IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 19, 2020, accepted April 16, 2020, date of publication April 20, 2020, date of current version May 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2989052

Visual Trunk Detection Using Transfer Learning and a Deep Learning-Based Coprocessor

ANDRÉ SILVA AGUIAR[1,2], FILIPE NEVES DOS SANTOS[1], ARMANDO JORGE MIRANDA DE SOUSA[1,3], PAULO MOURA OLIVEIRA[2], AND LUIS CARLOS SANTOS[1,2]

[1]INESC Technology and Science (INESC TEC), 4200-465 Porto, Portugal
[2]Department of Engineering, University of Trás-os-Montes e Alto Douro (UTAD), 5000-801 Vila Real, Portugal
[3]Faculty of Engineering, University of Porto (FEUP), 4200-465 Porto, Portugal

Corresponding author: André Silva Aguiar (andre.s.aguiar@inesctec.pt)

This work was supported in part by the National Funds through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT), within project under Grant UIDB/50014/2020, in part by the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation—COMPETE 2020 under the PORTUGAL 2020 Partnership Agreement—and through the Portuguese National Innovation Agency (ANI) as a part of project "ROMOVI: POCI-01-0247-FEDER-017945."

ABSTRACT Agricultural robotics is nowadays a complex, challenging, and exciting research topic. Some agricultural environments present harsh conditions to robotics operability. In the case of steep slope vineyards, there are several challenges: terrain irregularities, characteristics of illumination, and inaccuracy/unavailability of signals emitted by the Global Navigation Satellite System (GNSS). Under these conditions, robotics navigation becomes a challenging task. To perform these tasks safely and accurately, the extraction of reliable features or landmarks from the surrounding environment is crucial. This work intends to solve this issue, performing accurate, cheap, and fast landmark extraction in steep slope vineyard context. To do so, we used a single camera and an Edge Tensor Processing Unit (TPU) provided by Google's USB Accelerator as a small, high-performance, and low power unit suitable for image classification, object detection, and semantic segmentation. The proposed approach performs object detection using Deep Learning (DL)-based Neural Network (NN) models on this device to detect vine trunks. To train the models, Transfer Learning (TL) is used on several pre-trained versions of MobileNet V1 and MobileNet V2. A benchmark between the two models and the different pre-trained versions is performed. The models are pre-trained in a built in-house dataset, that is publicly available containing 336 different images with approximately 1,600 annotated vine trunks. There are considered two vineyards, one using camera images with the conventional infrared filter and others with an infrablue filter. Results show that this configuration allows a fast vine trunk detection, with MobileNet V2 being the most accurate retrained detector, achieving an overall Average Precision of 52.98%. We briefly compare the proposed approach with the state-of-the-art Tiny YOLO-V3 running on Jetson TX2, showing the outperformance of the adopted system in this work. Additionally, it is also shown that the proposed detectors are suitable for the Localization and Mapping problems.

INDEX TERMS Deep learning, transfer learning, convolutional neural networks, tensor processing unit.

I. INTRODUCTION

The research and development of robotic solutions for the agriculture sector have been growing [1], [2]. The need for automatic machines in this area is increasing since farmers increasingly recognize its impact in agriculture [3]. Robots are now used for a variety of tasks such as planting, harvesting, environmental monitoring, supply of water and nutrients, and others [4]. In this context, developing solutions that allow robots to navigate safely in these environments is essential. To do so, localizing the robotic platform in real-time is required. In vineyards built in steep slope hills, the use of the GNSS is, in most cases, unavailable due to signal blockage and multi-reflection. Thus, several solutions redundant to GNSS have been developed. In particular, Simultaneous Localization and Mapping (SLAM) and Visual Odometry

The associate editor coordinating the review of this manuscript and approving it for publication was Gianluigi Ciocca.

(VO) approaches are in many cases adopted [5]–[8]. In these cases, to give the robot knowledge about the vineyard patterns is a smart solution. The vine trunks can be used as landmarks for the SLAM, and to build a vineyard map. There are solutions to perform these tasks using range sensors [5], [9] or camera systems [10], [11], based on traditional methods such as Kalman Filters (KF), image processing, and others. However, to the best of our knowledge, the use of DL [12] to detect vine trunks is still nonexistent in the literature. The use of this approach is interesting since it provides artificial intelligence to the robot while being, in many cases, an accurate solution. Convolutional Neural Networks (CNN) shown the greatest performance in several contests in machine learning and pattern recognition [13], [14]. This procedure, however, assumes that the training and test data must be in the same feature space, and have the same distribution [15]. However, in some real-world scenarios, data collection can be challenging, as well as time expensive. So, learners that can be trained with data easily collected from different domains are, in some cases, required [16]. In other words, the learning procedure is performed, transferring knowledge from a given task that was already learned. This methodology is called TL [17].

While computing SLAM using a Central Processing Unit (CPU) of a given machine, it is wiser to minimize the CPU resources consumption. For example, the landmark detection task can be executed in a second processing unit. In the case of trunk detection using CNNs with DL or TL, several devices can be used, such as Graphical Processing Units (GPU), TPUs, Vision Processing Units (VPU), and others [18]. This configuration allocates a dedicated device for object detection, maximizing the frame rate of the robot navigation. To do so, the CNNs can be trained and executed using several frameworks. One of the most popular is Tensorflow [19]. This tool allows to create, train, and execute models that can be transferred to heterogeneous devices. Also, this framework supports deployment in embedded and mobile devices with Tensorflow Mobile and Tensorflow Lite. It is possible to convert Tensorflow models to the Lite or Mobile versions using the framework. There are also CNNs optimized for mobile and embedded systems such as the MobileNets [20], and SqueezeNet [21].

This work aims to perform DL-based object detection to:

- Detect high-level visual features in vineyards (vine trunks), in a low-power and high-performance manner;
- Present a reliable visual landmark input to SLAM systems in the vineyard context.

To do so, this work proposes an accurate, cheap, and fast trunk detection in steep slope vineyard context. To achieve these specifications, a single camera and an Edge TPU are used. The Edge TPU is provided by Google's USB Accelerator [22]. It is a small, high-performance, and low power unit suitable for image classification, object detection, and semantic segmentation. This device provides high-performance ML inferencing for TensorFlow Lite models. The proposed approach performs object detection on this device to detect vineyard trunks, using TL. This is done using a few pre-trained versions of MobileNet V1 and MobileNet V2. These are CNNs developed for mobile and embedded vision applications. A benchmark between the two models and the different pre-trained versions is performed, both in terms of processing time and detection precision. The models are pre-trained in a built in-house dataset. Results show that this configuration allows accurate and fast trunk detection, without spending the CPU resources. When compared to Tiny YOLO-V3 [23], the architecture proposed in this work outperforms it both in terms of inference accuracy and runtime performance.

The rest of the paper is described as follows. In the next section, the related work is reviewed. Section III contains the materials used in this work. In particular, the CNN models used and their architecture, and the description of the Edge TPU used. Section IV contains the approach adopted in this work, such as the data collection method, and the training procedure adopted. Section V presents the proposed system results using the built in-house dataset, and the respective analysis and discussion. Finally, the work is summarized in Section VI.

II. RELATED WORKS

At the best of our knowledge, DL has not yet been applied to trunk detection. Even so, image classification and object detection based on DL techniques are widely present in the agriculture sector. Intensive and time expensive tasks are being replaced by automatic machines, endowed with artificial intelligence. These machines are performing operations in the agriculture context such as plant disease detection, weed identification, seed identification, fruit detection and counting, obstacle detection, and others [24]–[26].

