Universal Barcode Detector via Semantic Segmentation

Andrey Zharkov*†, Ivan Zagaynov*†
* *R&D Department*
ABBYY Production LLC
Moscow, Russia
{*andrew.zharkov, Ivan.Zagaynov*}*@abbyy.com*
† *Phystech School of Applied Mathematics and Informatics*
Moscow Institute of Physics and Technology (National Research University)

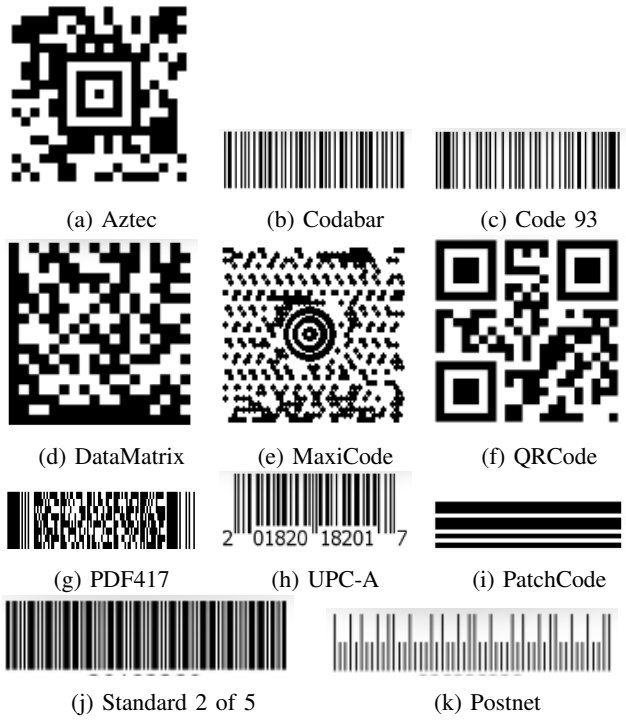
***Abstract*—Barcodes are used in many commercial applications, thus fast and robust reading is important. There are many different types of barcodes, some of them look similar while others are completely different. In this paper we introduce new fast and robust deep learning detector based on semantic segmentation approach. It is capable of detecting barcodes of any type simultaneously both in the document scans and in the wild by means of a single model. The detector achieves state-of-the-art results on the ArTe-Lab 1D Medium Barcode Dataset with detection rate 0.995. Moreover, developed detector can deal with more complicated object shapes like very long but narrow or very small barcodes. The proposed approach can also identify types of detected barcodes and performs at real-time speed on CPU environment being much faster than previous state-of-the-art approaches.**

I. Introduction

Starting from the 1960s people have invented many barcode types which serve for machine readable data representation and have lots of applications in various fields. The most frequently used are probably UPC and EAN barcodes for consumer products labeling, EAN128 serves for transferring information about cargo between enterprises, QR codes are widely used to provide links, PDF417 has variety of applications in transport, identification cards and inventory management. Barcodes have become ubiquitous in modern world, they are used as electronic tickets, in official documents, in advertisement, healthcare, for tracking objects and people. Examples of popular barcode types are shown in Fig. 1.

There are two main approaches for decoding barcodes, the former uses laser and the latter just a simple camera. Through years of development, laser scanners have become very reliable and fast for the case of reading exactly one 1D barcode, but they are completely unable to deal with 2D barcodes or read several barcodes at the same time. Another drawback is that they can not read barcodes from screens efficiently as they strongly rely on reflected light.

Popular camera-based reader is a simple smartphone application which is capable of scanning almost any type of barcode. However, most applications require some user guidance like pointing on barcode to decode. Most applications decode only one barcode at a time, despite it is possible to decode all barcodes in the image. It may become important when we need to scan barcodes from some official documents where might be a number of them.



