Improving Data Matrix mobile recognition via fast Hough transform and adaptive grid extractors

E.O. Rybakova [1], E.E. Limonova [1,2], P.V. Bezmaternykh [1,2]
[1] Smart Engines Service LLC, Prospekt 60–Letiya, Oktyabrya, 9, Moscow, 117312, Russia;
[2] Federal Research Center "Computer Science and Control" of Russian Academy of Sciences, Ulitsa Vavilova, 40, Moscow, 119333, Russia

Abstract

The Data Matrix is a barcode symbology originally designed for industrial needs. Today, its symbols are increasingly found on everyday products such as pharmaceutical packaging, electronic components, food labels, and clothing tags. This widespread usage presents a challenge: reading Data Matrix symbols from images captured by mobile cameras in uncontrolled environments. The reading process mainly consists of three steps, namely barcode localization, segmentation and decoding. In this work, we focus on the precise localization and segmentation of Data Matrix barcodes. We introduce a new method that involves the localization of the finder pattern using fast Hough transform and subsequent iterative segmentation to extract the encoded message. Our approach demonstrates superior localization quality, as measured by the mean Intersection over Union metric (0.889), and achieves better recognition accuracy (0.903) compared to open–source solutions for reading Data Matrix barcodes, such as libdmtx (0.665), ZXing (0.569), and ZXing–cpp (0.858). Our method requires only 35 milliseconds for computations on an ARM device, enabling real–time processing. It is significantly faster than libdmtx (10 seconds), ZXing (610 milliseconds), although it is slightly slower than ZXing–cpp (6.65 milliseconds).

Keywords: barcode localization; barcode segmentation; Data Matrix; fast Hough Transform; mobile recognition.

Citation: Rybakova EO, Limonova EE, Bezmaternykh PV. Improving Data Matrix mobile recognition via fast Hough transform and adaptive grid extractors. Computer Optics 2025; 49(6): 1182-1190. DOI: 10.18287/COJ1804

Introduction

Designed to be machine–readable, barcodes are well–suited for transferring information to camera–equipped devices, which has led to their widespread use and made barcode recognition on mobile devices a routine practice. It means that modern barcode readers have to:

- function effectively with images captured in uncontrolled conditions from mobile cameras,
- operate on the user's devices without Internet access,
- provide fast and accurate recognition.

It makes barcode recognition a challenging problem.

A general pipeline for barcode recognition through image processing was outlined as early as 1999 [1] and has not conceptually changed since then. In summary, this pipeline consists of four steps: (*a*) detecting the region of interest (ROI), (*b*) precise localization of the barcode, (*c*) segmenting the barcode, and (*d*) decoding the message.

There are many different types of barcodes, known as symbologies. Some of them, like QR codes, are common and have extensive datasets available for training and testing recognition algorithms [2, 3, 4]. As a result, various practical methods support working with QR codes and demonstrate high accuracy. However, for some symbologies, such as Data Matrix (DMX) codes (see example in Fig. 1), the training and testing data are often limited or not publicly accessible despite their growing usage among end–users. Today, these codes are commonly used in various applications and can be found in pharmaceutical packaging, metal parts, clothing, bottles, and other products.

In this paper, we propose a new method for precise localization and consecutive grid extraction of Data Matrix barcodes captured by mobile devices. By grid extraction, we refer to the process of mapping the segmented barcode region onto its two–dimensional module grid. Of course, there are a number of existing methods. However, these methods have only been thoroughly tested and compared for QR codes due to a lack of realistic public data for DMX codes. So, there is limited experimental evidence on their accuracy and operating time in this problem. Fortunately, at the SPRA 2024 conference, the SE–DMTX–SYN–1000 dataset [5] was introduced. It contains semi–synthetic complex data for the precise localization of Data Matrix codes, which can be used to estimate localization and decoding accuracy.

We propose a new method based on image processing techniques, which first localizes the DMX quadrilateral using several approaches, including the fast Hough transform [6], and then applies iterative segmentation for bit matrix extraction. We evaluate it on the SE–DMTX–SYN–1000 dataset and demonstrate high localization accuracy compared to open–source libraries for reading barcodes, such as libdmtx, Zxing, and Zxing–cpp. The resulting decoding accuracy is also superior to other methods. Moreover, our method has a competitive running time of about 35 ms on an ARM–based device, which proves its value for mobile barcode recognition.

a) b)

