2020 IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
April 15-16, Ponta Delgada, Portugal

Detection of Data Matrix Encoded Landmarks in Unstructured Environments using Deep Learning

Tiago Almeida
IEETA, DEM
University of Aveiro
3810-193 Aveiro Portugal
tm.almeida@ua.pt

Vitor Santos
IEETA, DEM
University of Aveiro
3810-193 Aveiro, Portugal
vitor@ua.pt

Bernardo Lourenço
DEM
University of Aveiro
3810-193 Aveiro, Portugal
bernardo.lourenco@ua.pt

Pedro Fonseca
IT, DETI
University of Aveiro
3810-193 Aveiro, Portugal
pf@ua.pt

***Abstract*—Visual based navigation in industrial environments has been a challenge because of the complex and unstructured nature of the surroundings that Automated Guided Vehicles (AGV) and autonomous robots have to traverse in a continuous routine in many production facilities. Since natural landmarks are still a huge challenge due to the highly dynamic configuration of the environment and the varying nature of these landmarks, simple and low cost artificial landmarks appear as a potential solution. The one exploited in this work uses simple sheets of paper with Data Matrix encoded markers spread in the environment trying to create a constellation of landmarks in such a way that several of them are always visible to a set of cameras on board the robot or AGV. All codes are unique, which makes robot continuous localization a much simpler challenge whenever at least two or three landmarks are visible. When using traditional vision techniques in large images of the scene, one of the most demanding parts is to detect the landmarks to further process them. For that purpose, this paper proposes a technique based on deep learning that efficiently detects these special landmarks in images. A dedicated dataset was created and a Faster R-CNN architecture was adapted and trained for that purpose. The results show that almost all markers was detected in the images with a processing speed larger more than one order of magnitude than traditional techniques, including in demanding situations of poorer illumination or partial occlusions.**

***Index Terms*—deep learning, object detection, Data Matrix, AGV, self-localization, unstructured environments.**

I. Introduction

Automation is undeniably a prevalent domain in industrial daily tasks, mainly in two forms: collaboratively with workers or, in certain cases, autonomously, promoting machines to do the heavy work relieving human operators. One example of these types of applications is transportation in the shop floor, which are activities with very low added values and that consume human resources. That is where AGVs and autonomous robots perform a relevant role in the automatic transportation of parts, products and raw materials.

One of the most relevant problems associated to AGVs is guidance and localization in the industrial premises. Usually, in most real industrial practices, this is done by tracking floor magnetic lines or marking "wires" for the vehicle to follow. There are several disadvantages of this solution such as the poor flexibility of the path tracks, and therefore alternatives have been studied. Following the requests of companies interests in visual based localization of AGVs, approaches were carried out with the intention of using visual landmarks of easy installation and maintenance in [1] and [2]. The landmarks conceived by these works were based on Data Matrix codes [3] which are easy to create, build and place in the environment and, hopefully, easy to detect in camera images to allow a reliable robot self localization as those works describe in detail. Besides the word "landmark", the names "label", "passive beacon" or "marker" are frequently used in the literature, and will also be interchangeably used throughout this document.

There is one library in *Python* named *libdmtx*, that encodes and decodes Data Matrix labels. This library exhibits quite reliable results, but in a manufacturing facility the standards are usually higher. Thus, we propose in this work a Deep Learning (DL) method that outperforms the most robust known application to detect Data Matrix labels or markers. Since the existing library is based on classical techniques, it lacks high-processing speed and effectiveness in some special scenarios. Deep Learning has been proving to be an accurate tool used in Computer Vision based applications, such as object detection. So, our approach intends to supersede some of the existing limitations.

II. Related Work

The problem described in this work is inserted on a well-known computer vision branch, which is object recognition. This is the combination of two different tasks: image classification, which consists of the object class prediction, and object localization, whose output is a bounding box that locates the object in the image. In [4]–[6] the authors deployed the most used DL architectures to deal with the object recognition problem. While two of them considered single shot detection models (YOLO [6] and SSD [4]) and achieved fast object detection, the other one, Faster R-CNN [5], presents more accurate results. Thus, in this type of architectures, there is an interesting trade-off between the processing time to perform accurately and the maximum achievable performance.