To detect tomato plant diseases and pests, Fuentes *et al.* [27] reported a performance comparison between several families of detectors combining them with different CNN models. This work focuses on identifying the infection status, the symptom location, patterns of the leaf, type of fungus, and color and shape of the leaf. The results are generated and compared with and without data augmentation. Similarly, to detect and identify apple leaf diseases Liu *et al.* [28] created a novel CNN architecture based on AlexNet. The network was trained using 13,689 images and is used to detect four common apple diseases. The overall accuracy of the network is 97.62%, which consists of an improvement of 10.83% compared with AlexNet. Barré *et al.* [29] propose a CNN-based plant identification system called LeafNet. This work aims to have a system that learns features from leaf images capable of identifying plant species using them. The method was tested in several datasets such as LeafSnap, Flavia, and Foliage, outperforming hand-crafted-based systems. Potena *et al.* [30] used two CNNs to perform crop and weed identification. The first, a lightweight CNN, is used to segment images in order to extract 3D pixel projections of points that belong to green vegetation. The second, a deeper CNN, is used to classify these pixels to classify the crop and weed.

This configuration allows real-time crop and weed detection on top of an unmanned ground vehicle (UGV). Also, for weed detection, in [31] an unsupervised data labeling approach is proposed. This work uses unmanned aerial vehicle (UAV) images to identify inter-row weeds that constitute the training dataset for a CNN. The network is used to detect the crop and weeds in the images. The performance obtained is comparable to the traditional approaches with supervised data labeling. Ashqar *et al.* [32] use a CNN to classify plant seedlings. In this work, a dataset with approximately 5,000 images with 12 plant species is used. This approach achieved an accuracy of 99.48%. To detect different apple growth stages in orchards, Tian *et al.* created an improved version of YOLO-V3 [23]. Their architecture is prepared for variations in illumination, complex backgrounds, and overlapping apples. The dataset uses augmented images to increase the amount of training data. Results show that accurate and real-time performance is achieved using high resolution images. In this context, many works use CNNs to count fruit. For example, in [33], two CNNs are used to count both apples and oranges. The first extracts the candidate regions of the image, and the second implements a counting algorithm for each region. The performance of the approach is analyzed using both images recorded during the day and the night. Results show that this pipeline presents well behavior using a limited dataset size. Similarly, Deep Count [34] proposes a fruit counting approach. In this work, a modified version of Inception-Resnet [35] is used. The network is trained on synthetic data and evaluated on real data. Fruits are counted even under shadow, occluded by branches, and foliage, or if there is overlap between fruits. The method presents and accuracy of 91% on real data, and 93% on synthetic data. CNNs can also be applied to image segmentation, and this can be used in agriculture. For example, in [36], roots in soil are segmented using U-Net [37]. In this case, the labeling procedure is time expensive. All the images pixels considered to belong to a root, have to be manually and individually annotated. Each image annotation takes, on average, 30 minutes. So, this work uses 50 training images and is evaluated in 867 images. Results show that the system produces segmentations with higher quality than the manual annotations. In [38], DL is used to perform obstacle detection in agricultural fields. The obstacle is standardized and it is detected with a precision of 99.9% in row crops, and 90.8% in grass mowing.

TL applications are far more rare in the agricultural sector. Despite this, few works in this area are reported. For instance, in [39], a CNN is pre-trained with a large and general dataset, with approximately 1000 classes, to initialize the weights. Then, the network is retrained in order to detect 9 diseases on tomatoes. A dataset with 14,828 images of tomato leaves is used. Similarly, a TL technique is also used by Mohanty *et al.* [40] to detect plant diseases. Here, a public dataset with 54,306 images is used to retrain two CNNs, in order to identify 26 diseases. To detect plant species, Ghazi *el al.* [41] use TL on pretrained popular CNN architectures. To increase the training dataset size and reduce the chance of overfitting, the original data was augmented with operations such as rotation, translation, scaling, and reflection. The system presents an accuracy of 80%. In [42], DL and TL are used to extract land information from UAV imagery. Firstly, a CNN is used to exclude linear features, such as roads and bridges. Secondly, the feature extraction procedure is used to extract the desired information using TL. TL can also be used to segment images. For example, in [43], semantic segmentation is applied using a TL technique to identify different crop types. In this work, three datasets are used to compare the classification performance using different retraining efforts. Training data is fully and partially labeled at the pixel level. Results show that TL, even with partially labeled data, presents high accuracy. Douarre *el at.* [44], use TL to segment soil roots in X-Ray tomography data. To retrain the network, simulated training data is used. Results show that soil and root are well segmented, even with shallow contrast between them.

Despite DL being widely used in agriculture, as described in this section, vine trunk detection using CNNs is still a gap of the state-of-the-art. This work proposes to fill this gap, with a low-power and high-performance DL-based trunk detection, suitable for real-time applications in robotics.

III. MATERIALS

In order to achieve high runtime performance in robotics navigation, edge inference was chosen to perform visual detection of vine trunks. Edge inference is the use of a particular Application Specific Integrated Circuit (ASIC) accelerator to deploy a Neural Network (NN) based on a given training dataset. Using this approach, the extraction of landmarks for a SLAM problem is computed using a dedicated device and can achieve high levels of performance with low power costs. In this work, Google's Edge TPU was used. A TPU is a coprocessor designed by Google that is usually connected to a host CPU. In the ideal case, the TPU device implements all the inference operations. Otherwise, the host CPU can perform some of them, but this will slow down the process. Using this configuration, to perform the detection, MobileNets [20] with Single Shot MultiBox Detector (SSD) [45] were used.

A. GOOGLE EDGE TPU

Google's Coral USB Accelerator (Fig. 1), provides an Edge TPU machine learning accelerator coprocessor. It is connected via USB to a host computer, allowing high-speed inference. This device is compatible with Tensorflow Lite, a lightweight version of TensorFlow designed for mobile and embedded devices, and can perform image classification, object detection, and semantic segmentation. To perform such tasks, the Edge TPU uses 8-bit quantized models. So, when training a 32-bit float model from scratch, it has to be either quantized using either *quantization aware training* or *post training quantization*. The first approach simulates the effect of 8-bit values during the training process using quantization nodes in the NN graph. The second does not modify the NN structure and is applied after training. However, it is

FIGURE 1. Google Coral USB Accelerator [22].

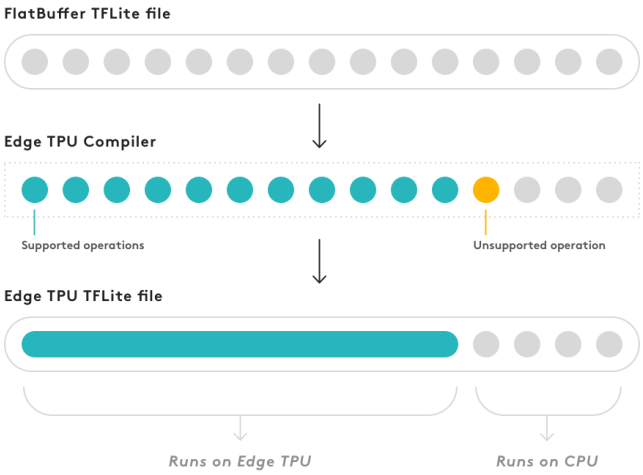


FIGURE 2. Edge TPU model compilation scheme [46].

less accurate than the first method. Alternatively, pre-trained models that are already quantized can be used if compatible with the Edge TPU. The device supports a range of operations and is most likely compatible with models designed for mobile devices, using the SSD architecture. After training the model, the *Edge TPU compiler* is used to assign inference operations to the device and the host CPU, as represented in Fig. 2. The compiled model differs from the original Tensorflow Lite model in the first operation of the graph. CPU processes the operations from the first non-supported operation of the TPU, until the end of the graph. The inference will be as faster as higher it is the number of operations assigned to the Edge TPU. Table 1 the MobileNet V1 compilation results after retraining, on this work. The table shows the supported operations performed by the model on the Edge TPU. When an unsupported operation is found, all the following ones are deployed by the host CPU, represented as *Custom* in Tab. 1.

