FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Robot navigation in vineyards based on the visual vanish point concept

José Maria Queirós Rodrigues Sarmento

MIEEC Master Thesis

Supervisor: Filipe Batista Neves dos Santos

Co-Supervisor: Armando Jorge Sousa

Co-Supervisor: André Silva Aguiar

July 27, 2021

Robot navigation in vineyards based on the visual vanish point concept

José Maria Queirós Rodrigues Sarmento

MIEEC Master Thesis

July 27, 2021

Resumo

A navegação autónoma na agricultura é muito desafiante, pois geralmente ocorre ao ar livre, onde há terreno acidentado, iluminação natural não controlada, cenários orgânicos em constante mudança e, às vezes, ausência de sinal para o Sistema Global de Navegação por Satélite (SGNS). Nesta dissertação, com a finalidade de navegar entre vinhedos é proposto um sistema de navegação baseado no controlador Proporcional, Integrativo e Derivativo (PID). Para tal, irá ser utilizada uma câmera e uma placa de desenvolvimento (Google Coral Dev Board mini) que possui uma Unidade de Processamento Tensorial (UPT). A orientação é possivel através da combinação de dois métodos para encontrar o centro do vinhedo: primeiro, estimando o ponto de fuga e, segundo, calculando a média da posição das duas detecções de base do tronco mais próximas. Então, o erro angular é determinado pela percepção monocular do ângulo. Para obter a posição do tronco da videira, é usada a detecção de objetos usando Redes Neurais Convolucionais (RNC) baseadas em *Deep Learning* (DL). As RNCs foram treinadas usando *Transfer Learning* (TL), que requer um conjunto de dados menor relativamente a métodos convencionais de treino. Para tanto, foi criado um conjunto de dados de 4221 imagens considerando técnicas de recolha de imagem, anotação e augmentação de imagens. Os resultados mostram que nosso sistema é capaz de detectar o ponto de fuga com uma média do erro absoluto de 0,48 graus e pode ser considerado para controle autónomo. Para avaliar o controlador proposto, uma simulação visual da vinha é criada utilizando o Gazebo. O controlador conjunto proposto é capaz de navegar entre uma vinha reta simulada, com um valor eficaz do erro de 1,17 cm. Além disso, foi também testada a simulação de uma vinha curva que foi modelada a partir de uma vinha real da região do Douro, onde o robô foi capaz de guiar com um valor eficaz do erro de 17,22 cm.

Abstract

Autonomous navigation in agriculture is very challenging as it usually takes place outdoors where there is rough terrain, uncontrolled natural lighting, constantly changing organic scenarios and sometimes the absence of a Global Navigation Satellite System (GNSS). In this work, a single camera and a Google coral dev Board Edge Tensor Processing Unit (TPU) setup is proposed to navigate among a woody crop, more specifically a vineyard, using a Proportional Integrative Derivative (PID)-based controller. Guidance is provided by combining two methods to find the center of the vineyard: First, by estimating the vanishing point and second, by averaging the position of the two closest base trunk detections. Then, by monocular angle perception, the angular error is determined. For obtaining the trunk position in the image, object detection using Deep Learning (DL) based Convolutional Neural Networks (CNN) is used. The CNNs were trained using Transfer Learning (TL), which requires a smaller dataset than conventional training methods. For this purpose, a dataset of 4221 images was created considering image collection, annotation and augmentation techniques. The results show that our framework can detect the vanishing point with an average of the absolute error of 0.48 degrees and can be considered for autonomous control. To evaluate the proposed controller, a visual vineyard simulation is created using Gazebo. The proposed joint controller is able to navigate a simulated straight vineyard with an Root Mean Squared (RMS) error of 1.17 cm. Moreover, a simulated curved vineyard modeled after the Douro region is tested in this work, where the robot was able to steer with a RMS error of 17.22 cm.

Agradecimentos

Em primeiro lugar quero agradecer à Faculdade de Engenharia do Porto e ao INESC-TEC a possibilidade de integrar o projeto de investigação em causa.
Ao meu orientador e co-orientadores, Filipe Neves dos Santos, Armando Sousa e André Aguiar pelo excelente apoio, disponibilidade e dedicação ao presente trabalho.
Gostaria ainda de agradecer a todos aqueles que dentro do INESC-TEC me prestaram auxílio e de alguma forma contribuíram para a realização deste trabalho.
Por último, à minha família e amigos um agradecimento especial por todo o apoio e carinho durante todo este percurso.

José Sarmento

Contents

List of Figures

List of Tables

Abbreviations

AP	Average Precision
BB	Bounding Box
DL	Deep Learning
EKF	Extended Kalman Filter
FN	False Negative
FOV	Field Of View
FP	False Positive
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
IOU	Intersection Over Union
LiDAR	Light Detection And Ranging
NN	Neural Network
PID	Proportional Integrative Derivative
PV	Precision Viticulture
CNN	Convolutional Neural Network
TL	Transfer Learning
RMS	Root Mean Square
ROS	Robot Operating System
RSSI	Received Signal Strength Indication
SSD	Single Shot Detection
TP	True Positive
TPU	Tensor Processing Unit
VP	Vanishing Point
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Context

Agricultural production differs from other contexts, because it is affected by uncontrolled inputs, such as climate, which affect the productivity of the system and constantly change the structures and characteristics of the environment (Marinoudi et al., 2019). Humans are easily capable of identifying and working around these changing scenarios. Still, some of these jobs can be demanding and depending on the terrain and weather, it may be unsuitable or unpleasant for humans to do this work. The preceding factors lead to a decline in the agricultural workforce in well-developed countries as low productivity and hard, specialized work cannot justify the choice. Currently, there is a shift to more stable, well-paying jobs in industry and services. Driven by rising labour costs, interest in automation has increased to find labour-saving solutions for crops and activities that are more difficult to automate (Christiaensen et al., 2021). An example is precision viticulture (PV), which aims to optimize vineyard management, reduce resource consumption and environmental impact, and maximize yield and production quality through automation, with robots playing a major role as monitoring tools, for example, as reviewed by Ammoniaci et al. (2021).

This work aims to develop a vision-based guidance system for woody crops, more precisely for vineyards. This type of crops brings several challenges to autonomous robots. To automate them, the first step is the ability to navigate between vine rows autonomously, which requires taking into account the rough terrain that prevents the use of odometry for long periods of time, the dense vegetation and harsh landscape that can obstruct the Global Navigation Satellite System (GNSS) signal, the outdoor lighting and the constantly changing appearance of the crops, and, depending on the season, the degree of noise added by loose vine whips and grass (Santos et al., 2021). This dissertation proposes a single-camera visual guidance system that combines two methods to find the center of the vineyard rows. First, by estimating the vanishing point by approximating the base of the trunks to lines and then intersecting them. Second, by averaging the position of the two closest base trunk observations. Then, by observing the acquired horizontal position, an angular error is determined for each method to the center of the image. To obtain the base of the trunks a Convolutional Neural Network (CNN) model trained with transfer learning was used.

The proposed system consists of a single camera and an Edge Tensor Processing Unit (TPU) Dev Board (Coral) capable of deep learning inference. Also, a visual vineyard simulation is created to develop and test the controller.

1.2 Motivation

Navigation in vineyards is quite complex as it is usually in rough terrain, on steep slopes and under the influence of weather and light. This means that there are constantly changing light conditions and, combined with irregular vine rows, presents a real challenge for any vision based guidance system. This dissertation intends to investigate the use of vanishing point detection to estimate angular orientation from a monocular vision system. Study Deep Learning object detection from training procedure to edge deployment. And explore the ROS2 to create a visual steering guidance node.

1.3 Objectives

The main objective of this dissertation is to develop a monocular, vision-only guidance system for use in vineyards. To do this, this dissertation intends to:

- Obtain the base trunk detections from an inference algorithm in real-time from a single camera input;
- Develop a new method to locate and guide the robot in between vineyards, making use of vanishing point detection on vine rows to obtain a reference point and obtain an angular error trough monocular angular perception;
- Develop a visual guidance controller;
- Evaluate vineyard base trunk detection;
- Evaluate Vanishing Point detection performance;
- Create a simulated environment in order to test the visual steering algorithm and evaluate its performance;
- Validate and compare results with other state of the art approaches.

1.4 Contributions

The master thesis contributions are:

- A vineyard base trunk dataset with 4221 images on the summer and winter seasons, including augmentation procedures. (https://doi.org/10.5281/zenodo.5038646)

- Two Gazebo worlds for vineyard simulation. A straight vineyard and a curved one, the latter inspired by the vineyards of the region Portugal Douro. (https://gitlab.inesctec.pt/agrob/Agrob4Simulation/-/tree/ros2)
- Two papers:
  - "Autonomous Robot Visual-only Guidance in Agriculture using Vanishing Point Estimation" in EPIA 2021 Conference on Artificial Intelligence. (Accepted);
  - "Robot navigation in vineyards based on the visual vanish point concept" in IRIA 2021. (Submitted).

1.5 Organization

The remainder of this paper is organized as follows. Chapter 2 contains a survey of related work. Chapter 3 details the hardware used and all the steps to obtain the vanishing point estimate and controller implementation. Chapter 4 presents the methodology used to evaluate the proposed solution and discusses the results. Finally, Chapter 5 concludes this work and discusses future work.

Chapter 2

State of Art

In this chapter:

- The topic Deep Learning (DL) for the purpose of object detection and the concept of transfer learning are examined and their advantages discussed;
- The vanishing point detection is presented, starting with conventional methods, then the state of the art for vanishing point detection using deep learning techniques is presented;
- The problem of Autonomous Navigation is presented and the various guidance techniques and solutions are discussed along with some simulations of the agricultural environment.

2.1 Deep Learning

2.1.1 Object detection

Deep Learning (DL) object detection is an extremely useful tool for agricultural automation. However, most object detection techniques that achieve high accuracy use Convolutional Neural Networks (CNN), which are mostly operated via cloud-based services due to the high demands on these powerful CNNs. And from the point of view of a mobile robot working under limited network and hardware conditions, and e.g. in the case of object detection based navigation that requires a fast response, cloud computing is not the ideal solution. Edge computing solutions do not need to upload and download data compared to cloud-based solutions, which is not only faster but also more secure and reduces costs. Recently, has there been a rise in faster and less demanding CNNs such as Single Shot Detection (SSD) (Liu et al., 2016) and You Only Look Once (YOLO) (Redmon et al., 2016) also dedicated low-profile hardware for inference for example Nvidia Jetson Nano (Fig. 2.1a), Intel Neural Compute Stick 2 (Fig. 2.1b) and Google coral Dev Board Edge TPU (Fig. 2.1c). Both will enable advances in object detection for mobile robots by providing object detection at the edge.

Both Hennessy et al. (2021) and Ahmad et al. (2021) use object detection for the purpose of locating weeds, Fig. 2.2c shows weed detection by the first author. In the first, it is also important

(a) Nvidia jetson nano, from Nvidia.

(b) Intel Neural Compute stick 2, from Intel.

(c) Google coral Dev board, from Google.

Figure 2.1: Examples of dedicated low-profile hardware for inference.

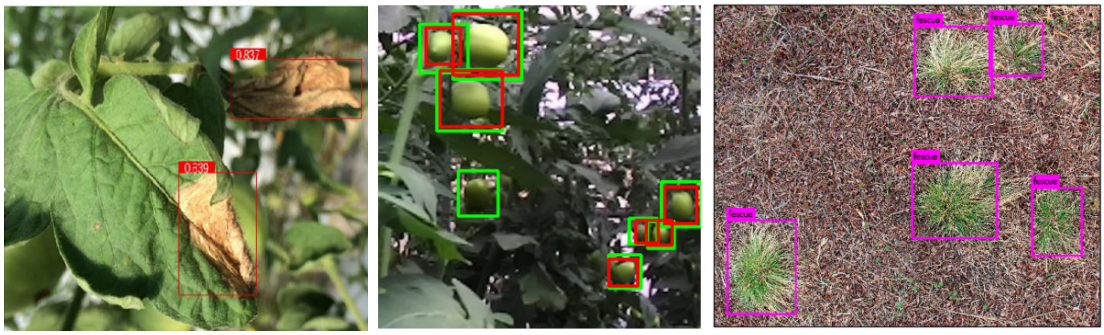
to note that for the spring season, YOLO-V3 clearly outperforms YOLO-V3 Tiny, also in this work is highlighted that the Tiny variant is less affected by the resolution decrease, which will result in faster inference. Also, for the purpose of image classification after detection, both of them propose a variety of CNNs, since this is not so relevant for this dissertation, we will focus only on the object detection part.

With the goal of counting objects, Vasconez et al. (2020) and Buzzy et al. (2020) use object detection to identify and count. The first, detects and counts a variety of fruits (avocado, apple and lemons), in Fig. 2.2e the avocado detection is shown. In this first work, a Faster RCNN and an SSD Mobilenet are evaluated, both prove to be good for counting fruit. However, MobileNet is faster than Faster RCNN and takes 60 ms for inference while the second one took 220 ms. The second work uses object detection to count leaves, Fig. 2.2f. It is also important to mention that compared to a Faster RCNN, the author concludes that Tiny-YOLOv3, improves the inference time, F1 score (94%) and false positive rate (FPR), while other measures such as and AP are degraded, which means that Tiny YOLO-V3 has higher accuracy and fewer false positive events than Faster RCNN.

Chen et al. (2021) proposes a drone coupled with a Nvidia Jetson Nano TX2 to detect Fruit Tree Pests with Deep Learning. Thus, real-world performance is evaluated at the edge. In this work, a comparison between YOLO-V3 and Tiny YOLO-V3 is done, which is very interesting as it shows that the mAP for YOLO-V3 Tiny, although smaller than YOLO-V3, not by a large margin, as the first model achieves a value of 93% and the second of 89%. The Tiny YOLO-V3 is actually faster and runs at 8.71fps while the other model only manages 2.96fps. Since the end application is real-time object detection, the second model is the clear choice.

Both Wang and Liu (2021) and Magalhães et al. (2021) use object detection in the context of tomato crops. In the first, diseases are detected in tomato leaves (Fig. 2.2a). In the second, tomato is detected at all life cycle stage (Fig. 2.2b). In this last work, a variety of SSD and YOLO architectures are compared. The results show that compared to SSD Inception v2, SSD ResNet 50, SSD ResNet 101 and YOLOv4 Tiny, the MobileNet v2 has the best performance. Also, this article discovers the best confidence to increase Average Precision and F1 score for each network.