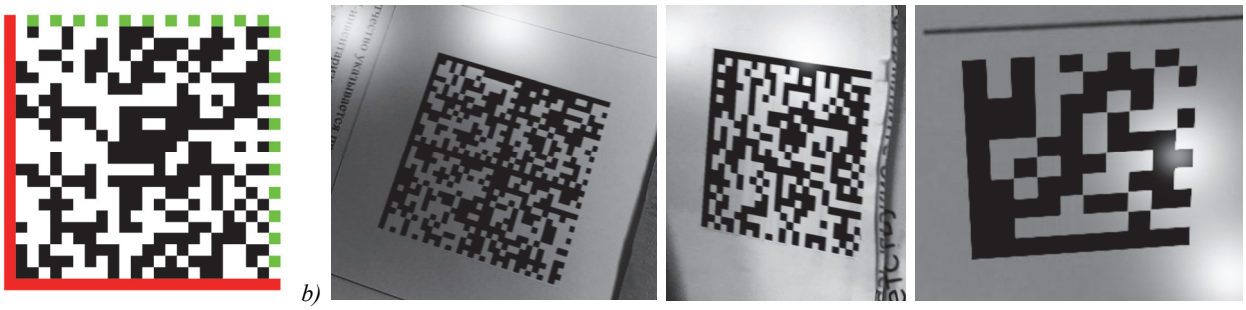


Fig. 1. a) The Data Matrix structure: finder pattern (red) and timing pattern (green); b) sample images from SE–DMTX–SYN–1000 dataset

Problem description

The instance of a barcode is known as a symbol. Its logical elements of dark and light colors are called modules. Each of them contains one bit of information defined by color. The Data Matrix specification [7] includes two mandatory groups of modules: the finding pattern (L–pattern) and the timing pattern. These elements are designed to establish the orientation of the symbol. The orientation knowledge is necessary to correctly construct a matrix of module values and decode the message. Also, the specification states the presence of the quiet zone, i.e., an area of plain color around the Data Matrix symbol. This zone serves to simplify code detection. A standard structure of Data Matrix code is illustrated in Fig. 1a.

This paper addresses the precise localization and bit matrix extraction for Data Matrix symbols within images captured by a handheld camera. The camera is assumed to have no optical aberrations and can be modeled as a pinhole camera. The Data Matrix symbol is expected to be visible and occupy at least 25 % of the image area. We can achieve it through specific capturing conditions or a Data Matrix ROI detector, such as the one described in [8]. Also, we focus on the common scenario where the Data Matrix symbols are on flat surfaces. It means the symbol's border in the image is a convex quadrilateral [9].

We define the outcome of the precise localization process as an oriented quadrilateral $Q = \{p_1, p_2, p_3, p_4\}$, where $p_j \in \mathbb{R}^2$ for $j = \{1, \dots, 4\}$. The vertices are ordered in a clockwise manner from the top–left to the bottom–left vertex with respect to the barcode coordinate system. Let I represent the input image of the Data Matrix symbol. The localization problem can then be formulated as follows:

$$\mathcal{L}(I) = Q, \tag{1}$$

where $\mathcal{L}$ is a localization method to be developed. The result of the localization step is subsequently used to calculate the projective transform $\mathcal{H}$ such that

$$\mathcal{H}(G) = Q, \tag{2}$$

where G is a grid quadrilateral defined in the barcode coordinate system, corresponding to the unit square of the ideal module grid. Applying $\mathcal{H}$ to grid nodes allows the construction of a distorted grid in the input image I, and further estimates of module colors, which is the step of barcode segmentation. Finally, we obtain a matrix of 0's and 1's, i.e., a bit matrix, encoding the DMX message.

Related work

Research on Data Matrix symbol localization includes classical image processing methods, as well as techniques based on feature extraction and machine learning. Due to the widespread use of QR codes, the majority of published papers in the literature focus on them. In contrast, there has been significantly less research on Data Matrix codes, although some studies do address this topic. For instance, in [10], the author presented a method for locating and decoding industrial Data Matrix codes.

Classical image processing techniques for Data Matrix symbol localization primarily leverage geometric characteristics, focusing on finder pattern detection and contour analysis, because Data Matrix codes have a distinguishable "L"–shaped finder pattern. Most studies require preliminary image binarization [11, 12, 13], simplifying and accelerating recognition. However, binarization demands careful fine–tuning, which becomes challenging under uncontrolled conditions typical of mobile cameras, including uneven illumination, flares, and projective distortions from camera tilt or rotation [10] that reduce the effective resolution of barcode modules and make thresholding unreliable. To avoid these issues, methods operating directly on grayscale images are preferable.

In 2021, Kollar and Pivarčiová [14] comprehensively evaluated various methods that rely heavily on detecting finder patterns either by binarization with subsequent connection of foreground points or by edge detection and connection of edge points. Huang et al. [15] proposed techniques that detect solid finder pattern via line segment detector (LSD) algorithm and employ barcode border fitting via Random Sample Consensus (RANSAC) [16] algorithm. Another

approach for fitting a dotted timing pattern is to use Radon or Hough transforms that focus less on the connectivity of segments. Thus, Donghong et al. [17] proposed an algorithm based on the Radon transform, and Chenguang et al. [1] introduced an algorithm based on the Hough transform. But both noted for their high computational complexity. It can be solved with the fast Hough Transform via Brady–Yong algorithm, which is widely used for computationally efficient image processing [6, 18, 19, 20].

