Lightweight neural network-based pipeline for barcode image preprocessing

P.K. Zlobin [1,2], V.A. Karnaushko [1], D.M. Ershova [1], R. Sánchez-Rivero [3], P.V. Bezmaternykh [1,2], D.P. Nikolaev [1,2]
[1] Smart Engines Service LLC, 117312, Russia, Moscow, 60-th Anniversary of October avenue, 9;
[2] Federal Research Center "Computer Science and Control" of RAS, 119333, Russia, Moscow, Vavilova st., 44 b.2;
[3] Advanced Technologies Application Center (CENATAV), Playa P.C.12200, Havana, Cuba, 7A, #21406 Siboney

Abstract

Barcode scanning greatly benefited from deep learning research, as well as the image processing stages included in its workflow. These stages commonly handle pre-processing tasks like localizing barcode symbols in the input image, identifying their type, and normalizing the found regions. They are especially important when there is no a priori knowledge of input image capturing conditions. Thus, a case of multiple barcode recognition within a unique image drastically differs from a single barcode processing in video stream via smartphone. We assess how accuracy of these stages affects the accuracy of the whole barcode scanning as its best and propose a lightweight neural network-based pipeline implementing tasks listed above. To perform this assessment and evaluate the performance of the proposed pipeline elements, we conduct a series of experiments using the set of popular open source scanners, including OpenCV , WeChat , ZBar , ZXing and ZXing-cpp over the SE-barcode and Dubska datasets. These experiments reveal how the proposed pipeline can be configured for optimum speed and accuracy performance depending on the objective and the chosen scanner.

Keywords: barcode scanning, image processing, deep learning.
Citation: Zlobin PK, Karnaushko VA, Ershova DM, Sánchez-Rivero R, Bezmaternykh PV, Nikolaev DP. Lightweight neural network-based pipeline for barcode image preprocessing. Computer Optics 2025; 49(6): 1071-1080. DOI: 10.18287/COJ1759.

Introduction

Barcodes are graphically encoded messages specially designed for machine reading. Nowadays, barcode scanning is widely used to reduce manual data entry errors and improve its speed. They are ubiquitous in retail [1, 2], healthcare [3, 4], and payment technologies [5, 6]. In many cases, people have to scan a barcode via mobile cameras in poorly controlled environments [7 – 9]. It presents extra challenges for barcode scanners (or readers), including faulty camera positioning, insufficient or uneven lighting, capturing multiple items at once, or the lack of information about the barcode type (or symbology) being processed. The barcode scanning software is getting more and more complicated in order to overcome these issues.

Currently, there are plenty of such software solutions, including proprietary and open source ones. These solutions clearly divide their processing into several stages (see Fig. 1). First, these scanners locate barcode items or symbols within the input image; then, they determine their symbologies and segment the symbol images into separate logical primitives or modules. After segmentation, the system populates and processes scan lines (for linear barcodes like UPC and EAN) and matrices (for 2D codes like QR and Aztec codes) using the procedure standardized in corresponding symbology specifications. This process includes metainformation modules interpretation, raw message bytes extraction, errors detection and correction, and original message recovering.

Well-established segmentation and decoding routines in the later stages change infrequently, unlike the first stages, which heavily involve image processing tasks. The latest success in artificial neural network training (ANNT), especially deep learning (DL), stimulated research in the barcode area. Tasks of symbology identification, barcode normalization, and primarily barcode localization became an object of keen interest [10 – 12]. Unfortunately, the vast majority of the proposed NNT models are too large to operate on the mobile devices. Thus, the problem of specialized lightweight model creation is particularly relevant.

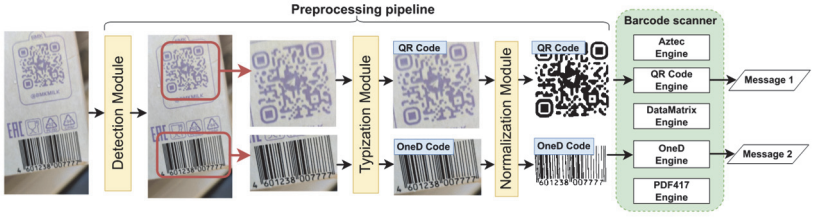


Fig. 1. Common barcode reading pipeline

In this paper, we explore how accuracy of solutions for image processing tasks from the first stages affects the end-to-end barcode scanning accuracy as its best. We also present a lightweight ANNT-based pipeline which covers tasks from these first stages. It preprocesses the input image for further segmentation and decoding by a barcode scanner. We

analyze this pipeline integration with the predefined set of popular barcode scanners and assess the overall performance of such integrations over the relevant barcode datasets.

The remainder of this paper is organized as follows. Section 1 summarizes previous studies dedicated to the problems from barcode recognition pipeline and overviews existing barcode datasets and some popular barcode scanners surveyed lately in this work. Section 2 presents the methodology used in this paper. Sections 3, 4, 5 describe the details of proposed lightweight detector, typization and normalization modules respectively. Section 6 demonstrates how some barcode scanners operate over the described datasets. Finally, last section draws the conclusions of this paper.

1. Related work

1.1. Barcode image preprocessingtasks

The first task in the general barcode scanning pipeline is to identify the regions of interest (ROI) within the image that contain barcode symbols and therefore reduce the image area for further analysis.

Numerous studies have already examined this task. Conventional image processing techniques, such as image gradient analysis, are the foundation of the initial group of methods [13, 14]. The next group is based on pre-trained feature detection. For instance, it includes an adaptation of the Viola-Jones approach [15, 16], originally designed for the face detection problem. Nowadays, the most representative is the group containing the DL-based models [17 – 19]. "You Only Look Once" (or YOLO) based models are the most commonly used ones because of their high detection accuracy and ability to handle objects of different shapes and areas in real-time.