Aguiar et al. (2020) detects the trunks of vineyards using Deep Learning, Fig. 2.2d. In this



(a) Tomato leaf disease detection, from Wang and Liu.

(b) Tomato detection, from Magalhães et al.

(c) Fescue detection, from Hennessy et al..

(d) Vine trunk detection, from Aguiar et al.

(e) Fruit detection for counting, from Vasconez et al.

(f) Leadf detection for counting, from Buzzy et al.

Figure 2.2: Various uses of object detection using Deep learning in Agriculture.

work, different CNNs were evaluated for both NVIDIA Jetson Nano and Google USB Accelerator to make a comparison in terms of average precision and runtime. The latter showed both better average precision and runtime performance. However, the former showed more compatible operations, which allow more CNNs to be supported.

Table 2.1 shows the results of all previously reviewed papers on object detection with Deep Learning for a simple comparison in terms of AP and F1 score, if provided by the corresponding author.

Table 2.1: Results from all deep learning object detection reviewed papers in terms of Average Precision (AP) or Mean Average Precision (mAP) and F1-score.

Refference	Purpose	Results
Hennessy et al.	Weeds detection	AP: 75.83% and 75.46% F1: 0.62 and 0.62 for Yolo-V3 and Tiny Yolo-V3, respectively.
Ahmad et al.	Weeds detection	AP: 89.89% for Yolo-V3.
Buzzy et al.	Leaf detection and count	AP: 58.30% and 60% F1: 0.94 and 0.9 for Tiny Yolo-V3 and Faster RCNN, respectively.
Wang and Liu	Tomato leaf diseases detection	AP: 86.6%,84.1% and 87.8% F1: 0.89,0.87 and 0.90 for Mobilenet-V2, Tiny Yolo-V3 and Faster RCNN, respectively.
Chen et al.	Fruit tree pests detection	mAP: 93% and 89% for Yolo-V3 and Tiny Yolo-V3, respectively.
Magalhães et al.	Tomato detection	mAP: 51.46% and 46.62% F1: 0.66 and 0.59 for Mobilenet-V2 and Resnet-50, respectively.
Vasconez et al.	Fruit detection and count	AP: 66% and 90% for Mobilenet-V1 and Faster RCNN , respectively.
Aguiar et al.	Vine trunk detection	AP: 49.74%, 52.98% and 32.56% F1: 0.61, 0.59 and 0.51 for Mobilenet-V1, Mobilenet-V2 and Tiny Yolo-V3, respectively.

2.1.2 Transfer Learning

To achieve object recognition with Deep Learning, a CNN must be trained, which usually requires many iterations and a large dataset. Transfer learning is a technique that takes a previously trained model and uses it as is for a similar or different purpose or retrains either the final layers or the entire model with a new dataset, taking into account the already established layers and weights trained with a previous dataset. This process is usually referred to as fine-tuning. In this subsection, we review the performance of this training method compared to full network training.

Aguiar et al. (2021) propose an object detection CNN for the purpose of recognizing grapevine stems. In this work, both the training from scratch and the fine-tuned training are evaluated. The results show that for the same dataset (VineSet), the fine-tuned training can achieve comparable results to the training from scratch with half the iterations. The results can be viewed in Table 2.2.

Ramdan et al. (2020) makes a comparison between training from scratch, transfer learning and fine tuning benchmarking for tea leaf disease detection. The results show, similar to Aguiar et al.

Table 2.2: Performance comparison, in AP (%) and F1 scores of trained models using VineSet, is performed by fine-tuning and training from scratch with two different numbers of training epochs, table by Aguiar et al.

CNN	**From scratch**(50k)		**From scratch**(100k)		**Fine tuned**(50k)	
	AP(%)	F1	AP(%)	F1	AP(%)	F1
SSD MobileNet-V1	68.44	0.685	85.93	0.834	84.16	0.841
SSD MobileNet-V2	60.44	0.639	83.70	0.812	83.01	0.808
SSD Inception-V2	58.05	0.658	76.77	0.849	75.78	0.848

(2021), that fine-tuning training outperforms training from scratch (see Table 2.3). Moreover, the results also show that fine-tuning the entire network is always beneficial in terms of performance.

Table 2.3: Comparison of accuracy (%) on test set of various training schemes, Table from Ramdan et al.

CNN	**From scratch**	**Partially fine tuned**	**Fully fine tuned**
ResNet	73.33	19.73	94.05
VGGNet	84.86	86.04	91.26
Xception	76.85	56.49	91.71

2.2 Vanishing Point detection

A vanishing point is the intersection of parallel lines in three-dimensional space when represented in two-dimensional space. As can be seen in Figure 2.3a, it is possible to see that the lines are parallel when we look at them from the perspective of the "top view". When the image is viewed from inside the vineyard, Figure 2.3b, the two vegetation lines converge to a single point called the vanishing point (VP). In this dissertation an estimate of the vanishing point is used to guide a robot. For the purpose of the study and the context, some methods for vanishing point estimation are presented in this section. Since vanishing point estimation is an image processing problem, first, more traditional methods are discussed, and second, since problems based on vision seem to be solved by machine learning, vanishing point estimation using machine learning techniques will be reviewed in the second subsection.

(a) View from above of Qta. da Aveleda vines.

(b) Desired result from vanishing point extraction on Qta. da Aveleda vines.

Figure 2.3: Vanishing Point detection.

2.2.1 Conventional Vanishing Point detection

Both García-Faura et al. (2019) and Zhou et al. (2017) propose an algorithm that estimates the vanishing point using edge detection, followed by a RANSAC-based algorithm for multiple model estimation and classification. Both of them use a contour detection proposed by Arbeláez et al. (2011), which combines local and global information to detect contours and has the addition of distinguishing the relevance of edges, unlike others such as the most commonly used Canny Edge Detection. The main difference between them is in the grouping part of the edges. Zhou et al. (2017) uses a J-link, while García-Faura et al. (2019) proposes a T-link. Although they are really similar since they are both RANSAC-based, the results presented in García-Faura et al. (2019) shows that the García-Faura et al. (2019) approach present best results as seen in Table 2.4.

Table 2.4: Average estimation error (with a 95% confidence interval), Table from García-Faura et al.

Reference	**AVA dataset**	**Flickr dataset**
García-Faura et al. (2019)	0.345021 ± 0.022162	0.187300 ± 0.021993
Zhou et al. (2017)	0.291836 ± 0.020709	0.161252 ± 0.020075

In Kondo et al. (2017) is proposed a road edge detection using hough transform to extract the lines and a Fast M-estimator to estimate the vanishing point. In the Table 2.5 is possible to observe the results compared to least-squares estimation, as we can see there is a significant improvement and overall good performance expect in the evening situation which has the least illumination. The rate is given by the success or non-success of the vanishing point estimate. In this case, a detection is considered successful if a vanishing point is successfully identified within a radius of 10px.

Table 2.5: Rate of detection by situations (%), Table from Kondo et al.

Situation	**least-squares**	**Fast M-estimation**
Daytime	63	82
Evening	24	28
Night	79	91
Tunnel	85	94
Average	62.8	73.8

2.2.2 Deep Learning Vanishing Point detection

Chang et al. (2018) proposes an AlexNet, with some minor modifications to the two last layers. The vanishing point detection is treated as a CNN classification problem, which means that since the number of outputs of the network is 225, these are the possible locations for the vanishing point. The supervised training is made using Mat-ConvNet toolkit using over 1 Million images collected from google street view (790,069 for training and 263,356 for testing). Each image of the data set has a vanishing point location label, out of 255 possible ones.

Liu et al. (2020) presents D-VPnet, a single-shot deep CNN-base network with a MobileNet as its backbone, for the detection of dominant VPs in natural scenes. For the purpose of training the whole network the Parallel Line base Vanishing Point (PLVP) dataset was used. For evaluation it utilizes a metric that measures the consistency between the detected edges and vanishing point hypothesis.

In Han et al. (2020) a vanishing point detection in orchards is proposed using a CNN. The algorithm proposed is a sliding window that is running along the image and inputs into the cnn the output of the network is the the probability of path or tree for each window. From the border of path and tree classifications a line is created for each row. From the intersection of the two row lines results the vanishing point. To train the network a data set was created from video taken in the orchard, resulting in 3000 images for training, 1000 for validation and 800 for testing.

In Liu et al. (2021) detects vanishing point for unstructured roads using a modified HRNet as the backbone for features extraction, a multiscale heatmap supervised learning is employed to increase the accuracy of vanishing point estimation and lasty using the strategy of coordinate regression a high-precision vanishing point coordinates are obtained. The coordinate regression module used in this work is based on YOLO. This work presents an unstructured roads dataset

(URVP), labeling 5255 images from 10000 unstructured roads obtained using flickr and Google image tools. The proposed metric is the difference from the position of the estimated vanishing point to the ground truth, divided by diagonal size of the image.

Table 2.6 shows the results of all previously reviewed papers on Vanishing point detection using Deep Learning approaches.

Table 2.6: Results for the discussed Deep learning Vanishing point detection approaches.

Reference	Enviorment	Proposed training Data-Set	Results
Chang et al. (2018)	Roads	1,053,425 images from google street view	99% accuracy in recovering the horizon line and 92% in locating the vanishing point within a ±5-degree range
Liu et al. (2020)	Natural Scenes	4,382 images from PLVP created by the authors	method achieved a consistency error of less than or equal to 5 in the proposed evaluation metrix, in 98.69% of the images. The author claims to outperform Zhou et al..
Han et al. (2020)	Orchard	4800 Images taken from video recording of orchard rows	the classification accuracies(CA) increased to 0.98% and 0.96% in training and validation, and the result of the classification with the test set showed a CA of 0.92%
Liu et al. (2021)	Unstructured Roads	5355 labeled images from URVP created by the authors	The author achieves a 0.034875 mean error on the proposed metric.

2.3 Autonomous navigation between rows

To enable a robot to be able to autonomously navigate between rows, the most relevant aspects of guidance are the row following and change aspects. In this section, approaches to these problems are reviewed and different methods for obtaining the position relative to the row are presented and also the results are shown for a latter comparison with the method developed in this dissertation

2.3.1 Laser Scanner

Laser Scanner guidance approach, usually consists of an Extended Kalman Filter (EKF) that uses a laser scanner or Light Detection And Ranging (LIDAR) for the update state and a kinematic model for the prediction, as described by Riggio et al. (2018), Costley and Christensen (2020), Bergerman et al. (2015). Although they all use a similar configuration of the Kalman filter, they vary the sensors used to estimate the position in the prediction state.

For the guidance between row there are various approaches. Riggio et al. (2018) propose a system that uses the laser scanner to detect the distance from the robot to the right row of vines, for the purpose of creating a and following a line using a pure pursuit controller. Similarly, Costley and Christensen (2020) follows one row but also makes use of a Global Positioning System (GPS)-aided system. In this work a comparison is made between the use of only dead-reckoning and with LiDAR in no GPS signal location. Bergerman et al. (2015) proposes a system that detects the two rows and creates a center line between the rows of the orchard.

The row change is done in a similar way in Riggio et al. (2018) and Bergerman et al. (2015), using only odometry. This is only possible because it is assumed that the distance to be travelled is short and the propagated error is not relevant. As can be seen in the Table 2.7 that shows all results, this method is successful in switching rows.

One of the most common problems with laser scanner guidance is the presence of tall weeds in the path or hanging branches, which can lead the robot to follow a wrong path. Since the laser scanner cannot distinguish what is feature or noise, it is imperative to find a way to filter out the noise. Riggio et al. (2018) introduces a method that removes points that are considered noise since they deviate from the expected location of the vine stem as seen in Figure 2.4. Although the method has good results, it relies on knowledge of the distance between the vine stems, which is not possible in all cases. In Bergerman et al. (2015) was developed an iterative algorithm that can filter the "outliers" by recalculating the parallel lines in each iteration until the mean of the squares of the point- to-line distance is smaller than a defined threshold or number of iterations as seen in Figure 2.5.

Table 2.7 shows the results of all previously reviewed papers on laser scanner guidance approaches.

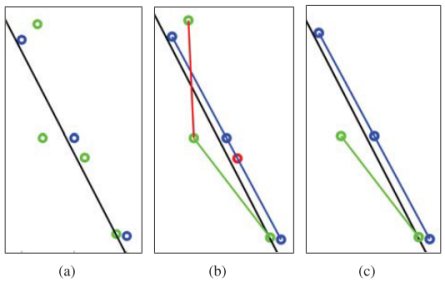


Figure 2.4: Blue points, lines are the considered points and green and red are the rejected and the black line is the predicted position of the row,Image from Riggio et al.

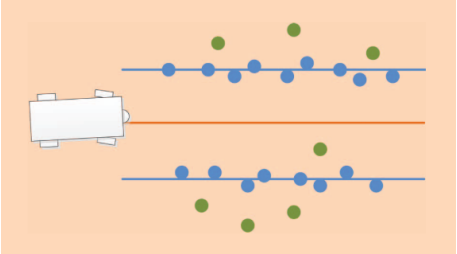


Figure 2.5: Orange line is the center line, green points are the outliers and blue points are the ones used to calculate the center line, image from Bergerman et al.

Table 2.7: Laser scanner based guidance methods results.

Reference	Type of crop	Results
Riggio et al.	Vineyard	Follows the simulated vineyard in ROS at the targeted distance. It is able to navigate a real vineyard (including executing u turns) with a mean value of the error of 0.05 m and its value is within 0.1 m more than 93% of the time.
Costley and Christensen	Orchard and Vineyard	Results show a 90.49% X posistion estimation improvement when using the LiDAR vs when using only odometry.
Bergerman et al.	Orchard	The vehicle has traversed over 350 km in research and commercial orchards and nurseries. The system has a maximum error of 0.5 m laterally and 1% of the row length longitudinally. Is able to detect obstacles such as people and bins. Has reliable speed control from 0.05 to 2 m/s.

2.3.2 Beacon Trilateration

Beacon localization estimates the position based on Received Signal Strength Indication (RSSI), i.e., the farther away the robot is, the weaker the signal becomes. Taking advantage of this property, it is possible to derive the robot position relative to the beacons , i.e. using trilateration.

