Deep Semantic Segmentation at the Edge for Autonomous Navigation in Vineyard Rows

Deep Semantic Segmentation at the Edge for Autonomous Navigation in Vineyard Rows

Diego Aghi[1], Simone Cerrato[2], Vittorio Mazzia[3,*] and Marcello Chiaberge[4]

***Abstract*— Precision agriculture is a fast-growing field that aims at introducing affordable and effective automation into agricultural processes. Nowadays, algorithmic solutions for navigation in vineyards require expensive sensors and high computational workloads that preclude large-scale applicability of autonomous robotic platforms in real business case scenarios. From this perspective, our novel proposed control leverages the latest advancement in machine perception and edge AI techniques to achieve highly affordable and reliable navigation inside vineyard rows with low computational and power consumption. Indeed, using a custom-trained segmentation network and a low-range RGB-D camera, we are able to take advantage of the semantic information of the environment to produce smooth trajectories and stable control in different vineyards scenarios. Moreover, the segmentation maps generated by the control algorithm itself could be directly exploited as filters for a vegetative assessment of the crop status. Extensive experimentations and evaluations against real-world data and simulated environments demonstrated the effectiveness and intrinsic robustness of our methodology.**

I. INTRODUCTION

Over the last years, the agriculture industries have been focusing their resources on the study and development of new technologies to increase productivity, cut costs, and ease farmers' jobs by reducing the need for humans for labor-intensive tasks. The wave of innovation brought by the advent of precision agriculture and digital farming has gradually introduced robotics and artificial intelligence into agricultural processes to improve product quality and management. In this context, self-driving systems for agricultural machineries have been a great breakthrough towards the accomplishment of the objectives mentioned above. Indeed, once endowed the proper equipment, these robotic vehicles can harvest [1], spray [2], seed [3] and irrigate [4] as well as collect data for inventory management [5].

Concerning the autonomous navigation in vineyards and orchards, some of the proposed solutions employ GPS devices along with three-dimensional LIDARs and Inertial Measurement Units (IMU) [6], while others exploit only 2D LiDARs [7] [8]. Nonetheless, they face difficulties to reach a large-scale implementation due to the elevated sensors costs and the unreliability in particular situations, which will be discussed later in this work. These issues made the research community move its focus on different kinds of technologies. Indeed, the state-of-the-art approaches for row crop scenarios with thick and high canopies are based on machine vision [9], and deep learning [10]. However, in [9] they use a multispectral camera for image acquisition which significantly raises hardware costs, while in [10] the controller provided is not proportional, and it is limited to 3 basics commands.

In this article, we present a novel low-cost and at the edge motion control system for autonomous navigation in vineyard rows. It exploits semantic segmentation properties to provide a proportional controller that drives the robotic platform along the whole row without colliding with the vine plants. Moreover, it fuses RGB images and depth information of the observed scene to overcome illumination issues and provide a robust control in challenging situations without relying on expensive sensors and GPS signals. Finally, it is possible to directly take advantage of segmentation maps generated by the deep neural network for a vegetative real-time assessment of the crop.

All of our training and testing code and data[1] are open source and publicly available[2].

A. Related Work

Generally, autonomous systems designed to navigate along vineyards or orchards rows employ high-precision GPS receivers and enhancement accuracy techniques [11] or a combination of laser-based sensors and GPS devices [12], [13]. However, in this particular environment, the canopies on the sides of the row reduce the GPS accuracy affecting its reliability [14], [15]. Modern solutions mix multiple sensors such as GPS, inertial navigation systems (INS), wheel encoders, and LIDARs [16] to locate more precisely the mobile platform during the navigation. Nevertheless, the usage of many sensors leads to higher system costs.

Regarding more affordable approaches, in [17] they propose a cost-effective monocular visual odometry (VO) algorithm. Nonetheless, as highlighted by the authors, VO systems show poor performance on long distances due to the accumulating error, and they require an absolute reference integration to preserve the necessary accuracy for a safe autonomous navigation.

*Corresponding author

[1]Diego Aghi is with the Department of Electronics and Telecommunications, Politecnico di Torino, Turin, Italy 10124 diego.aghi@polito.it

[2]Simone Cerrato is with the Department of Electronics and Telecommunications, Politecnico di Torino, Turin, Italy 10124 simone.cerrato@polito.it

[3]Vittorio Mazzia is with the Department of Electronics and Telecommunications, Politecnico di Torino, Turin, Italy 10124 vittorio.mazzia@polito.it

[4]Marcello Chiaberge is with the Department of Electronics and Telecommunications, Politecnico di Torino, Turin, Italy 10124 marcello.chiaberge@polito.it

[1]10.5281/zenodo.4601472

[2]https://github.com/MrD1360/deep_segmentation_vineyards_navigation

On the other hand, the ubiquity of deep learning in the latest technology advancements led to the development of a variety of intelligent systems for multiple applications in precision agriculture [18]. Indeed, in literature can be found crop type classifiers [19]–[21] and crop yield estimators [22]–[24] as well as disease detectors for plants [25], leaves [26] and fruits [27].