Creating lightweight models for barcode detection is a popular area of research, as shown in recent papers [20, 21]. However, the definition of a “lightweight” model varies significantly: the model presented in [20] has 2.86 M trainable parameters, while the one in [21] only has 140K.

The vast majority of barcode scanners use an approach where they sequentially run the set of specialized internal scanners in the found region. Every internal scanner handles a single family of barcode symbologies. Without a priori knowledge about barcode type within the ROI, the scanner has to try to recognize every supported symbology. This suggests that preliminary identification of barcode type allows us to run only one or a small subset of internal scanners, significantly speeding up overall processing time.

Unlike the barcode detection problem, the symbology typization problem as a separate task is poorly addressed in the literature. Zharkov et al. mentioned this problem in their popular study [19], but gave too few details about it. The authors claim to have proposed a model capable of confidently determine the 21 types of one-dimensional and two-dimensional barcodes. The authors do not address the topic of the complexity of classifying visually similar one-dimensional barcodes.

The next stage of barcode image processing frequently involves some image normalization techniques. According to this study, image normalization is defined as an image transformation process in such a way as if it was originally obtained under normal conditions. These techniques are broadly categorized into color and geometric ones [22].

Color normalization involves adjusting the color intensity of each image pixel to a specific range. In case of barcodes, it is often reduced to the binarization procedure. Although barcodes are typically designed to have high contrasted structure and well-distinct elements, the task of their binarization is surprisingly challenging when capturing conditions are not well-controlled. Because general purpose binarization methods fail in such scenarios, numerous specialized methods, including symbology-oriented ones have already been proposed [23, 24]. While error correction codes can tolerate some incorrectly binarized pixels, it is not always sufficient and the task remains crucial.

Geometric normalization involves correcting perspective distortions, rotating the barcode, and adjusting the size of the barcode modules. The first two problems are well-studied, comparing to the last one. This scaling aims to bring the module size within a defined range, speeding up image processing and enhancing decoding accuracy by optimizing the module size for each barcode scanner. The variability in barcode module size poses a significant challenge to scanner performance in practical scenarios. Consequently, processing barcode images with huge module sizes can substantially slow down the recognition process and potentially exhaust device memory.

To address these issues, employing the ROI image normalization module is advisable. This module standardizes the barcode area for optimal engine processing without compromising crucial data. Although color normalization approaches are widespread [25], the literature lacks sufficient attention to the problem of geometric barcode normalization.

1.2. Barcode datasets selection

Today, there are several datasets for barcode reading evaluation. They vary greatly in presented symbologies, capturing conditions, domain application, nature of images, and ground truth granularity. In paper [12], the authors present a detailed exploration of existing barcode datasets.

Not every barcode dataset suits for our needs. So we use "Muenster" [26], "Artelab" [27, 28], "ZVZ" [29] as training data for our detection and typization models. They contain images with one or many barcodes of different symbologies captured in varying environmental conditions. We don't use popular datasets "InventBar" (Fig. 2*a*.) and "ParcelBar" because they lack variety in presented symbologies and capturing conditions are very similar, and their ground truth

contains only barcode bounding box and message value. We don't use "Synthetic Barcode Datasets" because images in it are not real, but synthetically generated, they look unnatural (Fig. 2*b*.).

For the testing, we use two datasets. The first one is $\mathcal{B}$, which was published by Dubska et al. and presented in paper [30]. It contains realistic images of high-resolution QR codes photographed under different environmental conditions. For such large images, a preliminary detection module is especially important. This dataset's ground truth only provides the coordinates of each barcode's center and its distance to the most distant vertex; the coordinates of all vertices are not specified.

The "SE-barcode" dataset $\mathcal{R}$, presented in paper [21] by Ershova et al., is the second one. It contains a set of barcode images captured via a mobile device camera, which are further replaced with synthetic images in semi-realistic manner. Every image includes at least one barcode and the pixel dimensions of barcode modules exhibit significant variability, thereby facilitating extensive research. The ground truth provides comprehensive information about every presented barcode, including vertex coordinates, symbology type, and the original encoded message.

a) *b)*

Fig. 2. Example of barcode images from Synthetic Barcode Datasets (a) and InventBar (b) datasets

Today, there is a large number of barcode generators that are publicly available. Unfortunately, they cannot be directly used in the context of our problem for several reasons: a) most generators are designed with an emphasis on creating only one-dimensional or two-dimensional barcodes; b) almost all public generators produce a black and white image, whereas our task requires grayscale images, or even colorized versions; c) the output images represent just isolated symbols, lacking any background or context.

Thus, "out-of-the-box" image generators are not applicable for benchmarking barcode scanners, although they are helpful for producing initial valid barcode images to be modified and integrated into backgrounds and contexts [31].

1.3. Barcode scanners selection

Today, there are is a multitude of different barcode scanners. Unfortunately, not every scanner is suitable for our needs. We evaluate only those scanners which are able to handle at least one 1D and one 2D barcode symbologies simultaneously. In this work, we assess only open-source solutions.

ZXing (ver 3.5.3, "Zebra Crossing") is an open-source barcode scanning library written in Java. It supports the most popular set of 1D and 2D symbologies and is capable to deal with multiple codes within the image. It was supported by Google inc., however now its development is currently suspended. It is widely used in barcode benchmarking and it is still a default choice in many scenarios.