Reiser et al. (2017) proposes a beacon mapping system for guidance. First, the robot followed a predefined path to determine the locations of the sensor nodes based on their RSSI. Subsequently, the recorded and georeferenced RSSI data were evaluated and mapped. Based on the evaluated node locations, an optimised route was generated. A laser scanner was also used for obstacle detection and avoidance. The results of the beacon positioning using RSSI showed a deviation of 0.6 m.

A Bluetooth system with multiple antennas is proposed by Reis et al. (2018), which relies on RSSI to provide range estimates for each set of antennas. Guidance is provided by relative

position to the beacons relative to a map. For map generation, data was recorded not only from the beacons but also from a Posyx system and odometry. The results show that the application of a Extended Kalman filter reduces the standard deviation by 25%. The results also show that the orientation and position of the antenna affects the performance. Using a specific orientation and position resulted in a lower mean absolute error of 1.35m.

Results from previously described Beacon based guidance methods can be reviewed in Table 2.8.

Table 2.8: Beacon based guidance methods results.

Reference	Type of crop	Results
Reiser et al.	Vineyard	Trilateration showed better location results than the one provided by DGNSS data at all nodes.The best achieved location showed a deviation with DGNSS of 1.2 m and with RSSI trilateration of 0.6 m compared to the actual position.
Reis et al.	Vineyard	Relative to only odometry position estimation, the system applying the EKF presented a 25% decrease in standard deviation. There was also concluded that distance from beacon can be obtained with a mean absolute error of 1,35m.

2.3.3 Vision-Based

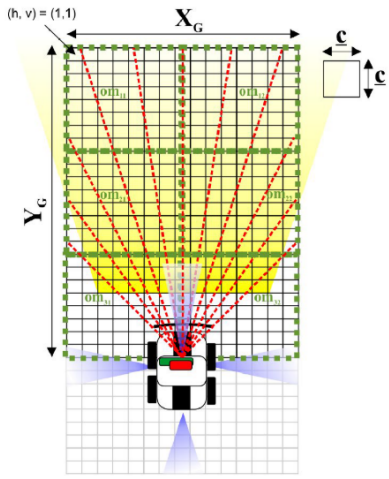
Image processing based guidance is a cheaper and more adaptable solution, but introduces high complexity due to the large amount of input information and the additional problem of occlusion, illumination variation, and recording equipment noise. All this complexity can lead to high processing time. For these reasons, there are several approaches to this problem.

2.3.3.1 Stereo Vision

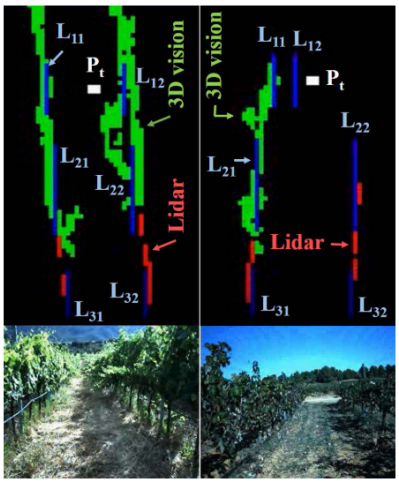
Stereo vision maps the information of depth obtained by the stereoscopic vision from the cameras into distance points (occupation grid map) or encodes the information of distance in the image (disparity map).

The use of occupation grid maps for navigation is proposed by Rovira-Mas et al. (2021), who developed a stereo vision-based perception system. That generates a 3D point cloud from the stereo camera, which is converted into a density grid for easier interpretation, Fig. 2.6a. The interpretation of the grid has been simplified to a cell with six regions. In this case, the position relative to the rows is not only calculated by estimating the row lines from the occupation grid map, but for more robustness, a LiDAR and an ultrasound sensor also play a role in the control, as shown in Fig. 2.6b, green lines represent the vineyard row estimation obtained from 3D vision, and in red the lidar measurements. As can be seen, the lidar overlays the 3D vision and makes

the system more robust. The results show that the fusion of the three technologies provides more consistent and fail-safe guidance.



(a) Density grid from Rovira-Mas et al.

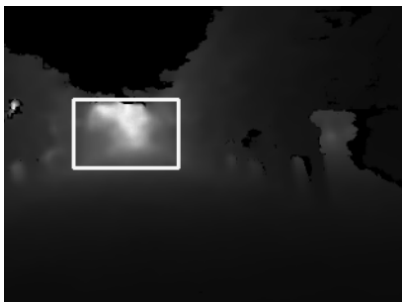


(b) Row estimation from 3D Vision and Lidar measurments from Rovira-Mas et al.

Figure 2.6: Images from Rovira-Mas et al.

The use of a disparity map, Figure 2.7a, for navigation can be found in Aghi et al. (2020). Here, a system is presented that uses a stereo camera to detect the end of the row. Since the camera is mounted in the center of the robot, a proportional controller is able to convert the horizontal pixel distance into angular velocity, which results in the guidance of the robot towards the desired point (Fig. 2.7b). This paper also presents a parallel algorithm to identify the orientation of the robot in three classes (left, center or right) using a deep learning approach. The algorithm is expected to be robust to different lighting and weather conditions by using a training dataset with different weather conditions.

Results from the discussed stereo-vision guidance methods are shown in Table 2.9 .



(a) Disparity map highlighting the furthest point, that is the end of the corridor.

(b) The red square represents the end of the row, and the red line the error from the middle frame to the end of the row.

Figure 2.7: Images from Aghi et al.

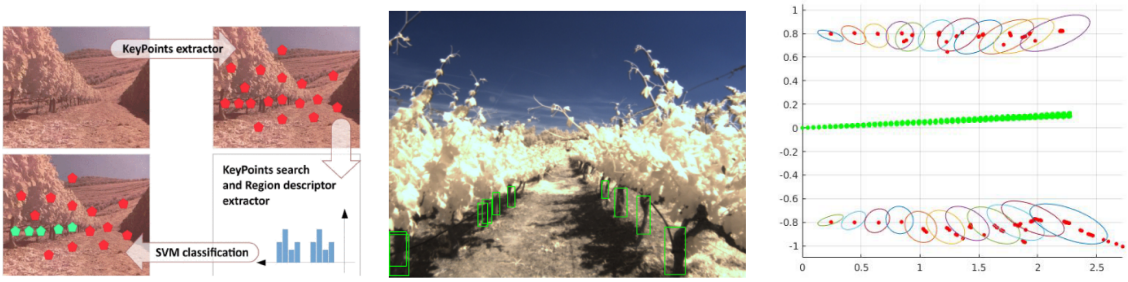
Table 2.9: Stereo-vision based guidance methods results.

Reference	Type of crop	Results
Rovira-Más et al.	Vineyard	The robot was successfully guided along the rows with an average deviation from the center line of 7.6 cm.
Aghi et al.	Vineyard	The system was able to perform an autonomous navigation along the given paths with the low resolution of (640 × 480) in different vineyard rows but similar weather conditions.

2.3.3.2 Monocular Vision

Monocular vision, unlike stereo vision, cannot directly determine distances. Nevertheless, it is possible to determine the relative position of the robot with a single camera, different approaches are reviewed in this subsection. This approach will be later pursued in this dissertation.

The use of Vine trunk detection is proposed by Santos et al. (2021). In this work, a navigation stack for localization and navigation is proposed using LiDAR, beacon localization and vision feature extraction to perform high-level feature detection with the goal of creating a feature-based map. Feature extraction using vision is done in two ways. The first method proposed in this paper is called ViTruDe, which extracts keypoints using convetional image processing methods and detects vine trunks using a support vector machine (SVM) classifier, Fig. 2.8a. The second approach is a CNN-based approach that uses deep learning object detection, Fig. 2.8b. After feature extraction, the features are mapped using the other sensors, and localization is possible by feature localization, Fig. 2.8c.



(a) ViTruDe features extaction (b) CNN-based features extaction (c) Position of the trunks estimation using Extended Kalmann Filter

Figure 2.8: Images from Santos et al.

The authors Sharifi and Chen (2015) make use of a classification technique based on graph partitioning theory the image is separate into terrain, trees and sky classes Figure 2.9b. Then, using the Hough transform the boundaries lines are extracted and used to create the center line as seen in Figure 2.9c. Using the generated path and information from GPS, compass and other tools, the mobile robot computes intermediate points to correct its trajectory and navigate itself in the orchard.

(a) Input Image.

(b) Segmented areas: terrain (brown), trees (green), sky (blue).

(c) boundaries(pink) and centerline(red).

Figure 2.9: Images from Sharifi and Chen.

Han et al. (2020) propose a path score map is generated by using CNN-based classification to separate the image into tree and path classes. From the segmented path area, the boundary lines are determined by linear regression and the intersection of the two lines is the vanishing point indicating the end of the row (Fig. 2.10). For guidance, the system relies on the center of the image, which represents the current direction of the robot, and concludes that the correction to be made is the one resulting from the relative distance to the vanishing point. This guidance method is very relevant and a similar method is used in this dissertation.

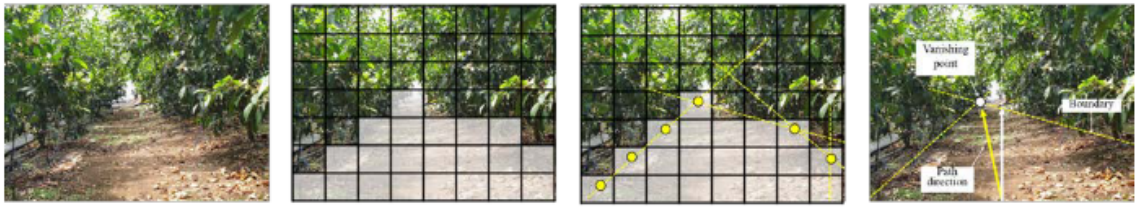
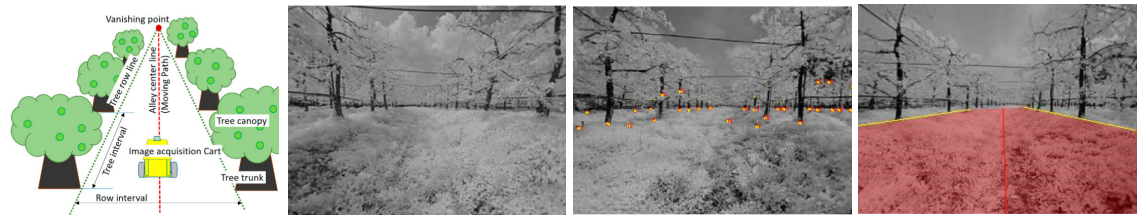


Figure 2.10: Path detection process from Han et al..

Lyu et al. (2018) is applied a Naive Bayesian classification to detect the boundary between trunk and the path, from this features a line is approximated and the center line is estimated making use of the vanishing point, as seen in Fig. 2.11. After the center line estimation the robot adjusts its steering angle in order to follow the center line.



(a) Center line estimation method.

(b) Input Image.

(c) Extracted local lowest points of white segments in ROI area.

(d) Estimated two boundary yellow lines and alley center red line.

Figure 2.11: Images from Lyu et al..

A slightly different approach is taken by Radcliffe et al. (2018), where the authors propose the detection of the boundaries between the trees and the sky. The detection is achieved by segmentation by color as seen in Figure 2.12.

Figure 2.12: extracted border line from the sky and tree row, Image from Radcliffe et al..

Table 2.10 shows the results of all previously reviewed papers on Monocular vision based guidance approaches.

Table 2.10: Monocular vision based guidance methods results.

Reference	**Type of crop**	**Results**
Santos et al. (2021)	Vineyard	A localization and mapping procedure based on high-level visual features which is reliable under GNSS signals blockages is achieved.
Sharifi and Chen	Orchard	the proposed approach classifies the orchard elements in image (ground, trees and sky) very well and is able to extract the border lines between terrain and trees and finally generates the desired central line for the robot to follow.
Han et al.	Orchard	The performance of the proposed framework is comparable to that of other technologies, while its memory cost is lower than those of other deep-learning-based frameworks such as object detection and segmentation. The average errors were 0.056 for lateral distance and 8.1° for angular orientation.
Lyu et al.	Orchard	The results are expressed in degrees different from those presented previously, -0.2° mean error and a standard deviation of 2.5° it is also compared to a human manual decision and showed lower mean error and Standard Deviation.
Radcliffe et al.	Orchard	the vehicle completed an entire orchard row with a maximum deviation of 3.5 cm.

2.3.4 Agricultural environment simulation

The design of a guidance system requires experimentation, not only to easily understand all the variables of the problem, which facilitates the development process, but also to validate the proposed solution. In this section, some simulations of agricultural scenarios are reviewed.

Iqbal et al. (2020) proposes LiDAR guidance for cotton row crops. The effectiveness of the proposed system is evaluated in a simulated agricultural environment in the Gazebo simulator, Fig. 2.13. The results showed that the RMS error was 0.0778 m.

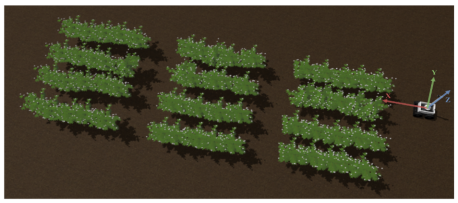


Figure 2.13: Cotton row crop Gazebo simulation, image from Iqbal et al.

Riggio et al. (2018) uses a gazebo simulation to test the proposed algorithm and evaluate its efficiency and robustness before testing it in the real environment. The gazebo simulation is shown in Fig. 2.14a, as can be seen only simple mast geometries are shown, since the intention is for the LiDAR to detect the trunk. The real scenario can be seen in Fig. 2.14b.

(a) Gazebo simulation of vineyard trunks. (b) Real robot in natural enviorment.

Figure 2.14: Images from Riggio et al..

Mengoli et al. (2020) as it uses the Hough transform to extract tree-row lines from an orchard, it needs a visual input. For this purpose the system was tested in a simulated orchard environment using Gazebo, as can be seen in Fig. 2.15.