Feature–based methods utilize invariant or regional image features, including geometric, color, or gradient–based descriptors [11]. Gradient–based features have been prominently employed for contour extraction and tracking due to their robustness against environmental variability [12, 21]. Connected component clustering methods exploit the rectangular shape of Data Matrix symbols to facilitate cluster rejection, streamlining localization tasks [10, 22]. Preliminary keypoint detection also reduces computational demands, improving method efficiency [23].

In recent years, machine learning approaches, in particular deep learning, have been introduced that demonstrate high performance in complex scenarios. Dong and Zhang [12] presented a real–time localization algorithm in 2020. This algorithm integrates classical contour detection with machine learning–based validation. It effectively detects multiple Data Matrix symbols simultaneously. In 2023, Loh et al. [24] demonstrated the potential of hybrid strategies by integrating deep learning–based object detection with YOLOv5 and traditional image processing techniques, achieving both accuracy and real–time capabilities suiTab. for industrial applications. In 2024, Xu et al. [25] combined YOLOv5–based coarse localization with precise localization to detect L–shaped dashed edges accurately, significantly improving robustness against interference and distortion. Similarly, Matuszczyk and Weichert [26] employed deep learning models for recognizing low–contrast Data Matrix codes in additive manufacturing contexts, demonstrating superior accuracy over traditional image processing techniques. Nevertheless, machine learning approaches require large labeled training datasets, which are limited for Data Matrix barcodes. It means that images for these datasets are usually generated and may lack real scenarios, like poor illumination, flares, low resolution, etc.

Moreover, neural network–based methods can slow down processing, which is often problematic for mobile devices. On the other hand, Data Matrix codes possess excellent visual characteristics, making them easily distinguishable in images. This quality makes classical methods still valuable for precise localization when the region of interest has already been identified. A significant issue in existing research is the lack of comparisons using public data. In this paper, we evaluate a classical approach to precise localization and segmentation of Data Matrix codes using a new public dataset, SE–DMTX–SYN–1000. We demonstrate that this approach achieves high quality and speed, even when handling complex data.

The proposed method

Precise localization

Here, we focus on the task of precise localization, assuming that the region of interest (ROI) has already been identified. Further in text, methods work with either grayscale or binary images. Where binarization is mentioned, we use two alternatives: global and hybrid. Global binarization uses a single threshold derived from the overall brightness histogram of the image. This method is quick but can be sensitive to uneven lighting conditions. In contrast, hybrid binarization divides the image into smaller blocks, calculates a local threshold for each block, and then merges the results. This approach enhances robustness against shadows and brightness gradients.

Finder pattern detection

First, a segment detector is applied to a grayscale image to identify line segments. We use the Canny edge detector to identify edges and apply minimum and maximum length restrictions. The detected segments are then grouped into pairs of finder pattern candidates based on geometric criteria and subjected to three rejectors designed to eliminate false matches:

1. Length–based rejector

Both segments in a candidate pair (s_1, s_2) must have approximately equal lengths within a predefined tolerance:

$$\frac{(\min(\|s_1\|,\|s_2\|))}{(\max(\|s_1\|,\|s_2\|))} < T_l, \tag{3}$$

where $\| s_1 \|$ and $\| s_2 \|$ are lengths of segments s_1, s_2, $T_l \in (0,1]$ is the length ratio threshold. If the ratio of their lengths exceeds the specified threshold, the pair is discarded.

2. Distance–based rejector

If the two segments s_1 and s_2 are positioned too far apart relative to each other, the pair is excluded from further consideration, i.e., we require

$$\rho(s_1, s_2) < T_d, \tag{4}$$

where ρ is the minimum distance between segments.

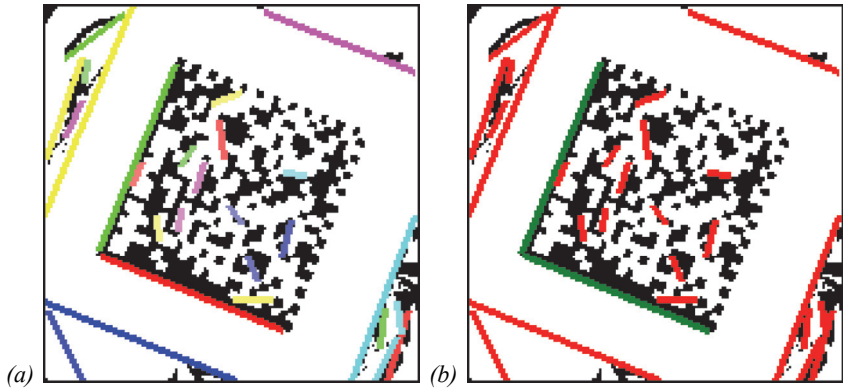
3. Angle–based rejector

The angle between the straight lines on which the segments s_1, s_2 lie is measured. If its value is not within the specified range, then the pair is rejected. The following condition must hold:

$$\angle(s_1, s_2) \in [\pi/2 - T_a; \pi/2 + T_a] \quad (5)$$

for some threshold T_a.