The detection of Data Matrix labels using Deep Learning is not very common in the literature. However, there is a work developed in [7] that describes an approach based on semantic segmentation whose aim is to detect both 1D and 2D barcodes of many different types (including the one that is the

978-1-7281-7078-7/20/$31.00 ©2020 IEEE

core of the problem in this paper). This work lacks examples of challenging environments, since the tested benchmarks are composed of product barcodes, which is not the scope of the problem being addressed in this paper. Moreover, there are several implementations related to the detection of the most common simple barcode-based markers. Works in [8], [9] used YOLO to perform their study, while the authors in [10], [11] deployed the Faster R-CNN architecture. Finally, in [12] the authors developed the BarcodeNet based on PSPNet [13].

In summary, there are already several methods to detect simple barcodes, but these are prepared to be performant on very limited environments, since their purpose is more related to detect product tags. Our scope is different, since we use these tags to encode their own localization in more unstructured environments, such as manufacturing facilities, and are usually relatively small patches in large images acquired with larger fields of view.

III. Proposed Approach

In this work, we propose the usage of one convolutional neural network trained to detect Data Matrix markers placed in random environments. We use the Faster R-CNN architecture because it gives quite accurate results, which is a requirement for industry facility environments. Since there are no benchmarks/datasets for this purpose, a dataset was created to perform the neural network training.

The Faster R-CNN was trained through Detectron2, a research platform for object detection and segmentation [14]. The library was built on top of *PyTorch* framework and enables the training of several state-of-the-art models such as Faster R-CNN, Mask R-CNN, RetinaNet, and DensePose. For our dataset to be known by the framework used, we had to register it, which allowed to use the input features as well as the respective labels during the network training.

The dataset was created using an online toolbox[1], by uploading the images that compose the training set and doing the respective labeling (procedure described in Sec. III-A).

A. A dataset of Data Matrix images

In order to create a representative dataset, different environments were chosen (Fig. 1). One of them was a common laboratory room with several objects spread around. It is not a common scenario for an AGV but it helps to introduce different types of equipment, which allows the dataset to be more generic. Another environment was attempted to be similar to a usual AGV's scenario; it is a workshop with machinery and common structures from a manufacturing space. The images of these two environments were acquired with the same camera setup.

The dataset is composed of 156 frames, which is equally distributed by the two environments described above. The full set was labeled with bounding boxes, corresponding to a Data Matrix class.

There are several training examples that take into account different placements of the Data Matrix markers, which allows the dataset to represent a wide variability of transformations. Two examples of this dataset specificity are seen in Fig. 2.

[1] http://labelbox.com

Fig. 1. Samples of images used in the training set. The top image represents a visually cluttered environment (many different objects) and the bottom image represents an environment associated to a manufacturing facility.

In order to evaluate the model, one test set in two different scenarios (Fig. 3) was created. Similar to the procedure described for the training set, this test set was also labeled on the same online platform. However, the images of those two different scenarios were acquired with different camera setups. One of the environments is a hallway, which is an organized space with a small diversity of objects, while the other one is a different part of the workshop used in the training set, which is the most challenging environment.

Finally, some challenging images (Fig. 4) were used in this test set. These scenarios were specially prepared to evaluate the model performance in difficult environments, i.e. scenarios containing objects similar to the target object Data Matrix.

B. Faster R-CNN

The Faster R-CNN architecture (Fig. 5) is composed of two modules: a deep fully convolutional network and the Fast R-CNN detector [15]. The first, Region Proposal Network (RPN), hypothesizes object locations trough anchor boxes, each one with a score, based on a feature map created through the input image. The latter takes those object proposals to output a bounding box estimation and a confidence related to the object class.

The module responsible for producing region proposals has a backbone network to create feature maps at different

Fig. 2. Additional samples of images used in the training set. These two images show different planes and orientations for the Data Matrix codes. This allows the network to be more generic in the inference stage.

Fig. 3. Samples of images used in the test set: from visually neat to more filled and cluttered scenarios.

Fig. 4. Example of challenging images used in the test set which include objects that could confuse the detection, like chessboards or ArUco markers.

scales. In this work we used the ResNet50 [16] to perform this stage of the whole model. After that, a small network ($n \times n$ convolutional layer) slides over the feature maps, predicting multiple possible proposals for each unit of the "sliding window" and returning a lower-dimensional feature. This feature is the input of two 1×1 convolutional layers: the classifier layer (cls) that yields the probability of a proposal bounding a Data Matrix label, and the regression layer (reg) that encodes the coordinates of each proposal. The full layout of the RPN module can be seen in Fig. 6.