B. The Broader Project

The proposed high-level controller for autonomous navigation in the vineyard rows is part of a broader project of our research group that aims at developing an affordable self-driving system for vineyard parcels. Given a global path made of georeferenced waypoints [28], we exploit a dual local planner to overcome the lack of accuracy of the GPS-based localization filter inside the vine rows. Indeed, the signals provided by the satellites are very sensitive to obstacles encountered during the satellite-receiver path (e.g. lush vegetation of vineyards) and that makes the navigation inside vineyard rows a very challenging task [14], [15], [29]. More in detail, since outside the vineyard rows the GPS receiver (a Piksi Multi GNSS receiver by Swift Navigation) has a good view of the sky and better accuracy, a simple Dynamic Window Approach (DWA) is used to switch from one vine row to the next one, exploiting an Extended Kalman Filter (EKF) to fuse the data coming from an Inertial Measurement Unit (IMU) and a GPS receiver. On the other hand, the proposed motion controller exploits semantic information of the vineyard environment in order to navigate throughout the vine rows. It performs periodic checks with respect to the provided global path made of GPS points, obtaining a broader estimation of its position. The overall solution guarantees to autonomously navigate throughout the whole vineyard without expensive sensors and in case of poor quality Global Navigation Satellite System signals due to lush vegetation and thick canopies.
Eventually, all the algorithms have been developed ROS-compatible, in order to make easier the communication among them and to be easily deployed on our developing platform; the Jackal Unmanned Ground Vehicle (UGV) by Clearpath Robotics[3], shown in Fig. 1, that is fully supported by the Robot Operating System(ROS).

II. METHODOLOGY

We propose a control algorithm to perform real-time autonomous navigation along the vineyard rows that leverages the latest advancement on edge AI to obtain a continuous and reliable control with a low-range hardware setup, as described on the presentation video[4]. The devised system exploits the joint information of RGB and depth inputs, cutting costs of expensive sensors and overcoming issues experienced by solutions based on GPS devices.

The workflow of our proposal, as illustrated in Fig. 2, is very straightforward; first, the RGB-D camera placed over the mobile platform acquires the depth map $\boldsymbol{X}^t_{depth}$ and the

[3]https://clearpathrobotics.com/

[4]https://www.youtube.com/watch?v=tt_BfPQ6dIA

Fig. 1: The Jackal Unmanned Ground Vehicle in Grugliasco, Turin, North of Italy.

corresponding RGB frame $\mathbf{X}^t_{rgb}$ at a certain time instant t, where $\boldsymbol{X}^t_{depth} \in \mathbb{R}^{h \times w}$ and $\mathbf{X}^t_{rgb} \in \mathbb{R}^{h \times w \times c}$ with h, w and c as high, width and channels, respectively. Then, we feed a custom deep neural network for semantic segmentation that produces a binary mask as shown in the following equation,

$$\hat{\boldsymbol{X}}^t_{seg} = H_{seg}(\mathbf{X}^t_{rgb}, \Theta) \tag{1}$$

where $\hat{\boldsymbol{X}}^t_{seg} \in \mathbb{R}^{h \times w}$ is the segmentation map containing the semantic information of the input image $\mathbf{X}^t_{rgb}$ and Θ are the model variables leaved to discriminative learning.

$$\hat{\boldsymbol{X}}^t_{cumSeg} = \sum_{n=0}^{S} \hat{\boldsymbol{X}}^{t-n}_{seg} \tag{2}$$

Successively, in order to obtain a more stable and coherent information, we select S consecutive segmentation maps at times $\{t-S, ..., t\}$ and we fuse them as shown in (2). Then, we join information coming from the depth channel $\boldsymbol{X}^t_{depth}$ to dynamically reduce the line of sight of the processed scene. Indeed, that is a fundamental point to achieve an effective and reliable navigation inside curved vineyard rows. Moreover, the line of sight reduction is dynamically computed in order to suitably adjust the depth in different situations and discard as less information as possible from the segmented scene. That is achieved finding the maximum value in the depth matrix $max(\boldsymbol{X}^t_{depth})$ and generating a binary map $\boldsymbol{X}^t_{depthT}$, as follows:

$$X^t_{depthT_{\substack{i=0,...,h \\ j=0,...,w}}}(i,j) = \begin{cases} 0, & \text{if } (X^t_{depth})_{i,j} \geq d_{depth} \\ 1, & \text{if } (X^t_{depth})_{i,j} < d_{depth} \end{cases} \tag{3}$$

where $d_{depth} = l_{depth} * max(\boldsymbol{X}^t_{depth})$ and $l_{depth} \in \mathbb{R}$ is a scalar between 0 and 1. So, it is possible to limit the line of

