Journal of Intelligent & Robotic Systems (2021) 103: 13
https://doi.org/10.1007/s10846-021-01442-x

REGULAR PAPER

Comparative Analysis of Deep Neural Networks for the Detection and Decoding of Data Matrix Landmarks in Cluttered Indoor Environments

Tiago Almeida[1,2] · Vitor Santos[1] · Oscar Martinez Mozos[2] · Bernardo Lourenço[1]

Received: 20 August 2020 / Accepted: 17 June 2021 / Published online: 11 August 2021

Abstract

Data Matrix patterns imprinted as passive visual landmarks have shown to be a valid solution for the self-localization of Automated Guided Vehicles (AGVs) in shop floors. However, existing Data Matrix decoding applications take a long time to detect and segment the markers in the input image. Therefore, this paper proposes a pipeline where the detector is based on a real-time Deep Learning network and the decoder is a conventional method, i.e. the implementation in *libdmtx*. To do so, several types of Deep Neural Networks (DNNs) for object detection were studied, trained, compared, and assessed. The architectures range from region proposals (Faster R-CNN) to single-shot methods (SSD and YOLO). This study focused on performance and processing time to select the best Deep Learning (DL) model to carry out the detection of the visual markers. Additionally, a specific data set was created to evaluate those networks. This test set includes demanding situations, such as high illumination gradients in the same scene and Data Matrix markers positioned in skewed planes. The proposed approach outperformed the best known and most used Data Matrix decoder available in libraries like *libdmtx*.

Keywords Deep learning · Data matrix · Detection · Decoding · Localization

1 Introduction

The industrial demand and competitiveness foster the increase of new, more sophisticated, and effective techniques in daily industrial tasks [1, 2]. One important and valuable task for a manufacturing facility is the automatic transportation of components and materials [3–5].

Transportation tasks are being performed by AGVs, which are industrial robots that travel from point to point, usually by following a magnetic wire or stripe on the shop floor [6, 7]. Although this is a widely used technique, these methods present serious disadvantages in terms of performance and logistics such as fixed path tracks and limited flexibility, plus the fact that the performance may decline over time [8].

Hence, solutions based on landmarks have been proposed to solve or improve the major drawbacks presented by those systems; in that line, visual passive landmarks appear as a simple but very interesting solution [9–11], and one promising approach has been tried and exploited in the work of Bergamin [8], as shown in Fig. 1.

The landmarks (markers) are encoded with specific information, such as the respective world coordinates, which, once detected, are used to compute the robot localization through trilateration techniques. The overall technique showed very interesting results in the localization and navigation procedures, but had a limited performance in detecting the markers (Data Matrix) in the images from the environment. In this work, the Data Matrix landmarks were selected to the detriment of the other existing 1D/2D markers (barcodes

✉ Tiago Almeida
tiago.almeida@oru.se

Vitor Santos
vitor@ua.pt

Oscar Martinez Mozos
oscar.mozos@oru.se

Bernardo Lourenço
bernardo.lourenco@ua.pt

1 IEETA, DEM, University of Aveiro, 3810-193 Aveiro, Portugal

2 Center for Applied Autonomous Sensor Systems (AASS), Örebro University, 702 81 Örebro, Sweden

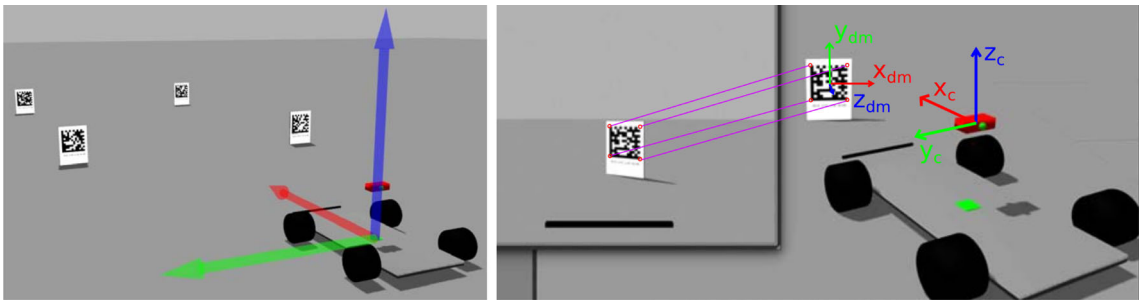

Fig. 1 Visual landmarks encoded with Data Matrix labels to perform robot localization using trilateration and related techniques [8]

and QR-codes, respectively) since they allow to have larger unit cells. Furthermore, they are considered a quite cheap, flexible, and robust solution according to the problem addressed in this work [8].

In that context, the work developed in [12] took that challenge of locating visual encoded landmarks (Data Matrix) in the environment using a DNN. That work proved that DL architectures to locate Data Matrix are capable of overcoming the barriers imposed by the traditional techniques, namely accuracy and processing time during the detection task. Even though the architecture presented in that work shows a good performance in most situations, it still presents high processing time (144ms per image in a single *Nvidia RTX2080ti GPU* just for the detection task). Therefore, in this work we try to develop a solution that balances effectiveness with low-latency. In this way, several different Deep Learning approaches for the marker detection task are studied, described, and assessed. They all have their own advantages and disadvantages, and the objective is to select the most suitable taking into account both accuracy and latency.

The remaining part of the paper is organized as follows. In Section 2, we exploit and discuss several previous works related to the one presented in this paper. In Section 3, we present our methodology and the different DNNs deployed, as well as each variant that they may have in the following sections. Section 4 enumerates and describes the baselines for the training of each network. Then, we characterize each experiment and the respective results throughout Section 5. Finally, in Section 6, we conclude the paper and define future research ideas based on the current work.

2 Related Work

The proposed self-localization approach stands out for a constellation of markers in the environment (a workshop). There are several works in the literature addressing robotics problems that make use of other techniques and landmarks. In [13, 14], the authors used ArUco markers and denoted many limitations in terms of the fiducial markers detection, which is crucial in the overall application. Recently, in [15], the authors designed a real-time solution for limited size landmarks detection. This technique would then be used in [16], where the authors do not present any relevant result regarding localization, but present and describe an overall architecture that allows to compute it. Moreover, in [17], the authors propose an algorithm based on classical techniques to detect ARTag markers. This system was not designed to localize the robot according to a world frame, but it yields the distance between the camera and the agent. Finally, in [18], the authors compute the robot pose through the recognition of customized circular landmarks displaced in the ceiling.

In the problem addressed in this paper, the detection of Data Matrix labels in wide images is a challenging task since the labels occur at multiple scales in cluttered and unstructured environments. The process of detecting and decoding these targets is very time-consuming and in some cases inaccurate for classical algorithms [19, 20], that is why DNNs started being devised to perform the detection stage [21]. Therefore, in this work, an important additional characteristic for the DL architecture is included: low-latency. This could be quite cumbersome to obtain because there is usually a trade-off between the performance of multi-scale objects detection and the latency of the network [22]. There are several studies related to the detection of this type of markers in [21, 23–25]. However, they all present results for structured environments and in limited situations. In [12], a Faster R-CNN architecture was proposed to detect Data Matrix landmarks in unstructured scenarios. This architecture is quite accurate in detecting objects at multiple scales and outperformed by far the traditional algorithm provided by the *libdmtx* Python's library in processing time. However, those improvements are not sufficient because for a system operating in real-time, this architecture is not the most suited. That is the main reason to extend the study to several other types of DL-based models, and investigate the real implication in a full pipeline, i.e. including the decoding stage.