This part of the Faster R-CNN model is trained through a loss function that takes into account the two tasks mentioned above, the box classification and regression layers (cls and reg).

After the RPN training, the features that correspond to objects are cropped and re-scaled (ROI pooling), and are feed throw the final classifier that predicts the class of the object and a fine-tuned bounding box estimation. During inference, the non-maximum suppression (NMS) algorithm is applied to filter out the best bounding box in terms of image location. This algorithm is directly related to the application of an IoU threshold. IoU is an evaluation metric for algorithms that provide bounding boxes and is given by the quotient between the overlap area and the union area of two bounding boxes. Therefore, NMS is based on two key steps:

- Select the box with the highest score.
- Compute the overlap of this box with all other boxes, and remove boxes that overlap significantly (IoU $\geq$ threshold, where IoU means "Intersection over Union" and threshold is the maximum value for each IoU).

Then, these steps are iterated until there are no more boxes with lower score than the currently selected box.

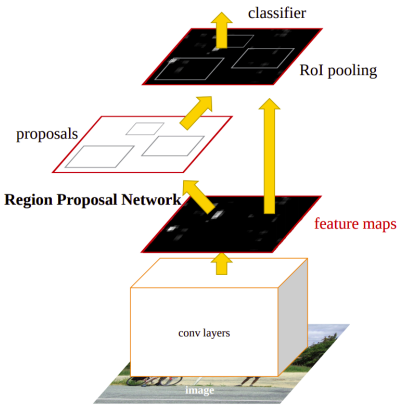

Fig. 5. Faster R-CNN architecture [5].

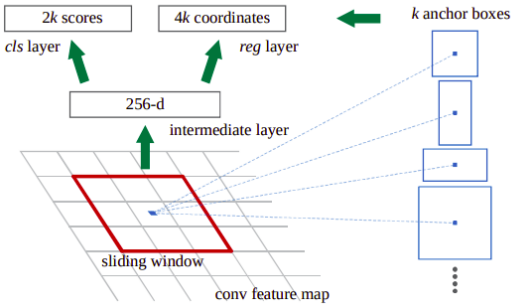

Fig. 6. The RPN layout [5]. Each unit of the sliding window is associated to k anchor boxes (proposals). Then, the classification layer outputs the probability of the target object is inside of each anchor box as well as anchor boxes' coordinates.

C. Implementation

In order to emulate an AGV, typically with lower heights, we used the robot of Fig. 7. Two different camera setups were used: one for the acquisition of the entire training set and some test set images with 8000×6000 pixels, and the other setup was used for the remaining test set images (4032×3024 pixels).

As mentioned before, we train both architecture modules on images of a single scale (8000×6000) as the original Faster R-CNN paper proposes, and the RPN module was initialized with the COCO pre-trained weights [17]. Moreover, in order to include more diversity of images in the dataset, some data augmentation techniques were performed such as image resizing and random flip.

The neural network was trained with a learning rate of 0.00025 during 4000 iterations and a batch size of two images in a single *NVidia RTX2080ti* GPU. The threshold of the NMS stage was set to 0.7.

IV. Localization Procedures

Once the labels are detected in the image, they can be extracted for decoding. As used in previous approaches [1], [2], the code yields absolute geometric coordinates in the shop

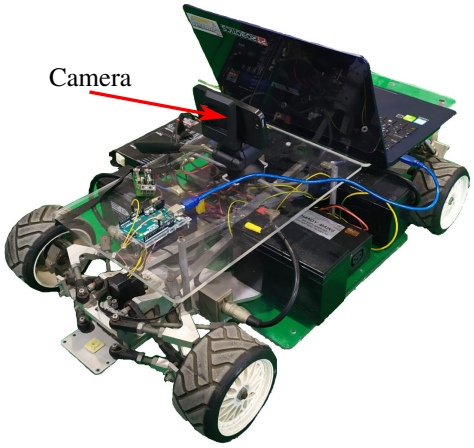

Fig. 7. The robot used to perform the acquisition of images for the training and test sets. However, two different cameras were used to acquire the images of the test set.

floor, which makes each of them unique and containing a meaningful information.

The traditional techniques for beacon-based localization in 2D, using trilateration, require two beacons whose distances are known; to solve the ambiguity, three beacons are actually needed; additionally, multilateration uses multiple beacons for enhanced localization. Similarly, triangulation techniques also require at least three beacons whose azimuths are used to calculate robot position.