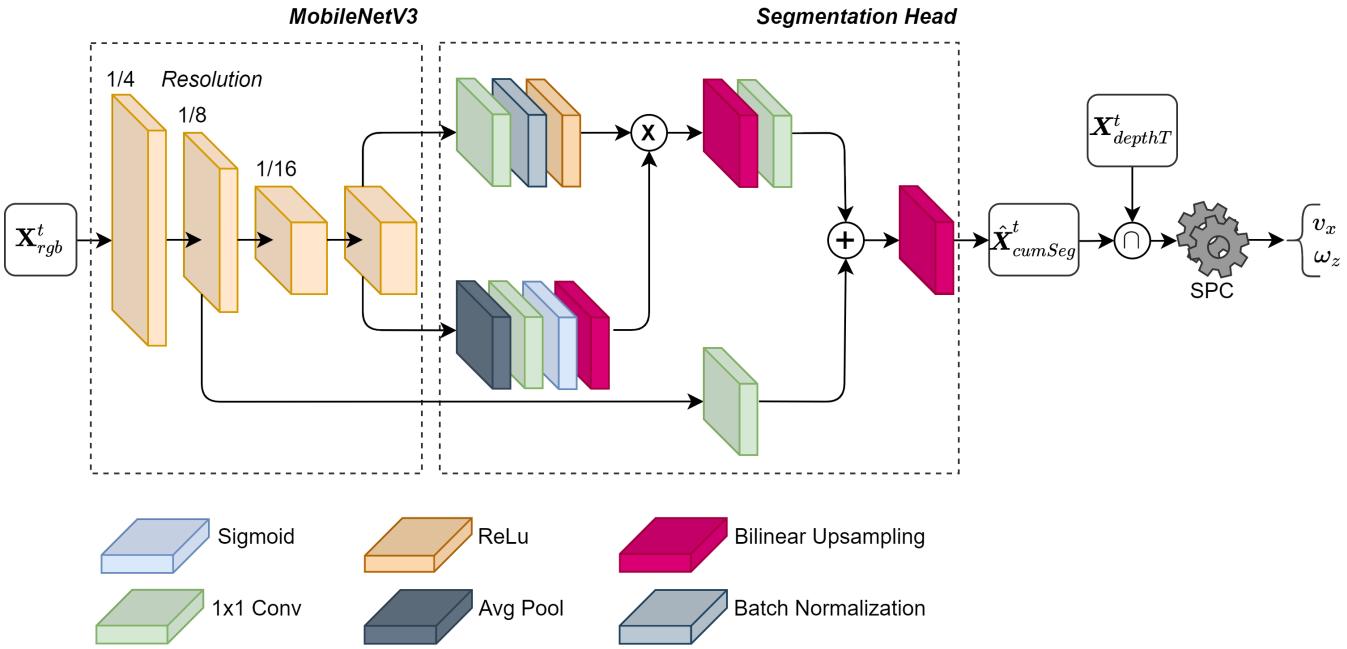


Fig. 2: The pipeline of our proposal. First, we feed the network with the RGB frame acquired by the camera to extract meaningful features. Next, the customized segmentation head provides the segmented frame combining features from different resolutions. Successively, we fuse S segmentation maps and intersect the resulting matrix with the thresholded depth map. Finally, the SPC takes as input the obtained binary map and it computes the control values for the autonomous vehicle.

sight with an interception operation between the cumulative output of the network and the binary map generated,

$$\boldsymbol{X}^t_{ctrl} = \sum_{n=0}^{S} \hat{\boldsymbol{X}}^{t-n}_{seg} \cap \boldsymbol{X}^t_{depthT} \tag{4}$$

where $\boldsymbol{X}^t_{ctrl} \in \mathbb{R}^{h \times w}$ represents the pre-processed input for the control algorithm. In the $\boldsymbol{X}^t_{ctrl}$ binary map 1 means obstacles and 0 free-space.

Finally, the actuation values for linear and angular velocity, v_x and ω_z, are calculated with a simple algorithm that extracts from the segmentation map a proportional control for the two variables.

$$v_x, \omega_z = \text{SPC}(\boldsymbol{X}^t_{ctrl}) \tag{5}$$

In (5), SPC is the segmentation to proportional control algorithm that extracts from the pre-processed segmentation map, $\boldsymbol{X}^t_{ctrl}$, a continuous control for the platform. In addition, v_x is the linear velocity along x axis with respect to the robot's frame and ω_z is the angular velocity around z axis with respect to the UGV's frame following the right-hand rule.

The overall solution can be easily integrated with a generic global path planner to accomplish an affordable and lightweight autonomous navigation along the whole vineyard parcel.

A. Segmentation Network Architecture

A segmentation network is usually composed of an initial feature extractor block and a final segmentation head that takes high-level representations and generates the corresponding segmentation map of the learned classes. Over the years, several deep learning backbones for feature extraction have been proposed. Among all, we carefully select a model that guarantees high accuracy levels by also containing hardware costs and computational load. In particular, we use MobileNetV3 [30] as network backbone with a custom segmentation head for generating the segmentation map predictions $\hat{\boldsymbol{X}}^t_{seg}$ at a certain time instance t. Due to the efficient design of the overall network, the footprint on the RAM is greatly reduced as well as the computational workload required for inference. Overall, the selected backbone is composed of 15 blocks. Each of them presents a linear block and an inverted residual structure introduced with the second version of the network, MobileNetV2 [31]. Moreover, attention modules based on squeeze and excitation [32] are implemented in some of the residual layers and with different non-linearity depending on the specific layer. As in [30], our proposed segmentation head is an upgrade of the reduced version of the Atrous Spatial Pyramid Pooling module [31], [33], [34] (R-ASPP) called Lite R-ASPP (LR-ASPP). It includes two branches connected to different resolutions in order to capture information from multiple-level features. More specifically, one layer applies atrous convolution [35] to the 1/16 resolution to extract denser features, and the other one is used to add a skip connection [36] from the 1/4 resolution to work with more detailed information. Due to the fact that in [30] they use a higher input dimension when performing semantic segmentation, we rescale the global average pooling layer setting the kernel size to 12×12 with strides (4,5). Additionally, to have equal input and output dimensions, we add a bilinear upsampling of a factor of 8

Algorithm 1 SPC algorithm

Input: $\boldsymbol{X}^t_{ctrl}$: Pre-processed segmented image
Output: v_x,ω_z: Continuous control commands

```
noise_reduction_function()
for i=1,··· w do
    c ←sum_colums(X^t_ctrl)
end for
removing_small_zero_clusters(c)
count_zero_clusters(c)
with previous_cluster_center as pcc:
if not anomaly_detection() then
    compute_cluster_center()
    v_x,ω_z ← control_function()
else if initial_state then
    remove_clusters_from_sides()
    max (cluster, key=len)
    compute_cluster_center()
    v_x,ω_z ← control_function()
else if pcc is in clusters or pcc is near clusters then
    select_new_cluster()
    compute_cluster_center()
    v_x,ω_z ← control_function()
end if
```

at the end of the segmentation head.