ZXing-cpp (ver 2.3.0) is a further mainly community-driven development of the ZXing library, oriented on runtime and detection performance improvements. It supports all the symbologies from ZXing and some extra ones. It is implemented in C++ and quickly increases its popularity. For our experiments we use an official Python-wrapper.

OpenCV (ver 4.11.0) is a well-established open source computer vision library. This library includes barcode scanning and is focused on recognizing one-dimensional barcodes and QR codes as the most popular 2D code. It is written in C++ and has wrappers for many other programming languages. For our experiments we use an official Python-wrapper.

ZBar (ver 0.23.93) is another popular open-source library. ZBar is written mostly in C and supports QR codes and 1D barcodes and is commonly used as a baseline in barcode benchmarking.

WeChat (ver 4.11.0) is a very popular communication program in China, reaching over 1 billion users. This program uses the QR code recognition module, which is shipped with the OpenCV library as one of its extra packages from OpenCV-Contrib. The version of WeChat is the same as OpenCV .

The tab. 1 shows the list of supported barcode symbologies per each barcode scanner listed above. Thus, the full list of used scanners is the following $\mathbb{S} = \{\mathrm{ZXing}, \mathrm{ZXing-cpp}, \mathrm{OpenCV}, \mathrm{ZBar}, \mathrm{WeChat}\}$.

2. Methodology

This section outlines the methodological approach employed in this study. We propose to add three modules of sequential image preprocessing (see Fig. 1): detection – $\mathcal{D}$, typization – $\mathcal{T}$, normalization – $\mathcal{N}$.

Tab. 1. Which symbologies are supported by barcode scanners (UPC/EAN includes UPC-A, UPC-E, EAN-8, and EAN-13).

Symbology	ZBar	ZXing	ZXing-cpp	OpenCV	WeChat
QR code	✓	✓	✓	✓	✓
Aztec	×	✓	✓	×	×
Data Matrix	×	✓	✓	×	×
PDF417	✓	✓	✓	×	×
ITF	✓	✓	✓	×	×
UPC/EAN	✓	✓	✓	✓	✓
Code39/Code128	✓	✓	✓	×	×

In experiment $\mathcal{E}_1$, taken as a baseline, we measure the default decoding accuracy of barcode scanners $s \in \mathbb{S}$ on original images from datasets $\mathcal{B}$ and $\mathcal{R}$. In experiments $\mathcal{E}_{2-4}$, we gradually incorporate modules emulating the functionality of ideal $\mathcal{D}_{GT}$, $\mathcal{T}_{GT}$ and $\mathcal{N}_{GT}$ modules, using ground truth responses as a benchmark. This evaluates the best possible accuracy with such a pipeline. In experiments $\mathcal{E}_{5-7}$, we gradually add the implementations of the proposed $\mathcal{D}$, $\mathcal{T}$ and $\mathcal{N}$ modules. These experiments provide insights about the resulting decoding accuracy and practical applicability of the proposed modules. The Fig. 3 depicts the series of proposed experiments.

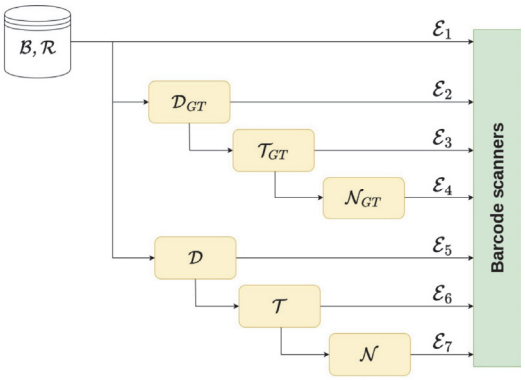


Fig. 3. The series of proposed experiments

For end-to-end evaluation, we measure the rate of correctly decoded barcodes: $Q = N_{correct}/N_{total}$, where N_{total} is the total number of barcodes on all images in this dataset and $N_{correct}$ is the number of barcode correctly recognized.

The performance of each pipeline module is evaluated via a combination of end-to-end and problem-specific metrics tailored to the detection, classification, and normalization tasks.

For barcode detection problem, we use well-established "Intersection Over Union" (IoU) and "Average Precision" (AP) metrics from a general object detection problem. IoU corresponds to the value of the Jaccard index, which is defined as:

$$IoU(C, M) = (area(C \cap M))/(area(C \cup M)), \quad (1)$$

where C is the detected barcode quadrangle, and M is the ideal quadrangle from the ground truth. AP defines the IoU threshold k for correct detection and calculate the rate of correctly detected barcode quadrangles:

$$AP = \sum_{k=0}^{k=n-1} (R(k) - R(k+1)) \cdot P(k), \quad (2)$$

where $P(k)$ defines the precision at threshold k, and $R(k)$ is the recall at threshold k, n is the number of thresholds.

To calculate the typization accuracy of barcode symbologies, we use a number of the most popular metrics: accuracy, precision, recall, which are expained in details in work [32].

We evaluate the time performance of the proposed pipeline using the following hardware and software configuration: 64 – bit Ubuntu 24.04 LTS operating system, AMD Ryzen 5 5600X CPU @3.7 GHz, 32 GB RAM. We denote the overall processing time as T in ms.

