević, "Text Line Segmentation with the Algorithm based on the Oriented Anisotropic Gaussian Kernel", *Journal of Electrical Engineering* vol. 64, no. 4, 2013, pp. 238–24.

Received 6 March 2018