There are many types of DNNs to perform object detection, but the most well-known and used ones are region

proposal networks represented, currently, by Faster R-CNN [26] and single-shot approaches, mainly composed of [27] and [28] — Single Shot Multibox Detector (SSD) and You Only Look Once (YOLO), respectively. The main differences between original single-shot methods [27, 28] and proposal networks [26] are the accuracy in locating objects (higher for proposal networks) and the overall processing time (lower for single-shot methods). Despite these characteristics, each model can be customized in two forms to influence both processing time and detection performance: the choice of backbones (or feature extractors) [29, 30] and the application of multi-scale detection techniques [31–34].

The function of the backbone network is to extract features from the input image [30]. This shows its importance in the final result since features with no semantic meaning imply networks' receptive fields empty of information [35]. Regarding Faster R-CNN and SSD, there are several conventional backbones that are used to perform the feature extraction task: VGG [36], ResNet [37], DenseNet [38], MobileNet [39], SqueezeNet [40], and ShuffleNet [41] are some of the available options. VGG was conceived in a work that proved that deeper networks could achieve better results than shallower networks. It is difficult to train from scratch as the straight layers connection, which may even cause vanishing gradient problems. After VGG, ResNet appeared with a novel layout for convolutional layers: residual blocks with skip connections, which allow an easier training of deeper networks. DenseNet is characterized by a novel transmission of semantic information between layers, in which every layer's result is passed throughout the following layers. Finally, the smallest architectures — MobileNet [39], SqueezeNet [40] and ShuffleNet [41] — provide faster predictions in preference to higher accuracy.

The multiple scales detection techniques are characterized by neural networks design choices and training details such as different ways of concatenating strong semantic information from multiple scales and the usage of a particular loss function that values the detection of more difficult classes. Regarding concatenation techniques, one of the most well-known approaches is the usage of Feature Pyramid Networks (FPNs) [31], which upsample semantic stronger feature maps and merge them to semantic weaker activation maps from the downsampling pathway. SSD also uses multi-scale feature maps but lacks semantic information, which is crucial for small object detection. Similarly, Feature Fused SSD [32] also adds semantic information from deeper layers to shallower feature maps through concatenation or element-sum modules. The former allows reducing the interference of a noisy background and the latter enhances the contextual information. YOLOv3 [42] improves also the detection of small objects by concatenating global features of multi-scale convolutional layers. One step further in the multi-scale object detection problem is the application of a Spatial Pyramid Pooling (SPP) to YOLO [33], which also fuses multi-scale local region features from the same convolutional layer. Finally, one training detail that can produce better results is the usage of focal loss [34], which penalizes more the miss-classifications of the most challenging classes.

Table 1 shows an overview of the main types of architectures useful for this work, and summarizes the main points of the networks implemented in this study, which are discussed in detail next.

3 Methodology and Deep Networks

The motivation behind the usage of a DNN to locate the landmarks in a full frame is that the *libdmtx* method performs much slower in full images than it does in small patches of the input image; so, the DL-based model locates the Data Matrix label bounding-box in the input image, and then each image patch is decoded by the *libdmtx*. Accordingly, the proposal in this paper consists of the development of the initial phase of the full pipeline, which can be divided into two stages: the first where the Data Matrix marker is located, and the second where the label is actually decoded. The pipeline's workflow is shown in Fig. 2 with a real example.

Indeed, the first stage of the pipeline is the focal point of this paper. In this way, throughout the document we discuss and describe the most suited DNN to perform the detection of the landmarks taking into consideration the balance between accuracy and latency. Hence, this study focuses on a comparison of several DL architectures with different backbones and training procedures for the Data Matrix detection task. The study starts with Faster R-CNN model, comparing the results of the ResNet50 FPN, ResNet50, and MobileNetV2 training, performed with the same data augmentation techniques. Then, faster predictions are provided by studying the influence of the usage of different feature extractors (ResNet50 and MobileNetV2) on SSD512. Finally, different versions of YOLO (v3 and v4) are trained, and the application of the Spatial Pyramid Pooling technique to the third version (YOLOv3 SPP) is also carried out.

This work was developed under the *PyTorch* framework and the code is publicly available[1].

3.1 Faster R-CNN

Region proposal architectures usually provide high-quality results with high-latency. Faster R-CNN is one example of these type of networks and is composed of two stages: the proposals computation after features extraction, and the final detection through Fast R-CNN detector [43]. The

[1] https://github.com/tmralmeida/data-matrix-detection-benchmark

Table 1 Summary of object detectors and implementations in this work

Types of detectors	Methods	MS detection	Implementations in this work
Region Proposal	Faster R-CNN	FPN —	ResNet50 FPN ResNet50, MobileNetV2
Single-Shot	SSD	MS FM FF	Resnet50, MobileNetV2 —
	YOLO	MS FM, SPP MS FM, SPP, PANet	YOLOv3, v3 SPP YOLOv4

MS multi-scale, *FPN* feature pyramid network, *FM* feature maps, *FF* feature fused, *SPP* spatial pyramid pooling

extraction of feature maps is performed by the backbone — a common classification DNN. Then, a small network slides over the feature maps, predicting multiple possible boxes for each of their cells through its output — a lower-dimensional feature. This output is fed to two 1×1 convolutional layers, which yield the probability and the encoded coordinates of each proposal. Finally, the most semantic valuable features (with higher objectness, that is a measurement that describes an object class) pass through an ROI pooling layer, which crop and re-scale the feature maps into fixed size feature maps. During inference, the non-maximum-suppression (NMS) algorithm filters out the best-located bounding boxes. This technique is transverse to all object detection algorithms that are described and used in this work.

3.2 SSD

Single-shot methods, like SSD and YOLO, can process the input faster, since the location and classification tasks are done in a single forward fashion. SSD, similarly to Faster R-CNN, has a conventional classification network (here we just used ResNet50 and MobileNetV2) that produces feature maps. Then, it skips the region proposal stage and yields final predictions at once. To do so, some extra layers