B. SINGLE SHOT MULTIBOX DETECTOR

To perform the vine trunk detection using the Edge TPU coprocessor, we chose the SSD [45] architecture since it is fully compatible with the device. To perform object detection, this architecture uses a feed-forward CNN producing a fixed-size collection of bounding boxes and attributing a score for each one of them. The CNN contains convolutional feature layers to the end of the truncated base network. These layers allow to detect objects at multiple scales, i.e., objects of different sizes in images with different resolutions.

TABLE 1. Output of the MobileNet V1 model compiler for Edge TPU.

Operator	Status
Logistic	Mapped to Edge TPU
Concatenation	Mapped to Edge TPU
2D Convolution	Mapped to Edge TPU
Depthwise 2D Convolution	Mapped to Edge TPU
Reshape	Mapped to Edge TPU
Custom	Uunsupported data type

C. MOBILENETS

Since in this work, a coprocessor is used to perform machine learning inference, using models suitable for mobile and embedded devices is a logical solution. Thus, MobileNets [20] were chosen. This set of models provide lightweight deep NN using depthwise separable convolutions. In other words, the model factorizes convolutions into depthwise, and 1×1 convolutions called pointwise convolutions. The first applies a single filter to the input channel, and the second applies a 1×1 convolution, combining the outputs of the first. The input of a CNN is a tensor with shape $D_f \times D_f \times M$, where D_f represents the input channel spatial width and height, and M is the input depth. After the convolution, a feature map of shape $D_f \times D_f \times N$ is obtained, where N is the output depth. In this context, MobileNets propose two hyper-parameters that allow the user to resize the model so that it meets the system specifications. There hyper-parameters are: *width multiplier* and *resolution multiplier*. The *width multiplier* α is used to thin the CNN uniformly at each layer. For a given value of $\alpha \in (0, 1]$, the number of inputs channels M becomes αM, as well as the number of output channels N becomes αN. *Width multiplier* reduces the computational cost and the number of parameters by α^2. The second hyper-parameter, *resolution multiplier* ρ, is also used to reduce the computational cost. This one is applied directly to the input image, setting its resolution. The values of $\rho \in (0, 1]$ are chosen in order to obtain typical input image resolutions. *resolution multiplier* also reduces the computational cost and the number of parameters by ρ^2.

In this work, two MobileNet versions provided by TensorFlow [19] are considered, MobileNet V1 and V2. Both models, already trained using the COCO dataset [47], were retrained to detect vine trunks. To analyze the impact of the *width multiplier* hyper-parameter, a version of MobileNet V1 that was pre-trained with a non-default value for this parameter was also retrained. Table 2 indicates the models considered. The *resolution multiplier* is set to its default value in all the experiments, so that the input resolution is 300×300.

TABLE 2. Pretrained models to perform vine trunk detection on the Edge TPU.

Model	Version	*Width Multiplier*	Input Resolution
MobileNet	1	1	300 × 300
MobileNet	1	0.75	300 × 300
MobileNet	2	1	300 × 300

In this work, other variations of the hyper-parameters were not tested since a TL procedure was adopted. This means that the NNs were already trained with specific values for the hyper-parameters, and to experiment other values with the desired impact, they have to be trained from scratch.

IV. METHODS

In this work, TL is addressed to perform DL-based object detection. TL main shortcoming is *negative transfer*, which happens when the pre-training data contributes to negative learning on the target application. To detect vine trunks, several models pre-trained with the COCO dataset [47] were retrained, and was verified that using such a vast dataset, the target detectors retrained do not suffer from *negative transfer*. In order to achieve a high-performance detector, that visually recognizes trunks in real-time, a training procedure over the CNNs was performed. To do so, a training dataset was created, using our robotic platform AgRob V16 [48], represented in Fig. 3. The dataset is publicly available at our repository (`http://vcriis01.inesctec.pt/`) with the `DS_AG_39` id. It contains camera images with both an infrared, and an infrablue filter, in two different vineyards. After collecting the data, the trunks were manually annotated on the images. Then, the training procedure was performed. The main steps of this work are represented in Fig. 4, and are detailed next.

FIGURE 3. AgRob V16 robotic platform.

A. DATA COLLECTION

To build the training dataset, two onboard cameras of our robotic platform, AgRob V16, collected data in two different vineyards, represented if Fig. 5. One of them is a Raspberry

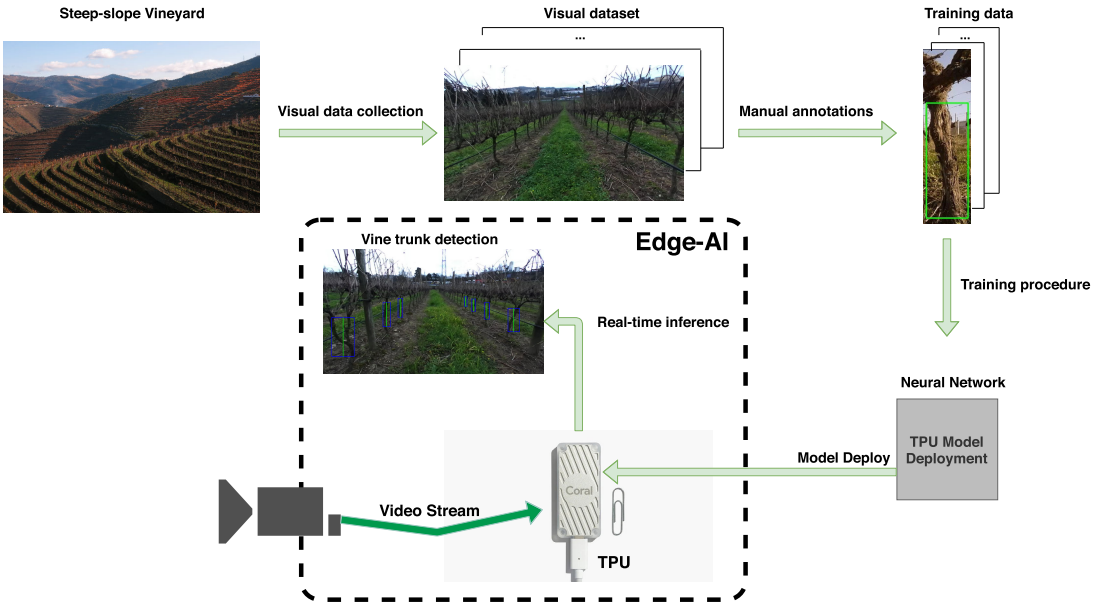


FIGURE 4. High-level design of the vine trunk detection framework.

(a)

(b)

FIGURE 5. Training data on two different vineyards, (a) and (b).

Pi camera with 640 × 480 resolution, with an infrared filter (Fig. 5(b)). The other is a Mako G-125C camera, with a resolution of 1292 × 964 resolution, and an infrablue filter (Fig. 5(a)). The dataset considers 336 different images and approximately 1,600 annotated trunks. It contains:

- Images with different resolutions.
- Two types of vine trunks, with and without foliage.
- Trunks covered by shadows.