3. Proposed detector model

Considering our task as a creating lightweight and universal barcode image preprocessing pipeline, as a base model, we chose the YOLO-Barcode, presented in 2024 [21]. We reproduced the proposed model and its training pipeline, excluding typization part, with using public datasets ZVZ, Muenster and ArteLab as a train and validation data. To improve the quality of the training data, we removed some images containing errors in the ground truth. During inference, we converted input images to grayscale and scaled them to a fixed 512×512 resolution. The proposed model predicts the confidence score and bounding box coordinates for each detected barcode within the image. To filter out the bounding boxes found by our detector, we fixed a confidence threshold of 0. 25 and used non-maximum suppression with a threshold of 0. 3 to remove duplicate responses for the same object.

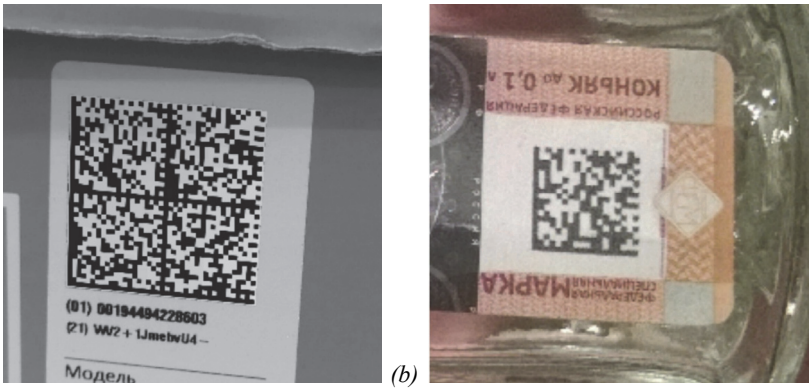
4. Proposed barcode symbology identification model

To achieve efficient and accurate classification, we propose a simple CNN-based model with the architecture outlined in Tab. 2. For training, we used cross-entropy as the loss function. As an input image, the model takes the ROI, found by detector and resized to 96×96 resolution. This resolution is enough to save distinctions between different barcode symbologies and small enough to keep the proposed model computationally efficient.

Tab. 2. Proposed CNN Architecture for barcodes symbology identification

Type	Kernel Size	Stride	Padding	Channels	Activation
Conv	5×5	1×1	2×2	12	ReLU
Conv	3×3	2×2	1×1	12	ReLU
Conv	3×3	1×1	1×1	16	ReLU
Conv	3×3	2×2	1×1	32	ReLU
Conv	3×3	2×2	1×1	32	ReLU
Conv	3×3	2×2	1×1	48	ReLU
Conv	3×3	2×2	1×1	64	ReLU
Conv	3×3	2×2	0×0	64	ReLU
Dense	–	–	–	5	SoftMax

Initially, we aimed to train the model using only publicly available datasets: ZVZ, Muenster and Artelab. Although the test dataset contained barcode samples with the same symbologies as in train datasets, in some cases their internal structure varied a lot and codes had completely distinct patterns. For instance, the ZVZ dataset contains DataMatrix samples only with one data region (Fig. 4*b*) while codes with four data regions are really widespread in other datasets (Fig. 4*a*). This difference in patterns drastically lowered type identification accuracy, so we had to expand the training set with some private synthetic data. By expanding the dataset with synthetic samples, we improved the model's ability to generalize across variations in barcode patterns.



(a) *(b)*

Fig. 4. Example of Data Matrix codes from SE-dataset (a) and ZVZ-dataset (b)

We applied on-the-fly data augmentation during training to further enhance model quality. For each epoch, a random combination of geometric transformations (crop, zoom, and rotation) and visual distortions (blur, brightness adjustments, and image auto-contrasting) augmented 95 % of the training data. We propose to use the following list of symbology groups: 1D, QR code, Aztec, Data Matrix and PDF417. We combined all one-dimensional symbologies into a single 1D group because of their high similarity; at low image resolution, correctly classifying them becomes a laborious task and greatly complicates the model, which contradicts the goal of this work.

5. Proposed normalization model

In practical barcode recognition scenarios, images are often captured under varying conditions, resulting in distortions that can significantly impair decoder performance. The normalization module in our pipeline addresses two primary categories of distortions: color-space inconsistencies (uneven illumination, shadows, glare) and geometric deformations (perspective skew, cylindrical wrapping, partial occlusions).

Traditional approaches to barcode normalization often employ fixed thresholding techniques such as Otsu's method and predetermined scaling factors. However, our analysis of the $\mathcal{R}$ dataset revealed that Otsu's method fails in approximately 38 % of real-world mobile-captured images due to nonlinear lighting gradients, while fixed scaling factors cause module aliasing in 22 % of cases. These observations motivated the development of our hybrid normalization approach.

5.1 Normalization challenges

The design of our normalization module addresses several key challenges:

- **Dynamic adaptation** to local contrast variations without prior knowledge of symbology-specific reflectance patterns

- **Module topology preservation** during geometric transformations to prevent decoder misinterpretation
- **Computational efficiency** for real-time operation on resource-constrained devices

5.2. Hybrid normalization architecture

Our proposed approach combines lightweight neural network components for parameter estimation with traditional image processing operations, creating a hybrid architecture that balances accuracy and computational efficiency. The pipeline consists of two main components: a color normalization submodule and a geometric normalization submodule, working together to address the identified challenges.

Color normalization submodule

The color normalization component addresses issues related to uneven illumination and poor contrast. We implement a miniature CNN with a parameter estimation approach rather than end-to-end image transformation.

Adaptive illumination correction

A lightweight 4-layer CNN processes the grayscale ROI to predict gamma correction parameters, local adaptive thresholding block size (8 – 32 pixels), and contrast-limited adaptive histogram equalization clip limit.