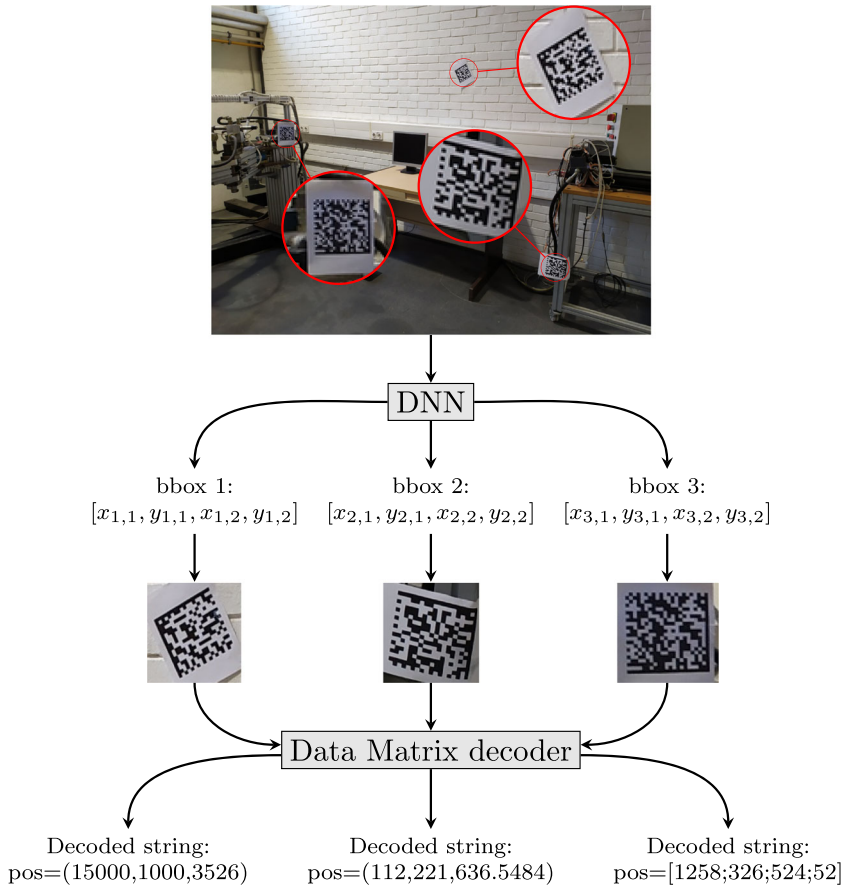

Fig. 2 Data Matrix decoding workflow. From top to bottom: an input image is processed by a deep model (DNN), which yields the Data Matrix locations in the image. Finally, the image patches are decoded by the *libdmtx* library. The encoded strings depicted in this image are arbitrary and just for illustration purposes

are attached to the backbone yielding multi-scaled feature maps. Moreover, each of these extra layers provides a fixed set of detection predictions using convolutional filters. Finally, the model outputs the score for each category and the location of the boxes that bound the targets.

3.3 YOLOv3, YOLOv3 SPP, and YOLOv4

The YOLOv3 network [42], differently from the architectures presented so far, has a custom feature extractor — DarkNet53. This is also a convolutional neural network and, similarly to SSD, predicts three multi-scaled feature maps. It has 106 layers and has an interesting particularity that improves the object detection results: it concatenates feature maps of shallower layers (with low-level features) to the result of upsampled and deeper feature maps (FPN approach). This provides activation maps more representative of global features of different-sized objects. Moreover, the application of SPP implies an additional block after the input's downsampling, which pools and concatenates multi-scale local region features (through max pool layers). This enables the usage of both global and local multi-scale features for the object detection task. Finally, the detection is performed by applying 1×1 convolutional detection filters on the three different feature maps.

On the other hand, YOLOv4 [29] is composed of a Cross Stage Partial (CSP [44]) Darknet53 with an SPP module, a path-aggregation network (PANet [45]), and a YOLOv3 head. CSP networks have similar basis and purposes to a DenseNet. Therefore, this architecture enhances the features reuse by reducing the amount of repeated gradient information observed in a DenseNet. To do so, it divides the base feature map, then one part of the channels passes through a partial dense block and the remaining part undergoes to the final partial transition layer. After the activation maps production, the only difference between YOLOv3 and YOLOv4 in terms of architecture's layout is the global features concatenation. In YOLOv4, instead of the FPN technique, a custom PANet approach is used [46]. PANet is simply an enhanced version of FPN; after the FPN's block composed of a top-down pathway with lateral connections, PANet also propagates low-level features through a bottom-up path augmentation block. This block allows the addition (concatenation for YOLOv4) of the FPN resulting features with the output of those feature maps with 3×3 convolutions, which yields an even better understanding of the low-level features.

3.4 Conventional Backbones

Backbones have a key role to play in the aforementioned type of architectures, since they are the activation maps producers, which contain the semantic value that allows to identify an object in the input image. Here, we describe and explain the backbones used in Faster R-CNN and SSD networks.

Faster R-CNN and SSD were trained with the same ResNet50 and MobileNetV2 backbones. They differ in both feature maps quality (in terms of semantic importance) and, subsequently, inference time. ResNet50 is a deep convolutional neural network that provides high accuracy by employing residual blocks with skip connections. These blocks propose to fit a residual mapping from the input's layer to the output, instead of directly trying to fit an underlying transformation. MobileNetV2 is a shallower, variable-width neural network, which is based on depthwise separable convolutions. It is not a common convolution in which the kernel and input depths are the same, but a combination of a depthwise and a pointwise convolutions. In a depthwise convolution, the input and the kernel are divided into channels and each kernel is separately applied to each input channel. A pointwise convolution implies the application of an 1×1 filter throughout the input channels. Hence, a depthwise separable convolution is composed of two stages: the depthwise convolution and a final 1×1 convolutional operation. Furthermore, ResNet50 FPN was also implemented in Faster-RCNN. The only difference comparing to the original ResNet50's layout is the strengthening of semantic weaker feature maps by concatenating them to semantic stronger ones. As mentioned before, this technique can provide a better performance by detecting multi-scaled objects.

4 Baselines

In order to compare and evaluate precisely the networks, they were all trained from scratch. The training and validation sets are the same as those conceived in [12], but at a quarter of the original size of the images (1500×2000) because the previous input shape did not correspond to what was going to be used in the final system and hampered the training of low-latency networks like SSD and YOLO. Moreover, data augmentation techniques were performed to increase the training input variance. In this way, the models can generalize for more situations, which means a more robust solution.

4.1 Faster R-CNN

The first baseline, Faster R-CNN, groups together ResNet50 FPN, ResNet50, and MobileNetV2 backbones. These models were trained with one of the geometric transformations: random crops with a final size of 480×640 or 960×1280, or just a resize of 750×1000 (half of the training input size). In addition, random brightness, random contrast and

horizontal flip were also applied. Further, the training was done for 200 epochs with an AdamW optimizer and a cosine annealing scheduler with a warm up of 100 iterations (the learning rate scheduler is common to all baselines). The learning rate was set to 10^{-3}, the weight decay to 10^{-4}, and the batch size to 4.

4.2 SSD512

The second baseline is composed of all SSD variants — ResNet50 and MobileNetV2. The augmentation performed here is the same as the one presented in [27], taking into account an input size of 512 × 512. These architectures were trained for 300 epochs with the AdamW optimizer. Finally, the learning rate was set to 10^{-3}, the weight decay to 4×10^{-5}, and the batch size to 16.

4.3 YOLO

The YOLO baseline is common to YOLOv3, YOLOv3 SPP, and YOLOv4. Here, two augmentation approaches were performed and compared: with and without mosaic augmentation (presented in YOLOv4 [29]), which can be seen in Fig. 3.

These YOLO approaches have in common the application of both random horizontal and vertical flip, an HSV color space augmentation, and input size of 672 × 896. The usage of mosaic augmentation (upper images in Fig. 3) allows the model to be more generic since it learns from 4 different contexts in one single image. One of the augmentation approaches, instead of applying mosaic augmentation, runs random affine transformations (bottom images in Fig. 3) such as image rotation ($[-1.98°, 1.98°]$), translation (not greater than 0.05 of both image dimensions), re-scaling (increasing or decreasing 5% of the original image size), and image shear ($[-0.641°, 0.641°]$). These networks were trained for 400 epochs with the SGD optimizer with a learning rate of 10^{-3}, a weight decay of 10^{-4}, and batch size of 4.

Fig. 3 Two different data augmentation methods applied during YOLO networks training: two examples of mosaic augmented images (upper images) and random affine transformed frames at the bottom

The two augmentation approaches provide two different training procedures for the YOLO baseline. However, just one of those yields the models that are evaluated on the test set. To do so, after the hyperparameters tuning, the two approaches are compared on the validation set. The average precision (AP) and the average recall (AR) results of the YOLO variants are shown in Tables 2 and 3, respectively. There, the numerical subscripts represent the IoU threshold, i.e. 50 means an IoU threshold of 0.50, and the letter subscripts correspond to the scale of the objects, i.e. "S" represents small objects (pixel area $< 32^2$), "M" medium (pixel area $\in [32^2, 96^2]$), and "L" large (pixel area $> 96^2$). Finally, the metric without numerical subscript means an average over multiple IoU thresholds within [0.50, 0.95] in steps of 0.05.