These are challenging conditions that confer variety and robustness to the training procedure.

B. DATA ANNOTATION

Given the training dataset, vine trunks were manually annotated. For maintain consistency, on the trunks that are less than approximately 3 meters from the robot and that belong to the corridor where the robot is located, were manually annotated. Figure 6 shows an example of this procedure. The annotation procedure output is a set of bounding boxes of different sizes, for each image. These are represented in a `.xml` file with the Pascal VOC annotation syntax, containing the label class considered, and the four corners location of each bounding box. The annotations are also publicly available (`http://vcriis01.inesctec.pt/`) with the training images. This data is the input for the training procedure, described below.

FIGURE 6. Annotation example referent to the training procedure.

C. RETRAINING PROCEDURE

The training procedure aims to create an Edge TPU compatible model, capable of detecting the vine trunks in real-time. To do so, Tensorflow and Tensorflow Lite were used, as represented in Fig. 7. The first step is to serialize the *ground truth* bounding boxes, so that Tensorflow can interpret it efficiently. The data serialization is performed using the *TFRecord* data type, which stores the data as a sequence of binary strings. This constitutes the input for the CNN retraining. This step uses the configurations present in Tab. 2, providing the ability to recognize a trunk to the CNN. After that, a Tensorflow model is generated, and, in order to save the model for posterior TL or retraining, the model is exported into a *frozen graph*. Since the Edge TPU only supports TensorFlow Lite models that are fully 8-bit quantized, the *frozen graph* is then converted into such a model. Finally, in order to assign operations to the Edge TPU device, the Tensorflow Lite model is compiled using the procedure described in Sec. III.

D. EVALUATION METRICS

In order to evaluate the CNN performance detecting vine trunks, the PASCAL Visual Object Classes (VOC) Challenge [49] was used. This method is used by many DL-based works to evaluate CNN performance, offering a fair comparison between the set of works present in the literature. To do so, given an annotated *ground truth* bounding box B_g, and a detected bounding box B_d, the IoU is firstly calculated as follows

$$IoU = \frac{m(B_g \cap B_d)}{m(B_g \cup B_d)} \tag{1}$$

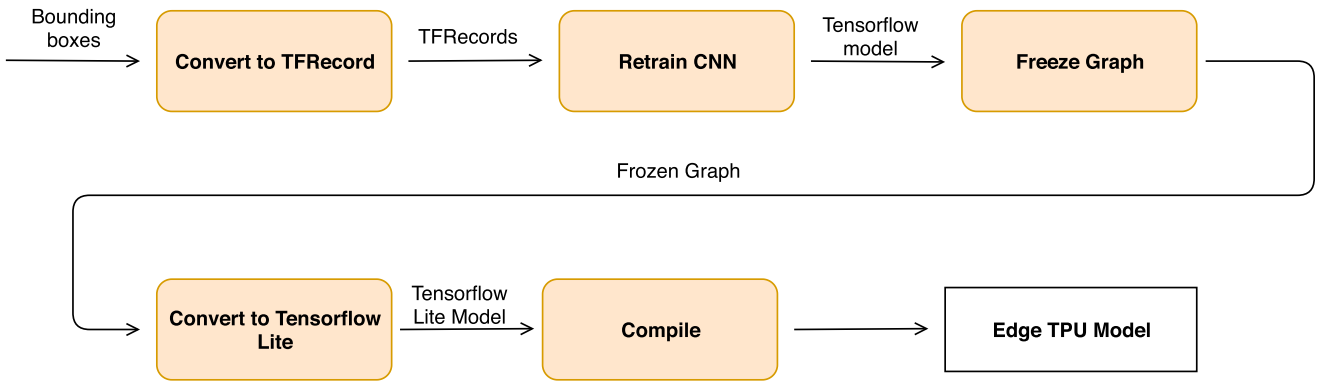


FIGURE 7. Training procedure flow.

where $m(x)$ denotes the area of x. This can be also understood, analysing Fig. 8. So, *IoU* represents the quotient between the area of overlap and the area of union between the *ground truth*, and the detection bounding boxes. Using this definition, three main concepts can be defined. For a given threshold value t, let us define:

- **True Positive** (TP): IoU $\geq t$, i.e., a correct detection.
- **False Positive** (FP): IoU $\leq t$, i.e., an incorrect detection.
- **False Negative** (FN): a *ground truth* is not detected.

It is worth noting that if more than one detection for a single *ground truth* is computed, only the one with the highest IoU is considered as TP, and all the others are FPs. This being said, with these three qualifiers it is possible to define two fundamental concepts. The first, *precision* p, is defined as the total number of **TPs** over all the detections. The second, *recall* r, is the total number of **TPs** over all the *ground truths*. Using these, is possible to plot a curve of the *recall* in function of the *precision*, $p(r)$. The evaluation considers that a suitable detector is the one that maintains the precision high for an increase in recall. With this consideration, the detector is evaluated computing the Average Precision (AP), interpolating the obtained curve, and calculating the area below the curve. Mathematically, this is expressed as follows

$$\sum_{r=0}^{1}(r_{n+1} - r_n)p_{interp}(r_{n+1}) \tag{2}$$

with

$$p_{interp}(r_{n+1}) = \max_{\tilde{r};\tilde{r}\geq r_{n+1}} p(\tilde{r}) \tag{3}$$

where $p(\tilde{r})$ is the measured precision at recall $\tilde{r}$.

V. RESULTS

In order to evaluate the trained NNs on top of Google's Edge TPU, a subset of the dataset previously described was extracted for testing. From the total of 336 images on the dataset, 45 images were used for the test procedure, with approximately 180 vine trunks. These images were randomly extracted from the dataset before training in order to be only used for validation. The performance of the three model configurations is compared using all the evaluation data, and the images of each vineyard individually. Thus, the global performance of the detector is evaluated, as well as its isolated performance in the images with both types of filters. Tiny YOLO-V3 was trained and evaluated using exactly the same training and testing data, respectively. So, a brief comparison is performed between this model running on Jetson TX2 and the proposed system on this work. Additionally, this Sec. shows an application of the proposed detectors to a Localization and Mapping system.

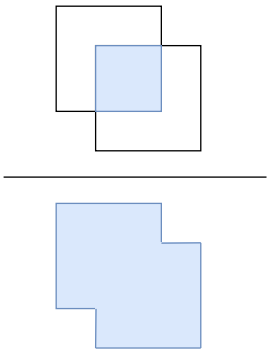


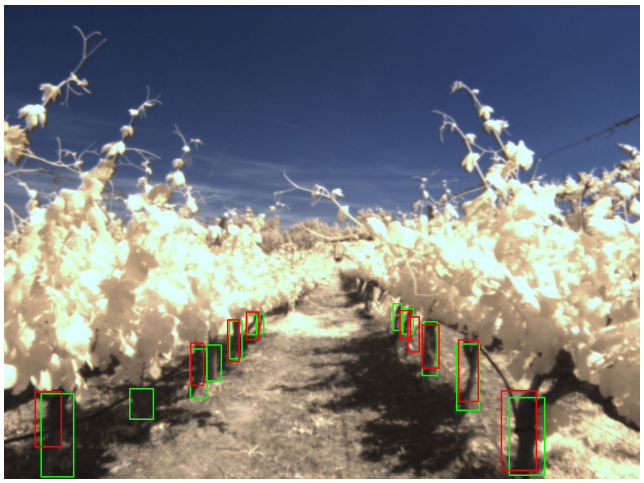
FIGURE 8. Interception over union representation.