B. SPC Algorithm

In Algorithm 1, is described a schematic representation of the overall segmentation to proportional algorithm that generates a continuous control for the platform starting from a pre-processed cumulative segmentation output $\boldsymbol{X}^t_{ctrl}$. Firstly, we remove the noise due to grass on the terrain that could mislead our model when predicting by analyzing the network output. We sum the values over rows of $\boldsymbol{X}^t_{ctrl}$ obtaining a column 1D-array $\boldsymbol{g}_{noise} \in \mathbb{R}^h$, then we select all the indices where $\boldsymbol{g}_{noise} < th_{noise}$, with th_{noise} as threshold. Finally, we set $\boldsymbol{X}^t_{ctrl(indices,:)} = 0$. Ideally, we would not have any ones in the top of the image and on the bottom, whilst the majority of them are supposed to be in the central belt.

After that, we sum the values over columns of the obtained matrix $\boldsymbol{X}^t_{ctrl}$, in order to generate the array $\boldsymbol{c} \in \mathbb{R}^w$, that contains the amount of detected vineyards for each column. Therefore, every zero in $\boldsymbol{c}$ is a potential empty space where to route the mobile platform. Next, we detect the clusters of zeros, which are the groups of consecutive zeros, in $\boldsymbol{c}$, and those with a length below a certain threshold are considered as unreliable, and therefore they are not taken into account when computing the control values. Successively, we begin evaluating the different scenarios. First of all, we perform anomaly detection by counting the number of clusters. In case we have just one, we move forward with the control functions. In contrast, when dealing with more clusters, the algorithm tries to identify which is the right one by using information from the previous iterations. More specifically, it checks if the previously computed abscissa used in the control functions is contained or near one of the current detected clusters. In case it is the first algorithm execution, there is no previous command; therefore, supposing that the mobile platform is placed pointing at the center of the vine row, we remove the clusters laying on the sides of the matrix $\boldsymbol{X}^t_{ctrl}$. After that, if there is still more than one cluster, we select the largest one. During the whole process, at any moment, if there are no valid clusters, the computed matrix $\boldsymbol{X}^t_{ctrl}$ is discarded and the next one is elaborated.

The identified cluster represents the obstacle-free space in which we can route the mobile platform to continue the navigation safely. The next step is to compute the linear and angular velocities to drive the mobile platform. To do so, we take the center of the selected cluster, that ideally corresponds to the center of the row in front of the UGV. After that, we estimate the velocities with two custom functions.

$$\omega_z = \begin{cases} -\omega_{z,max} \cdot \left[\frac{d^2}{(\frac{w}{2})^2}\right], & \text{if } d \geq 0 \\ \omega_{z,max} \cdot \left[\frac{d^2}{(\frac{w}{2})^2}\right], & \text{if } d < 0 \end{cases} \tag{6}$$

In (6) is represented the control function for the angular velocity whilst for the linear velocity we use (7) as in [37].

$$v_x = v_{x,max} \cdot \left[1 - \left[\frac{d^2}{(\frac{w}{2})^2}\right]\right] \tag{7}$$

where $\omega_{z,max}$ and $v_{x,max}$ are two constants which defines the maximum angular and linear velocity of the mobile platform respectively, w is the width of $\boldsymbol{X}^t_{ctrl}$ and d is defined as:

$$d = x_c - \frac{w}{2} \tag{8}$$

with x_c center coordinate of the selected cluster. The final control values sent to the actuators are calculated using the exponential moving average (EMA), formalized in (9), in order to prevent the mobile platform from sharp motion.

$$EMA_t = EMA_{t-1} \cdot (1 - \alpha_{EMA}) + \begin{bmatrix} v_x \\ \omega_z \end{bmatrix} \cdot \alpha_{EMA} \tag{9}$$

where t is the time step and α_{EMA} the multiplier for weighting the EMA.

III. EXPERIMENTAL RESULTS AND DISCUSSION

A. Dataset Creation

In order to create a dataset for training and testing the deep neural network, we carry out field surveys in two distinct agricultural areas in the North of Italy: Grugliasco near Turin in Piedmont region and Valle San Giorgio di Baone in the Province of Padua in the Veneto region. The data is collected at different times of the day, with diverse weather conditions, and they present a variety of terrain types and wine qualities. To have different perspectives inside the vineyard rows, we acquire several videos with only three fixed orientations: one pointing the camera at the center of the vineyard row and the other two pointing to the left and right sides, respectively. For training and testing, we select one frame every 25 in

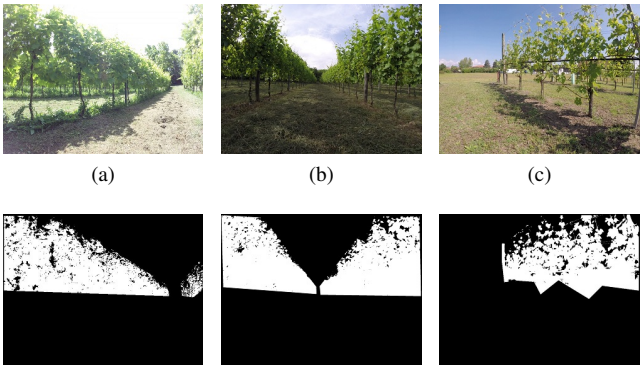


Fig. 3: In (**a**),(**b**) and (**c**) are shown three instances of the dataset, one for each video orientation. (**a**) is an example left samples, (**b**) center, and (**c**) right. (**d**),(**e**) and (**f**) are the corresponding binary masks for the supervised training. Dataset samples have been collected with different weather conditions and at different times of the day. The resulting heterogeneous training set is aimed at giving generality and robustness to the model.

order to work with quite different scenarios. Finally, to generate the masks for the supervised training, we exploit the collected images by first manually annotating them using an open-source software [38], and successively refining the obtained binary masks using a Gaussian mixture model [39] in order to train and evaluate the network with more accurate masks. Before feeding the network, all the acquired images are resized to a fixed input dimension, 224x224, and then normalized with values from 0 to 1.