In most cases, Tables 2 and 3 show better results from mosaic augmentation training for every model, thus, from now on, any YOLO result that is shown in this document was obtained through this data augmentation procedure.

4.4 Summary

To sum up, three baselines are set up, which correspond to each type of architecture (Faster R-CNN, SSD, and YOLO). Thus, the models of the same baseline (i.e. same type of architecture with different combinations of backbones) have common training procedures. The main hyperparameters discussed in the previous sections are summarized in Table 4.

5 Experiments

This work presents a collection of deep networks for Data Matrix detection. Since all models were tuned by using the validation set, this could not be considered as the set to perform the numerical comparisons and assessments. Therefore, a test set was created similarly to the sets conceived in [12]. This set of frames was processed only once by each model and that is why the networks results presented throughout this Section are unbiased and valid. Additionally, in this Section, we also report Data Matrix decoding results for the model that presented the best overall trade-off between detection and processing time in the test set.

Table 2 AP results of Data Matrix landmarks detection on the validation set for YOLO models with two different augmentation approaches

Augmentation	Variants	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Random affine transformation	v3	35.0	64.2	32.5	26.5	54.5	80.0
	v3SPP	36.4	66.5	36.6	26.7	56.6	80.0
	v4	37.1	63.8	40.8	27.3	58.4	90.0
Mosaic	v3	37.5	64.6	38.0	27.6	58.8	90.0
	v3SPP	38.3	64.0	40.0	28.0	61.7	80.0
	v4	39.2	62.7	42.8	28.9	62.6	90.0

5.1 A data set of Data Matrix images

The full dataset was manually labeled in an online toolbox[2] and it is composed of three different subsets: the training set, the validation set, and the test set. The first two were designed in [12], where the training set has 156 frames equally distributed by two different scenarios (a lab and a workshop), and the validation set is also divided into two environments — a hallway (158 frames) and a different workshop (66 frames). Therefore, both training and validation sets were collected in different scenarios to avoid biased results during neural networks training.

Additionally, the full test set was conceived during this work and it has 145 frames with 895 annotations. It comprises three scenarios, which are different from those used for the training and validation sets. One of them is a neat hall with overshadowed and over-lightened landmarks in different planes as can be seen in Fig 4. The second environment is a classroom laboratory with various electronic equipment arranged in an orderly manner (Fig. 5). Since the final robotic system is expected to operate in cluttered workshops, a third scenario was made very challenging with multiple machinery spread out all over the place and also with a more diversified range of materials (Fig. 6).

5.2 Detection results

The detection algorithms were evaluated in the set test described in Section 5.1. To do so, and analogously to [12] and Section 4.3, AP and AR were calculated. The results obtained for AP are presented in Table 5.

A broad view of the AP results shows that among all Faster R-CNN variants, the one with best results is ResNet50 FPN, as expected. Regarding SSD neural networks, the ResNet50 backbone is the most accurate one. As far as YOLO models are concerned, although all results were quite similar, YOLOv4 was the one that stood out most among the others.

In addition to AP, Table 6 shows the AR and the averaged processing time results throughout the test set.

The results presented in Table 6 are in line with the AP results. The most relevant result is the 75.3% provided by Faster R-CNN with ResNet50 FPN, but all YOLO models, especially YOLOv4, performed well above average with the strengthening factor of a lower processing time, yielding results at 47.6fps.

An overview of both Tables 5 and 6 shows that shallower backbones like MobileNetV2 struggle to achieve results comparable to the deeper networks, but in return they are much faster. Moreover, FPN/PANet techniques enable to improve the detection of small objects (directly comparing SSD and Faster R-CNN with YOLO results).

All YOLO variants provide quite interesting results, namely in terms of recall, which is the most appealing metric since the characteristics of the system are such that false positives can easily be ruled out (false negatives are easily discarded by the downstream decoding stage, but add computational overhead). Also, it seems that the SPP and the PANet modules (YOLOv3 SPP and YOLOv4) help to increase both AP and AR results. Overall, YOLOv4 shows the best trade-off between qualitative and latency results and, therefore, it should be the solution deployed in the final robotic system.

5.3 Qualitative detection results on the test set

Theoretical insights have shown that DL-based models generalization is one of the most critical and important points to consider. This is because biased models for training/validation sets do not have the expected outcome in real world applications. Therefore, we show some interesting visual results/comparisons between some of the models whose results were presented and discussed in the previous Section.

The first visual example (Fig. 7) represents three frames from the test set processed by the models, which obtained the best numerical results for each baseline, i.e. ResNet50 FPN from Faster R-CNN, ResNet50 from SSD, and YOLOv4 from YOLO. It is worth mentioning that the confidence threshold used for the YOLO variants in this

[2] http://labelbox.com

Table 3 AR results of Data Matrix landmarks detection on the validation set for YOLO models with two different augmentation approaches

Augmentation	Variants	AR	AR_S	AR_M	AR_L
Random affine transformation	v3	45.1	36.1	64.3	80.0
	v3SPP	46.7	37.8	65.8	80.0
	v4	47.3	37.2	68.9	90.0
Mosaic	v3	46.3	36.7	66.8	90.0
	v3SPP	46.7	35.9	69.7	80.0
	v4	47.7	36.7	71.1	90.0

Table 4 Summary of the main training hyperparameters for the 3 baseline DNNs

Baselines	Epochs	Optimizer	Learning rate	Weight decay	Batch size
Faster R-CNN	200	AdamW	10^{-3}	4×10^{-5}	4
SSD512	300	AdamW	10^{-3}	4×10^{-5}	16
YOLO	400	SGD	10^{-3}	10^{-4}	4

Fig. 4 Examples of two labeled test set images of the hall environment: one containing different types of lighting (left) and the other with several landmarks spread over different planes in the scene (right)

Fig. 5 Two labeled test set frames of the classroom scenario: one with electronic devices and landmarks in the desks (left) and the other with targets on shelves and walls (right)

Fig. 6 Examples of two labeled test set images of an "workshop" scene with several different objects and machinery

Table 5 AP results of Data Matrix landmarks detection on the test set

	Variants	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN	ResNet50 FPN	**72.2**	**90.6**	**85.4**	22.2	69.8	**81.5**
	ResNet50	61.8	85.5	73.0	< 1	54.8	79.4
	MobileNetV2	52.2	78.3	58.3	< 1	43.2	74.3
SSD 512	ResNet50	45.5	68.3	54.6	< 1	37.7	64.9
	MobileNetV2	25.5	44.1	27.0	< 1	12.8	52.9
YOLO	v3	59.4	85.7	70.7	43.5	70.4	77.0
	v3 SPP	59.4	86.3	69.8	**43.7**	70.6	76.9
	v4	60.6	84.6	70.4	42.7	**72.9**	78.7