A. DETECTION PERFORMANCE

Using the metrics described on Sec. IV, and the test set of images, the detectors were evaluated. Figure 9 shows an example of a detection, provided by MobileNet V1, with $\alpha = 1.0$, for both vineyards. The green bounding boxes represent the *ground truth*, and the red ones, the detections.

Figures [10-12] represent the *precision* $\times$ *recall* curves $p(r)$, for all the considered configurations: the three retrained models, either with the *IoU* threshold t equals to 0.5 and 0.65, under the three evaluation sets. Table 3 summarizes the AP for all the configurations.

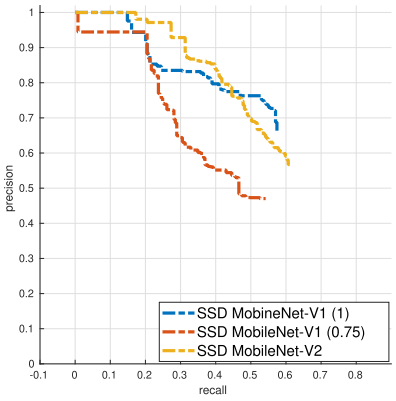
To evaluate the runtime performance of each detector, they were profiled using the *high resolution clock* from `chrono`



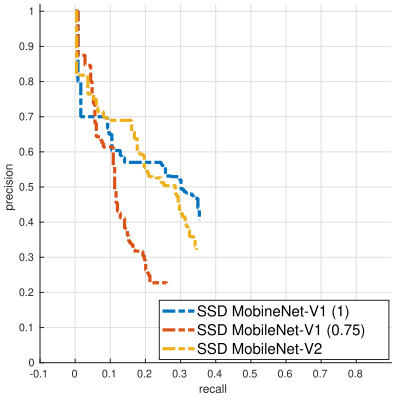
(a)

(b)

FIGURE 9. A result example for the two different vineyards, (a) and (b). The red boxes represent the detections, and the green ones the *ground truth*.



(a)



(b)

FIGURE 10. Interpolated AP p_{interp} results using all the training data and a IoU threshold of (a) 0.5 and (b) 0.65.

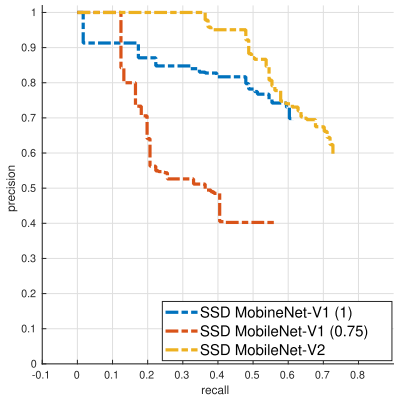
present in the *std* library. This measure consists of the time that the detector takes to return the resultant bounding boxes, since it receives the input image. Table 4 shows the obtained results for each detector.

To compare the proposed system with a state-of-the-art model, Tiny YOLO-V3 was training with the built in-house dataset and evaluated over the same 45 images using an IoU equals to 0.5. Table 5 summarizes the results obtained with this configuration, both in terms of AP and runtime performance.

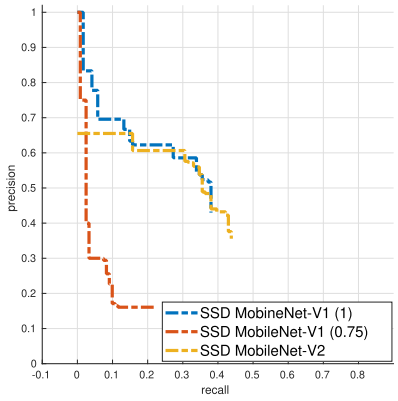
B. DISCUSSION

By analysis of Tab. 3, several conclusion can be extracted. Both for the global set of images, and the infrared filtered ones, MobileNet V2 is the best detector. For $t = 0.5$, the difference to the other detectors is significant. With $t = 0.65$, this margin tends to attenuate. On the infrablue filtered set of images, MobileNetV1 ($\alpha = 1.0$) is, not by far, the best detector. Despite this, the three detectors behave very similarly on this set of images. Comparing the performance of the detectors on the two filtered set of images, the conclusion is that the infrablue is, in general, more challenging. Except for MobileNet V1 ($\alpha = 0.75$), the other detectors behave better on the set with vineyard images using the conventional infrared filter. Obviously, increasing the IoU threshold t leads to a decrease in the average precision, as verified in all the studied cases. Globally, MobileNet V2 is the detector that presents the best performance, providing an AP of 52.98% on the global set of images, 62.95% on the infrared filtered images, and finally, 41.33% on the infrablue filtered ones, for a *width multiplier* $t = 0.5$. Comparing these results with the ones present in Tab. 5, it is visible that Tiny YOLO-V3 presents a lower AP than all the proposed configurations for the same value of t, showing an overall AP of 31.32%.

In terms of inference runtime performance, Tab. 4 shows the average inference per image for all the detectors, in milliseconds. These results were generated, taking into account all the evaluation images considered, computing the average inference time of each detector in all of them. They show that MobileNet V1 ($\alpha = 0.75$) is the fastest detector with

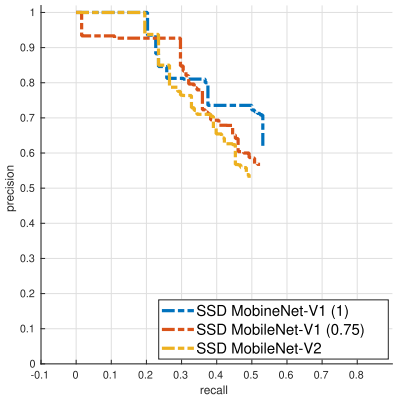


(a)

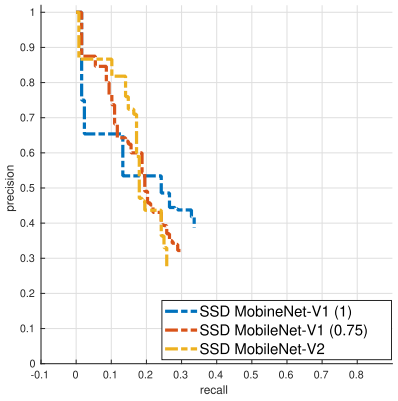


(b)

FIGURE 11. Interpolated AP p_{interp} results using the infrared training images and a IoU threshold of (a) 0.5 and (b) 0.65.



(a)



(b)

FIGURE 12. Interpolated AP p_{interp} results using the infrablue training images and a IoU threshold of (a) 0.5 and (b) 0.65.

20.5326 ms average inference per image, which corresponds to 48.7030 *frames per second*. MobileNet V1 ($\alpha = 1.0$), is slower than the previous one, but faster than MobileNet V2. The first achieves an average inference time of 21.1853 ms, which corresponds to 47.2025 *frames per second*. The second, MobileNet V2, presents 23.8238 ms of average inference time, and, consequently, 41.9748 *frames per second*. Thus, MobileNet V1 ($\alpha = 0.75$) can process 5 more images per second than MobileNet V2. Despite this, it is notorious that all the detectors being executed on top of the Edge TPU present high performance, being suitable for real-time usage. The architecture of this TPU, dedicated explicitly to processing CNNs, revealed to process DL-based object detectors with time performances that can be used in any visual SLAM system. The loop frequency of SLAM is, in most cases, not higher than 20 *frames per second*. Our detectors achieved time performances that can process more than twice this number of *frames per second*. Table 5 presents the average runtime performance of Tiny YOLO-V3 on top of Jetson TX2, resulting in an average inference time per image of 54.20 milliseconds. This result corresponds to a frame rate of 18.45 *frames per second*, which leads to the conclusion that the MobileNets run more than twice as fast over the Edge TPU than YOLO on top of the Jetson TX2.