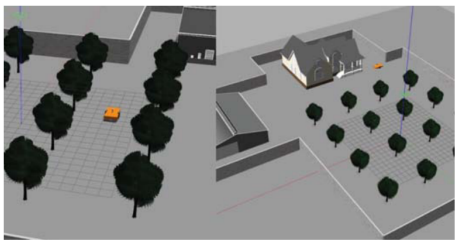
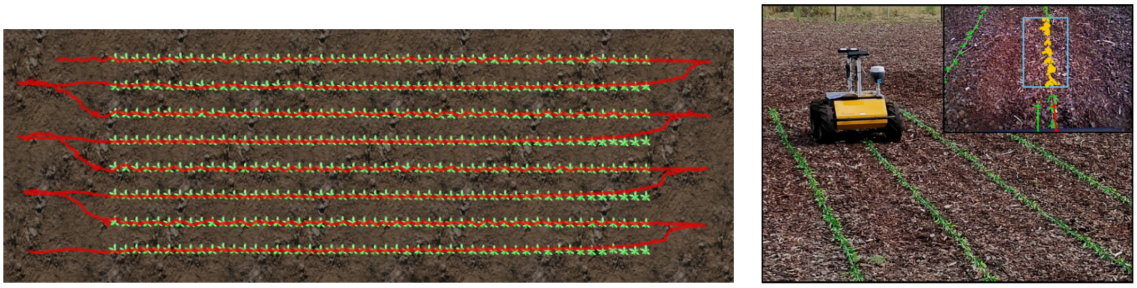


Figure 2.15: Orchard gazebo simulation, image from Mengoli et al..

Ahmadi et al. (2020) proposes a visual servoing controller to guide the robot along a crop. To validate the system, visual input is needed similar to the last example, a Gazebo simulation was created as seen in Fig. 2.16a. In this work, it can also be seen that a real solution was then tested in real scenario, Fig. 2.16b. The simulation results show an average distance to plant rows of 0.8 cm and a standard deviation of 1.15 cm. The real results show a deviation of 4 cm. This increase is due to the fact that the simulation, even if accurate, cannot simulate the randomness of real scenarios.



(a) Gazebo simulation, in red is the robot traveled path.

(b) Real robot implementation.

Figure 2.16: Images from Ahmadi et al..

Chapter 3

Autonomous robot guidance using the vanishing point

In this chapter:

- The Hardware will be introduced and justified;
- The vanishing point estimation approach will be presented;
- The guidance approach will be shown.

Fig. 3.1 describes the visual steering approach. from acquisition to guidance.

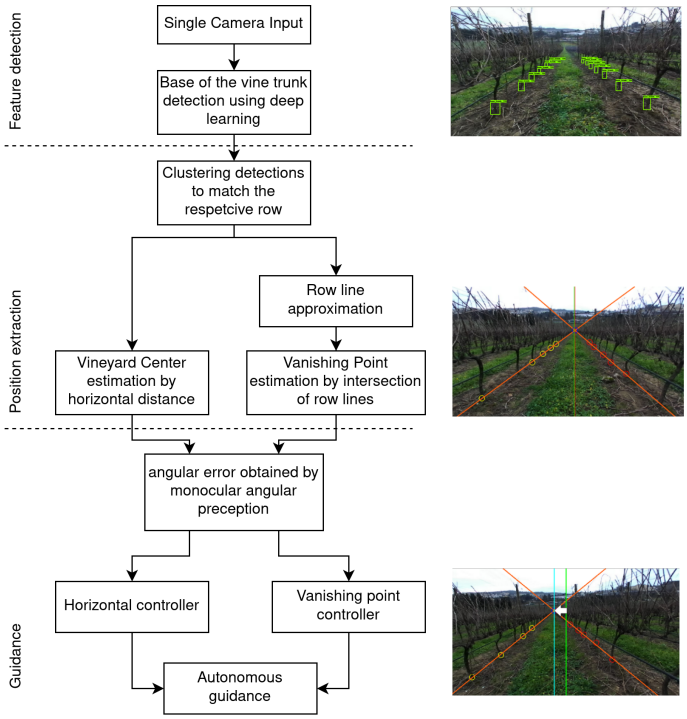


Figure 3.1: Flow chart of the visual steering system.

3.1 Hardware

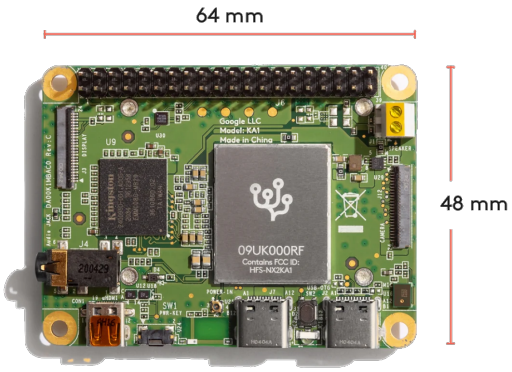


Figure 3.2: Google coral Dev Board Mini from Google.

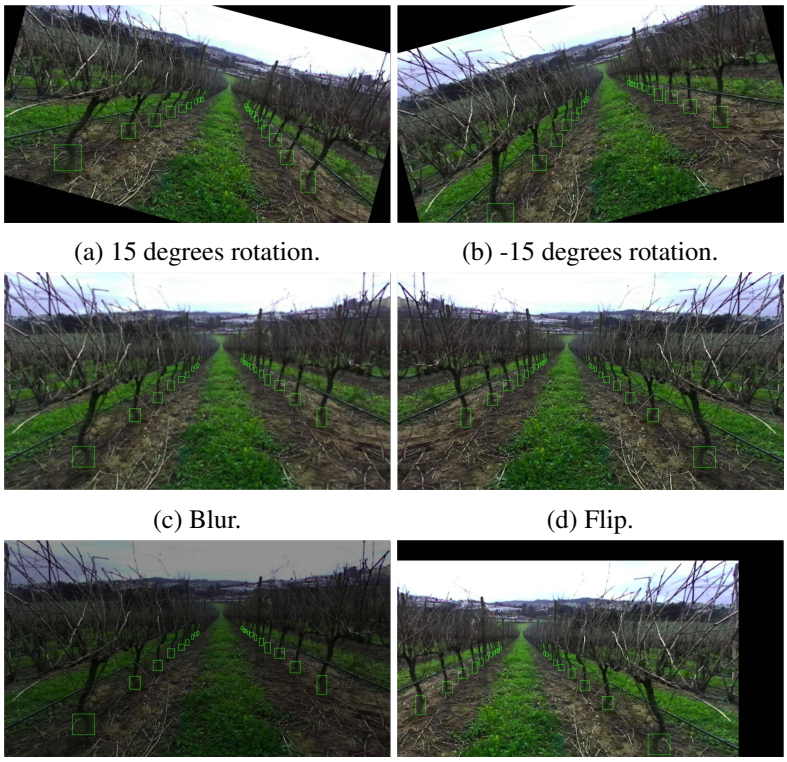
Considering the previously described requirements in the objectives (section 1.3), the hardware chosen is a Google Coral Dev Board Mini (Fig. 3.2) that has a Accelerator Module capable of performing fast machine learning inference at the edge (4 trillion operations per second), avoiding the need for cloud computing and allowing inference to be performed in real time and locally. This module is placed in a single board computer with a quad-core CPU (MediaTek 8167s), 2 GB RAM, 8 GB flash memory, a 24-pin dedicated connector for a MIPI-CSI2 camera and runs on Debian Linux. Thus, this device is not only able to run the inference algorithm, but also the visual steering. To do so, it receives images from a camera and uses the ROS2 framework to easily exchange between the distributed algorithms. In addition, everything previously described is possible on a small form factor device with low consumption.

3.2 Vanishing point detection

3.2.1 Trunk detection

To obtain the vine lines, the position of the vine trunks is determined. For this purpose, various neural networks are trained and then compared for object detection. The selected networks were MobileNet-V1 (Howard et al., 2017), MobileNet-V2 (Sandler et al., 2018), MobileDets (Xiong et al., 2020) and Resnet50 (He et al., 2016). Since the Google Coral Edge TPU has certain limitations in terms of compatible operations, not all NN architectures are ideal. Therefore, the selection was conditioned by already available trained networks supplied by Google and TensorFlow, which the TPU supports. To make the neural networks able to recognise the base of the trunks a dataset must be created. The dataset created consists of the annotation of 604 vineyard images recorded on-board of our agricultural robot (Santos et al. (2020)) in two different stages of the year (summer and winter), Fig. 4.2. The annotation resulted in 8708 base trunk annotations (3278 in summer, 5430 in winter). In the summer , vineyard have more vegetation, and thus, the base of the vine trunks is sometimes occluded. For this reason, the set of images taken in the summer have fewer annotated trunks in comparison with the winter ones. To increase the size of the

dataset, an augmentation procedure was performed (Figure 3.3), which consisted in applying different transformations to the already annotated dataset, namely: A 15 and -15 rotation was applied to the images to simulate the irregularity of the vineyards, a varying random multiplication was performed to vary the brightness of the images, a random blur was applied, a flip transformation was also implemented and also a random combination of transformations selecting three out of seven transformations (scale, angle, translation, flip, multiplication, blur and noise) and applying them in random order. The final dataset consisted of 4221 labelled images.



(a) 15 degrees rotation. (b) -15 degrees rotation.

(c) Blur. (d) Flip.

(e) Multiply. (f) Random combination.

Figure 3.3: Set of augmentation operations performed to create the final dataset.

The training method used was transfer learning, a technique that takes a previously trained model and re-trains either the final layers or the entire model with a new dataset considering the already established layers and weights trained by a previous dataset. In this case, the selected models were pre-trained using the COCO dataset, an object detection dataset trained to recognise 90 objects. The entire model was re-trained with the new base trunks detection dataset. During the training, an evaluation has to be performed for validation. For this purpose, the created dataset was divided as follows: 70% for the training, 15% for the validation and 15% for the benchmarking after the training, this last part will be explained later in Section 4.1.
To use the trained model in the Coral dev board mini, it must be fully quantized to 8-bit precision and compiled into a file compatible with the Edge TPU. To do this, the dataset that was in PASCAL VOC 1.1 was converted to TFRecord, TensorFlow's proprietary binary storage format.

Binary data takes up less space on disk, requires less time to copy, and therefore makes the re-training process more efficient. Next, all models were trained using Google Colaboratory and the TensorFlow framework. If possible, the model was quantized during training, which is often better for model accuracy. Finally, the model must be compiled into a file compatible with the Edge TPU, as described previously. For this, the Edge TPU compiler is used, since this compiler requires TensorFlow Lite, which is a version optimized for mobile and edge devices, the model trained with TensorFlow is converted to TensorFlow Lite and then compiled. The previously described workflow can be observed in Fig. 3.4.

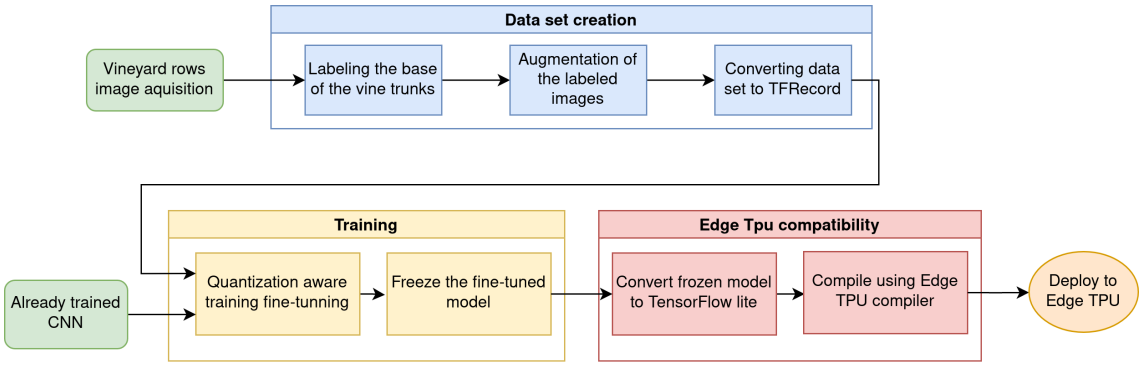


Figure 3.4: Workflow from data acquisition to deploying a fine-tuned CNN to the edge TPU.

3.2.2 Clustering

In order to assign each trunk with one vineyard row, since the detections do not know to which row a point belongs, a clustering procedure was implemented (Algorithm 1). The clustering first sorts by Y position the detection array given from an inference node. Since the array is sorted by Y position, it can be assumed that when the array is parsed, the largest X-distance between two consecutive points belongs to the ones closest to the robot. The result of the clustering algorithm is shown in Fig. 3.5.

Figure 3.5: Clustering step, the blue line is the center of the image, yellow and red circles represent left and right row respectively. The blue points represent the closest detections to the robot.

Algorithm 1: Clustering

Input : detection array : $trunk\_detections$, which contains the x and y coordinates for the center of the bounding box(BB).

Result: two detection arrays : $row\_detections_{Left}$ and $row\_detections_{Left}$, with the respective x and y coordinates for each row.