Highest scores for each metric are highlighted

Table 6 AR and fps results of Data Matrix landmarks detection on the test set

	Variants	AR	AR_S	AR_M	AR_L	fps
Faster R-CNN	ResNet50 FPN	**75.3**	39.6	73.0	**85.1**	5.4
	ResNet50	66.1	11.2	61.4	82.4	3.9
	MobileNetV2	56.6	< 1	49.4	78.0	7.7
SSD 512	ResNet50	49.0	< 1	41.9	69.1	48.3
	MobileNetV2	27.7	< 1	15.1	56.9	**67.1**
YOLO	v3	66.3	54.4	75.5	80.4	39.8
	v3 SPP	66.8	55.0	76.0	80.8	37.1
	v4	68.0	**55.2**	**78.0**	81.6	47.6

Highest scores for each metric are highlighted

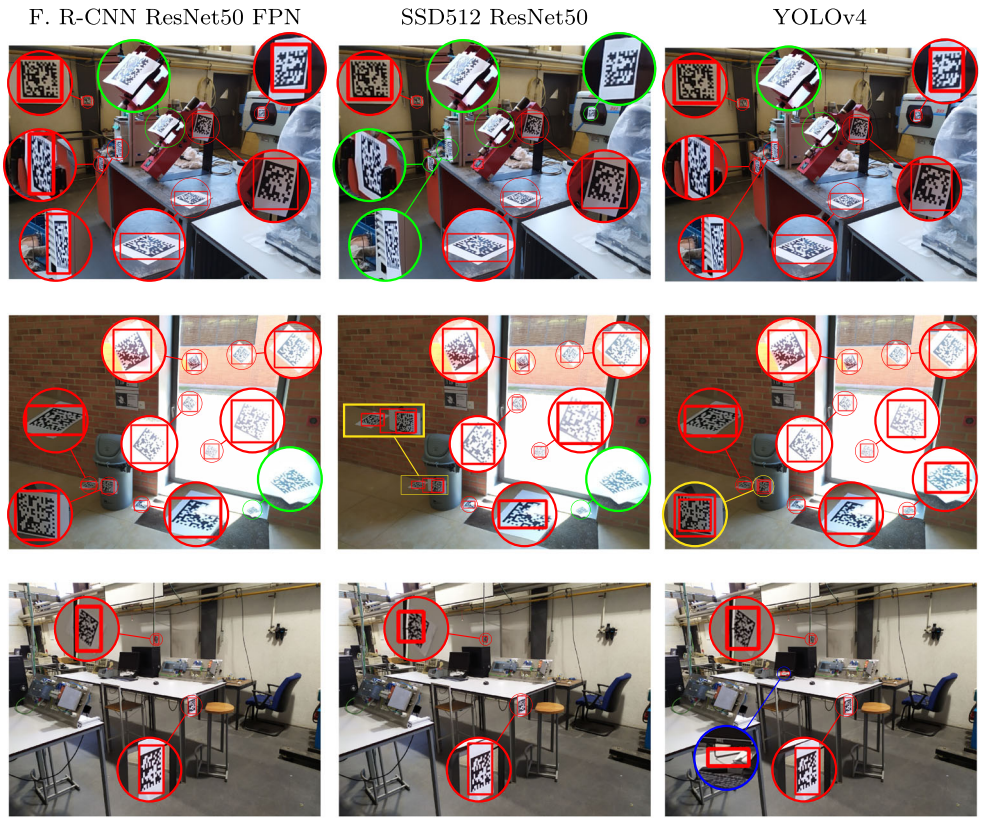

Fig. 7 Visual results from the test set, where each zoom color has a different meaning: red is a true positive, green corresponds to a false negative, blue pertains to the false positives, and yellow highlights ineffective NMS outcomes. Each row contains a frame that was processed by the best variant of each architecture. In the first row there is an example from the "workshop" scenario with machinery; the second row is an image from the hall environment, in which the landmarks are on different illumination conditions; finally, the images of the third row are from the classroom where there is one partial label correctly classified

subsection is not the same as the one used on the test set evaluation. In this way, the false positives are greatly reduced, thus presenting better overall results. Therefore, the YOLO minimum object confidence was changed from 0.001 to 0.05.

From these images, it is possible to infer that Faster R-CNN with the ResNet50 FPN as the backbone and YOLOv4 are the most accurate models with very similar predictions. SSD512, in the "workshop" image, faced problems inferring labels in skewed planes and also failed a label on the hall scenario with very poor illumination conditions.

The second example joins the results provided by the faster variants of each architecture, i.e. MobileNetV2 from Faster R-CNN and SSD512, and YOLOv4 from YOLO. The results can be seen in Fig. 8, which shows a clear example of why YOLOv4 is the best model to perform the Data Matrix detection in this context because, besides being qualitatively more performing than the other two models of each baseline, it is also a fast algorithm. Furthermore, we can also note that throughout the YOLOv4 results, the NMS threshold might not be optimally defined since there are several failures in the suppression of similar bounding boxes. Therefore, in the next Section we discuss the most suited value for this parameter, because it could be a bottleneck of the pipeline, since the decoder spends time trying to decode twice the same Data Matrix.

Table 7 Comparison between the proposed approach (YOLOv4 and *libdmtx*) and the existing *libdmtx* method

Methodology	YOLOv4 + libdmtx	*libdmtx*
Number of decoded Data Matrix	458 (1970)	458
time(s)	1984.6	22252.2

The proposed methodology yields the same number of decoded landmarks 11x faster than the existing standalone methodology, within 1970 detections

5.4 Decoding Results

Indeed, the best model to deploy is the YOLOv4 according to the trade-off between processing time and accuracy required by the application. In this section, we show a comparison between the standalone methodology of *libdmtx* and our hybrid proposal — YOLOv4 (detection) plus *libdmtx* (decoding). In Table 7, we show the decoding results throughout the test set of the two approaches producing the same amount of decoded markers — 458, which represents the maximum decoded landmarks provided by the *libdmtx* standalone approach.