In practice, these three criteria are sufficient to remove the vast majority of unsuitable segment pairs, leaving only those most likely to form valid finder patterns. An example is shown in Fig. 2.



(a) (b)

Fig. 2. Finder pattern detection example (binarized image): a) detected segments, b) finder pattern segments (green) and rejected segments (red)

Quadrilateral extraction

To construct a complete quadrilateral from a detected finder pattern, we work with binarized images and use three different approaches that are applied in cascade. Despite differences in implementations, all of these approaches share a common initial stage. In the first step, the fourth vertex P_{tr} (top–right corner) is computed from the three known vertices of the L–pattern: P_{tl} (top–left corner), P_{bl} (bottom–left corner), and P_{br} (bottom–right corner) (see Fig. 3). The position of the top–right corner is determined from the geometric relation

$$P_{\text{tr}} = P_{\text{tl}} + P_{\text{br}} - P_{\text{bl}}. \quad (6)$$

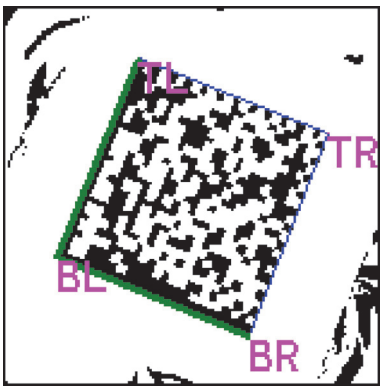


Fig. 3. Data Matrix quadrilateral with highlighted finder pattern

This is how an initial oriented quadrilateral is formed, which is then refined by three methods described below.

1. Edge refinement and intersection (CD1)

The initial quadrilateral is taken as the first approximation. The top and right edges are refined by searching along them for black–white transitions, after which the corner is shifted to align with the detected boundary lines. The intersection of refined edges defines the updated top–right corner $\hat{P}_{\text{tr}}$. Geometric proportion check

$$\| \hat{P}_{\text{tr}} - P_{\text{tr}} \|_2 \; < \; \| P_{\text{tl}} - P_{\text{bl}} \|_2 \; / \; 3 \quad (7)$$

and inside–image constraints are applied. If the conditions are not satisfied, the process terminates without producing a result; if all checks pass, a single final quadrilateral is returned.

2. Rectangular search in a limited region with validation (CD2)

Rather than refining edges locally, an expanded region of interest is created, within which a white rectangular contour corresponding to the quiet zone surrounding the Data Matrix code is identified under the assumption that the projective distortions are small. The detected quadrilateral must satisfy several constraints: a white rectangular contour must be present within the expanded region and its side length must satisfy $\|\cdot\| \geq 20$ px. If any of these constraints fail, the original quadrilateral from the L–pattern is used. If all passes, the detected contour is expanded by several small offsets, transformed using the current geometric transformation, and multiple candidates are generated.

3. Fast Hough transform line detection and intersection (CD3)

Two fixed regions of the image are selected: a top band and a right band. In each of them, the dominant straight line is found using the fast Hough transform (Fig. 4). These lines are intersected with the corresponding edges of the initial quadrilateral: the top line with the left and right edges to refine the top–left and top–right corners, and the right line with the top and bottom edges to refine the top–right and bottom–right corners. If reliable detection (determined by the presence of a distinct peak in the Hough space) of at least one of the lines fails, the original quadrilateral is used; exactly one result is always produced.

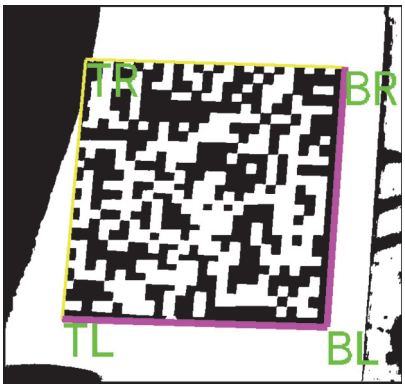


Fig. 4. An example of CD3 corner detection method. Yellow lines were found with the fast Hough transform

The first variant refines edges locally and uses intersection with geometric and boundary checks, producing at most one result. The second searches for a rectangle in a limited area, applies explicit scale and geometric constraints, and may produce multiple expanded candidates. The third applies the fast Hough transform to detect two lines in fixed bands, without scale or dimensional validation, and always produces exactly one result. Both the first and third variants rely on intersections, but the first derives edges from local analysis, while the third obtains them via Hough detection in predefined regions.