Overall, our dataset consists of 1538 RGB images with their corresponding binary mask, of which 1038 from the vineyards in Veneto region and 500 from the other rural area.

B. Network Training and Evaluation

For training and testing our segmentation network, we use the two surveyed rural areas. Indeed, a completely different area for testing allows to properly evaluate the generalization capabilities of the network not only in different weather conditions and times of the day but also in entirely different scenarios.

We train our model applying transfer learning [41] to the selected backbone. Indeed, rather than using randomly initialized weights, we exploit MobileNetV3 variables derived from an initial training phase on the 1k classes and 1.3M images of the ImageNet dataset [42]. That largely improves the final robustness of the model and its final generalization capability with a reduced number of training samples.

We use as loss function the intersection over unit (IoU) as shown in the following equation:

$$\mathcal{L}(\Theta) = \frac{1}{N}\sum_{i=0}^{N}\left(1 - \frac{\hat{\boldsymbol{X}}_{seg}^{(i)} \cap \boldsymbol{X}_{seg}^{(i)}}{\hat{\boldsymbol{X}}_{seg}^{(i)} \cup \boldsymbol{X}_{seg}^{(i)}}\right) \tag{10}$$

where N is the number of training samples, $\hat{\boldsymbol{X}}_{seg}^{(i)}$ is a prediction mask of the i-th input instance and $\boldsymbol{X}_{seg}^{(i)}$ is the corresponding ground truth.

As regularization, we adopt dropout [43] with a drop rate of $p = 0.2$ and early stopping using a 0.1 of the training as validation. We use an Nvidia RTX2080 GPU with 8GB of memory, 64GB of DDR4 SDRAM and CUDA 11 with TensorFlow 2.x. With our hardware configuration, the training phase of the segmentation head takes approximately 3 minutes.

The resulting network is optimized in order to reduce latency, inference cost, memory, and storage footprint. That is mainly achieved with two distinct techniques: model pruning and quantization. With the first one we simplify the topological structure, removing unnecessary parts of the architecture, or favors a more sparse model introducing zeros to the parameter tensors. On the other hand, with quantization, we reduce the precision of the numbers used to represent model parameters from float32 to float16. That can be accomplished after the training procedure with a post-training quantization procedure.

In Table I are summarized experimentation results with some reference architectures. It is possible to notice how weight precision and graph optimization have a high impact on the inference energy consumed by the network and its impact on the main memory of the tested device.

We evaluate our model using IoU metric with a threshold of 0.9 on the output logits. In this way, we take into account only the values that would be actually used by the SPC algorithm when performing autonomous navigation. On the test set, the final overall metric is 0.62 with a pixel accuracy of 92.7%. Moreover, considering as true positive only the test instances with a IoU score that exceeds some predefined

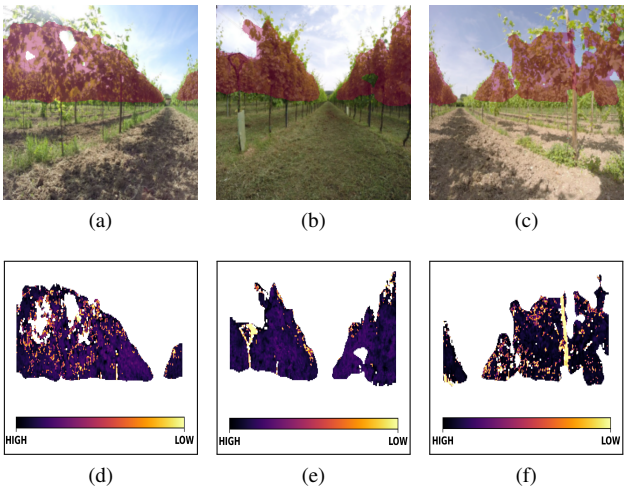


Fig. 4: Qualitative assessment of the segmentation capabilities of the network. In (**a**), (**b**) and (**c**) are presented three samples taken from the test site. On the other hand, (**d**), (**e**) and (**f**) show a vegetative assessment that can be derived directly using generated masks and RGB channels [40].

TABLE I: Comparison between different devices' energy consumption and inference performances.Graph optimization (G.O.) and weight precision (W.P.) reduction further increase the capability of our already efficient neural network architecture, helping to deal with energy, speed, size, and cost constraints.

Device	GO	WP	Latency [ms]	E_{net} [mJ]	Size [MB]
RTX 2080	N	FP32	28 ± 109	819	9.3
	Y	FP32	0.1 ± 0.3	52	7.4
	Y	FP16	0.1 ± 0.2	39	4.9
Cortex-A57	Y	FP32	111 ± 0.9	166	4.2
	Y	FP16	111 ± 2.3	165	2.2
Cortex-A76	Y	FP32	55.4 ± 10.6	210	4.2
	Y	FP16	65.3 ± 9.5	248	2.2

threshold, we compute the precision values for different IoU thresholds. In particular, precision at 0.4, 0.5, 0.6 and 0.7 are equal to 96.8%, 85.8%, 62.8%, and 24%, respectively. Fig. 4 allows to qualitative assess the accuracy of the network and its generalization capabilities with three segmentation maps generated from the test site. Moreover, as previously stated, it is possible to exploit the generated segmentation maps not only for navigation but also as a starting point for a vegetative assessment of the crop. Three false-color representations are presented in Fig. 4 as examples of vegetation index maps that can be derived directly from reflectances of RGB channels [40]. Nevertheless, other spectral bands can be used in conjunction with segmentation maps to extract valuable crop indices.