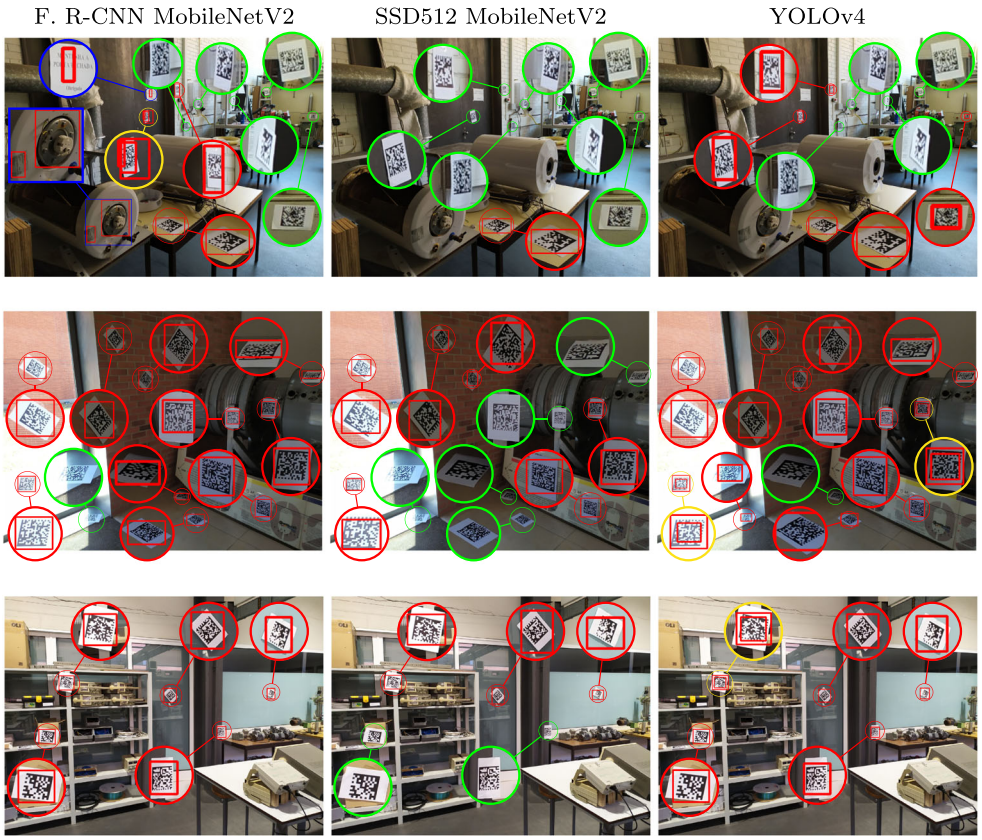

Fig. 8 Another set of visual results from different test set images, where the colors have the same meaning as before, i.e. red is a true positive, green corresponds to a false negative, blue pertains to the false negatives, and yellow zooms ineffective NMS results. In the first row there is an example from the "workshop" scenario where long distance detection is tested; the second row is an image from the hall environment, with several landmarks in different planes; finally, the third row of images are from the classroom where the targets are in shelves and walls

The confidence and NMS thresholds of the deep model also influence the effectiveness/speed trade-off of the full pipeline. In other words, if we set a higher threshold for the confidence of the model, the model would return much less possible markers, and consecutively, the decoding stage would be less requested, which turns the overall pipeline faster; similarly, in case of a lower NMS threshold, the number of inputs of the decoding stage would be less. It is worth mentioning that the impact of the NMS threshold is less than the one provided by the confidence limit since the unique NMS's objective is to suppress overlapped bounding boxes. Therefore, we also studied the influence of these factors by applying a grid search algorithm. For instance, the results obtained for the same number of decoding Data Matrix, presented in the Table above, correspond to a confidence threshold of 0.003 and a NMS limit of 0.6. This represents another advantage of DNNs on the detection stage of the pipeline as the user may set different thresholds according to the accuracy/processing time expectation: in case of having more decoded markers than necessary (e.g. in our localization problem, two decoded Data Matrix in each time step ensure a fully operational system), the user can increase the detections confidence and decrease the NMS limit in order to take advantage of a faster pipeline. In fact, the settings that allow more decoded targets, 479 decoded markers within 13065 detections in 13105.4s, correspond to the minimum confidence and the maximum NMS threshold used in the grid search experiment, i.e. 0.001 and 0.95 for the confidence and NMS thresholds, respectively.

In a nutshell, the inclusion of YOLOv4 in the detection stage of the overall pipeline provides faster and, in most of the settings (relying on the thresholds discussed above), better results. Finally, both number of decoded Data Matrix and time results presented so far are merely comparative, since a fraction of the Data Matrix labels placed in each scenario are arranged in such a way that it is impossible to decode them, as can be seen from the examples depicted in Fig. 9. In this way, both time and number of decoded markers were equally affected as there was time spent attempting to decode Data Matrix proposed by YOLOv4 that were undecodable. This means that the results presented here can be considered almost a lower bound when comparing to the real system because in this case (real application) the markers would be placed in such a way that are more likely to be decoded, so there would not be much time spent in undecodable landmarks.

Finally, as the developed system needs at least two decoded Data Matrix labels to compute the robot positioning [8] (more than two would allow an enhanced localization), this number can be used as an example to exploit a fairer (and faster) comparison between the two decoding methods (the classical and the proposed in this document). The average processing speed results of the proposed solution are presented in Fig. 10, taking into account that the classical algorithm produces the same results at 0.34fps.

This result shows that, in average, the proposed method provides decoded labels at 13.8fps when the two best scorer predictions were decoded. This has occurred in 43% of the test set frames. Moreover, when just one prediction of the top-confidence outputs is non-decodable, the pipeline yields results at 3.9fps (18% of the test set images). Finally, when there are two undecoded targets, it outputs the two decoded Data Matrix targets at 3.5fps (6% of the evaluated test set). The remaining situations are unrepresentative (i.e. scant) in the test set, considering that 27% of the test set frames did not produce 2 decoded Data Matrix labels. Hence, the most representative situations occur when the two labels are decoded through the first four predictions (0 to 2 undecoded predictions) — 67% of the test set. In the worst case scenario, the DL-based solution yields results 10x faster than the classical algorithm in average, but can reach up to 40x speed improvement.

5.5 Summary

In this section, we started by presenting the full dataset comprising heterogeneous subsets for the respective training, validation and test procedures. This allows us to ensure fair results and comparisons throughout the remaining parts of this work. Afterwards, we reported both qualitative and

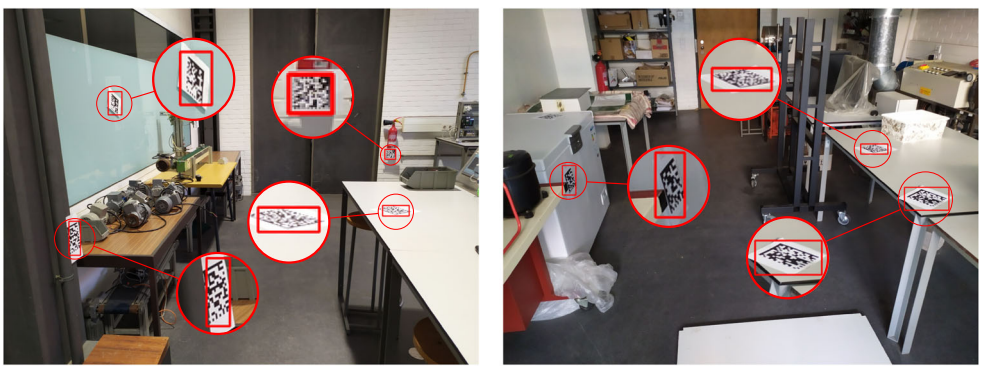

Fig. 9 Two examples where the markers are correctly detected, but their position in the environment will hardly allow them to be decoded with *libdmtx*

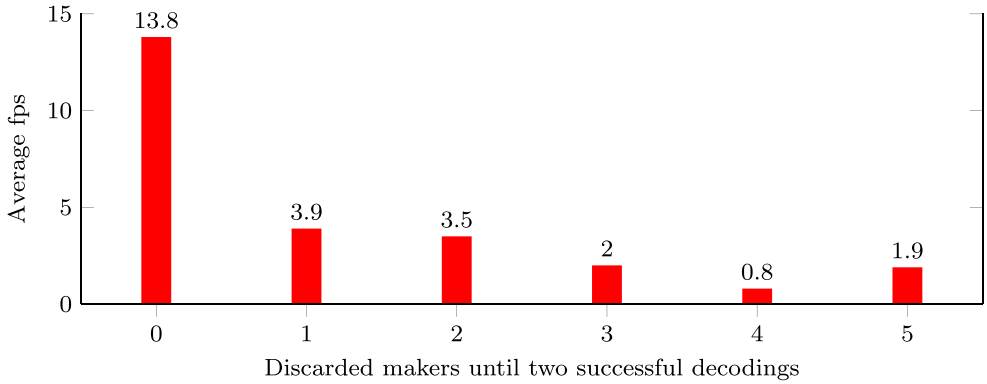

Fig. 10 Number of undecoded detected Data Matrix markers until two of them are successfully decoded, and the respective average processing rate for each case

numerical results for the detection of the landmarks in the test set. In doing so, one can infer that the most promising Data Matrix detector is the YOLOv4 according to the accuracy-latency trade-off (prime priority for this problem). Finally, considering YOLOv4 as the DNN in the pipeline described above, we propose a final comparison with the standalone *libdmtx* approach. Therewith, we found that our proposed method, YOLOv4 combined with *libdmtx*, is faster (and possibly better) than the standalone *libdmtx* when selecting the appropriate hyperparameters as shown in Section 5.4.