The network architecture consists of input layer, two convolutional layers with ReLU activation, global average pooling, and fully connected layers. This structure requires only 18,304 parameters — 85 % fewer than typical U-Net implementations — while achieving comparable illumination normalization.

Differentiable binarization

To enable end-to-end training while maintaining discrete output at inference time, we replace traditional thresholding operators with a differentiable approximation during training:

$$T(x) = 1/1 + e^{-k(x-\tau)} \tag{3}$$

Where k controls transition steepness (learned per image) and τ is the threshold predicted by the CNN. This allows end-to-end training using reconstruction losses while converging to hard thresholds at inference time.

Geometric normalization submodule

The geometric normalization component addresses issues related to perspective distortion and module size variations. It adapts images to optimal dimensions for each decoder.

Module size estimation

A regression head attached to the typization network predicts: a) horizontal modules per unit length m_x; b) vertical modules per unit length m_y; c) perspective distortion score $d_p \in [0,1]$.

These values drive adaptive resizing:

$$\text{Targetwidth} = \text{ROIwidth} \times m_x/\alpha, \tag{4}$$

$$\text{Targetheight} = \frac{\text{ROIheight} \times m_y}{\alpha}, \tag{5}$$

where α is the preferred module size: 3px for ZXing , 4px for ZBar .

Thin-plate spline transformation

For cases with significant perspective distortion ($d_p > 0.5$), we employ a lightweight Spatial Transformer Network (STN) to predict control point displacements:

$$\theta = f_{\text{STN}}(I_{\text{ROI}}) \tag{6}$$

This combination handles both affine and nonlinear distortions with significantly fewer parameters than full STN implementations.

5.3. Integration with decoding pipeline

To accommodate various performance requirements, our normalization module exposes three operational modes: Speed, Balanced, and Accuracy. These modes vary in their utilization of the color CNN and geometric transformation components, offering trade-offs between computational efficiency and normalization quality.

Barcode scanners receive normalized images in formats optimized for their specific requirements:

- **1D Barcodes**: Binarized, deskewed, 4px/module
- **2D Codes**: Grayscale, perspective-corrected, 3px/module

We evaluate our approach using several metrics: Decoding Success Rate (DSR), Geometric Preservation Score (GPS), Color Consistency Index (CCI), and processing time. Experimental results demonstrating the effectiveness of our normalization approach are presented in Section 6.

6. Evaluation results and discussion

In this section, we present the evaluation results obtained by the methodology described in section 2. Two tables are used to display these findings. Tab. 3 assesses the potential accuracy and time we can obtain with such a pipeline and

correlates to experiments 1 – 4. Tab. 4 assesses the actual accuracy and time using the pipeline assembled from the suggested $\mathcal{D}$, $\mathcal{T}$ and $\mathcal{N}$ modules and relates to experiments 5 – 8.

Tab. 3. Potential pipeline's performance on Dubska and SE-Barcode datasets

$\mathcal{E}$	$\mathcal{D}_{GT}$	$\mathcal{T}_{GT}$	$\mathcal{N}_{GT}$	Dubska									
				WeChat		ZBar		ZXing		ZXing-cpp		OpenCV	
				Q,%	T, ms	Q,%	T, ms	Q,%	T, ms	Q,%	T, ms	Q,%	T, ms
1	×	×	×	58.	92.3	76.7	404.5	57.8	1471.8	**88.3**	**44.4**	43.1	528.2
2	✓	×	×	80.9	120.3	78.2	60.7	66.8	497.3	**84.4**	**14.7**	64.9	109.5
3	✓	✓	×	80.9	102.0	78.2	45.5	66.5	141.1	**84.3**	**2.7**	64.9	95.6
4	✓	✓	✓	**92.4**	19.4	87.8	11.8	85.9	109.9	88.9	**0.7**	66.4	32.2
$\mathcal{E}$	$\mathcal{D}_{GT}$	$\mathcal{T}_{GT}$	$\mathcal{N}_{GT}$	SE-Barcode									
				WeChat		ZBar		ZXing		ZXing-cpp		OpenCV	
				Q,%	T, ms	Q,%	T, ms	Q,%	T, ms	Q,%	T, ms	Q,%	T, ms
1	×	×	×	47.7	**49.0**	61.1	303.1	52.4	2155.5	**85.8**	73.3	34.9	232.5
2	✓	×	×	52.0	50.0	61.7	15.7	69.5	225.9	**83.6**	**4.7**	41.9	29.6
3	✓	✓	×	52.0	7.4	61.2	8.4	69.5	92.2	**83.6**	**1.5**	41.9	18.5
4	✓	✓	✓	52.6	7.3	56.3	6.0	70.4	104.6	**81.3**	**0.9**	40.9	15.0

In Experiment $\mathcal{E}_1$, taken as a baseline, we measure the decoding accuracy of barcode scanners on original images comprising one or more barcodes for each scanner $s \in \mathbb{S}$ without any external image preprocessing.

In experiments $\mathcal{E}_{2-4}$ we gradually add the imitation of the ideal modules $\mathcal{D}_{GT}$, $\mathcal{T}_{GT}$, $\mathcal{N}_{GT}$. The barcode ROI is calculated based on the ground truth coordinates of the quadrangles. Some barcode scanners may not correctly localize barcode if one or more of its corners are placed next to the image's side. To compensate for this effect on the final recognition result, we extend the ROI area by 0.1 relative to the expansion side. The number of images in $\mathcal{B}$ and $\mathcal{R}$ datasets are used to normalize all time measurement results.