Then we finally adjust the corners by shifting P_{tl}, P_{br} along the diagonal $P_{\text{tl}}P_{\text{br}}$ (and P_{tr} along $P_{\text{bl}}P_{\text{tr}}$) inside the Data Matrix region until reaching foreground pixels, i.e. the pixels of the barcode modules that carry information. In such a way, we can correct small corner position errors before further processing.

Symbol segmentation

At this stage, all detected quadrilaterals are segmented to extract bit matrices. We launch four different grid extractors in cascade to obtain the encoded message, which is sent to the decoder. If the decoding is successful and the message corresponds to the DMX code, we exit processing.

Before performing grid extraction, we need to determine the dimensions of the barcode, i.e., the number of modules along its rows and columns, as well as the corresponding module size. In this stage, we estimate these parameters by analyzing the binarized image and the detected quadrilateral borders of the code. We measure the number of black–to–white transitions along the top and right edges of the quadrilateral, adjust the counts to match valid Data Matrix sizes, and classify the code shape as either square or rectangular based on the ratio of these counts.

Grid extraction methods described below operate by mapping an ideal rectangular grid of size (w, h), where each cell corresponds to one module of the Data Matrix code, onto the detected quadrilateral region in the source image. In the first three methods, this mapping is implemented as a projective transformation, providing a geometric correspondence between grid coordinates and image coordinates. The subsequent steps differ between methods in how they handle local distortions, determine sampling positions within each cell, and decide the final binary value of the module.

1. Simple grid extractor (GE1)

The method receives a binarized source image S, the grid dimensions $(w,\ h)$, and a projective transformation that maps coordinates of an ideal rectangular grid to the detected quadrilateral in the source image. An "ideal" grid cell layout is assumed, where each cell corresponds to one module of the Data Matrix code. For each grid cell, its geometric center is projected into the source image, and the binary value of the cell is determined by inspecting the corresponding pixel in S. This approach directly transfers the geometric structure of the ideal grid to the image, without attempting to correct for potential local misalignments. It is simple and fast, and works well when the projective transformation is accurate and the distortion of the printed or imaged code is minimal.

2. Adaptive grid extractor (GE2)

The method receives a binarized source image S, the grid dimensions (w, h), and a projective transformation mapping the ideal grid to the detected quadrilateral in the source image. Unlike the simple grid extractor, here the extraction is performed in a block–wise manner: the grid is divided into smaller contiguous regions according to the expected number of rows and columns specified by the barcode version, and for each block an independent local projective transformation is computed. This allows the method to correct local geometric distortions that cannot be fully compensated by a single global transformation.

Within each block, the sampling position for each grid cell is initially obtained by projecting its ideal location into the source image. This position is then refined by adjusting it according to the observed characteristics of the local image region, aiming to align more closely with the true center of the corresponding module. The results from all blocks are merged into the final binary grid, benefiting from both local geometric corrections and adaptive sampling.

3. Grey grid extractor (GE3)

The method receives a grayscale image G, the grid dimensions (w, h), and a mapping between the ideal grid and the detected quadrilateral in the source image. The grid is sampled in the grayscale domain rather than from a binarized image, allowing the extraction process to account for subtle intensity variations before thresholding. For each grid cell, the ideal cell center is projected into the source image, and the average grayscale value is computed over a small neighborhood around this position to reduce noise.

The sampled grayscale grid is then locally thresholded: for each cell, its intensity is compared to the mean intensity of its surrounding cells, and it is marked as black if it is darker than this local average. This local adaptive thresholding makes the method more robust to uneven illumination, shadows, or gradual shading across the code, since binarization is performed relative to the local background rather than using a constant global threshold.

4. Simplified adaptive grid extractor (GE4)

The method receives a binarized image S, the grid dimensions (w, h), and the detected quadrilateral in the source image. It detects the grid structure to establish correspondences between ideal grid coordinates and their positions in the source image, then divides the grid into blocks like the adaptive grid extractor. For each block, a local projective transform maps the center of each cell to the source image, and the binary value at that location is copied into the output grid.

Experiments

Data and metrics

We are utilizing the recently introduced SE–DMTX–SYN–1000 dataset [5], which can be accessed at ftp://smartengines.com/se–dmtx–syn–1000. This dataset includes Data Matrices that cover at least 25% of the total image area. It comprises 1,000 barcodes set against naturally complex backgrounds, making it challenging for open–source readers. Sample images from the dataset are displayed in Fig. 1*b*.

To evaluate localization accuracy, we utilize the metrics outlined in [5]:

1. Mean Intersection Over Union (**mIoU**)

The mean value of the Jaccard index is defined as:

$$IoU\ (Q, M) = \frac{area\ (Q \cap M)}{area\ (Q \cup M)},$$

where Q is the detected Data Matrix quadrilateral, and M is the ideal quadrilateral from the ground truth.