6 Conclusion

This work describes, assesses and compares several DNNs when performing Data Matrix detection task. The model that had the best performance in this work is the one whose average precision, recall and processing speed form the best combination. Therefore, YOLOv4 was considered the best network to detect this type of landmarks. However, this DL-based model only represents the detection part of the entire decoding pipeline. Thus, the paper evaluates/compares the decoding system (with YOLOv4 followed by the classical decoder) with the original full-frame decoder from *libdmtx*. This comparison proved that the proposed method outperforms by far the classical algorithm in terms of processing time.

Nevertheless, during the decoding system evaluation, bottlenecks were found in the system, such as: trying to decode false positives and non-decodable Data Matrix markers. In order to deploy a very robust robotic self-localization system, in the future, four situations should be studied to suppress the bottlenecks mentioned, or to improve the overall self-localization system:

- During YOLOv4 training, the objectness confidence yielded by the model become the probability of the prediction being decodable. This custom YOLOv4 would be more confident if the object is decodable, and as the decoder acts from the most confident to the least confident locations, it would decode the two best scorer predictions more times than it does right now.
- Alternatively, the conception of a decoding DL-based network. The bottleneck of our approach is the decoder stage, thus a DL network that locates and decodes Data Matrix in a single fashion, would decrease the latency of the overall system by discarding non-decodable markers and returning only the results of decodable labels.
- The detection model can be modified to output a warped bounding box, instead of a regular one. This output would be used downstream to perform a homographic transformation of the marker image, producing a better input to the decoding stage. There are several methods that can provide such an output, such as the ExtremeNet [47].
- The usage of a tracking system. This solution allows to only run the pipeline studied in this document if one of the labels stops being tracked. Therefore, the overall latency of the system would decrease.

Finally, as the overall contribution, this document proposes a pipeline for Data Matrix decoding based on the YOLOv4 detector, after the study of several different DL models to perform the detection task. This method is faster and potentially better (according to the confidence and NMS thresholds), but still has bottlenecks resulting from the decoder that can be improved/suppressed with the replacement of the entire pipeline by a single DL-network that locates and decodes Data Matrix, or more simply, with the usage of a tracking system allied to a new loss function that would value more the decodable objects, or even with the deployment of a detection network that yields also the homographic transformation of each bounding box.

Author Contributions

- Tiago Almeida: Coding and writing
- Vitor Santos: Writing and review
- Oscar Martinez Mozos: Review
- Bernardo Lourenço: Test set acquisition and review

Funding Open access funding provided by Örebro University. This work was partially supported by Project SeaAI-FA_02_2017_011, Project PRODUTECH II SIF- POCI-01-0247-FEDER-024541, by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation, and by the Spanish Ministerio de Ciencia, Innovación y Universidades under project RobWell (RTI2018-095599-A-C22).

Declarations

Consent to Participate The authors consent to participate in this work.

Competing Interests The authors declare that they have no conflict of interest.

Availability of Data and Material Code at github.com/tmralmeida/data-matrix-detection-benchmark

References

1. Li, D., Ouyang, B., Wu, D., Wang, Y.: Artificial intelligence empowered multi-AGVs in manufacturing systems. arXiv:1909.03373 (2019)
2. Shi, Q., Zhang, Y.-L., Li, L., Yang, X., Li, M., Zhou, J.: Safe: Scalable automatic feature engineering framework for industrial tasks (2020)
3. Oyekanlu, E.A., Smith, A.C., Thomas, W.P., Mulroy, G., Hitesh, D., Ramsey, M., Kuhn, D.J., Mcghinnis, J.D., Buonavita, S.C., Looper, N.A., Ng, M., Ng'oma, A., Liu, W., Mcbride, P.G., Shultz, M.G., Cerasi, C., Sun, D.: A review of recent advances in automated guided vehicle technologies: Integration challenges and research areas for 5g-based smart manufacturing applications. IEEE Access **8**, 202312–202353 (2020)
4. Bhosekar, A., Isik, T., Eksioglu, S., Gilstrap, K., Allen, R.: Simulation-optimization of automated material handling systems in a healthcare facility. arXiv: Optimization and Control (2020)
5. Cui, W., Wu, C., Zhang, Y., Li, B., Fu, W.: Indoor robot localization based on multidimensional scaling. Int. J. Distrib. Sens. Netw. **11**(8), 719658 (2015)
6. Sabattini, L., Digani, V., Secchi, C., Cotena, G., Ronzoni, D., Foppoli, M., Oleari, F.: Technological roadmap to boost the introduction of agvs in industrial applications. In: 2013 IEEE 9th International Conference on Intelligent Computer Communication and Processing (ICCP), pp. 203–208 (2013)
7. Fellan, A., Schellenberger, C., Zimmermann, M., Schotten, H.D.: Enabling communication technologies for automated unmanned vehicles in industry 4.0. In: 2018 International Conference on Information and Communication Technology Convergence (ICTC), pp. 171–176 (2018)
8. Bergamin, M.: Indoor localization using visual information and passive landmarks. Master Thesis, Universities of Padova and Aveiro. http://tesi.cab.unipd.it/50395/1/bergamin_marco.pdf (2015)
9. Betke, M., Gurvits, L.: Mobile robot localization using landmarks. IEEE Trans. Robot. Autom. **13**(2), 251–263 (1997)
10. Okuyama, K., Kawasaki, T., Kroumov, V.: Localization and position correction for mobile robot using artificial visual landmarks. In: The 2011 International Conference on Advanced Mechatronic Systems, pp. 414–418 (2011)
11. Zhang, X., Zhu, S., Wang, Z., Li, Y.: Hybrid visual natural landmark-based localization for indoor mobile robots. Int. J. Adv. Robot. Syst. **15**(6) (2018)
12. Almeida, T., Santos, V., Lourenço, B., Fonseca, P.: Detection of data matrix encoded landmarks in unstructured environments using deep learning. In: 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 74–80 (2020)
13. Babinec, A., Jurišica, L., Hubinský, P., Duchoň, F.: Visual localization of mobile robot using artificial markers. Procedia Eng. **96**, 1–9 (2014). Modelling of Mechanical and Mechatronic Systems
14. Mutka, A., Miklic, D., Draganjac, I., Bogdan, S.: A low cost vision based localization system using fiducial markers. IFAC Proc. Vol. **41**(2), 9528–9533 (2008). 17th IFAC World Congress
15. Romero-Ramirez, F.J., Muñoz-Salinas, R., Medina-Carnicer, R.: Speeded up detection of squared fiducial markers. Image Vis. Comput. **76**, 38–47 (2018)
16. Mantha, B., Garcia de Soto, B.: Designing a reliable fiducial marker network for autonomous indoor robot navigation. In: Al-Hussein, M. (ed.) Proceedings of the 36th International Symposium on Automation and Robotics in Construction (ISARC), pp. 74–81. International Association for Automation and Robotics in Construction (IAARC), Banff (2019)
17. Annusewicz, A., Zwierzchowski, J.: Marker detection algorithm for the navigation of a mobile robot. In: 2020 27th International Conference on Mixed Design of Integrated Circuits and System (MIXDES), pp. 223–226 (2020)
18. Zhong, X., Zhou, Y., Liu, H.: Design and recognition of artificial landmarks for reliable indoor self-localization of mobile robots. Int. J. Adv. Robot. Syst. **14**(1), 1729881417693489 (2017)
19. Karrach, L., Pivarčiová, E.: The analyse of the various methods for location of data matrix codes in images. In: 2018 ELEKTRO, pp. 1–6 (2018)
20. Dai, Y., Liu, L., Song, W., Du, C., Zhao, X.: The realization of identification method for datamatrix code. In: 2017 International Conference on Progress in Informatics and Computing (PIC), pp. 410–414 (2017)
21. Hansen, D.K., Nasrollahi, K., Rasmusen, C.B., Moeslund, T.B.: Real-time barcode detection and classification using deep learning. In: Proceedings of the 9th International Joint Conference on Computational Intelligence - Volume 1: IJCCI, pp. 321–327. SciTePress (2017)
22. Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., Qu, R.: A survey of deep learning-based object detection. IEEE Access **7**, 128837–128868 (2019)
23. Zhang, H., Shi, G., Liu, L., Zhao, M., Liang, Z.: Detection and identification method of medical label barcode based on deep learning. In: 2018 Eighth International Conference on Image Processing Theory, Tools and Applications (IPTA), pp. 1–6 (2018)
24. Yang, Q., Golwala, G., Sundaram, S., Lee, P., Allebach, J.: Barcode detection and decoding in on-line fashion images. Electron. Imaging **2019**(8), 413–1–413–7 (2019)