(a) Aztec (b) Codabar (c) Code 93

(d) DataMatrix (e) MaxiCode (f) QRCode

(g) PDF417 (h) UPC-A (i) PatchCode

(j) Standard 2 of 5 (k) Postnet

Figure 1: Some examples of different barcodes

In this work, we introduce segmentation based barcode detector which is capable of locating all barcodes simultaneously no matter how many of them are present in the image or which types they are, so the system does not need any user guidance. The developed detector also provides information about most probable types of detected barcodes thus decreasing time for reading process.

II. Related Work

The early work in the domain of barcode detection from 2D images was motivated by the wide spread of mobile phones with cameras. [1] proposes a method for finding 2D barcodes

via corner detection and 1D barcodes through spiral scanning. [2] introduces another method for 1D barcode detection based on decoding. Both approaches however require certain guidance from the user.

In more recent papers authors pay more attention on developing solutions which can be done automatically with less user guidance. [3] finds regions with high difference between x and y derivatives, [4] calculates oriented histograms to find patches with dominant direction, [5] relies on morphology operations to detect both 1D and 2D barcodes, reporting high accuracy on their own data. The work of Sörös *et al.* [6] is notable as they compare their own algorithm with other works mentioned in this paragraph. They demonstrate that their approach is superior on the same dataset WWU Muenster Barcode Database (Muenster). Their algorithm is based on the idea that 1D barcodes have many edges, 2D barcodes have many corners, while text areas have both many edges and many corners.

The work of Cresot *et al.*, 2015 [7] is a solid baseline for 1D barcode detection. They evaluated their approach on Muenster and on extended ArTe-Lab 1D Medium barcode database (Artelab) provided by Zamberletti *et al.* [8] outperforming him on both datasets. The solution in [7] seems to outperform [6] despite it is hard to compare as they were evaluated on different datasets using slightly different metrics. Cresot's algorithm detects dark bars of barcodes using Maximal Stable Extremal Regions (MSER) followed by finding imaginary perpendicular to bars center line in Hough space. In 2016 Cresot *et al.* came with a new paper [9] improving previous results using a new variant of Line Segment Detector instead of MSER, which they called Parallel Segment Detector. [10] proposes another bars detection method for 1D barcode detection, which is reported to be absolutely precise in real-time applications.

In the recent years neural networks show very promising results in many domains including Computer Vision. However, at the moment of writing there are only a few research works which utilize deep learning for barcode detection. The first is already mentioned [8] where neural network analyzes Hough space to find potential bars. More recent and promising results are obtained in [11] where authors use YOLO (You Only Look Once) detector to find rectangles with barcodes, then apply another neural network to find the rotation angle of that barcode, thus after that they are able to rotate the barcode and pass it to any barcode recognizer simplifying the recognition task. They showed new state-of-the-art (SOTA) results on Muenster dataset. However, their solution can not be considered real-time for CPUs. Moreover, YOLO is known to have problems with narrow but very long objects, which can be an issue for some barcode types.

III. Detection via segmentation

Our approach is inspired by the idea of PixelLink [12] where authors solve text detection via instance segmentation. We believe that for barcodes the situation when 2 of them are close to each other is unusual, so we do not really need to solve instance segmentation problem therefore dealing with semantic segmentation challenge should be enough.

PixelLink shows good results capturing long but narrow lines with text, which can be a case for some barcode types so we believe such object shape invariance property is an additional advantage.

To solve the detection task we first run semantic segmentation network and then postprocess its results.

A. Semantic segmentation network

Barcodes normally can not be too small so predicting results for resolution 4 times lower than original image should be enough for reasonably good results. Thus we find segmentation map for superpixels which are 4x4 pixel blocks.

Detection is a primary task we are focusing on in this work, treating type classification as a less important sidetask. Most of barcodes share a common structure so it is only natural to classify pixels as being part of barcode (class 1) or background (class 0), thus segmentation network solves binary (super)pixel classification task.