2. Mean Average Precision (**mAP**)

The rate of correctly detected barcode quadrilaterals determined for some IoU threshold T:

$$mAP\ _T = \frac{|\{Q: IoU\ (Q,M) \geq T\}|}{|\{Q\}|},$$

where Q defines the detected Data Matrix quadrilateral, and M is the ideal quadrilateral from the ground truth.

3. Mean Maximum Point Distance (**mMPD**)

Mean maximum distance between the corresponding vertices of Q and M computed in the coordinate system of Q:

$$MPD\ (Q, M) = \max_i ||r_i - H(m_i)||, r_i \in R, m_i \in M,$$

where $R = \{(0,0), (1,0), (1,1), (0,1)\}$ is a unit square, M is quadrilateral from ground truth, and H is a homography to transform Q into R: $H(Q) = R$. So, we estimate the maximum residual between the points of Q and M.

4. Recognition accuracy (**ACC**)

The rate of correctly decoded barcodes.

Experimental setup

We consider the following methods for Data Matrix localization and recognition:

- libdmtx [27] library,
- Zxing [28] library,
- Zxing–cpp [29] library,
- the proposed method.

We implemented the proposed method in C++ and integrated Zxing–cpp into it for decoding the bit matrix to estimate the final recognition accuracy.

We launched the experiments on Jetson AGX Orin module with ARM Cortex–A78E v8.2 central processing unit for time measurements.

Method evaluation

The results of DMX processing are demonstrated in Tables 1, 2. The mIoU values from Tab. 1 indicate that the proposed method achieves the highest localization accuracy. It is followed by Zxing–cpp, libdmtx, and Zxing, which were unable to localize a lot of DMX instances. In terms of mAP, all tested implementations achieve a perfect score of 1.0 at the 0.7 threshold. At 0.9, libdmtx reaches 1.0, while Zxing–cpp and the proposed method achieve near–perfect scores of 0.998 and 0.995, respectively. At the 0.95 threshold, Zxing–cpp leads with a score of 0.995. High mAP values are also associated with low mMPD values. The accuracy is not so high: the highest value is 0.903 for our method, 0.858 for Zxing–cpp, whereas others are significantly lower. Fig. 5 demonstrates some examples where Zxing–cpp failed to recognize but the proposed method (version from the last row of Tab. 2) succeeded. Regarding processing time, the proposed method is the second, with 34 ms, while Zxing–cpp operates in 6.65 ms. However, Zxing takes 610 ms, and libdmtx takes 10310 ms, making them considerably slower. The substantial delay observed with libdmtx may be attributed to its Python implementation and the possible non–optimal parameter setting that leads to its inefficient processing time.

Tab. 1. The quality of barcode readers on SE–DMTX–SYN–1000 dataset.

Library	mIoU	$mAP_{0.7}$	$mAP_{0.9}$	$mAP_{0.95}$	mMPD	ACC	T, ms
Zxing	0.542	**1.0**	0.965	0.603	0.023	0.569	610
Zxing–cpp	0.847	**1.0**	0.998	**0.995**	**0.009**	0.858	**6.65**
libdmtx	0.653	**1.0**	**1.0**	0.981	0.015	0.665	10310
Ours	**0.889**	**1.0**	0.996	0.987	0.066	**0.903**	34.31

Tab. 2. Quality evaluation of the proposed method variations on SE–DMTX–SYN–1000 dataset.

Variants							mIoU	$mAP_{0.7}$	mAP	$mAP_{0.95}$	mMPD	ACC	T, ms
CD1	CD2	CD3	GE1	GE2	GE3	GE4							
	✓		✓	✓	✓	✓	0.827	**1.0**	**1.0**	**0.9964**	0.06548	0.839	**21.16**
✓		✓	✓	✓	✓	✓	0.871	**1.0**	0.9977	0.9898	0.06616	0.884	28.98
✓	✓	✓	✓				0.853	**1.0**	0.9976	0.9896	0.06551	0.866	24.33
✓	✓	✓	✓	✓			0.871	**1.0**	0.9988	0.9875	0.06587	0.884	27.78
✓	✓	✓	✓	✓	✓		0.873	**1.0**	0.9977	0.9864	0.06584	0.886	28.14
✓	✓	✓	✓	✓	✓	✓	**0.889**	**1.0**	0.9966	0.9878	**0.06627**	**0.903**	34.31

Tab. 2 shows an ablation study of the proposed method. At first, we enable quadrilateral corner detectors sequentially while keeping the grid extractor configuration fixed. It results in a gradual increase in quality, along with an increase in runtime. Similarly, when the corner detectors are kept fixed and the grid extractors are enabled sequentially, we observe the same trend. This suggests that all the described options effectively complement each other.