25. Zharkov, A., Zagaynov, I.: Universal barcode detector via semantic segmentation. CoRR arXiv:1906.06281 (2019)
26. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. CoRR arXiv:1506.01497 (2015)
27. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C.-Y., Berg, A.C.: SSD: single shot multibox detector. CoRR arXiv:1512.02325 (2015)
28. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: Unified, real-time object detection. CoRR arXiv:1506.02640 (2015)
29. Bochkovskiy, A., Wang, C.-Y., Liao, H.: Yolov4: Optimal speed and accuracy of object detection. arXiv:2004.10934 (2020)
30. Benali Amjoud, A., Amrouch, M.: Convolutional neural networks backbones for object detection. In: El Moataz, A., Mammass, D., Mansouri, A., Nouboud, F. (eds.) Image and Signal Processing, pp. 282–289. Springer International Publishing, Cham (2020)
31. Lin, T.-Y., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. CoRR arXiv:1612.03144 (2016)
32. Cao, G., Xie, X., Yang, W., Liao, Q., Shi, G., Wu, J.: Feature-fused SSD: fast detection for small objects. CoRR arXiv:1709.05054 (2017)
33. Huang, Z., Wang, J.: DC-SPP-YOLO: dense connection and spatial pyramid pooling based YOLO for object detection. CoRR arXiv:1903.08589 (2019)
34. Lin, T.-Y., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. CoRR arXiv:1708.02002 (2017)
35. Chen, Y., Yang, T., Zhang, X., Meng, G., Xiao, X., Sun, J.: Detnas: Backbone search for object detection. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32, pp. 6642–6652. Curran Associates, Inc. (2019)
36. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 (2014)
37. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR arXiv:1512.03385 (2015)
38. Huang, G., Liu, Z., Weinberger, K.Q.: Densely connected convolutional networks. CoRR arXiv:1608.06993 (2016)
39. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR arXiv:1704.04861 (2017)
40. Iandola, F.N., Moskewicz, M.W., Ashraf, K., Han, S., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. CoRR arXiv:1602.07360 (2016)
41. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. CoRR arXiv:1707.01083 (2017)
42. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv:1804.02767 (2018)
43. Girshick, R.B.: Fast R-CNN. CoRR arXiv:1504.08083 (2015)
44. Wang, C.-Y., Liao, H.-Y.M., Yeh, I.-H., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W.: Cspnet: A new backbone that can enhance learning capability of cnn. arXiv:1911.11929 (2019)
45. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. CoRR arXiv:1803.01534 (2018)
46. Tan, M., Pang, R., Le, Q.: Efficientdet: Scalable and efficient object detection. arXiv:1911.09070 (2019)
47. Zhou, X., Zhuo, J., Krähenbühl, P.: Bottom-up object detection by grouping extreme and center points. CoRR arXiv:1901.08043 (2019)

Tiago Almeida is a PhD student in Computer Science at Örebro University in the Wallenberg AI, Autonomous Systems and Software Program in Sweden. Tiago's research interests are real-world problems involving Artificial Intelligence (mainly Machine Learning) and Human-Robot Interaction (HRI). Therefore, his work has included Deep Learning methods for Computer Vision problems (road space segmentation and object detection) and Inverse Reinforcement Learning for behavioral patterns recognition. Tiago received his Master's degree in Mechanical Engineering at the University of Aveiro, Portugal, with a thesis addressing the problem of detecting the road free space onboard an autonomous car. This work resulted in one Conference paper (ROBOT 2019) and one Journal paper (RAS 2020). Thereafter, Tiago was a research fellow in the Produtech II SIF 24541 project, whose main goal was to develop a mobile platform capable of computing its localization through the detection of artificial landmarks. The outcomes of this project were one Conference paper (ICARSC 2020) and one Journal paper (JINT 2021).

Vitor Santos obtained a 5-year degree in Electronics Engineering and Telecommunications in 1989, at the University of Aveiro, Portugal, where he later obtained a PhD in Electrical Engineering in 1995, and the Habilitation in Mechanical Engineering in 2018. He was awarded fellowships for research in mobile robotics during 1990-1994 at the Joint Research Center, Italy. He is currently Associate Professor at the University of Aveiro and has carried out research activity on mobile robotics, autonomous driving, advanced perception and humanoid robotics. He founded the ATLAS project for mobile robot competition that achieved 6 first prizes in the annual Autonomous Driving competition and has coordinated the development of ATLASCAR, the first real car with autonomous navigation capabilities in Portugal. He is one of the founders of the Portuguese Robotics Open in 2001 and co-founder of the Portuguese Society of Robotics in 2006.

Oscar Martinez Mozos is currently Associate Professor at Örebro University and faculty member in the Wallenberg AI, Autonomous Systems and Software Program in Sweden. From 2016 to 2019 he was a senior researcher at the Technical University of Cartagena, Spain. From 2013 to 2015 he was a senior lecturer at the University of Lincoln, UK. From 2010 to 2012 he was a postdoctoral fellow of the Japan Society for the Promotion of Science (JSPS) at Kyushu University, Japan, where he keeps a position as a collaborative researcher. From 2009 to 2010 he was a postdoctoral fellow at the University of Zaragoza, Spain. He got his MSc (2005) and PhD (2008) from the University of Freiburg, Germany. He applies solutions based on artificial intelligence to problems in different areas including robotics, autonomous systems, health, and quality of life technologies.

Bernardo Lourenço graduated with an MSc. in Mechanical Engineering in 2018 at the University of Aveiro. After graduation, and during the development of this research work, he was a research fellow in the fields of Computer Vision and Robotic Systems at the Department of Mechanical Engineering at the University of Aveiro. He has participated in multiple conferences, such as the 2019 and 2020 ICARSC Conference and the ROBOT2019 Conference. Currently, he works at everis, an NTTData company, in the field of data engineering. His research interests are deep learning, computer vision, programming, and robotic systems.