C. Motion Controller Evaluation

As already introduced in Section III-A, all collected images are acquired with three fixed orientations of the vineyard rows. That allows us to have more flexibility when extrapolating statistical value to assess the controller performances. More specifically, for each row and for each type of video in the test set, we compute the mean value and standard deviation for the three most meaningful variables of the controller: the abscissa, the linear, and angular velocities. Grouping the calculated values by video class, we obtain results summarized in Table II. The outcomes are referred to a frame dimension of 224x224 with maximum linear and angular velocities equal to 1 m/s and 1 rad/s, respectively. Furthermore, we set $l_{depth} = 0.5$, $th_{noise} = 0.03 \cdot max(\boldsymbol{g}_{noise})$, $\alpha_{EMA} = 0.1$, and we fuse $S = 3$ segmentation maps before feeding the SPC. In addition, for each video orientation of the test set, we compute the fault rate percentage (FR) of iterations where the control algorithm could not provide any command.

Finally, we run the optimized neural network along with the presented controller on the embedded Jackal's PC. It is mainly composed of a CPU Intel Core i3-4330TE @ 2.4 GHz and a DDR3 RAM of 4GB. In such conditions, the Intel Realsense D435i provides both RBG images and depth map at 30 FPS, while the optimized deep neural network is able to process the RGB images on the CPU at 22 FPS, and the controller generates velocity commands at 5 Hz.

TABLE II: Overall motion controller evaluation

Class	Metrics	Abscissa	v_x[m/s]	ω_z [rad/s]	FR [%]
Center	μ_{center}	111.15	0.9926	0.0052	0.0
	σ_{center}	5.05	0.0005	0.0005	
Left	μ_{left}	44.42	0.6434	0.3566	0.04
	σ_{left}	7.67	0.0084	0.0083	
Right	μ_{right}	184.93	0.5971	-0.4026	0.26
	σ_{right}	12.98	0.0114	0.0129	

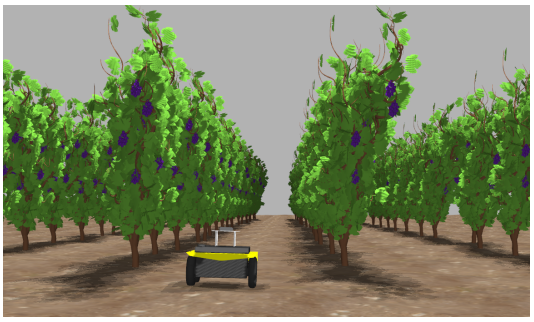
All considered, the overall performances of the deployed algorithm are very promising. Indeed, five output commands per second are greatly sufficient taking into account the slow dynamic of the vehicle.

D. Simulation Environment Results

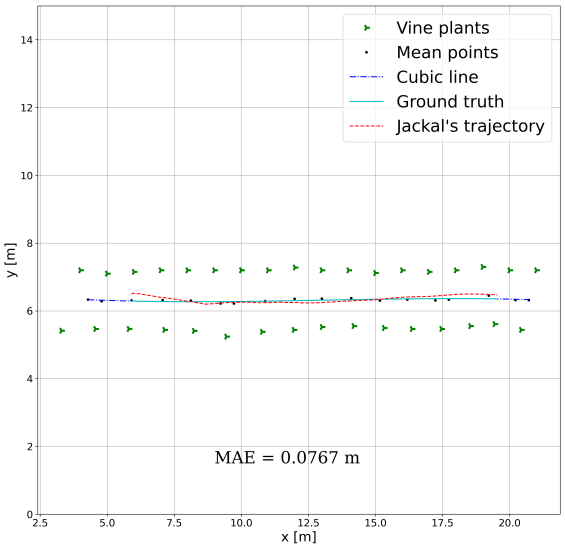
In addition to previous results, the controller algorithm is tested in two custom simulated environments. As a simulator engine, we use Gazebo[5], which is ROS-compatible and allows us to customize the simulation environments and exploit several plugins to simulate sensors(e.g. cameras). The simulation model of the Jackal UGV is provided by Clearpath Robotics through URDF file, which contains specification about mechanical structure and links related to the robot platform. On the other hand, the simulation environments are designed from scratch; drawing a completely custom vine plant and arranging the vine rows on a plane in order to realistically reproduce the vineyard geometry, as shown in Fig. 5a and Fig. 6a. The inter-row distance ranges from 1.70 meters to 2.00 meters, while the vine plants' distance in the same row ranges from 0.70 meters to 1.0 meters from each other. The simulated plane reproduces the bumpy and uneven terrain of a real vineyard exploiting the heightmap option of the SDF format for model building in Gazebo. The involved simulated camera is an Intel Realsense d435i, that by means a Gazebo plugin is able to provide RGB frames and depth map of what the UGV is seeing. It is placed 10 centimeters up and with 0 degrees of tilt with respect to Jackal's plate.
The real trajectory followed by the Jackal has been obtained using a Gazebo plugin (*libgazebo_ros_p3d*), which provides the real pose of the robot inside the simulation environment so that it is possible to perform additional analysis on the followed path. The procedure to measure the ability of the proposed controller to maintain the vine rows centrality is very straightforward. First, the midline between two vine rows is obtained averaging the Euclidean distances of the points belonging to a vine row with respect to the nearest point related to the other vine row, and approximating such distances with a cubic line. Subsequently, taking the midline as ground truth and the Jackal's trajectory, the Mean Absolute Error (MAE) is computed to measure the quality of the path.