Besides the techniques used in [1], purely deterministic, or [2], probabilistic using a Kalman Filter, the landmark detection in the approach proposed in this paper is now more performant, allowing the detection of labels in more positions and orientations. This means that not only "well placed" markers are useful, but also the ones with other orientations since the region detection with the R-CNN is now more accurate and powerful.

This enhanced detection capability opens the way for future versions of this work to detect also the perspective distortion of the labels. When that becomes true, a single label can be enough for localization if, for example, absolute orientation is also encoded in the Data Matrix information. This potential new source of information will allow further refinements of the solutions exploited in [1], [2]. It is clear that a single image may have uncertainties in the detection, therefore, more refined techniques may be needed for improved accuracy, but that is also left open for future work and developments.

V. Experiments and Results on Label Detection

We evaluate our detection method with the test set mentioned in Sec. III-A. This set consists of 224 images, in which 158 images correspond to the hallway scenario, while 66 frames were acquired in the workshop environment. Additionally, we followed the COCO dataset metrics [17]. These metrics are the average precision (AP) and average recall (AR), and they characterize the performance of an object detector trained with the COCO dataset. Since the scope of our work

is also object detection, those metrics fit perfectly on this problem. Just as a fast reference, knowing that T_P = True positives, F_P = False positives and F_N = False negatives, precision is given by $P = \frac{T_P}{T_P+F_P}$, and recall by $R = \frac{T_P}{T_P+F_N}$.

Table I shows the results obtained by evaluating the test set. Both metrics present in the table are averaged over multiple IoU thresholds (threshold $\in [0.50; 0.95]$ in steps of 0.05, which is henceforward noted by $[0.50 : 0.05 : 0.95]$). For AP, a constant IoU threshold of 0.50 (AP@0.5) is also computed, which is the most common value used in this type of architectures, but also for 0.75 (AP@0.75) and 0.95 (AP@0.95) thresholds. Moreover, these metrics were also applied across different scales of objects (small, medium, large or all). The objects are considered small when their area is less than 32^2 pixels, medium when its area is between 32^2 and 96^2 pixels, and large when the area is greater than 92^2 pixels. Finally, for the AR metric, the maximum number of detections ($maxDets$) was also varied taking the values $\{1, 10, 100\}$.

TABLE I
RESULTS OF DATA MATRIX LANDMARK DETECTION ON THE TEST SET.

	IoU Thresholds	Scales	maxDets	AP/AR values
AP	[0.50 : 0.05 : 0.95]	all	100	0.619
	0.50	all	100	0.876
	0.75	all	100	0.730
	0.95	small	100	0.364
	[0.50 : 0.05 : 0.95]	medium	100	0.565
	[0.50 : 0.05 : 0.95]	large	100	0.737
AR	[0.50 : 0.05 : 0.95]	all	1	0.288
	[0.50 : 0.05 : 0.95]	all	10	0.669
	[0.50 : 0.05 : 0.95]	all	100	0.670
	[0.50 : 0.05 : 0.95]	small	100	0.438
	[0.50 : 0.05 : 0.95]	medium	100	0.611
	[0.50 : 0.05 : 0.95]	large	100	0.786

In table I, the most relevant result is 0.876 for AP@0.5. In addition, for larger objects, as expected, both metrics are higher, which means that the model is more accurate when detecting larger objects (when the Data Matrix landmarks are closer to the camera). Finally, it is worth mentioning that the recall is higher than the precision, implying that the number of false positives (FP) is higher than the number of false negatives (FN). In other words, the model detects almost all the Data Matrix landmarks, but also detects some other objects that are not. Nonetheless, this is preferable, since a small number of false detections is not problematic, as the Data Matrix decoder is resilient to this problem, and the false negatives are obviously never accounted for and are missed information. Hence, in this context, higher recall is preferable to higher precision.

The running time for obtaining these results is $144\,\mathrm{ms}$ per image, which corresponds to $7\,\mathrm{fps}$ in the same hardware used to perform the model training. In order to compare the performance of this novel approach for Data Matrix labels detection against the classic algorithms used for the same task, we also run the test set on those classic algorithms based on *libdmtx* library. This analysis led to the following conclusions:

- Only $45\,\%$ of the test set frames were accurately processed by the classic algorithm.
- The running time for obtaining these results is $4.97\,\mathrm{s}$ per image, which means that the classic algorithm is almost 40 times slower than the model that we use in this paper.