The time and accuracy measurements of barcodes recognition using various barcode scanners allowed to reveal interesting findings. The sequential addition of $\mathcal{D}_{GT}$, $\mathcal{T}_{GT}$, $\mathcal{N}_{GT}$ modules has resulted in an overall reduction of processing time for most scanners at each step of their addition. However, WeChat scanner is the exception by demonstrating a slowing down of the process, which may be due to the peculiarities of its architecture or image processing algorithms. The decoding quality has improved for WeChat , ZBar , ZXing and OpenCV , which confirms the utility of the proposed pipeline. At the same time, ZXing-cpp indicated a decrease in recognition accuracy, which may be due to insufficient adaptation of its algorithms to the modified data. These results highlight the importance of taking into account the specifics of each scanner when optimizing barcoding recognition processes.

Since we imitate the ideal detector and symbology identification module in these experiments, the process time of each module equals to 0 milliseconds. In reality, the total process time of preprocessing modules and a recognition scanner is calculated by the formula:

$$T = T_{\mathcal{D}} + T_{\mathcal{T}} + T_{\mathcal{N}} + T_P, \tag{7}$$

where $T_{\mathcal{D}}$ – is the average $\mathcal{D}$ module time per original image from $\mathcal{B}$ and $\mathcal{R}$, $T_{\mathcal{T}}$ is the average $\mathcal{T}$ module time process, $T_{\mathcal{N}}$ is the average $\mathcal{N}$ module time, T_P is the average processing time of an image containing barcodes inside barcode reader.

Tab. 4. Proposed pipeline's performance on Dubska and SE-Barcode datasets. The average times for $\mathcal{D}$ (11.6 ms), $\mathcal{T}$ (0.72 ms), and $\mathcal{N}$ (25.3 ms) are included in total time spent on barcode scanning.

$\mathcal{E}$	$\mathcal{D}$	$\mathcal{T}$	$\mathcal{N}$	Dubska									
				WeChat		ZBar		ZXing		ZXing-cpp		OpenCV	
				Q,%	T, ms	Q,%	T, ms	Q,%	T, ms	Q,%	T, ms	Q,%	T, ms
1	×	×	×	58.6	92.3	76.7	404.5	57.8	1471.8	**88.3**	**44.4**	3.1	28.2
2	✓	×	×	**79.0**	21.0	5.2	02.3	2.6	02.8	**9.0**	**8.0**	1.1	51.8
3	✓	✓	×	76.0	1.7	2.9	3.4	0.3	53.4	**6.3**	**7.0**	8.0	35.8
4	✓	✓	✓	**80.9**	9.8	6.7	8.3	4.0	48.1	6.3	**8.2**	7.3	6.9
$\mathcal{E}$	$\mathcal{D}$	$\mathcal{T}$	$\mathcal{N}$	SE-Barcode									
				WeChat		ZBar		ZXing		ZXing-cpp		OpenCV	
				Q,%	T, ms	Q,%	T, ms	Q,%	T, ms	Q,%	T, ms	Q,%	T, ms
1	×	×	×	47.7	**49.0**	61.1	303.1	52.4	2155.5	**85.8**	73.3	34.9	232.5
2	✓	×	×	52.8	71.3	60.1	38.1	66.8	557.5	**78.1**	**19.6**	41.8	50.1
3	✓	✓	×	51.5	24.0	58.4	27.8	64.7	166.4	**76.9**	**14.6**	41.1	36.2
4	✓	✓	✓	57.5	46.1	60.3	43.5	77.2	135.2	**81.3**	**38.5**	49.4	52.2

In this study, barcode image preprocessing modules $\mathcal{D}$, $\mathcal{T}$ and $\mathcal{N}$ were developed and implemented. These modules, although inferior in speed and accuracy to modules imitating ideal $\mathcal{D}_{GT}$, $\mathcal{T}_{GT}$ and $\mathcal{N}_{GT}$, demonstrate a significant improvement over the basic approach where preprocessing modules were absent. The recognition accuracy has increased for WeChat , ZBar , ZXing and OpenCV scanners. This confirms the effectiveness of the proposed solutions in improving image processing quality. However, ZXing-cpp 's accuracy has decreased. This might be due to its algorithms being less resistant to changes made at the preprocessing stage.

As for the decoding time, the implementation of proposed modules has led to its decrease for all decoders except WeChat. WeChat 's processing time is only slightly reduced when adding $\mathcal{T}$, which indicates that the symbology recognition engines may be overrun or not run optimally. These results highlight that the preprocessing modules developed can significantly improve the performance of the barcode scanner, although their effectiveness may vary depending on the scanner being used. Special attention should be paid to the normalization module. Despite the increase in overall recognition time, decoding accuracy is greatly improved on all engines.

To assess the accuracy of the proposed $\mathcal{D}$ module, we evaluated $\mathcal{B}$ and $\mathcal{R}$ datasets using precision, recall, F1 score, AP, and mean IoU metrics. The results are presented in Tab. 5.

Tab. 5: Detection results on Dubska and SE-Barcode datasets

Detection metric	Dubska	SE-Barcode
Precision	88.39	89.24
Recall	90.08	90.70
F1 score	89.22	89.96
Average Precision	86.56	90.74
Mean IoU	0.75	0.79

Comparing the results on two datasets allows to conclude that the detection module effectively meets the task of locating barcodes, although its performance can vary depending on the complexity of the input data.

Attention should be paid to the accuracy of $\mathcal{T}$ as it affects the performance of further barcode recognition. We demonstrate the accuracy of the typization $\mathcal{T}$ in Tab. 6.