The first simulation environment consists of straight vine rows, where the proposed controller has achieved excellent results obtaining a MAE$= 0.0767$ m over several trials. While, the second simulated vineyard is composed of vine

[5]http://gazebosim.org/

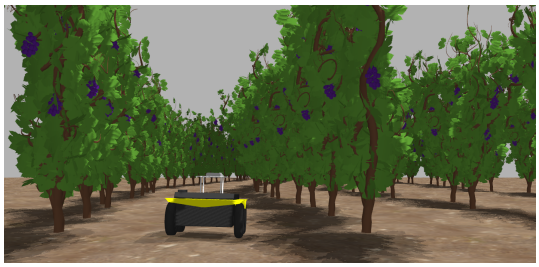


(a)

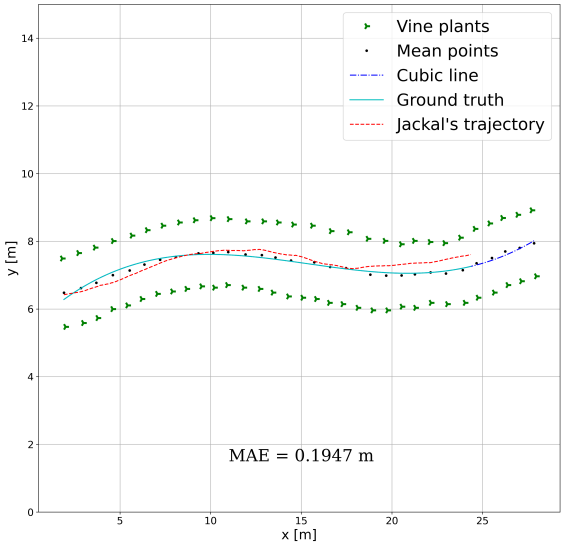


(b)

Fig. 5: (**a**) shows the straight vine rows of the first simulation environment. While (**b**) summarizes the results coming from the first simulation environment. The most important aspects are: the ground truth (cyan line) and the Jackal's pose over time (dash red line). The black points represent the mid-distance between vine rows, while the blue line is an approximation of such points.



(a)



(b)

Fig. 6: (**a**) shows the slight curvature of vine rows in the second simulated environment. (**b**) contains the relevant results obtained from the second simulation environment. The black points represent the mid-distance between vine rows, while the blue line is an approximation of such points. Moreover, this chart allows to visualize the comparison between the ground truth (cyan line) and the UGV's trajectory (dash red line).

rows with curvatures, where the controller slightly worsens the performances achieving a MAE of 0.1947 m. Nevertheless, results are still very promising, taking into account the challenging environment.

IV. CONCLUSIONS

We introduced a novel low-cost motion controller algorithm driven by edge AI image segmentation that showed very promising results in motion control robustness and segmentation accuracy, solely relying on low-range equipment. It can be easily deployed on a variety of mobile platforms thanks to the low computational load required by the overall processing framework. Finally, generated semantic information of the navigation environment can be directly exploited for a vegetative assessment of the crop. Further works will aim at assessing our proposed module inside a broader navigation framework devised by our research group for affordable autonomous vineyard navigation.

ACKNOWLEDGMENT

This work has been developed with the contribution of the Politecnico di Torino Interdepartmental Centre for Service Robotics PIC4SeR[6] and SmartData@Polito[7].

REFERENCES

[1] C. W. Bac, E. J. van Henten, J. Hemming, and Y. Edan, "Harvesting robots for high-value crops: State-of-the-art review and challenges ahead," *Journal of Field Robotics*, vol. 31, no. 6, pp. 888–911, 2014.

[2] R. Berenstein, O. B. Shahar, A. Shapiro, and Y. Edan, "Grape clusters and foliage detection algorithms for autonomous selective vineyard sprayer," *Intelligent Service Robotics*, vol. 3, no. 4, pp. 233–243, 2010.

[3] J. Katupitiya, R. Eaton, and T. Yaqub, "Systems engineering approach to agricultural automation: new developments," in *2007 1st Annual IEEE Systems Conference*. IEEE, 2007, pp. 1–7.

[4] D. Kohanbash, A. Valada, and G. Kantor, "Irrigation control methods for wireless sensor network," in *2012 Dallas, Texas, July 29-August 1, 2012*. American Society of Agricultural and Biological Engineers, 2012, p. 1.

[6]https://pic4ser.polito.it

[7]https://smartdata.polito.it

[5] N. Virlet, K. Sabermanesh, P. Sadeghi-Tehran, and M. J. Hawkesford, "Field scanalyzer: an automated robotic field phenotyping platform for detailed crop monitoring," *Functional Plant Biology*, vol. 44, no. 1, pp. 143–153, 2017.

[6] F. Callegati, A. Samorì, R. Tazzari, N. Mimmo, and L. Marconi, "Autonomous tracked agricultural ugv configuration and navigation experimental results," 2018.

[7] P. M. Blok, K. van Boheemen, F. K. van Evert, J. IJsselmuiden, and G.-H. Kim, "Robot navigation in orchards with localization based on particle filter and kalman filter," *Computers and Electronics in Agriculture*, vol. 157, pp. 261–269, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169918315230

[8] F. B. Malavazi, R. Guyonneau, J.-B. Fasquel, S. Lagrange, and F. Mercier, "Lidar-only based navigation algorithm for an autonomous agricultural robot," *Computers and Electronics in Agriculture*, vol. 154, pp. 71–79, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169918302679

[9] J. Radcliffe, J. Cox, and D. M. Bulanon, "Machine vision for orchard navigation," *Computers in Industry*, vol. 98, pp. 165–171, 2018.

[10] D. Aghi, V. Mazzia, and M. Chiaberge, "Autonomous navigation in vineyards with deep learning at the edge," in *International Conference on Robotics in Alpe-Adria Danube Region*. Springer, 2020, pp. 479–486.