We also provide some interesting qualitative results obtained by the Faster R-CNN model on the test set. The first example (Fig. 8) is the detection of partial landmarks. Here, besides the fact that the markers are placed in poor conditions or only part of the Data Matrix code is captured, the network was capable of returning valid detection results.

Fig. 8. Two scenarios with partially visible landmarks.

Two other examples can be seen in Fig. 9, which represent the detection of a marker in a floor reflection and under contrasting lighting conditions: markers in dimly lit spaces and placed in a window.

Regarding the workshop environment, there are also some interesting qualitative results. Two examples are shown in Fig. 10, which demonstrates the model resilience to different planes/orientations of the target objects.

On the other hand, there are also detection errors due to the presence of chessboard/ArUco images in some frames, as shown in Fig. 11. These two objects are similar to the Data Matrix labels and are not very representative in the dataset, thus, the model somehow has an expected behavior in these cases. Anyway, these objects are not very common in a manufacture facility (the normal environment for an AGV), so this is not a critical problem. In addition, these objects would be false positives, which is not problematic for the application. The inclusion of objects similar to Data Matrix labels on the dataset, such as barcodes, QR codes, ArUcos or chessboards could be used to minimize this type or errors.

Fig. 9. Two examples of landmark detection in challenging scenarios: an unexpected detection on floor reflection and a poorly illuminated scene.

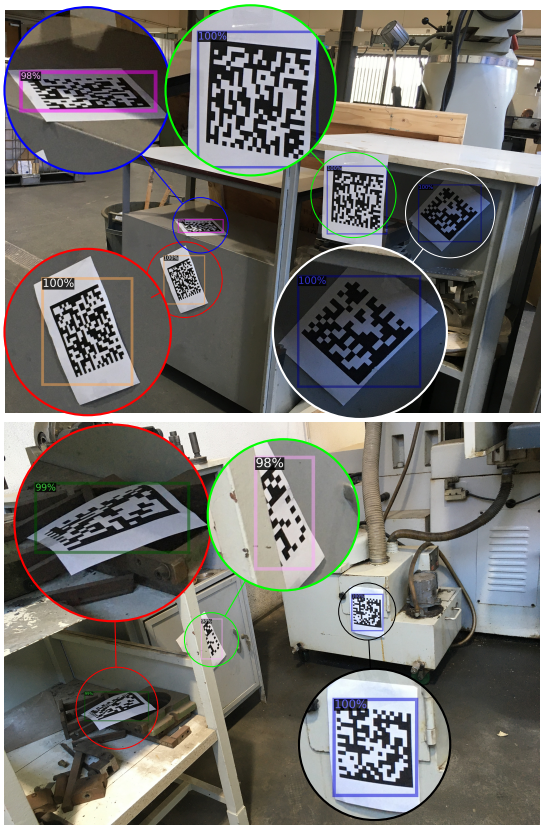

Fig. 10. Two examples of detection in challenging scenarios on the workshop environment: markers in variate planes/orientations and a partial label.

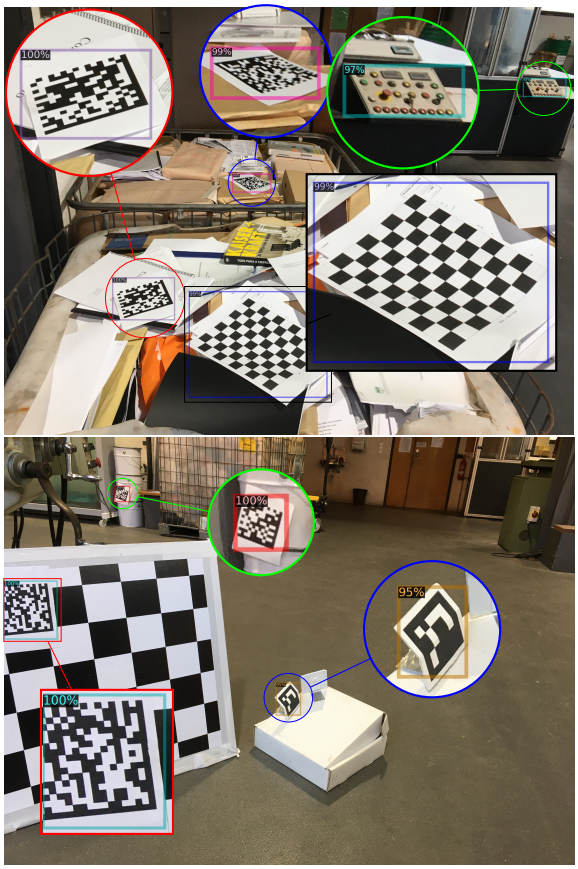

Fig. 11. Two examples of detection in very challenging scenarios on the workshop environment where detection error occur due to the presence of similar to Data Matrix labels.