L_d : largest distance between consecutive points

X_p,Y_p : X and Y coordinates for the center of the BB, respectively

ΔX_d : Variation of X coordinates between consecutive positions of $trunk\_detections$

$\Delta X_{left}(X_p)$ and $\Delta X_{right}(X_p)$: Variation of X coordinates between last position of $row\_detections_{Left}$ or $row\_detections_{Right}$ and a certain X_p

$trunk\_detections$.sort(by Y_p);

Finding Largest consecutive distance:

foreach Det_i *in* $trunk\_detections$ **do**

if $\Delta X_d > L_d$ **then**

  $L_d = \Delta X_d$;

  if $Det_i.X_p < Det_{i-1}.X_p$ **then**

    $row\_detections_{Left0}$=Det_i;

    $row\_detections_{Right0}$=Det_{i-1};

  else

    $row\_detections_{Left0}$=Det_{i-1};

    $row\_detections_{Right0}$=Det_i;

  end

end

;

foreach Det_i *in* $trunk\_detections$ **do** **if** $X_{left}(Det_i.X_p) < X_{left}(Det_i.X_p)$ **then**

  $row\_detections_{Right}$.append(Det_i);

else

  $row\_detections_{Left}$.append(Det_i);

end

;

3.2.3 Row line approximation

After obtaining the two clusters of base trunk detections, a line must be fitted for each row in order to find the vanishing point later. To do this, a least squares approximation is performed. Given the center of the bounding box of the detection (x, y) and the function F (equation 3.1), where $(\phi_1(x), \phi_2(x), ...\phi_k(x))$ are given functions and coefficients $(c_1, c_2, ..., c_k)$ are the parameters of the function F.

$$F(x; c_1, c_2, ..., c_k) = (c_1 \cdot \phi_1(x) + ... + c_k \cdot \phi_k(x)) \tag{3.1}$$

Since the function to be approximated is a line, F is written as the line equation:

$$F(x;c_1,c_2) = c_1 + c_2 \cdot x \tag{3.2}$$

The least squares method consists in finding the coefficients $(c_1, c_2, ..., c_k)$ for some quantity (q) that minimizes the square of the distance (d) between a point and its approximation, equation 3.3.

$$q(c1,c2) = \sum d^2 = \sum_{i=1}^{n} [y_i - (c_1 + c_2 \cdot x_i)]^2 \tag{3.3}$$

A minimum is found when the following condition is met :

$$\frac{\partial q}{\partial c_j} = 0, j = 1,2,...,k; \tag{3.4}$$

Then find the partial derivative for each coefficient:

$$\frac{\partial q}{\partial c_1} = c_1 \cdot \sum_{i=1}^{n} 1 + c2 \cdot \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} y_i \tag{3.5}$$

$$\frac{\partial q}{\partial c_2} = c_1 \cdot \sum_{i=1}^{n} x_i + c2 \cdot \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} x_i \cdot y_i \tag{3.6}$$

And equal them to zero to find the minimum condition:

$$c_1 \cdot \sum_{i=1}^{n} 1 + c2 \cdot \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i \tag{3.7}$$

$$c_1 \cdot \sum_{i=1}^{n} x_i + c2 \cdot \sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} x_i \cdot y_i \tag{3.8}$$

Finally since it is known that c1 is the y zero intersection and c2 is the slope. The system of equations must be solved in order to find the best fit coefficients. This was done by placing c1 in evidence in equation 3.7, obtaining equation 3.9. Then by replacing equation 3.9 in equation 3.8, equation 3.10 is obtained. Now it is possible to compute the slope and then y zero intersect.

$$c_1 = \frac{\sum_{i=1}^{n} y_i - c2 \cdot \sum_{i=1}^{n} x_i}{N} \tag{3.9}$$

$$c_2 = \frac{N \cdot \sum_{i=1}^{n} x_i \cdot y_i - \sum_{i=1}^{n} y_i \cdot \sum_{i=1}^{n} x_i}{N \cdot \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2} \tag{3.10}$$

In Fig. 3.6 is show the the approximated line to each row, represented by an orange line.

3.2.4 Vanishing Point Estimation

After determining the locations of the base of the vines, the clustering algorithm has separated the points for each row and performed the least squares approximation: The last step to obtain the

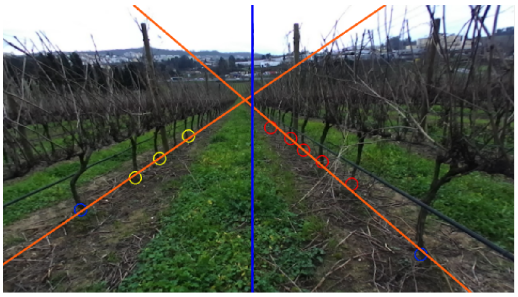


Figure 3.6: Linear approximation step, the blue line is the center of the image, the orange lines represent the best fit calculated for each row.

vanishing point is to intersect the two row lines, the intersection results in a point (the vanishing point). Since the steering is only corrected horizontally, the position of the vanishing point x is as shown in Fig. 3.7.

Figure 3.7: Intersection step, the blue line is the center of the image, the green line indicates the horizontal position of the estimated vanishing point.

3.3 Visual-only guidance

3.3.1 Horizontal distance estimation

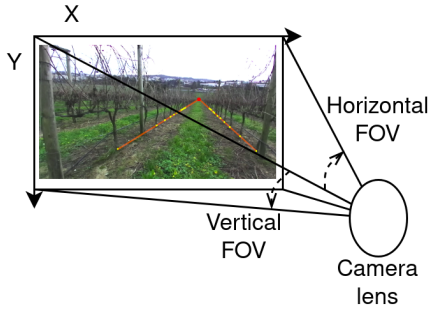
To make the system more robust, the error is determined in two ways:

- Vanishing point estimation;
- Average horizontal distance between the vines.

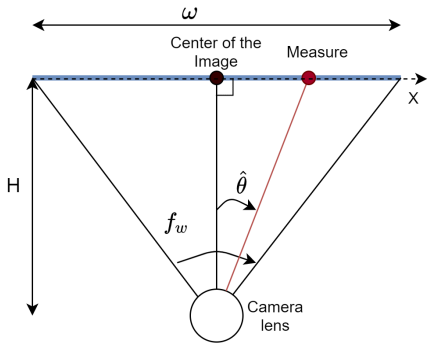
The reason for this second method is that while developing the vanishing point controller, it was observed that if there was a significant deviation from the center, the vanishing point controller was unable to effectively return to center. To ensure that the system always returns to the center, the middle of the vineyard is estimated by averaging the horizontal position of the lower left and right detection points obtained from clustering (blue circles in Fig. 3.8).

Figure 3.8: Horizontal error, green line is the measured average horizontal distance between the selected points (blue circles).

3.3.2 Monocular angular perception



(a) Camera acquisition, vertical and horizontal perspective.



(b) Camera acquisition, only horizontal perspective.

Figure 3.9: Representation of the angle estimation method.

For the guidance system to work, the horizontal distance in pixels between the reference and the measurement must be converted into an angular error so that it can later be the input to an angular velocity controller. To do this, it is necessary to take into account how the image is acquired by the camera, this process is shown in Fig. 3.9. From the observation of Fig. 3.9b, we can see that the angular error ($\hat{\theta}$) can be obtained from the displacement of the measurement towards the center of the image (Δ_x) and distance to the object (H) with the equation 3.11.

$$\hat{\theta} = \arctan(\frac{\Delta_x}{H}) \tag{3.11}$$

$$\tan(\frac{-}{2}) = \frac{\frac{\omega}{2}}{H} \tag{3.12}$$

As the distance to the object is unknown, the angular error is always relative to the center of the image and the adjacent line (H) of $triangle_1$ (center of the image, camera and measure) is the same as that of $triangle_2$ (center of the image, camera and half width of the image). It is interesting to have equation 3.11 dependent of the f_w and not H. For that purpose, equation 3.12, shows the relation between the image horizontal field of view (f_w), the width in pixels (ω) and the distance to the object (H), is placed in order of H resulting in equation 3.13.

$$H = \frac{\omega}{2 \cdot \tan(\frac{f_w}{2})} \tag{3.13}$$

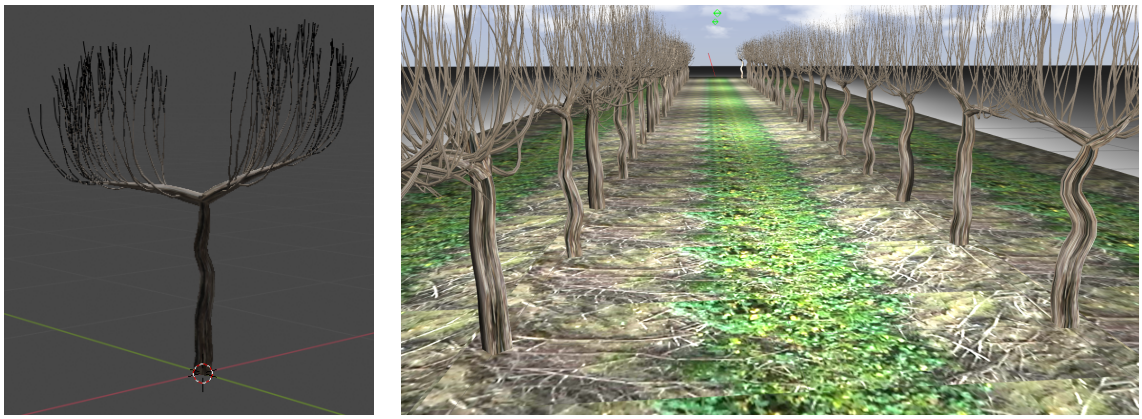
Then substituting H into equation 3.11, equation 3.14 is obtained.

$$\hat{\theta} = \arctan\left(\frac{2 \cdot \Delta_x \cdot \tan\left(\frac{f_w}{2}\right)}{\omega}\right) \tag{3.14}$$

3.4 Simulator

Since the guidance system depends on visual input, an interactive graphical simulation was needed to evaluate its performance. Two scenarios were developed in Gazebo, a simulator that provides the ability to accurately and efficiently simulate robots in complex indoor and outdoor environments. First, a straight line to set and test the system controller in a best-case scenario. Then, a curved vineyard scenario was created to simulate not so straight and complex vineyards.

3.4.1 Straight vineyard



(a) Vine tree model. (b) Gazebo simulation of a straight vineyard.

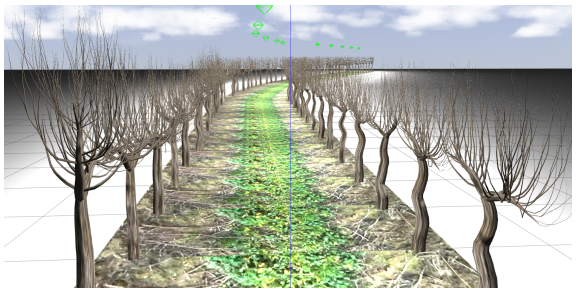
Figure 3.10: Gazebo straight vineyard.

As the system relies on the detection of the base of the vine trunk, the main object to be simulated is the vine. For this purpose, Blender, a 3D design software, was used in conjunction with an add-on program called Modular Trees, which allows the user to enter the tree specification and generate multiple trees based on the defined parameters. An example of the generated tree can be seen in Figure 3.10a. This image also shows the texture of the vine trunk that was applied to the mesh. To create a vineyard, the generated vine meshes and a vineyard terrain were then added to Gazebo as shown in Figure 3.10b.

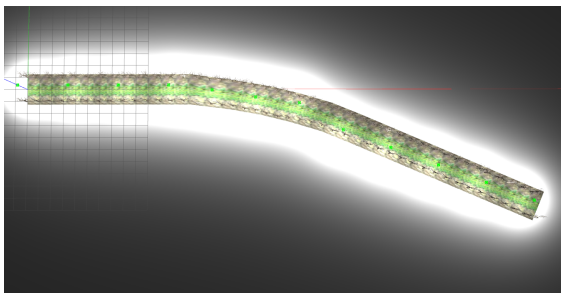
3.4.2 Curved vineyard

(a) Satalite photo of Douro vineyard, with a 10 meter scale.

(b) Mask of of Douro vineyard, with a 10 meter scale.



(c) Gazebo simulation of a curved vineyard.



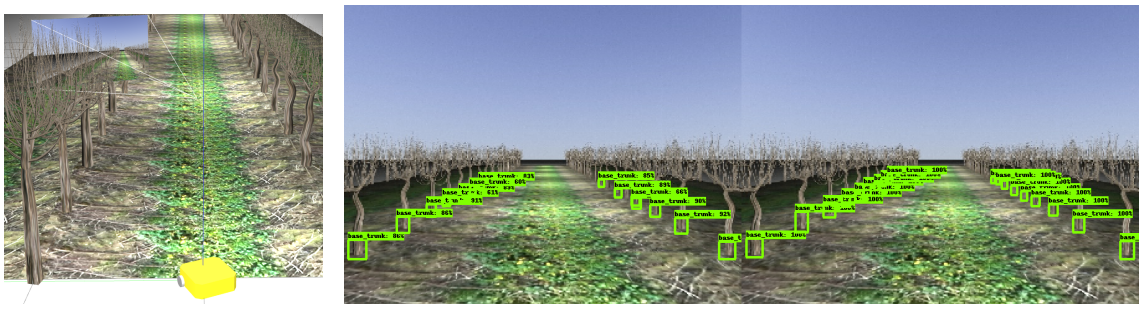
(d) Gazebo simulation of a straight vineyard (up view).

Figure 3.11: Gazebo curved vineyard.

Since the purpose of this second scenario is to mimic a harsh and natural vineyard environment, a satellite image of a vineyard of the Douro region from Portugal was used as reference (Fig. 3.11a). The relevant information of the rows was mapped and then rearranged using the trunks designed for the straight vineyard to follow the real curvature of the vineyard Figs. 3.11c and 3.11d.

3.4.3 Additional steps

A robot with a camera is needed to test the control system, the TurtleBot3 waffle-pi simulation was used for this purpose (Fig. 3.12a) as the platform was readily available, already had a camera plug-in installed, and was compatible with ROS2. Since the system relies on DL object detection to detect the base of the trunks, and although the vine trunks are made to resemble the real ones, when testing with the previously trained models, it was found that the created dataset needed images of the simulator. For that purpose new images were labeled, augmented and added to the previously created dataset. Finally, the MobileDets model was trained. Fig. 3.12 shows the detections for the trained method and the respective labeled image.



(a) TurtleBot3 with a 25º inclination relative to the vines.

(b) Base trunk detection (left portion of the image) against labeled ground truth (right portion of the image).

Figure 3.12: Images from the simulated vineyard in Gazebo.

3.5 Controller

To steer the robot the angular error must be adequately converted to angular speed. To do so, a Proportional Integrative Derivative (PID) controller will be studied, the equation for this controller is:

$$\dot{\theta}(t) = K_p\theta(t) + K_i \int \theta(t)dt + K_d\frac{d\theta}{dt} \tag{3.15}$$

Where θ is the angular error, $\dot{\theta}$ in angular speed; K_p, K_i and K_d are the proportional, integrative and derivative gains, respectively. These gains are later calibrated or set equal to zero, with the associated term deleted if it is not needed.