[11] O. Ly, H. Gimbert, G. Passault, and G. Baron, "A fully autonomous robot for putting posts for trellising vineyard with centimetric accuracy," in *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*, 2015, pp. 44–49.

[12] S. J. Moorehead, C. K. Wellington, B. J. Gilmore, and C. Vallespi, "Automating orchards: A system of autonomous tractors for orchard maintenance," in *Proceedings of the IEEE International Conference of Intelligent Robots and Systems, Workshop on Agricultural Robotics*, 2012.

[13] S. Hansen, E. Bayramoglu, J. C. Andersen, O. Ravn, N. Andersen, and N. K. Poulsen, "Orchard navigation using derivative free kalman filtering," in *Proceedings of the 2011 American Control Conference*. IEEE, 2011, pp. 4679–4684.

[14] M. S. N. Kabir, M.-Z. Song, N.-S. Sung, S.-O. Chung, Y.-J. Kim, N. Noguchi, and S.-J. Hong, "Performance comparison of single and multi-gnss receivers under agricultural fields in korea," *Engineering in Agriculture, Environment and Food*, vol. 9, no. 1, pp. 27–35, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1881836615300136

[15] S. Marden and M. Whitty, "Gps-free localisation and navigation of an unmanned ground vehicle for yield forecasting in a vineyard," in *Recent Advances in Agricultural Robotics, International workshop collocated with the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*, 2014.

[16] P. Astolfi, A. Gabrielli, L. Bascetta, and M. Matteucci, "Vineyard autonomous navigation in the echord++ grape experiment," *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 704–709, 2018.

[17] S. Zaman, L. Comba, A. Biglia, D. R. Aimonino, P. Barge, and P. Gay, "Cost-effective visual odometry system for vehicle motion control in agricultural environments," *Computers and Electronics in Agriculture*, vol. 162, pp. 82–94, 2019.

[18] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169917308803

[19] N. Kussul, M. Lavreniuk, S. Skakun, and A. Shelestov, "Deep learning classification of land cover and crop types using remote sensing data," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 5, pp. 778–782, 2017.

[20] A. K. Mortensen, M. Dyrmann, H. Karstoft, R. N. Jørgensen, R. Gislum, *et al.*, "Semantic segmentation of mixed crops using deep convolutional neural network." in *CIGR-AgEng Conference, 26-29 June 2016, Aarhus, Denmark. Abstracts and Full papers*. Organising Committee, CIGR 2016, 2016, pp. 1–6.

[21] V. Mazzia, A. Khaliq, and M. Chiaberge, "Improvement in land cover and crop classification based on temporal features learning from sentinel-2 data using recurrent-convolutional neural network (r-cnn)," *Applied Sciences*, vol. 10, no. 1, p. 238, 2020.

[22] Q. Wang, S. Nuske, M. Bergerman, and S. Singh, "Automated crop yield estimation for apple orchards," in *Experimental robotics*. Springer, 2013, pp. 745–758.

[23] K. Kuwata and R. Shibasaki, "Estimating crop yields with deep learning and remotely sensed data," in *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2015, pp. 858–861.

[24] V. Mazzia, L. Comba, A. Khaliq, M. Chiaberge, and P. Gay, "Uav and machine learning based refinement of a satellite-driven vegetation index for precision agriculture," *Sensors*, vol. 20, no. 9, p. 2530, 2020.

[25] R. Calderón, J. A. Navas-Cortés, and P. J. Zarco-Tejada, "Early detection and quantification of verticillium wilt in olive using hyperspectral and thermal imagery over large areas," *Remote Sensing*, vol. 7, no. 5, pp. 5584–5610, 2015.

[26] M. López-López, R. Calderón, V. González-Dugo, P. J. Zarco-Tejada, and E. Fereres, "Early detection and quantification of almond red leaf blotch using high-resolution hyperspectral and thermal imagery," *Remote Sensing*, vol. 8, no. 4, p. 276, 2016.

[27] Y. Tian, G. Yang, Z. Wang, E. Li, and Z. Liang, "Detection of apple lesions in orchards based on deep learning methods of cyclegan and yolov3-dense," *Journal of Sensors*, vol. 2019, 2019.

[28] V. Mazzia, F. Salvetti, D. Aghi, and M. Chiaberge, "Deepway: a deep learning waypoint estimator for global path generation," 2021.

[29] S. Cerrato, "Gps-based autonomous navigation of unmanned ground vehicles in precision agriculture applications," Master's thesis, Politecnico di Torino, 2020.

[30] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1314–1324.

[31] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[32] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[33] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv preprint arXiv:1412.7062*, 2014.

[34] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[35] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[36] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[37] D. Aghi, V. Mazzia, and M. Chiaberge, "Local motion planner for autonomous navigation in vineyards with a rgb-d camera-based algorithm and deep learning synergy," *Machines*, vol. 8, no. 2, p. 27, 2020.

[38] A. Dutta and A. Zisserman, "The VIA annotation software for images, audio and video," in *Proceedings of the 27th ACM International Conference on Multimedia*, ser. MM '19. New York, NY, USA: ACM, 2019. [Online]. Available: https://doi.org/10.1145/3343031.3350535

[39] D. A. Reynolds, "Gaussian mixture models." *Encyclopedia of biometrics*, vol. 741, pp. 659–663, 2009.

[40] L. Pádua, P. Marques, J. Hruška, T. Adão, E. Peres, R. Morais, and J. J. Sousa, "Multi-temporal vineyard monitoring through uav-based rgb imagery," *Remote Sensing*, vol. 10, no. 12, p. 1907, 2018.

[41] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big data*, vol. 3, no. 1, pp. 1–40, 2016.

[42] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.