Barcodes are relatively simple objects and thus may be detected by relatively simple architecture. To achieve real-time CPU speed we have developed quite simple architecture based on dilated and separable convolutions (see Table I). It can be logically divided into 3 blocks:

1) **Downscale Module** is aimed to reduce spatial features dimension. Since these initial convolutions are applied to large feature maps they cost significant amount of overall network time, so to speed up inference these convolutions are made separable.
2) **Context Module**. This block is inspired by [13]. However, in our architecture it serves for slightly different purpose just improving features and exponentially increasing receptive field with each layer.
3) Final classification layer is 1x1 convolution with number of filters equal to 1+n_classes, where n_classes is number of different barcode types we want to differentiate with.

We used ReLU nonlinearity after each convolution except for the final one where we apply sigmoid to the first channel and softmax to all of the rest channels.

We have chosen the number of channels $C = 24$ for all convolutional layers. Our experiments show that with more filters model has comparable performance, but with less filters performance drops rapidly. As we have only a few channels in each layer the final model is very compact with only 32962 weights.

As the maximal image resolution we are working with is 512x512, receptive field for prediction is at least half an image which should be more than enough contextual information for detecting barcodes.

Table I: Model architecture, C=24 is the number of channels and N is the number of predicted classes (barcode types)

	Downscale Module			Context Module						Final
Layer	1	2	3	4	5	6	7	8	9	10
Stride	2	1	2	1	1	1	1	1	1	1
Dilation	1	1	1	1	2	4	8	16	1	1
Separable	Yes	Yes	Yes	No	No	No	No	No	No	No
Kernel	3x3	3x3	3x3	3x3	3x3	3x3	3x3	3x3	3x3	1x1
Output Channels	C	C	C	C	C	C	C	C	C	1+N
Receptive Field	3x3	7x7	11x11	19x19	35x35	67x67	131x131	259x259	267x267	267x267

B. Detecting barcodes based on segmentation

After the network pass we get segmentation map for superpixels with $1 + n\_classes$ channels. For detection we use only the first channel which can be interpreted as probability being part of barcode for superpixels.

We apply the threshold value for probability to get detection class binary labels (barcode/background). In all our experiments we set this threshold value to 0.5. We now find connected components on received superpixel binary mask and calculate bounding rectangle of minimal area for each component. To do the latest we apply *minAreaRect* method from *OpenCV* library (accessed Dec 2018).

Now we treat found rectangles as detected barcodes. To get detection rectangle on original image resolution we multiply all of its vertices coordinates by the network scale 4.

C. Filtering results

To avoid a situation when a small group of pixels is accidentally predicted as a barcode, we filter out all superpixel connected components with area less than threshold T_{area}. The threshold value should be chosen to be slightly less than minimal area of objects in the dataset on the segmentation map. In all of our experiments we used value $T_{area} = 20$.

D. Classification of detected objects

To determine barcode type of detected objects we use all of the rest $n\_classes$ channels from segmentation network output. After softmax we treat them as probabilities of being some class.

Once we found the rectangle we compute the average probability vector inside this rectangle, then naturally choose the class with the highest probability.

IV. Optimization scheme

A. Loss function

The training loss is a weighted sum of detection and classification losses

$$L = L_{detection} + \alpha L_{classification} \tag{1}$$

Detection loss $L_{detection}$ itself is a weighted sum of three components: mean binary crossentropy loss on positive pixels L_p, mean binary crossentropy loss on negative pixels L_n, and mean binary crossentropy loss on worst predicted k negative pixels L_h, where k is equal to the number of positive pixels in image.

$$L_{detection} = w_p L_p + w_n L_n + w_h L_h \tag{2}$$

Classification loss is mean (categorical) crossentropy computed by all channels except the first one (with detection). Classification loss is calculated only on superpixels which are parts of ground truth objects.

As our primary goal is high recall in detection we have chosen $w_p = 15$, $w_n = 1$, $w_h = 5$, $\alpha = 1$. We also tried several different configurations but this combination was the best among them. However, we did not spent too much time on hyperparameter search.