3.5.1 Discretization

The Controller will be implemented digitally, to do so, equation 3.15 must be in the discrete domain. First, the laplace transform of equation 3.15 was taken resulting in:

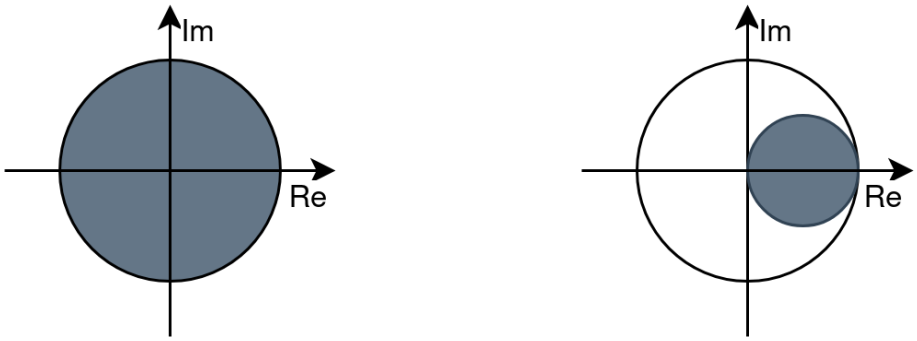
$$\dot{\theta}(s) = K_p\theta(s) + \frac{1}{s}\cdot K_i\cdot\theta(s) + s\cdot K_d\cdot\theta(s) \tag{3.16}$$

Since the control is given by the sum of the three terms (proportional, integrative, and derivative), the equation is divided into three parts for simplicity, where P(s) is the proportional term, I(s) is the integrative term, and D(s) is the derivative. The sum of these three terms is the control signal, equation 3.17.

$$\dot{\theta}(s) = P(s) + I(s) + D(s) \tag{3.17}$$

Then both the billinear (equation 3.18) and backward Euler (equation 3.19) methods are used to place the equation into the discrete domain. At first glance, considering the figure 3.13, it can be seen that by using the billinear transformation the entire left plane is mapped onto the unit circle, whereas with the backward Euler method there is a loss in the mapping. But it will be seen further that this method has distinct advantages over the first.

$$s = \frac{2}{T_s}\frac{z-1}{z+1} \quad (3.18) \qquad s = \frac{z-1}{z \cdot T_s} \quad (3.19)$$



(a) (b)

Figure 3.13: Mapping of half-left-plane of the laplace domain into the Z domain by billinear transform (a) and by backwards Euler (b) methods.

The Z-domain representations of P, I and D terms are show respectivly in equations 3.20, 3.21 and 3.23 for the billinear transformation and equations 3.20, 3.22 and 3.24 for the backwards Euler.

$$P[Z] = K_p \cdot \theta(Z) \quad (3.20)$$

$$I(Z) = \frac{T_s \cdot k_i}{2}\frac{1+z^{-1}}{1-z^{-1}} \cdot \Theta(Z) \quad (3.21) \qquad I(Z) = k_i \cdot \frac{T_s}{1-z^{-1}} \cdot \Theta(Z) \quad (3.22)$$

$$D(Z) = \frac{2 \cdot k_d}{T_s}\frac{1-z^{-1}}{1+z^{-1}} \cdot \Theta(Z) \quad (3.23) \qquad D(Z) = k_d \cdot \frac{1-z^{-1}}{T_s} \cdot \Theta(Z) \quad (3.24)$$

Finally, making use of the Z-transform time shift property : $u[k-1] = U(Z) \cdot z^{-1}$ the differences equations are obtained for the P,I,D terms as shown respectivly in equations 3.25, 3.26 and 3.28 for the billinear transformation and equations 3.25, 3.27 and 3.29 for the backwards Euler.

$$p[k] = K_p * \theta(k) \quad (3.25)$$

$$i[k] = \frac{T_s * k_i}{2} * (\theta[k] + \theta[k-1]) + i[k-1] \quad (3.26) \qquad i[k] = K_i * Ts * \theta[k] + i[k-1] \quad (3.27)$$

$$d[k] = \frac{2 * k_d}{T_s} * (\theta[k] - \theta[k-1]) - d[k-1] \quad (3.28) \qquad d[k] = K_d \frac{\theta[k] - \theta[k-1]}{Ts} \quad (3.29)$$

As can be seen in the difference equation 3.28 obtained by billinear transformation, the previous instance of d is taken into account, leading to an undesirable derivative error integration that can potentially throw the system out of stability. On the other hand, the difference equation for the derivative term obtained by backward Euler does not depend on the past derivative errors. A simulation was performed with the PD controller using billinear and backward Euler, and the results are shown in Fig. 3.14. As can be seen, the error in the billinear case is larger and more

erratic than in the backward Euler case for the same gains. It is also noticeable that the error does not decrease as fast when the sign is changed as in the backward Euler case, which is due to the fact that in the former case the subtraction of the last derivative term takes place.

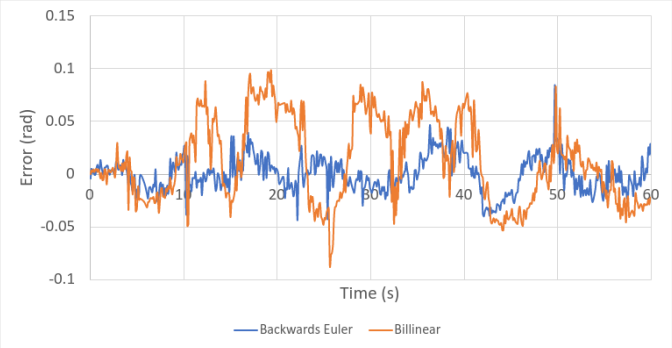


Figure 3.14: Billinear and Backwards Euler PD controller simulation in a straight line, k_p=0.3 and k_d=0.03 for both.

So for the PID controller equation 3.30 is used, this equation is the sum of the three difference equations (3.25,3.27 and 3.29).

$$\dot{\theta}[k] = K_k\theta[k] + K_i * Ts * \theta[k] + i[k-1] + K_d\frac{\theta[k]-\theta[k-1]}{Ts} \tag{3.30}$$

3.5.2 Implementation

For the system to work, a few things need to be considered. First, since the controller only outputs an angular velocity, a linear velocity must be set so that the robot can move forward. For this, a constant velocity of 0.3 m/s is specified, which only applies when the system is receiving trunk detections. Secondly, to avoid large variations or unreachable speeds, an output threshold is set to limit the angular velocity to 0.6 rad/s. Considering that there are cases where the vanishing point cannot be found but horizontal control is still possible, the system is designed to allow both controllers to operate independently, the operation of the controller is shown in the fluxogram from Fig. 3.15. In this figure it can also be seen that when the system detects that the two detected slopes have the same sign, it concludes that it is facing a wall and corrects accordingly.

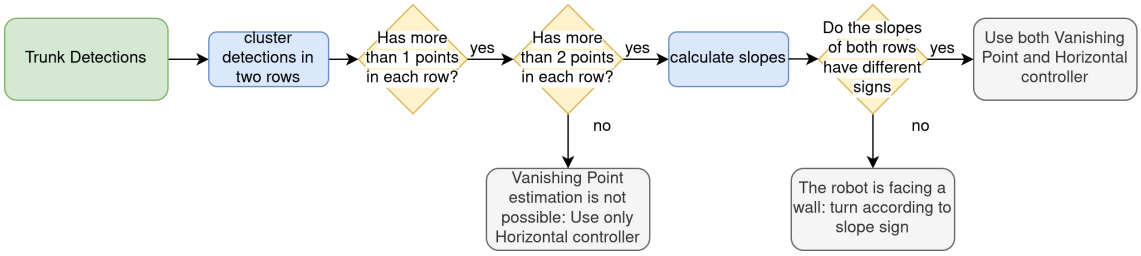


Figure 3.15: Fluxogram of the implemented controller.

3.5.3 Calibration

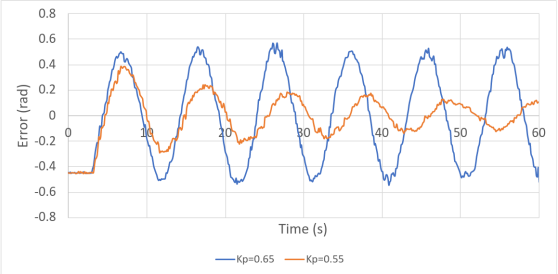
The controller was calibrated by analyzing the step response. For this purpose, the robot was placed in the center of the vineyard simulator with a rotation of 25 degrees (Fig. 3.12a). This allowed the system to see both rows of the vineyard, allowing for control, while also introducing a large error.

Only the vanishing point is considered first to calibrate the proportional gain, and then the resulting gains are tested and adjusted for the horizontal controller if necessary. The proportional gain was adjusted, inspired by the Ziegler and Nichols (1993) calibration method, by increasing the proportional gain (K_p) until the system reached its stability limit, that is the maximum possible actuation before instability, and places the system in an always moving state between a given point. In Fig. 3.16a we can observe two cases where the system oscillates around a point, in orange (K_p=0.55) the oscillations of the system decrease, which means that it is underdamped, while in blue (K_p=0.65) the system maintains the same amplitude at each oscillation, this is the stability limit. Then, the gain is set to half of the gain that caused the stability limit and as seen in Fig. 3.16c to reduce the overshoot the gain was decrease to K_p equal to 0.3. Now for the horizontal controller, the proportional gain was set to the same value as for the vanishing point controller (K_p=0.3), and as shown in Fig. 3.16d, this gain was decreased to 0.25 to reduce the overshoot.

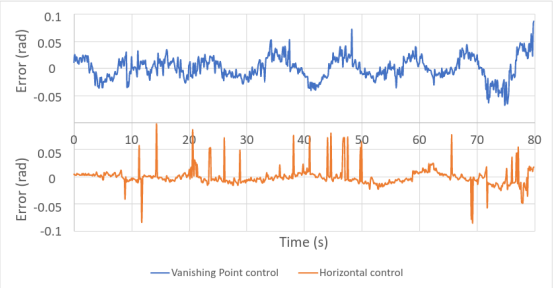
Since for both the VP and the horizontal controller, Figs. 3.16c and 3.16d, no significant steady-state error is observed, the integral part of the controller is turned off so k_i is zero for both controllers.

To further reduce the overshoot of the system without affecting the settling time, according to Li (2007), a derivative term was added. The initial gain of this term for both controllers was set to 10% of the proportional gain and increased until the overshoot was reduced to a minimum and the Root Mean Square (RMS) error reached a minimum. In Figs. 3.16e and 3.16f are the step responses for all tested derivative gains for the vanishing point and the horizontal controllers. Tables 3.1 and 3.2 show the relative percentage decrease in RMS error measured by the only proportional controller for all derivative gains for the step response and in the straight line for the vanishing point and horizontal controllers. The straight line test consists of placing the robot in the middle of the vineyard and letting it steer without additional disturbances to assess whether the derivative term is to erratic. The ideal case is a robot with a fast response, but not too erratic.

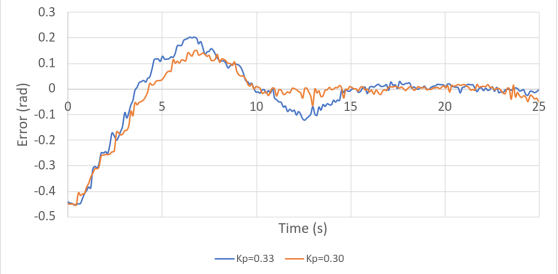
The step response evaluates the rapid response, and the straight line test measures the erraticness. Thus, in the ideal case, the RMS error is smallest for both tests. This has not been verified for either case. In the vanishing point case, considering the table 3.1, the best step response is observed when K_d is equal to 0.15, albeit with a small distance from K_d equal to 0.12. On the other hand, it is clear that for the straight line, the lowest RMS error is recorded when K_d is equal to 0.12, so the gain is set to the latter value. For the horizontal controller, it is easy to conclude from the step response that the selected K_d gain must be 0.115. However, looking at the straight



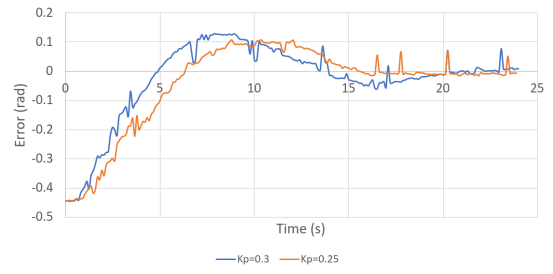
(a) VP_c Proportional gain calibration



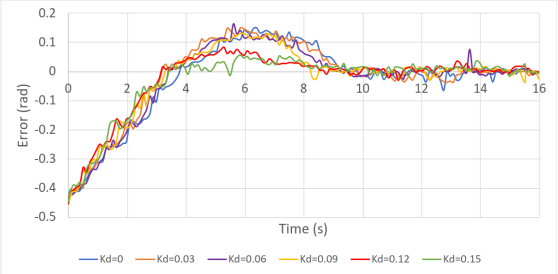
(b) Straight line test.



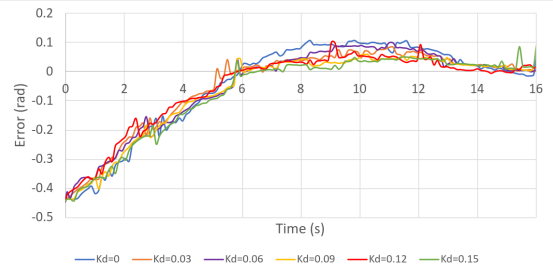
(c) Step response for various VP_c proportional gains.



(d) Step response for various H_c proportional gains.



(e) Step response for various VP_c derivative gains.



(f) Step response for various H_c derivative gains.

Figure 3.16: Tests for the purpose of calibration for the Vanishing Point controller (VP_c) and Horizontal controller (H_c).

line test recorded in table 3.2, it is possible to observe no significant reduction in the error, which means that the derivative term is detrimental for the horizontal controller. This can be explained by considering Fig. 3.16b and comparing the straight line test of the vanishing point control (blue line) with the horizontal control (orange line). It can be seen that the latter has highly fluctuating error peaks that cause the derivative term to increase to a very large value. Therefore, for the horizontal control, the derivative gain (K_d) will be zero. The proposed controller diagram is shown in Fig. 3.17.

Table 3.1: Error decrease in % for different derivative gains for the vanishing point controller, the RMS error decrease is relative to the proportional gain results, 0.139754 and 0.022356 for the step and straight line test, respectively.

	Error decrease in (%) for Derivative Gain of:				
Test scenario	**0.03**	**0.06**	**0.09**	**0.12**	**0.15**
Step	2.51	1.75	8.50	17.61	17.69
Straight line	5.75	14.75	11.06	16.15	3.01

Table 3.2: Error decrease in % for different derivative gains for the horizontal controller, the RMS error decrease is relative to the proportional gain results, 0.161927 and 0.017222 for the step and straight line test, respectively.