Overall, the evaluation demonstrates that the proposed method allowed us to noticeably increase localization and recognition accuracy for natural–looking Data Matrix from uncontrolled environments.

Fig. 5. Examples of images that our method successfully recognized, whereas Zxing–cpp failed

Conclusion

In this paper, we proposed a new method for the accurate recognition of distorted Data Matrix symbols and demonstrated its running time and quality using the publicly available dataset SE–DMTX–SYN–1000. We performed finder pattern localization based on segment detection and used the fast Hough Transform to identify the quadrilateral of the barcode. We extracted several quadrilateral candidates and processed them iteratively to extract bit matrices for further decoding.

Experimental evaluation of localization accuracy and processing time on an ARM device demonstrated that our method completes computations in just about 35 milliseconds, which is significantly faster than Zxing (610 milliseconds), libdmtx (10 seconds), and slightly worse than Zxing–cpp (6.65 milliseconds). Additionally, our method demonstrated the best results according to mean Intersection over Union (0.889) and final decoding accuracy (0.903).

Our research indicates that classical image processing techniques can be effectively employed to locate Data Matrix barcodes within complex datasets accurately. The processing speed on the ARM processor makes this approach well–suited for real–time processing on modern mobile and edge devices.

The method also allows generalization to other barcode symbologies in the following way. The supporting elements can be identified using symbology–specific methods, while segmentation and decoding can be accomplished using the proposed method.

The main limitations of the proposed approach concern challenging scenarios with low–quality images. Since the dataset used in our experiments does not contain extremely degraded samples, such cases were beyond the scope of this study. Future work will include a more detailed error analysis and investigation of potential improvements.

References

[1] E. Ouaviani, A. Pavan, M. Bottazzi, E. Brunelli, F. Caselli, M. Guerrero, A common image processing framework for 2D barcode reading, in: Image Processing And Its Applications, 1999. Seventh International Conference on (Conf. Publ. No. 465), Vol. 2, IET, 1999, pp. 652–655. DOI: 10.1049/cp:19990404

[2] A. Zharkov, A. Vavilin, I. Zagaynov, New benchmarks for barcode detection using both synthetic and real data, in: International workshop on document analysis systems, Springer, 2020, pp. 481–493. DOI: 10.1007/978–3–030–57058–3_34

[3] I. Szentandrási, A. Herout, M. Dubská, Fast detection and recognition of QR codes in high–resolution images, in: Proceedings of the 28th spring conference on computer graphics, 2012, pp. 129–136. DOI: 10.1145/2448531.2448548

[4] P. Bodnár, T. Grósz, L. Tóth, L. G. Nyúl, Efficient visual code localization with neural networks, Pattern Analysis and Applications, vol. 21, 2018, pp. 249–260. DOI: 10.1007/s10044–017–0619–6

[5] E. Limonova, P. Zlobin, P. Bezmaternykh, Generation of semisynthetic natural–looking 2D barcodes for localization problems, in: Fifth Symposium on Pattern Recognition and Applications (SPRA 2024), vol. 13540, SPIE, 2025, p. 135400I. DOI: 10.1117/12.3056438

[6] D. Nikolaev, S. Karpenko, I. Nikolaev, P. Nikolaev, Hough transform: Underestimated tool in the computer vision field, Proceedings – 22nd European Conference on Modelling and Simulation, ECMS 2008, pp. 238–243. DOI: 10.7148/2008–0238

[7] Information technology – Automatic identification and data capture techniques – Data Matrix barcode symbology specification, Edition 3, International Organization for Standardization, 2024.

[8] D. M. Ershova, A. V. Gayer, P. V. Bezmaternykh, V. V. Arlazarov, YOLO–Barcode: towards universal real–time barcode detection on mobile devices, Computer Optics, vol. 48, n. 4, 2024, pp. 592–600. DOI: 10.18287/2412–6179–CO–1424

[9] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, 2nd Edition, Cambridge University Press, New York, NY, USA, 2003.

[10] A. Y. Kruchinin, Industrial Data Matrix barcode recognition with random tilt and rotate the camera, Computer Optics, vol. 38, n. 4, 2014, pp. 856–864. DOI: 10.18287/0134–2452–2014–38–4–865–870

[11] Z. Liu, X. Guo, C. Cui, Detection algorithm of 2D barcode under complex background, Int. Proc Comput Sci Inf Technol, vol. 53, n. 1, 2012, pp. 116–117.

[12] Y. Dong, T. Zhang, A real–time algorithm for multiple Data Matrix codes localization, in: Advances in Guidance, Navigation and Control: Proceedings of 2020 International Conference on Guidance, Navigation and Control, ICGNC 2020, Tianjin, China, October 23–25, 2020, Springer, 2022, pp. 2477–2487.