B. Data augmentation

For augmentation we do the following:

1) with (p=0.1) return original nonaugmented image
2) with (p=0.5) rotate image random angle in [-45, 45]
3) with (p=0.5) rotate image on one of 90, 180, 270 degrees
4) with (p=0.5) do random crop. We limit crop to retain all barcodes on the image entirely and ensure that aspect ratio is changed no more than 70% compared to the original image
5) with (p=0.7) do additional image augmentation. For this purpose we used "less important" augmenters from heavy augmentation example from *imgaug* library [14].

V. Experimental results

A. Datasets

The network performance was evaluated on 2 common benchmarks for barcode detection - namely WWU Muenster Barcode Database (Muenster) and ArTe-Lab Medium Barcode Dataset (Artelab). Datasets contain 595 and 365 images with ground truth detection masks respectively, resolution for all images is 640x480. All images in Artelab dataset contain exactly one EAN13 barcode, while in Muenster there may be several barcodes on the image.

For training we used our own dataset with both 1D barcodes (Code128, Patch, EAN8, Code93, UCC128, EAN13, Industrial25, Code32, FullASCIICode, UPCE, MATRIX25,

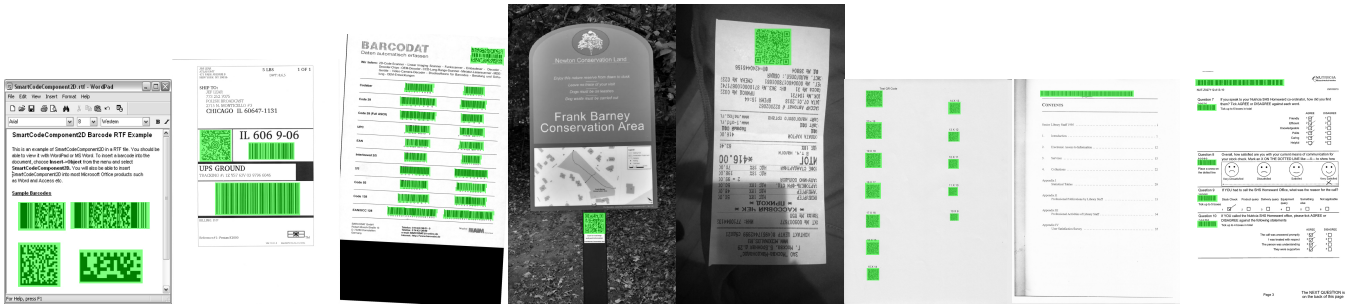


Figure 2: Detection examples from test set

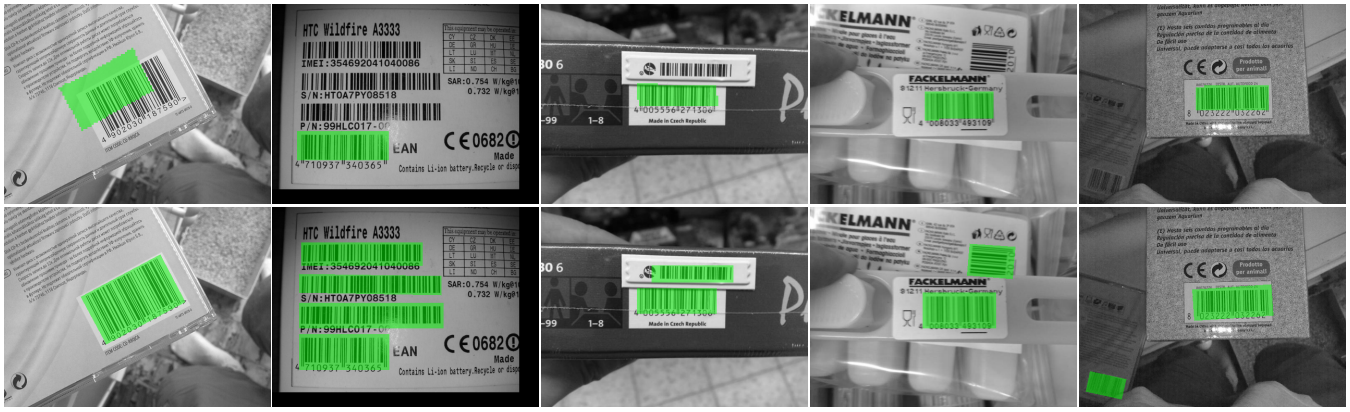


Figure 3: Detection examples: markup issues on Artelab and Muenster dataset. Markup on the top, detections results below