	Error decrease in (%) for Derivative Gain of:				
Test scenario	**0.025**	**0.055**	**0.085**	**0.115**	**0.145**
Step	11.70	9.84	7.15	15.45	4.96
Straight line	0.88	-4.18	-2.11	-9.67	-4.31

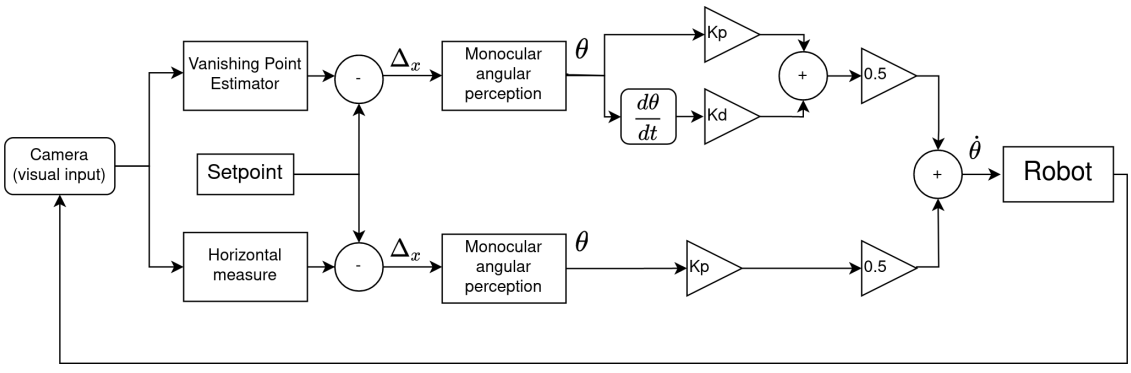


Figure 3.17: Visual steering controller diagram, Δ_x is the displacement of the measurement towards the center of the image discussed in section 3.3.2, θ is the angular error and $\dot{\theta}$ the angular velocity.

Chapter 4

Results

4.1 Methodology

To evaluate the base trunk detection, as referred in Section 3.2.1, a portion of the proposed dataset was reserved to later perform an unbiased benchmark of the detections using the most commonly used metrics (Precision, Recall, Average Precision (AP) and F1 Score) such as reviewed by Padilla et al. (2020). To evaluate the vanishing point estimation, a ground truth was created by manually labeling images extracted from a video recording of the vineyards in the winter and then in the summer. Then, by comparing the horizontal error between the ground truth and the obtained estimate, the error in pixels is obtained. Since the guidance system will deal with angles, it is interesting to evaluate the error in angles. To do that, eq. 3.14 is used to obtain the angular error relative to the center of the image from the ground truth and also for the obtained estimate, then by subtracting both errors, the angular error between the two is obtained. To evaluate the proposed controller for vineyard guidance. First, using the straight line gazebo simulation scenario (section 3.4.1), two tests were conducted. First, all configurations of the controller were tested in a straight line without disturbances, comparing the center position of the vineyard to the robot position topic published by the simulator. Then, to observe the response to potential perturbations, a script was developed to apply a specific angular perturbation to the controller topic at specific time intervals (Fig. 4.6a). Second, using the the curved vineyard (section 3.4.2), was evaluated the distance to the center. This scenario is more complex than in the straight case, since the first follows a natural pattern and cannot be described by a simple mathematical function. Therefore, the simulated robot was manually guided to the center of the vineyard and the x and y positions of the robot were recorded in the simulator and later approximated by a 5th order polynomial function, as shown in Fig. 4.7a, which will serve as ground truth.

4.2 Deep learning analysis

4.2.1 Object detection evaluation metrics

Since the metrics used depend on the terms True Positives (TP), False Positive (FP) and False Negative (FN), the concept of Intersection Over Union (IOU) is used, which, as the name suggests, is the ratio of the intersection of the detected and the ground truth bounding boxes with the respective union ($IOU = \frac{\text{area of overlap}}{\text{area of union}}$) as shown in Fig. 4.1.

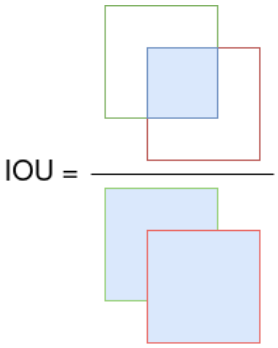


Figure 4.1: IOU visual representation.

Then the value can be compared to a specific threshold t, for example:

- if IOU $\geq$ t then is a TP;
- if IOU $\leq$ t then is a FP;
- if IOU = 0, then is FN.

A brief explanation of the metrics follows. Starting with Precision, equation 4.1, which is the percentage of TP for a given validation threshold, *i.e.*, if the number of FP is low for a given threshold, Precision is high.

$$P = \frac{TP}{TP+FP} = \frac{TP}{\text{all detections}} \tag{4.1}$$

Recall, equation 4.2, is the percentage of TP in agreement with all labeled data, *i.e.*, if the number of FN is high, Recall is low.

$$R = \frac{TP}{TP+FN} = \frac{TP}{\text{all ground truths}} \tag{4.2}$$

Average Precision (AP), equation 4.3 where P is precision and R is recall, differs from Precision in the sense that it does not refer to a specific threshold, *i.e.* Average Precision is calculated for all possible thresholds and is equal to the area under the curve of Precision-Recall curve.

$$AP = \int_0^1 P(R)dR \tag{4.3}$$

The F1 score, equation 4.4, is calculated from Recall and Precision, the higher it is, the higher the Precision and Recall values are.

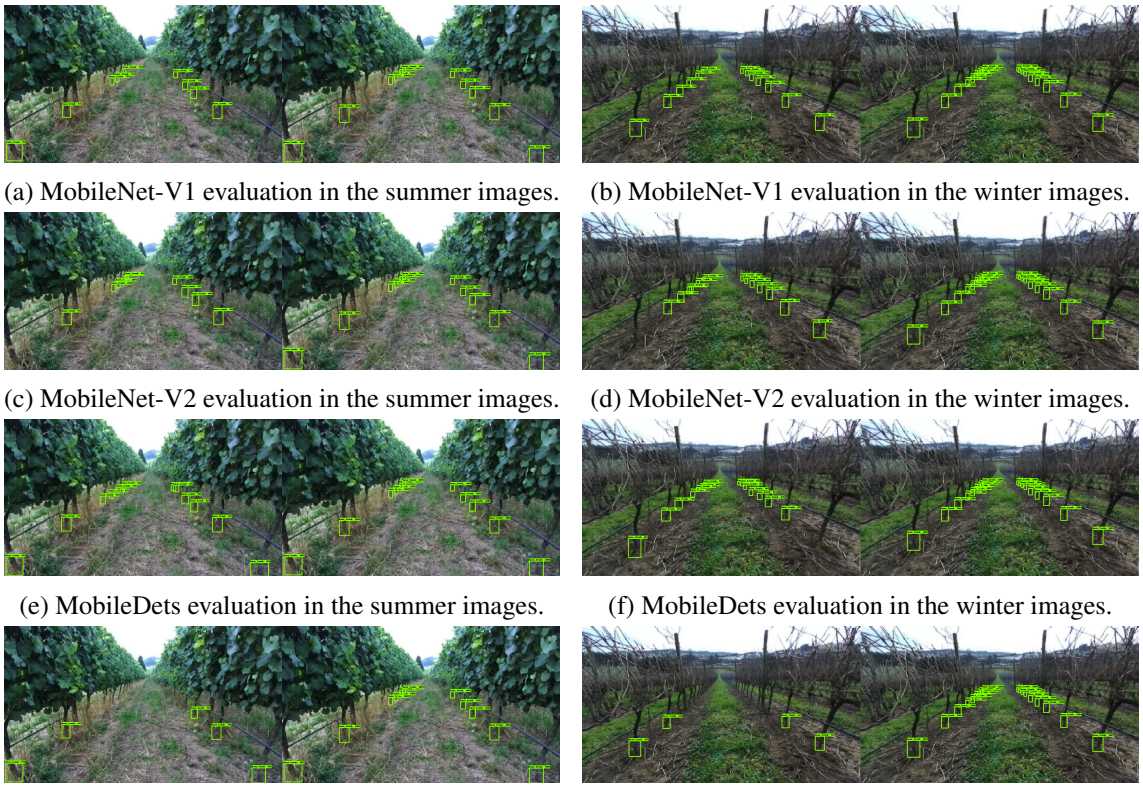
$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})} \tag{4.4}$$

4.2.2 Base trunk detection

The results of all re-trained models can be viewed in Table 4.1, in term of object detection metrics, and in Figs. 4.2a to 4.2h, are shown the detections and respective ground truths. Mobilenet V1 and V2 are very similar. The most noticeable difference can be seen between Figs. 4.2a, 4.2b and Figs. 4.2c, 4.2d, where V2 has fewer detections indicating a higher number of False Negatives, which can also be confirmed by the lower Recall value. Mobiledets is characterized by the large number of FP, as can be seen it has the lowest Precision. Although this number is very high, most of the cases are duplicate detections and unlabeled trunks detected by this model, as can be seen in Figs. 4.2e, 4.2f. Even though the second variant is beneficial, the duplicate detections can be harmful to the vine rows line approximation. From the vanishing point estimation perspective, this model is good because it has a higher number of detections. However, for the same reason, it requires more complexity to ensure that the set of harmful FP can compensate the system. ResNet50 has very assertive results because it has a very small number of FP and thus has the highest Precision. On the other hand, it has a very high number of FN, as shown by the Recall value, which is the lowest of all. So, for this application, it may not be the most interesting solution because, as can be seen in Figs. 4.2g and 4.2h, it has a very low number of detections, which may cause the linear regression algorithm not to approximate the vine line very well.

Table 4.1: Deep Learning-based vanishing point detection evaluation.

Model	**Precision** (%)	**Recall** (%)	**AP** (%)	**F1 Score** (%)
MobileNet-V1	94.55	77.36	77.08	85.10
MobileNet-V2	98.02	70.41	70.25	81.95
MobileDets	88.13	77.46	76.58	82.45
ResNet50	98.85	31.20	31.09	47.43



(a) MobileNet-V1 evaluation in the summer images.
(b) MobileNet-V1 evaluation in the winter images.
(c) MobileNet-V2 evaluation in the summer images.
(d) MobileNet-V2 evaluation in the winter images.
(e) MobileDets evaluation in the summer images.
(f) MobileDets evaluation in the winter images.
(g) ResNet50 evaluation in the summer images.
(h) ResNet50 evaluation in the winter images.

Figure 4.2: Base trunk detections, in each image the left and right portions represent the detections and the ground truth respectively.

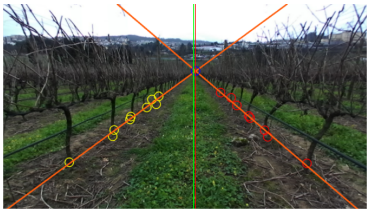
4.3 Vanishing point estimation

Figs. 4.3a to 4.3c and 4.4a to 4.4c show vanishing point estimates by the trained model in winter and summer, respectively. Figs. 4.3d to 4.3f and 4.4d to 4.4f show the histograms of the vanishing point estimation errors compared to the ground truth for winter and summer, respectively. Table 4.2 shows the average error and the maximum and minimum values in degrees by trained model and season. The average error was determined by summing all absolute error values and dividing by the total to evaluate the magnitude of the error rather than its relative position. For the winter annotations, the recorded error is so small that the error associated with the labeling of the images is relevant, which means that a comparison between the different models is not possible, although it also means that all models have good results and the vanishing point estimation algorithm is robust. The vanishing point estimation performed by Chang et al. (2018) was made within a range of ± 5 degrees. All of the previously mentioned models combined with the vine line estimation and intersection outperform the method proposed by Chang et al. (2018), with the largest range recorded in the winter evaluation being ± 2 degrees, even tho having a significantly smaller data set. In contrast, the results for the summer annotations were not so satisfactory. As can be seen in Table 4.2, there is an increase in the average error. Moreover, the number of estimates with an error larger than 10px ($\approx 1°$) is very frequent for MobileNet-V2 and MobileDets, as can be seen in the

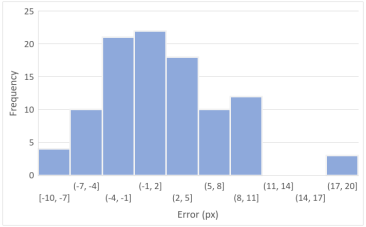
histograms of the error, although, as previously discussed, some annotation errors must be taken into account. Moreover, it can be observed that when comparing the dispersion of the histograms of the vanishing point estimation error from winter (Figs. 4.3d to 4.3f) to the summer (Figs. 4.4d to 4.4f), where the range of values is [-20;20]px and [-100;100]px, respectively, that the latter is significantly larger than the former. The main reason for such deviations in the summer is well represented in Fig. 4.5a, where it can be observed that in the left row there is a large amount of noise caused by tall weeds near the trunk, which consequently causes the line approximation not to converge to the real base of the trunk line. Also in this figure it can be seen that only two points are detected in the right row, although this is not very troublesome in this case, it is beneficial to have as many detections as possible. As mentioned in section 3.5.2, if the vanishing point estimation is not possible when one of the rows has less than two detections, it will be filtered. In Fig. 4.5 some examples of these occurrences are shown. In Fig. 4.5b there are no detections in the left row but four on the right, so in this case the system mistakenly thinks it is facing against a single vine row. In Fig. 4.5d it can be seen that there is a lot of tall grass in the left row, so in this case only one point was detected, unlike the right row which has three points, although as previously discussed, if a row has less than two points, estimation is impossible and this case is filtered. Similarly to the last case, reduced detections due to the occlusion of the base trunks by a loose vine, lead to the filtering of this case. In all the previous cases, it is also important to note that the clustering algorithm successfully separated left and right points, even when there was only one point for a given row. The algorithm as it stands will not provide as robust guidance in the summer as it does in the winter. To improve reliability, a larger dataset may prove useful. If this is not sufficient, a more complex system with other sensors may be proposed.

(a) MobileNet-V1 5px Vanishing Point Error in the winter.

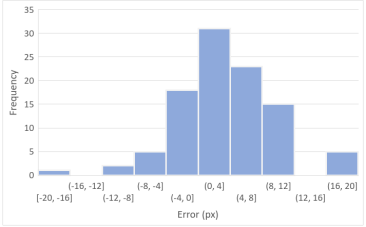
(b) MobileNet-V2 8px Vanishing Point Error in the winter.