[13] L. Karrach, E. Pivarčiová, Recognition of Data Matrix codes in images and their applications in production processes, Management Systems in Production Engineering, 2020. DOI: 10.2478/mspe–2020–0023

[14] L. Karrach, E. Pivarčiová, Comparative study of Data Matrix codes localization and recognition methods, Journal of Imaging, vol. 7, n. 9, 2021, p. 163. DOI: 10.3390/jimaging7090163

[15] Q. Huang, W.-S. Chen, X.-Y. Huang, Y.-Y. Zhu, Data Matrix code location based on finder pattern detection and bar code border fitting, Mathematical Problems in Engineering, vol. 1, 2012, p. 515296.

[16] M. A. Fischler, R. C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Communications of the ACM, vol. 24, n. 6, 1981, pp. 381–395.

[17] T. H. H. Donghong, C. Xinmeng, Radon transformation applied in two dimensional barcode image recognition, Journal of Wuhan University, vol. 5, n. 584, 2005.

[18] P. V. Bezmaternykh, D. P. Nikolaev, V. L. Arlazarov, High–performance digital image processing, Pattern Recognition and Image Analysis, 2023, pp. 743–755. DOI: 10.1134/S1054661823040090

[19] P. V. Bezmaternykh, Text image normalization using fast Hough transform, ITiVS, vol. 4, 2024, pp. 3–16. DOI: 10.14357/20718632240401

[20] I. Kunina, E. I. Panfilova, M. Povolotskiy, Zebra–crossing detection on road images using dynamic time warping, Proceedings of ISA RAS, vol. 68, 2018, pp. 23–31.

[21] O. Pârvu, A. G. Balan, A method for fast detection and decoding of specific 2d barcodes, in: Proceedings of the 17th Telecommunications forum TELFOR, 2009, pp. 1137–1140.

[22] L. Karrach, E. Pivarčiová, The analyse of the various methods for location of Data Matrix codes in images, in: 2018 ELEKTRO, 2018, pp. 1–6. DOI: 10.1109/ELEKTRO.2018.8398250

[23] L. K. Leong, W. Yue, Extraction of 2D barcode using keypoint selection and line detection, in: Advances in Multimedia Information Processing, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 826–835. DOI: 10.1007/978–3–642–10467–1_73

[24] S. H. Loh, P. C. Teh, J. J. Sim, C. K. Tai, K. H. Yeap, Y. J. Lee, A. U. Mazlan, Decoding dot peen Data Matrix code with deep learning capability for product traceability, Applications of Modelling and Simulation, vol. 7, 2023, pp. 38–48.

[25] Y. Liu, Y. Song, G. Gu, J. Luo, T. Wang, Q. Jiang, A Data Matrix code recognition method based on l–shaped dashed edge localization using central prior, Sensors, vol. 24, n. 13, 2024, p. 4042. DOI: 10.3390/s24134042

[26] D. Matuszczyk, F. Weichert, Reading direct–part marking Data Matrix code in the context of polymer–based additive manufacturing, Sensors, vol. 23, n. 3, 2023, p. 1619. DOI: 10.3390/s23031619
[27] libdmtx – Open Source Data Matrix Software, https://github.com/dmtx/libdmtx , (Accessed 07.10.2025).
[28] ZXing ("Zebra Crossing") barcode scanning library for Java, Android, https://github.com/zxing/zxing , (Accessed 07.10.2025).
[29] C++ port of ZXing, https://github.com/zxing–cpp/zxing–cpp , (Accessed 07.10.2025).

About authors

Ekaterina Olegovna Rybakova, (b. 1999) obtained her Specialist's degree from the Faculty of Mechanics and Mathematics, Lomonosov Moscow State University in 2023. Since 2021, she has worked as a programmer at Smart Engines Service LLC. Research interests: mathematical programming, computational optimization.
E–mail: *e.rybakova@smartengines.com*

Elena Evgenevna Limonova, (b. 1993) completed Master's degree (2017) in Applied Mathematics and Physics at Moscow Institute of Physics and Technology. Received her Ph.D. degree in Computer Science in 2023 at FRC "Computer Science and Control" RAS. Currently, she works as a researcher at the FRC "Computer Science and Control" RAS and as a programmer at Smart Engines Service LLC. Research interests: computer vision, artificial neural networks, image recognition on mobile devices. E–mail: *limonova@smartengines.com*

Pavel Vladimirovich Bezmaternykh, (b. 1987), received a specialist degree in applied mathematics from the Moscow Institute of Steel and Alloys in 2009. Since 2016 he is employed at Smart Engines Service LLC, and since 2019 he is employed at the FRC "Computer Science and Control" of RAS. Research interests: image processing, barcode recognition, document recognition. E–mail: *bezmpavel@smartengines.com*.

Received August 13, 2025. The final version – November 19, 2025