Tab. 6: Typization results on Dubska and SE-Barcode datasets

Typization metric	Dubska	SE-Barcode
Precision	96.28	88.87
Recall	88.93	72.51
F1 score	92.46	79.86

The module $\mathcal{N}$ allows to normalize barcode until all its modules are reduced to a same size. After the $\mathcal{N}$, the barcode occupies the entire area of the image. As noted above, some barcode scanners do not decode well barcodes that take up an overwhelming part of the image, so we expand the image like in $\mathcal{E}_2$.

Conclusion

In this paper, we explored a barcode image preprocessing pipeline assembled from detection, type identification, and normalization modules. The outcome of this pipeline is integrated with the input for widely-used barcode scanners. Our detection model $\mathcal{D}$, based on the YOLO architecture, demonstrates high accuracy in detecting barcodes even under difficult capturing conditions, regardless of the total number of barcodes in the image. The results of the recognition speed comparison using the detection module confirm a considerable saving in computing time. The use of the type identification module $\mathcal{T}$ allows for a futher reduction in average time. It also helps to filter out barcodes that are not supported by the scanners.

The proposed hybrid normalization approach significantly improves decoding success rates by addressing both color-space inconsistencies and geometric deformations through a lightweight parameter estimation NNT. By combining ANNT components with conventional image processing techniques, the proposed normalization module $\mathcal{N}$ achieves by maximum 24 % improvement in decoding accuracy while maintaining reasonable inference times.

The recognition pipeline allows configuring the modules and barcode scanners to achieve the best time performance/accuracy balance.

In our further work, we will focus on the normalization stage of the pipeline to improve image input of the barcode scanners and provide some extra hints for their fast and successful recognition. This includes determining the optimal barcode dimensions for each scanner used in benchmark.

References

[1] Kjellberg H, Hagberg J, Cochoy F: Thinking market infrastructure: barcode scanning in the US grocery retail sector, 1967–2010. In: Thinking infrastructures, 2019:207–232

[2] Melek CG, Sönmez EB, Varl S: Datasets and methods of product recognition on grocery shelf images using computer vision and machine learning approaches: An exhaustive literature review. Engineering Applications of Artificial Intelligence 133 2024. DOI: 10.1016/j.engappai.2024.108452
[3] Fernandez WP, Xian Y, Tian Y.: Image-based barcode detection and recognition to assist visually impaired persons. In: 2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), 2017:1241–1245
[4] Patra S.: Healthcare Distribution Alliance -Barcoding Requirement for Serialized Product. International Journal of Engineering Research & Technology 11(7), 2022:353–358
[5] Eren BA: QR code m-payment from a customer experience perspective. Journal of Financial Services Marketing p. 1 2022
[6] Rafferty NE, Fajar AN: Integrated QR payment system (QRIS): cashless payment solution in developing country from merchant perspective. Asia Pacific Journal of Information Systems 32(3), 2022:630–655
[7] Brylka R, Schwanecke U, Bierwirth B: Camera Based Barcode Localization and Decoding in Real-World Applications. In: 2020 International Conference on Omni-layer Intelligent Systems (COINS), 2020:1–8. DOI: 10.1109/COINS49042.2020.9191416
[8] Chernov TS, Razumnyy NP, Kozharinov AS, Nikolaev DP, Arlazarov VV: Image quality assessment for video stream recognition systems. ITiVS (4), 2017:71–82. DOI: 10.1117/12.2309628.
[9] Gallo O, Manduchi R: Reading 1D Barcodes with Mobile Phones Using Deformable Templates. IEEE Transactions on Pattern Analysis and Machine Intelligence 33(9), 2011:1834–1843. DOI: 10.1109/TPAMI.2010.229
[10] Kutiyanawala A, Qi X, Tian J: A simple and efficient approach to barcode localization. In: 2009 7th International Conference on Information, Communications and Signal Processing (ICICS) 2009:1–5. DOI: 10.1109/ICICS.2009.5397472.
[11] Ren, Y., Liu, Z.: Barcode detection and decoding method based on deep learning. In: 2019 2nd International Conference on Information Systems and Computer Aided Education (ICISCAE). 2019:393–396. DOI: 10.1109/ICISCAE48440.2019.217911
[12] Wudhikarn, R., Charoenkwan, P., Malang, K.: Deep Learning in Barcode Recognition: A Systematic Literature Review. IEEE Access 10, 2022:8049–8072. DOI: 10.1109/access.2022.3143033
[13] Ouaviani, Pavan, Bottazzi, Brunelli, Caselli, 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, 1999:652–655. DOI: 10.1049/cp:19990404
[14] Sörös G, Flörkemeier C: Blur-resistant joint 1D and 2D barcode localization for smartphones. In: Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia. 2013:1–8. DOI: 10.1145/2541831.2541844
[15] Belussi L, Hirata N: Fast QR Code Detection in Arbitrarily Acquired Images. In: 2011 24th SIBGRAPI Conference on Graphics, Patterns and Images. 2011:281–288. DOI: 10.1109/SIBGRAPI.2011.16
[16] Usilin, S.A.: Using greedy strategy of Viola-Jones cascade choosing for improving performance of multi-class object detection in video stream. Trudy ISA RAN (Proceedings of ISA RAS) 67(1), 2017:75–82
[17] Hansen DK, Nasrollahi K, B. Rasmusen C, Moeslund TB. Real-Time Barcode Detection and Classification using Deep Learning. Proceedings of the 9th International Joint Conference on Computational Intelligence 2017. DOI: 10.5220/0006508203210327
[18] Zhang L, Sui Y, Zhu F, Zhu M, He B, Deng Z. Fast Barcode Detection Method Based on ThinYOLOv4. Communications in Computer and Information Science 2021:41–55. DOI: 10.1007/978-981-16-2336-3_4.
[19] Zharkov A, Zagaynov I. Universal Barcode Detector via Semantic Segmentation. 2019 International Conference on Document Analysis and Recognition (ICDAR) 2019:837–43. DOI: 10.1109/ICDAR.2019.00139
[20] Chen J, Dai N, Hu X, Yuan Y. A Lightweight Barcode Detection Algorithm Based on Deep Learning. Applied Sciences 2024;14:10417. DOI: 10.3390/app142210417
[21] Ershova DM, Gayer AV, et al. YOLO-Barcode: towards universal real-time barcode detection on mobile devices. Computer Optics 2024;48:592–600. DOI: 10.18287/2412-6179-CO-1424
[22] Konovalenko I, RMS coordinate discrepancy as accuracy criterion of images normalization at optical document recognition. Informatsionnye protsessy, 20(3), 2020: 215-230.
[23] Chen R, Yu Y, Xu X, Wang L, Zhao H, Tan H-Z. Adaptive Binarization of QR Code Images for Fast Automatic Sorting in Warehouse Systems. Sensors 2019;19:5466. DOI: 10.3390/s19245466
[24] Yang Z, Xu H, Deng J, Loy CC, Lau WC. Robust and Fast Decoding of High-Capacity Color QR Codes for Mobile Applications. IEEE Trans on Image Process 2018;27:6093–108. DOI: 10.1109/TIP.2018.2855419.
[25] Liao L, Li J, Lu C. Data Extraction Method for Industrial Data Matrix Codes Based on Local Adjacent Modules Structure. Applied Sciences 2022;12:2291. DOI: 10.3390/app12052291
[26] Wachenfeld S, Terlunen S, Jiang X. Robust 1-D Barcode Recognition on Camera Phones and Mobile Product Information Display. Lecture Notes in Computer Science 2010:53–69. DOI: 10.1007/978-3-642-12349-8_4.
[27] Zamberletti, A., Gallo, I., Albertini, S.: Robust Angle Invariant 1D Barcode Detection 2013
[28] Zamberletti A , Gallo I, Carullo M, Binaghi E. (2010). Neural Image Restoration for Decoding 1-D Barcodes using Common Camera Phones. VISAPP 2010 - Proceedings of the International Conference on Computer Vision Theory and Applications. 1 2010:5-11.
[29] Zharkov A, Vavilin A, Zagaynov I. New Benchmarks for Barcode Detection Using Both Synthetic and Real Data. Lecture Notes in Computer Science 2020:481–93. DOI: 10.1007/978-3-030-57058-3_34.
[30] Szentandrási I, Herout A, Dubská M. Fast detection and recognition of QR codes in high-resolution images. Proceedings of the 28th Spring Conference on Computer Graphics 2012:129–36. DOI: 10.1145/2448531.2448548
[31] Zlobin P. The system for synthesizing problem-oriented data packages in the task of reading barcodes. Informatsionnye protsessy 24(4), 2024:438-448. 10.53921/18195822_2024_24_4_438.
[32] Juba, B., Le, H.S.: Precision-recall versus accuracy and the role of large data sets. In: Proceedings of the AAAI conference on artificial intelligence. vol. 33, 2019:4039–4048