Code39, IATA25, UPCA, CODABAR, Interleaved25) and 2D barcodes (QRCode, Aztec, MaxiCode, DataMatrix, PDF417), being 16 different types for 1D and 5 types for 2D Barcodes, 21 type in total. Training dataset contains both photos and document scans. Example images from our dataset can be found in Fig. 2. Dataset consist of 17k images in total, 10% of it was used for validation.

B. Training procedure

We trained our model with batch size 8 for 70 epochs with learning rate 0.001 followed by additional 70 epochs with learning rate 0.0001

While training we resized all images to have maximal side at most 1024 maintaining aspect ratio and make both sides divisible by 64. We pick and augment/preprocess 3000 images from the dataset, then group them into batches by image size, and do this process repeatedly until the end of training. After that we pick next 3000 images and do the same until the end of the dataset. After we reach the end of the dataset, we shuffle image order and repeat the process.

We trained three models: *Ours-Detection (all types)* (without classification on entire dataset), *Ours-Detection+Classification (all types)* (with classification on entire dataset), *Ours-Detection (EAN13 only)* (without classification on EAN13 subset of 1000 images).

C. Evaluation metrics

We follow common evaluation scheme from [7]. Having binary masks G for ground truth and F for found detection results the Jaccard index between them is defined as

$$J(G, F) = \frac{|G \cap F|}{|G \cup F|}$$

Another common name for Jaccard index is "intersection over union" or IoU which follows from definition. The overall detection rate for a given IoU threshold T is defined as a fraction of images in the dataset where IoU is greater than that threshold

$$D_T = \frac{\sum_{i \in S} I(J(G, F) \geq T))}{|S|}$$

where S is set of images in the dataset and I is indicator function.

However, one image may contain several barcodes and if one of them is very big and another is very small D_T will indicate error only on very high threshold, so we find it reasonable to evaluate detection performance with additional metrics which will average results not by images but by ground truth barcode objects on them.

For this purpose we use recall R_T, defined as number of successfully detected objects divided by total number of

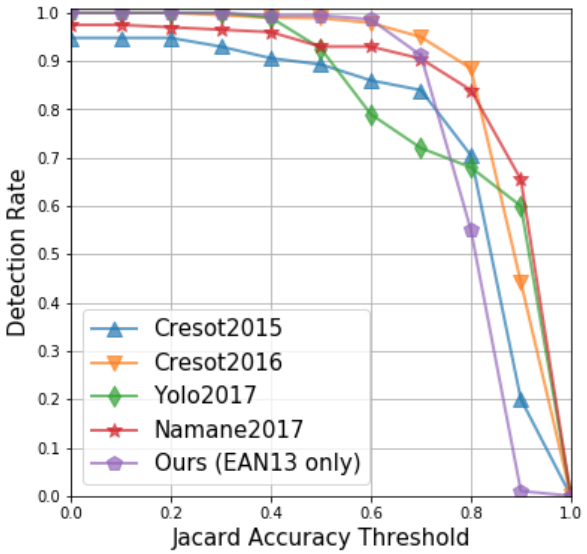


Figure 4: Detection rate for different Jaccard index thresholds

objects in the dataset

$$R_T = \frac{\sum_{i \in S} \sum_{G \in SG_i} I(J(G, F(G)) \geq T))}{\sum_{i \in S} |SG_i|}$$

where SG_i is set of objects on ground truth on image i and $F(G)$ is found box with highest Jaccard index with box G. The paired metric for recall is precision, defined as the number of successfully detected objects divided by total number of detections

$$P_T = \frac{\sum_{i \in S} \sum_{G \in SG_i} I(J(G, F(G)) \geq T))}{\sum_{i \in S} |SF_i|}$$

where SF_i is set of all detections made per image i.