TABLE 3. Summary of the detector performance.

Model	Overall (%)		Infrared (%)		Infrablue (%)	
	t		t		t	
	0.5	0.65	0.5	0.65	0.5	0.65
MobileNet V1 ($\alpha = 1.0$)	49.74	21.19	51.66	25.07	46.11	19.36
MobileNet V1 ($\alpha = 0.75$)	39.78	12.97	33.73	3.89	43.37	18.81
MobileNet V2 ($\alpha = 1.0$)	52.98	21.09	62.95	28.15	41.33	18.16

Globally, the main conclusions that can be taken from this analisys are:

- MobileNet V2 is the most accurate detector, from the set of three detectors analyzed.
- MobileNet V1 ($\alpha = 0.75$) is the fastest detector, but globally the least accurate one.
- Tiny YOLO-V3 is shows lower AP and lower runtime performance running on top of Jetson TX2 than the MobileNets on the Edge TPU.

These conclusions confirm the *a-priori* knowledge about the detectors and their hyper-parameters. MobileNet V2 is an improvement to the first version of the MobileNets,

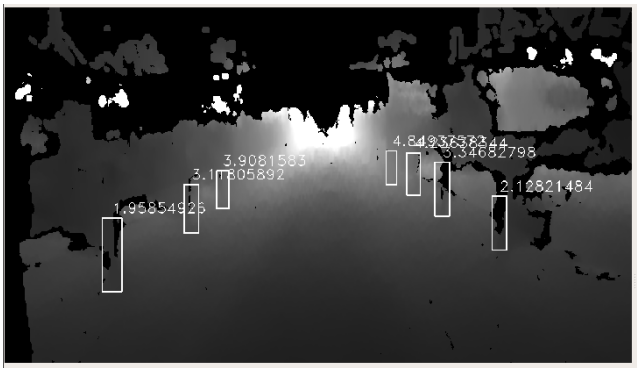
TABLE 4. Runtime performance (ms) of the different retraining configurations performed.

Model	Average inference time per image (ms)	Inference time standard deviation (ms)
MobileNet V1 ($\alpha = 1.0$)	21.1853	0.536463
MobileNet V1 ($\alpha = 0.75$)	20.5326	0.518247
MobileNet V2 ($\alpha = 1.0$)	23.8238	0.662605

TABLE 5. Tiny YOLO-V3 AP and runtime performance results for an IoU equals to 0.5.

Model	Inference average precision			Average inference time per image (ms)
	Overall (%)	Infrared (%)	Infrablue (%)	
Tiny YOLO-V3	31.32	36.40	28.25	54.20

(a)



(b)

FIGURE 13. (a) Disparity map constructed using a stereo camera system and (b) vine trunk detections projected on it with the respective depth information.

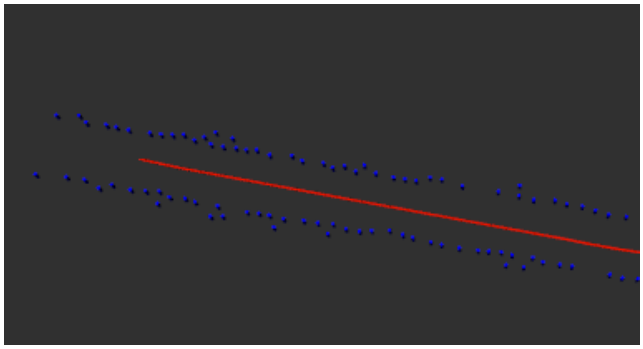


FIGURE 14. 2D vine trunk mapping using the proposed detectors.

introducing new features to the original architecture. Thus, the improvement in comparison to the first version was expected. Also, the behavior of the MobileNets with the variation of the *width multiplier* hyper-parameter, was verified. With $\alpha = 0.75$, MobileNet V1 is, in general, less accurate but faster than using a higher value for α. Additionally, Tiny YOLO-V3, a lightweight version of YOLO-V3, showed an overall lower performance than the proposed configurations on this work, even being executed in a much *high-cost* device. The optimized architecture of the Edge TPU for CNN models suitable for embedded devices leads to a much higher frame rate and a more effective inference performance.

The state-of-the-art does not currently provide any work that detects vine trunks on images using NNs. This work presents a solution to this problem using a device that is yet not popular in the literature. This solution has strong and weak points, depending on the application that uses its final results. It can be concluded that:

- The AP results obtained were not as high as many DL works present in the literature. However, in this work was used a *low-cost* and *low-power* device, that uses *lightweight* and 8-bit *quantized* models - MobileNets. This leads to least accurate inference results but, at the same time, inference is performed with low power consumption, at high *frame rate*, with much less costs than works that use, for example, powerful GPUs.

As referenced before, the desired application for the proposed detector of this work is SLAM. In steep slope vineyards, GNSS-based signals such as the Global Positioning System (GPS) are not always available. So, redundant solutions to GPS have to be developed. The solution proposed in this work can be suitable for such an application. The navigation stack where the detector will be included imposes a minimum *frame rate* of 10 *frames per second*. This condition is ensured, as demonstrated before. Additionally, the detections can be used both on mapping and localization tasks, considering, for example, a stereo camera system. Figure 13 represents the application of the proposed detector on such a system. Using both cameras it is possible to compute a disparity map, as represented in Fig. 13(a), that provides depth information. Thus, the detections on the original stereo images can be

projected on this map. The depth of each trunk is calculated computing the median of the depth of all points inside each bounding box. Figure 13(b) represents the bounding boxes projection and the depth calculation for each one. Using the computed depths, and the implicit bearing information, it is possible to calculate the position of each trunk on the world, with a given standard deviation. This information can be further used both for the mapping and localization procedures. Figure 14 shows an example of a vineyard corridor map build using this information and real data from the robotic platform. The robot trajectory is represented in red. It is worth noting that this procedure does not consider yet a procedure to remove outliers.

VI. CONCLUSION

The SLAM problem is still an intensive research topic. The primary step of any SLAM method is to extract reliable features from the surrounding environment. In the context of steep slope vineyards, the vine trunks can constitute these features. In this work, a real-time DL-based approach to compute the visual detection of vine trunks is proposed. The Edge TPU provided by Google's USB Accelerator is used, performing TL to develop reliable trunk detectors. Two versions of the MobileNets were retrained, taking into consideration their hyper-parameters. The retraining process was performed using a built in-house dataset, that is publicly available. It contains 336 different images with approximately 1,600 annotated trunks and images belonging to two different vineyards. One of them presents camera images with the conventional infrared filter, and the images with an infrablue filter. Results show that our system achieves real-time vine trunk detection. Compared with the state-of-the-art model Tiny YOLO-V3 running on Jetson TX2, the configurations proposed in this work achieve higher inference accuracy and runtime performance. MobileNet V2 revealed to be the most accurate model.

In future work, the vine dataset will be increased both in size and in variability. To do so, the objectives are: to collect data relative to different vines, to add thermal camera images to the dataset, and to augment all the images, performing operations such as rotation, translation, rescaling, etc. Also, it is planned to retrain another kind of models to perform vine trunk detection, such as VggNet, ResNet, InceptionNet, etc. Similarly, it is intended to train NNs from scratch in order to be able to vary the hyper-parameters. Finally, DL-based semantic segmentation will be considered in order to extract the exact shape of each trunk and eliminate the background pixels that are present on the bounding boxes of our detectors.