VI. Conclusion and Future Work

This work presents an implementation of a Faster R-CNN model to detect landmarks with printed Data Matrix codes. This architecture performed accurately and with consistent results by detecting almost all the landmarks in the images tested, overcoming by far, in performance and frame rate, the traditional algorithms also tested.

These advantages bring more robustness and performance to the final AGV self-localization system. Therefore, it can create possibilities of new studies in order to improve what has been done in this work. Thus, future work should include the development of a neural network whose output are parallelograms and not simple rectangular bounding boxes as provided by the Faster R-CNN. If, in addition to the global position of the landmarks, the orientation in relation to a known reference was also encoded, by knowing the real dimension of the labels, a homography matrix can be used to obtain the transformation, hence, the relative placement of the robot with respect to the label. This could be a relevant breakthrough in the AGV self-localization problem, since its localization could be known by detecting only one single marker, or having more individual markers contributing to a finer global localization in challenging environments such as industrial shop floors.

ACKNOWLEDGEMENTS

This work was supported by Project PRODUTECH II SIF-POCI-01-0247-FEDER-024541, and partially supported by Project SeaAI-FA_02_2017_011.

REFERENCES

[1] L. Carrão, "*Sistema de Visão para Guiamento de AGV em Ambiente Industrial*," Master Thesis, University of Aveiro, 2014. [Online]. Available: http://hdl.handle.net/10773/13697

[2] M. Bergamin, "Indoor localization using visual information and passive landmarks," Master Thesis, Universities of Aveiro and Padova, 2015. [Online]. Available: http://tesi.cab.unipd.it/50395/1/bergamin_marco.pdf

[3] ISO, "Information technology – automatic identification and data capture techniques – data matrix bar code symbology specification," International Organization for Standardization, Geneva, CH, Standard ISO/IEC 16022:2006, 2006. [Online]. Available: https://www.iso.org/standard/57333.html

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015. [Online]. Available: http://arxiv.org/abs/1512.02325

[5] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[6] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015. [Online]. Available: http://arxiv.org/abs/1506.02640

[7] A. Zharkov and I. Zagaynov, "Universal barcode detector via semantic segmentation," *CoRR*, vol. abs/1906.06281, 2019. [Online]. Available: http://arxiv.org/abs/1906.06281

[8] Y. Xiao and Z. Ming, "1d barcode detection via integrated deep-learning and geometric approach," *Applied Sciences*, vol. 9, no. 16, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/16/3268

[9] D. Hansen, K. Nasrollahi, C. Rasmusen, and T. Moeslund, "Real-time barcode detection and classification using deep learning," in *IJCCI*, 01 2017, pp. 321–327.

[10] S. Mushfika, A. Islam Chowdhury, K. Zawad, T. Mahee, R. Ahmed, and N. Sakib, "An approach to facilitate business system by multiple barcode detection using faster rcnn," *International Journal of Applied Information Systems*, pp. 10–15, 12 2019.

[11] Q. Yang, G. Golwala, S. Sundaram, P. Lee, and J. Allebach, "Barcode detection and decoding in on-line fashion images," *Electronic Imaging*, vol. 2019, no. 8, pp. 413–1–413–7, 2019.

[12] Q. Zhao, F. Ni, Y. Song, Y. Wang, and Z. Tang, "Deep dual pyramid network for barcode segmentation using barcode-30k database," *CoRR*, vol. abs/1807.11886, 2018. [Online]. Available: http://arxiv.org/abs/1807.11886

[13] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *CoRR*, vol. abs/1612.01105, 2016. [Online]. Available: http://arxiv.org/abs/1612.01105

[14] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[15] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. [Online]. Available: http://arxiv.org/abs/1504.08083

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[17] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312