We found connected components for ground truth binary masks and treat them as ground truth objects.

We emphasize that all the metrics above are computed for the detected object *regardless its actual type*. To evaluate classification of the detected objects by type we use simple accuracy metric (number of correctly guessed objects / number of correctly detected objects). So if we find Barcode-PDF417 as Barcode-QRCode precision and recall will not be affected, but the classification accuracy will be.

D. Quantitative results

We compare our results with Cresot2015 [7], Cresot2016 [9], Namane2017 [10], Yolo2017 [11] on Artelab and Muenster datasets (Table II).

The proposed method is trained on our own dataset, however all other works which we compared with were trained on different datasets. As for the full reproducibility of other authors works on our dataset we have to follow the exactly same training protocol (including initialization and augmentations) to not underestimate the results we decided to rely on the numbers reported in works of other authors.

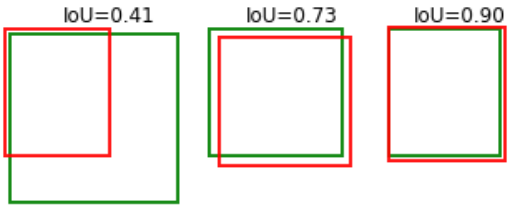


Figure 5: Example of Jaccard index computation for various bounding boxes

We outperformed all previous works in terms of detection rate on Artelab dataset with the model trained only on EAN13 subset of our dataset. According to the tables, detection rate of our model trained on entire dataset with all barcode types is slightly worse than model trained on EAN13 subset. The reason for this is not poor generalization but markup errors or capturing more barcodes than in the markup (i. e. non-EAN barcodes), see Fig. 3.

As it can be seen in Fig. 4 our model has a rapid decrease in detection rate for higher Jaccard thresholds. Aside from markup errors, the main reason for that is overestimation of barcode borders in detection, which is caused by prioritizing high recall in training, it makes high impact for higher Jaccard thresholds as Jaccard index is known to be very sensitive to almost exact match (Fig. 5).

On Table III we show comparison of our models by precision and recall. Our models achieve close to an absolute recall, meaning that almost all barcodes are detected. On the other hand precision is also relatively high.

E. Execution time

For our network we measure time on 512x512 resolution which is enough for most of applications. We do not include postprocessing time as it is negligible compared to forward network run.

The developed network performs at real-time speed and is 3.5 times faster than YOLO with darknet [11] on higher resolution on the same GTX 1080 GPU. In the Table IV we compare inference times of our model with other approaches. We also provide CPU inference time (for Intel Core i5, 3.20GHz) of our model showing that it is nearly the same as reported in Cresot2016, where authors used their approach in the real-time smartphone application. It is important since not all of the devices have GPU yet.

F. Classification results

Among correctly detected objects we measured classification accuracy and achieved 60% accuracy on test set.

Moreover, classification subtask does not damage detection results. As shown in Table III the results with classification are even slightly better, meaning that detection and classification tasks can mutually benefit from each other.

Table II: Result comparation on different datasets.

	Muenster		Artelab	
	Acc J_{avg}	Detection rate $D_{0.5}$	Acc J_{avg}	Detection rate $D_{0.5}$
Cresot2015	0.799	0.963	0.763	0.893
Cresot2016	-	0.982	-	**0.989**
Yolo2017	0.873	**0.991**	0.816	0.926
Namane2017	**0.882**	0.966	**0.860**	0.930
Ours-Detection (all types)	0.842	0.980	0.819	**0.989**
Ours-Detection (EAN13 only)	0.762	0.987	0.790	**0.995**

Table III: Precision and recall of our approach on different datasets, Jaccard index threshold set to 0.5.