ACKNOWLEDGMENT

The opinions included in this article shall be the sole responsibility of their authors. The European Commission and the Authorities of the Programme are not responsible for the use of information contained therein.

REFERENCES

[1] F. A. Auat Cheein and R. Carelli, "Agricultural robotics: Unmanned robotic service units in agricultural tasks," *IEEE Ind. Electron. Mag.*, vol. 7, no. 3, pp. 48–58, Sep. 2013.
[2] E. A. Skvortsov, E. G. Skvortsova, I. S. Sandu, and G. A. Iovlev, "Transition of agriculture to digital, intellectual and robotics technologies," *Economy Region*, vol. 14, no. 3, pp. 1014–1028, Sep. 2018.
[3] J. Billingsley, A. Visala, and M. Dunn, *Robotics in Agriculture and Forestry*. Berlin, Germany: Springer, 2008, pp. 1065–1077, doi: 10.1007/978-3-540-30301-5_47.
[4] J. J. Roldán, J. del Cerro, D. Garzón-Ramos, P. Garcia-Aunon, M. Garzón, J. de León, and A. Barrientos, *Robots Agriculture: State Art Practical Experiences*. London, U.K.: IntechOpen, 2018. [Online]. Available: https://app.dimensions.ai/details/publication/pub.1100266943 and [Online]. Available: https://www.intechopen.com/citation-pdf-url/56199
[5] F. B. N. D. Santos, H. M. P. Sobreira, D. F. B. Campos, R. M. P. M. D. Santos, A. P. G. M. Moreira, and O. M. S. Contente, "Towards a reliable monitoring robot for mountain vineyards," in *Proc. IEEE Int. Conf. Auto. Robot Syst. Competitions*, Apr. 2015, pp. 37–43.
[6] F. Auat Cheein, G. Steiner, G. Perez Paina, and R. Carelli, "Optimized EIF-SLAM algorithm for precision agriculture mapping based on stems detection," *Comput. Electron. Agricult.*, vol. 78, no. 2, pp. 195–207, Sep. 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169911001542
[7] M. Duarte, F. N. dos Santos, A. Sousa, and R. Morais, "Agricultural wireless sensor mapping for robot localization," in *Proc. 2nnd Iberian Robot. Conf.*, L. P. Reis, A. P. Moreira, P. U. Lima, L. Montano, and V. Muñoz-Martinez, Eds. Cham, Switzerland: Springer, 2016, pp. 359–370.
[8] A. Aguiar, F. Santos, L. Santos, and A. Sousa, "Monocular visual odometry using fisheye lens cameras," in *Progress in Artificial Intelligence*, P. Moura Oliveira, P. Novais, and L. P. Reis, Eds. Cham, Switzerland: Springer, 2019, pp. 319–330.
[9] S. Marden and M. Whitty, "GPS-free localisation and navigation of an unmanned ground vehicle for yield forecasting in a vineyard," in *Proc. Int. Workshop Collocated 13th Int. Conf. Intell. Auton. Syst. (IAS)*, 2014, pp. 1–10.
[10] J. Mendes, F. N. D. Santos, N. Ferraz, P. Couto, and R. Morais, "Vine trunk detector for a reliable robot localization system," in *Proc. Int. Conf. Auto. Robot Syst. Competitions (ICARSC)*, May 2016, pp. 1–6.
[11] J. M. Mendes, F. N. dos Santos, N. A. Ferraz, P. M. do Couto, and R. M. dos Santos, "Localization based on natural features detector for steep slope vineyards," *J. Intell. Robotic Syst.*, vol. 93, nos. 3–4, pp. 433–446, Mar. 2019.
[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.
[13] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608014002135
[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Red Hook, NY, USA: Curran Associates, 2012, pp. 1097–1105 [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-conv olutional-neural-networks.pdf
[15] S. Jialin Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[16] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *J. Big Data*, vol. 3, no. 1, Dec. 2016, doi: 10.1186/s40537-016-0043-6.
[17] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of Research on Machine Learning Applications*. Hershey, PA, USA: IGI Global, Jan. 2009.
[18] Y. E. Wang, G.-Y. Wei, and D. Brooks, "Benchmarking TPU, GPU, and CPU platforms for deep learning," arXiv, Cornell Univ., Ithaca, NY, USA, Tech. Rep. 1907.10701, 2019.
[19] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and M. Kudlur, "Tensorflow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Syst. Design Implement. (OSDI)*. Savannah, GA, USA, Nov. 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi
[20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv, Cornell Univ., Ithaca, NY, USA, Tech. Rep. 1704.04861, 2017.