Authors' information

Pavel Konstantinovich Zlobin, (b. 1996), received a Master degree in Information Systems and Technologies from MISiS in 2021. PhD student of the FRC "Computer Science and Control" of RAS. Since 2019 he is employed at Smart Engines Service LLC. He is an author of 5 publications. Research interests: computer vision, machine learning.
E-mail: *p.zlobin@smartengines.com*

Vladimir Alekseevich Karnaushko, (b. 2001), recieved a Master degree in Informatics and Computer Technologies from NUST MISIS in 2025. Since 2023 he is employed at Smart Engines Service LLC. Research interests: 3D graphics, computer vision, data analysis. Email: *v.karnaushko@smartengines.com*

Daria Mikhailovna Ershova, (b. 2000), PhD student at Moscow Institute of Physics and Technology. Graduated from Lomonosov Moscow State University in 2023, is a developer at Smart Engines Service LLC since 2021. Research interests: machine learning, computer vision, object detection. E-mail: *d.ershova@smartengines.com*

Rubén Sánchez-Rivero, (b. 1993), graduated in Software Engineering from the Havana University of Technologies "José Antonio Echeverría", Cuba, in 2017. He is a researcher at the Advanced Technologies Application Center (CENATAV). He is author of more than 5 scientific publications. Research interest: computer vision, digital image processing, document analysis. E-mail: *rsanchez@cenatav.co.cu*

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. He is an author of more than 25 scientific publications. Research interests: image processing, document recognition. E-mail: *bezmpavel@gmail.com*

Dmitry Petrovich Nikolaev, (b. 1978), was awarded Doctor of Sciences in Computer Science in 2023. Since 2007, he has been the Head of the Vision Systems Laboratory, Institute for Information Transmission Problems, RAS, Moscow, and he has been the CTO of Smart Engines Service LLC, Moscow, since 2016. Since 2016, he has been an Associate Professor with the Moscow Institute of Physics and Technology (State University). Now he is the Head of the Vision Systems Laboratory at FRC "Computer Science and Control" of the RAS. He has authored over 150 Scopus-indexed publications and 15 patents. His research interests encompass computer vision and color image understanding.
E-mail: *d.p.nikolaev@gmail.com*

Received: June 19, 2025. Final version: September 12, 2025