	Muenster		Artelab		Test Multiclass	
	Precision	Recall	Precision	Recall	Precision	Recall
Ours-detection (all types)	0.777	0.990	0.814	0.995	0.940	0.991
Ours-detection+classification (all types)	**0.805**	0.987	**0.854**	0.995	**0.943**	**0.994**
Ours-detection (EAN13 only)	0.759	**1.000**	0.839	**0.997**	-	-

Table IV: Inference time comparison

	Execution time (ms)	Resolution
Sörös [6]	73	960x723
Cresot16 [9]	40	640x480
Yolo17 [11]	13.6	416x416
Namane17 [10]	21	640x480
Ours (GPU)	3.8	512x512
Ours (CPU)	44	512x512

G. Capturing long narrow barcodes

Additional advantage of our detector is that it is capable of finding objects of any arbitrary shape and does not assume that objects should be approximately squares as done by YOLO. Some examples are provided in Fig. 2.

VI. Conclusion

We have introduced new barcode detector which can achieve comparable or better performance on public benchmarks and is much faster than other methods. Moreover, our model is universal barcode detector which is capable to detect both 1D and 2D barcodes of many different types. The model is very light with less than 33000 weights which can be considered very compact and suitable for mobile devices.

Despite being shallow (i.e. very simple, we didn't use any SOTA techniques for semantic segmentation) our model shows that semantic segmentation may be used for object detection efficiently. It also provides natural way to detect objects of arbitrary shape (e.g. very long but narrow).

Future work may include using more advanced approaches in semantic segmentation to develop better network architecture and increase performance.

References

[1] E. Ohbuchi, H. Hanaizumi, and L. A. Hock, "Barcode readers using the camera device in mobile phones," in *International Conference on Cyberworlds*, Nov 2004, pp. 260–265.

[2] S. Wachenfeld, S. Terlunen, and X. Jiang, "Robust recognition of 1-d barcodes using camera phones," in *19th International Conference on Pattern Recognition*, Dec 2008, pp. 1–4.

[3] O. Gallo and R. Manduchi, "Reading 1d barcodes with mobile phones using deformable templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 1834–1843, 2011.

[4] E. Tekin, D. Vasquez, and J. M. Coughlan, "S-k smartphone barcode reader for the blind," *J Technol Pers Disabil*, vol. 28, pp. 230–239, 2013.

[5] M. Katona and L. G. Nyúl, "Efficient 1d and 2d barcode detection using mathematical morphology," in *ISMM*, 2013.

[6] G. Sörös and C. Flörkemeier, "Blur-resistant joint 1d and 2d barcode localization for smartphones," in *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, ser. MUM '13. New York, NY, USA: ACM, 2013, pp. 11:1–11:8.

[7] C. Creusot and A. Munawar, "Real-time barcode detection in the wild," in *2015 IEEE Winter Conference on Applications of Computer Vision*, Jan 2015, pp. 239–245.

[8] A. Zamberletti, I. Gallo, and S. Albertini, "Robust angle invariant 1d barcode detection," in *2013 2nd IAPR Asian Conference on Pattern Recognition*, Nov 2013, pp. 160–164.

[9] C. Creusot and A. Munawar, "Low-computation egocentric barcode detector for the blind," in *IEEE International Conference on Image Processing (ICIP)*, Sep. 2016, pp. 2856–2860.

[10] A. Namane and M. Arezki, “Fast real time 1d barcode detection from webcam images using the bars detection method,” 07 2017.

[11] D. Kold Hansen, K. Nasrollahi, C. B. Rasmusen, and T. Moeslund, “Real-time barcode detection and classification using deep learning,” 01 2017, pp. 321–327.

[12] D. Deng, H. Liu, X. Li, and D. Cai, “Pixellink: Detecting scene text via instance segmentation,” in *AAAI*, 2018.

[13] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” *CoRR*, vol. abs/1511.07122, 2015.

[14] A. B. Jung, “imgaug,” https://github.com/aleju/imgaug, 2018, [Online; accessed 25-Dec-2018].