[21] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size," arXiv, Cornell Univ., Ithaca, NY, USA, Tech. Rep. 1602.07360, 2016.
[22] Google. (2019). *USB Accelerator*. Accessed: Jan. 3, 2020. [Online]. Available: https://coral.ai/products/accelerator
[23] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," arXiv, Cornell Univ., Ithaca, NY, USA, Tech. Rep. 1804.02767, 2018.
[24] L. Santos, F. N. Santos, P. M. Oliveira, and P. Shinde, "Deep learning applications in agriculture: A short review," in *Proc. 4th Iberian Robot. Conf.*, M. F. Silva, J. L. Lima, L. P. Reis, A. Sanfeliu, and D. Tardioli, Eds. Cham, Switzerland: Springer International Publishing, 2020, pp. 139–151.
[25] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Comput. Electron. Agricult.*, vol. 147, pp. 70–90, Apr. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169917308803
[26] A. Kamilaris and F. X. Prenafeta-Boldú, "A review of the use of convolutional neural networks in agriculture," *J. Agricult. Sci.*, vol. 156, no. 3, pp. 312–322, Apr. 2018.
[27] A. Fuentes, S. Yoon, S. Kim, and D. Park, "A robust Deep-Learning-Based detector for real-time tomato plant diseases and pests recognition," *Sensors*, vol. 17, no. 9, p. 2022, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/9/2022
[28] B. Liu, Y. Zhang, D. He, and Y. Li, "Identification of apple leaf diseases based on deep convolutional neural networks," *Symmetry*, vol. 10, no. 1, p. 11, 2018. [Online]. Available: https://www.mdpi.com/2073-8994/10/1/11
[29] P. Barré, B. C. Stöver, K. F. Müller, and V. Steinhage, "LeafNet: A computer vision system for automatic plant species identification," *Ecological Informat.*, vol. 40, pp. 50–56, Jul. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1574954116302515
[30] C. Potena, D. Nardi, and A. Pretto, "Fast and accurate crop and weed identification with summarized train sets for precision agriculture," in *Intelligent Autonomous Systems*, W. Chen, K. Hosoda, E. Menegatti, M. Shimizu, and H. Wang, Eds. Cham, Switzerland: Springer, 2017, pp. 105–121.
[31] M. Bah, A. Hafiane, and R. Canals, "Deep learning with unsupervised data labeling for weed detection in line crops in UAV images," *Remote Sens.*, vol. 10, no. 11, p. 1690, 2018. [Online]. Available: https://www.mdpi.com/2072-4292/10/11/1690
[32] B. A. M. Ashqar, B. S. Abu-Nasser, and S. S. Abu-Naser, "Plant seedlings classification using deep learning," *Int. J. Academic Inf. Syst. Res.*, vol. 3, no. 1, pp. 7–14, 2019.
[33] S. W. Chen, S. S. Shivakumar, S. Dcunha, J. Das, E. Okon, C. Qu, C. J. Taylor, and V. Kumar, "Counting apples and oranges with deep learning: A data-driven approach," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 781–788, Apr. 2017.
[34] M. Rahnemoonfar and C. Sheppard, "Deep count: Fruit counting based on deep simulated learning," *Sensors*, vol. 17, no. 4, p. 905, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/4/905
[35] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. (2017). *Inception-V4, Inception-Resnet and the Impact of Residual Connections on Learning*. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14806/14311
[36] A. G. Smith, J. Petersen, R. Selvan, and C. R. Rasmussen, "Segmentation of roots in soil with U-Net," arXiv, Cornell Univ., Ithaca, NY, USA, Tech. Rep. 1902.11050, 2019.
[37] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham, Switzerland: Springer, 2015, pp. 234–241.
[38] K. Steen, P. Christiansen, H. Karstoft, and R. Jørgensen, "Using deep learning to challenge safety standard for highly autonomous machines in agriculture," *J. Imag.*, vol. 2, no. 1, p. 6, 2016. [Online]. Available: https://www.mdpi.com/2313-433X/2/1/6
[39] M. Brahimi, K. Boukhalfa, and A. Moussaoui, "Deep learning for tomato diseases: Classification and symptoms visualization," *Appl. Artif. Intell.*, vol. 31, no. 4, pp. 299–315, Apr. 2017, doi: 10.1080/08839514.2017.1315516.
[40] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers Plant Sci.*, vol. 7, p. 1419, Sep. 2016. [Online]. Available: https://www.frontiersin.org/article/10.3389/fpls.2016.01419
[41] M. Mehdipour Ghazi, B. Yanikoglu, and E. Aptoula, "Plant identification using deep neural networks via optimization of transfer learning parameters," *Neurocomputing*, vol. 235, pp. 228–235, Apr. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231217300498
[42] H. Lu, X. Fu, C. Liu, L.-G. Li, Y.-X. He, and N.-W. Li, "Cultivated land information extraction in UAV imagery based on deep convolutional neural network and transfer learning," *J. Mountain Sci.*, vol. 14, no. 4, pp. 731–741, Apr. 2017.
[43] P. Bosilj, E. Aptoula, T. Duckett, and G. Cielniak, "Transfer learning between crop types for semantic segmentation of crops versus weeds in precision agriculture," *J. Field Robot.*, vol. 37, no. 1, pp. 7–19, Jan. 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21869
[44] C. Douarre, R. Schielein, C. Frindel, S. Gerth, and D. Rousseau, "Transfer learning from synthetic data applied to soil–root segmentation in X-ray tomography images," *J. Imag.*, vol. 4, no. 5, p. 65, 2018. [Online]. Available: https://www.mdpi.com/2313-433X/4/5/65
[45] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," 2015, *arXiv:1512.02325*. [Online]. Available: http://arxiv.org/abs/1512.02325
[46] Google. (2019). *Tensorflow Models on the Edge TPU*. Accessed: Jan. 5, 2020. [Online]. Available: https://coral.ai/docs/edgetpu/models-intro/
[47] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Computer Vision–ECCV*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer, 2014, pp. 740–755.
[48] L. Santos, F. Santos, J. Mendes, P. Costa, J. Lima, R. Reis, and P. Shinde, "Path planning aware of robot's center of mass for steep slope vineyards," *Robotica*, vol. 38, no. 4, pp. 684–698, 2020.
[49] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

ANDRÉ SILVA AGUIAR received the M.Sc. degree in electrical engineering (specialized in robotics) from the Faculty of Engineering, University of Porto, Portugal, in 2019. He is currently pursuing the Ph.D. degree in electrical and computers engineering with UTAD University. He is also a Researcher with the Centre for Robotics in Industry and Intelligent Systems, INESC TEC, Porto. His current research interests are focused on robotics navigation, simultaneous localization and mapping, and deep learning.

FILIPE NEVES DOS SANTOS received the degree (five years degree) in electrical and computer engineering from the Instituto Superior de Engenharia do Porto (ISEP), in 2003, the M.Sc. degree in electrical and computer engineering from the Instituto Superior Técnico (IST), Universidade Técnica de Lisboa, in 2007, and the Ph.D. degree in electrical and computer engineering from the Faculdade de Engenharia (FEUP), Universidade do Porto, Portugal, in 2014. Since 2003, he has been a formal Robotics Researcher, with a large experience on navigation systems for unnamed aerial and ground vehicles. He is currently a Senior Researcher and the Head of the Laboratory of Robotics and the IoT for Smart Precision Agriculture and Forestry, Centre for Robotics in Industry and Intelligent Systems (CRIIS), INESC TEC. He has more than 40 peer-reviewed articles in international conferences and international journals. Actually, his priority research fields are robotics for agriculture/forestry, human–robot interaction, perception, and semantic SLAM.

ARMANDO JORGE MIRANDA DE SOUSA received the Ph.D. degree in electrical and computer engineering (ECE) from the Faculty of Engineering, University of Porto (FEUP), in 2004. His thesis work was in the subarea of automation and robotics. He is currently a Professor with the ECE Department, FEUP, and also an Integrated Researcher with the Center for Intelligent and Industrial Systems, INESC TEC Interface Institute. In 2014, he received the international pedagogical certification "ING.PAED.IGIP" from the International Society for Engineering Education and Modern Engineering Pedagogy. His research areas include higher education side by side with more technical areas of automation and robotics. As a frequent participant in robotic contests, he has earned several national and international merits (examples: vice champion of Robotic Soccer in 2006, winner of the Autonomous Driving of Portuguese Robotics Open of 2019). He has also earned educational awards such as the University of Porto (UP) Excellence Award, in 2015, and ranked within the ten best at ECEL 2015 excellence ELearning awards. He has published more than 80 indexed peer-reviewed articles both in pedagogical issues and more technical areas. One of his most cited articles is "Robust 3/6 DoF self-localization system with selective map update for mobile robot platforms." He also has an active patent entitled "Device and method for identifying a cork stopper, and respective kit." He is also involved in educational and technical funded projects such as "IntelWheels 2" and "EIT CPP 101."

PAULO MOURA OLIVEIRA received the degree in electrical engineering from UTAD University, Portugal, in 1991, and the M.Sc. degree in industrial control systems and the Ph.D. degree in control engineering from Salford University, Manchester, U.K., in 1994 and 1998, respectively. He is currently an Associate Professor with Habilitation with the Engineering Department, UTAD University, and also a Senior Researcher with the Centre for Robotics in Industry and Intelligent Systems (CRIIS), INESC TEC Institute, Porto. He has authored more than 150 peer-reviewed scientific publications. His current research interests are focused on the fields of control engineering, robotics, evolutionary and nature inspired algorithms for single and multiple objective optimization problem solving, swarm optimization, intelligent control, PID control, and control engineering education.

LUIS CARLOS SANTOS received the integrated master's degree in electrical and computers engineering (specialized in robotics) from the Faculty of Engineering, University of Porto, Portugal, in 2017. He is currently pursuing the Ph.D. degree in electrical and computers engineering with UTAD University. He is also a Researcher with the Centre for Robotics in Industry and Intelligent Systems, INESC TEC, Porto. His current research interests are focused on agricultural robotic navigation, specifically in advanced path planning techniques of farming scenarios. His current research topics focus on the safety supervision of robotic software for agriculture.