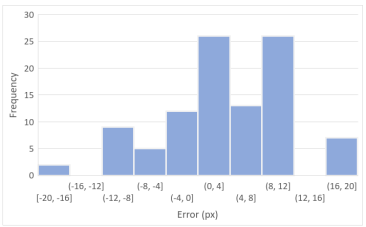
(c) MobileDets 6px Vanishing Point Error in the winter.



(d) MobileNet-V1 Vanishing Point Error distribution (winter).

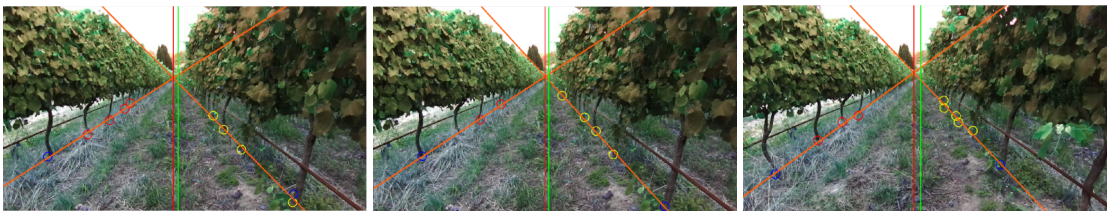


(e) MobileNet-V2 Vanishing Point Error distribution (winter).



(f) MobileDets Vanishing Point Error distribution (winter).

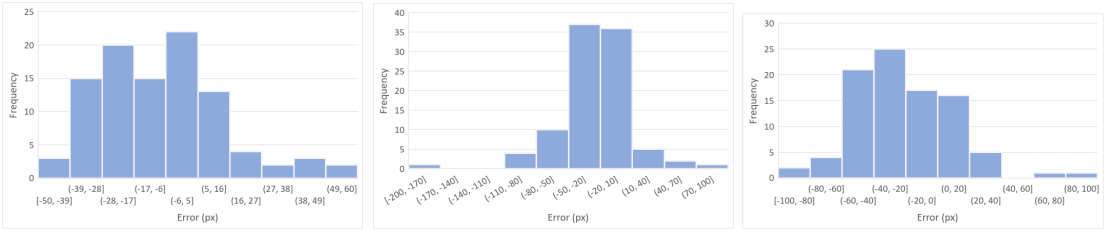
Figure 4.3: Vanishing Point estimation Evaluation in the winter. In figs. (a) to (c) The vertical green and red lines are the vanishing point ground truth and estimation respectively.



(a) MobileNet-V1 20px Vanishing Point Error in the summer.

(b) MobileNet-V2 10px Vanishing Point Error in the summer.

(c) MobileDets 20px Vanishing Point Error in the summer.



(d) MobileNet-V1 Vanishing Point Error distribution (summer).

(e) MobileNet-V2 Vanishing Point Error distribution (summer).

(f) MobileDets Vanishing Point Error distribution (summer).

Figure 4.4: Vanishing Point estimation Evaluation in the summer. In figs. (a) to (c) The vertical green and red lines are the vanishing point ground truth and estimation respectively.

Table 4.2: Deep Learning-based vanishing point detection average absolute error and min and max range in degrees by time of the year. ResNet50 was not implemented due to the incapability of quantization.

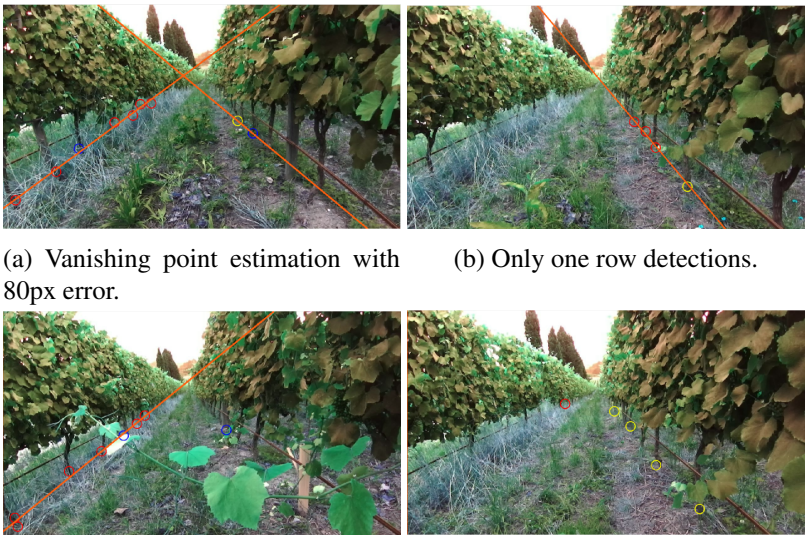
Model	**winter**		**summer**	
	mean (°)	**range (°)**	**mean (°)**	**range (°)**
MobielNet-V1	0.48	[-1:2]	1.67	[-6:5]
MobielNet-V2	0.51	[-2:2]	2.87	[-20:10]
MobileDets	0.69	[-2:2]	2.80	[-10:10]

4.4 Autonomous guidance evaluation

4.4.1 Straight line vineyard

The straight vineyard is the best environment to test each individual controller and observe their advantages and disadvantages and how they contribute to the joint controller. In Figs. 4.6b, 4.6c, and 4.6d, are plotted the error measurements, given the disturbances in Fig. 4.6a, for the vanishing point controller (VP), the horizontal controller (H), and the joint controller, respectively. The table 4.3 shows the recorded RMS error values for scenarios with and without disturbances for all controller configurations.

Compared to the horizontal controller, the vanishing point has a more precise control, which is confirmed by the lower RMS error without disturbances. It is also possible to observe the



(a) Vanishing point estimation with 80px error.

(b) Only one row detections.

(c) Occlusion by loose vine, only one detection on the right row.

(d) Occlusion by tall grass, only one detection on the left row.

Figure 4.5: Example of great error in estimation (a), and of filtered bad vanishing point estimation or lack off (b),(c) and (d).

advantages of the derivative part of the VP controller, with only VP control in Fig. 4.6b, and also in the joint control, Fig. 4.6d. However, when the VP control is disturbed, the system is not able to return to the center of the vineyard using vanishing point detection. This is caused by the robot being thrown off course and the vanishing point estimator incorrectly estimating the center of the vineyard due to perspective. The horizontal controller, on the other hand, excels at staying in the center of the vineyard. As shown in Fig. 4.6c, the controller returns to the zero position (center of the vineyard) even in the presence of disturbances. The joint controller shows the best results as it receives the positive attributes from both VP and the horizontal control. As shown in Fig. 4.6d, in contrast to Fig. 4.6b, where the position remained in a steady state error after the perturbation. In this case, it starts to sink, showing that it is trying to return to the center. As for the RMS error in a scenario without disturbances, we can also observe that the RMS error for the joint controller is the lowest among all configurations, which means that the junction of both controllers is beneficial and more robust. And when comparing to other state of the art guidance system in vineyards such as Rovira-Mas et al. (2021), that yields an average error of 7cm from the center line, our proposed system is able to outperform this as the RMS error for straight vineyards for the joint controller is 1.17cm. Although it is important to mention that Rovira-Mas et al. (2021) results are from a real scenario and not a simulation.

4.4.2 Curved vineyard

The curved vineyard is complex scenario that usually exist in mountainous regions, such as in the Douro region, Portugal. This simulated scenario will be used to evaluate the behaviour of all the proposed controllers configurations and if they are able to robustly guide in this complex scenario.

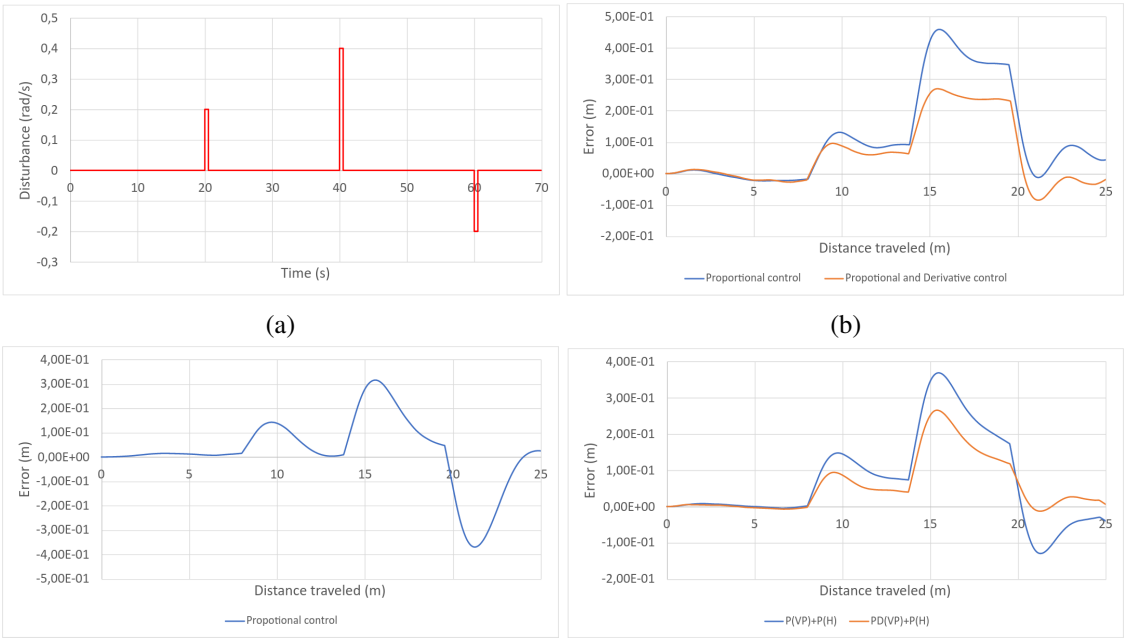


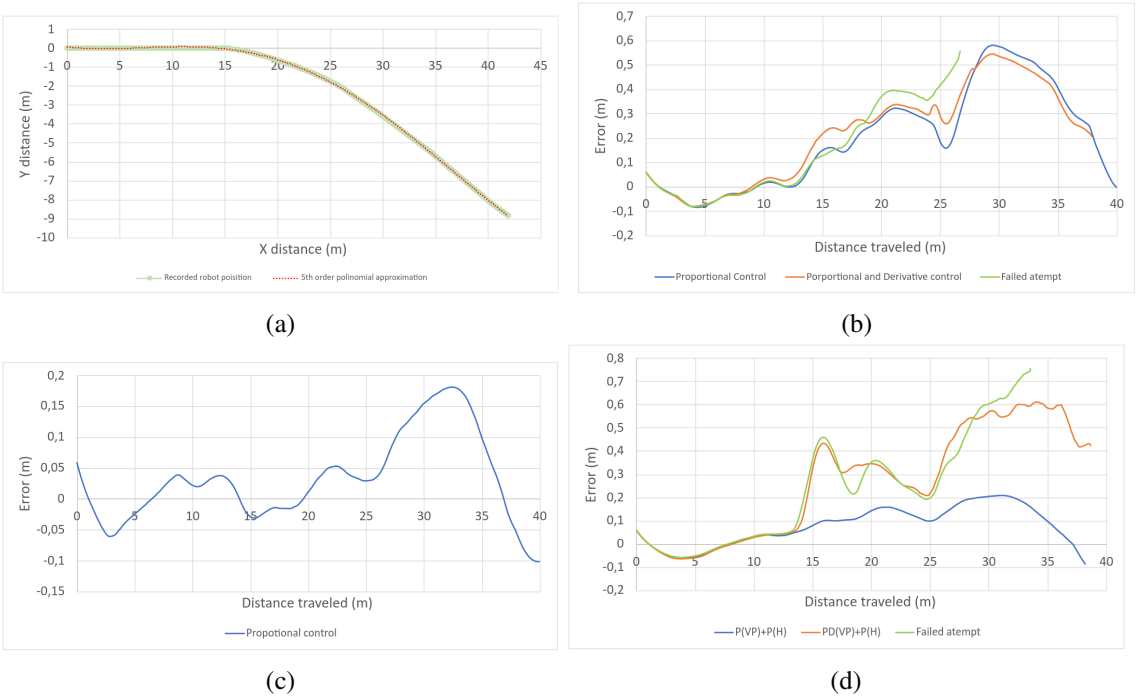
Figure 4.6: (a) is the disturbances signal and (b),(c),(d) are disturbances error response for vanishing point, horizontal and joint controller.

Figs 4.7b,4.7c and 4.7d display the error to the center of the vineyard, comparing the recorder position to the ground truth in Fig. 4.7a, for the vanishing point, horizontal and joint controller, respectively.

As can be seen from Figs. 4.7b, 4.7c and table 4.4, the vanishing point has very poor performance when compared to the horizontal controller with high RMS error and high maximum error reaching 0.5m. Considering that the simulated vineyard is 2m wide, this error is unacceptable. Moreover, it can be observed that a failed attempt was recorded where the vanishing point could not finish steering until the end of the vineyard. On the other hand, the horizontal control has very positive results, with a very low RMS value of 0.07m, it is able to guide the robot robustly through the vineyard, with a maximum error of 0.17m, also low, which is acceptable considering the complex scenario. The large discrepancy between these controllers can be explained by the number of detections needed for steering. Vanishing point control requires at least two points in each row for minimum steering capability. This can be difficult in high accentuated curves where at some point only a few trunks are visible, and if the robot is not perfectly centered, the occlusion will only get worse. With horizontal controller, only one point is needed in each row, which is ideal in this scenario, and the results confirm this. Analysing the joint controller, Fig. 4.7d and table 4.4, it is possible to observe that there is no advantage for the vanishing point control, since even in the best recorded scenario, with only proportional control, the RMS and the maximum error are significantly higher than only the horizontal controller.

Table 4.3: RMS error (cm) for all controller configurations.

	RMS error (cm) for controller configuration:				
Test scenario	**P(VP)**	**P(H)**	**P(VP)+P(H)**	**PD(VP)**	**PD(VP)+P(H)**
with disturbances	17.86	13.53	13.74	11.37	9.01
without disturbances	1.86	2.75	1.37	1.78	1.17



(a) (b)

(c) (d)

Figure 4.7: (a) is the curved vineyard ground truth and (b),(c),(d) are error to the center of the vineyard for vanishing point, horizontal and joint controller.

Table 4.4: RMS error (cm) and Max module of the error (cm) for all controller configurations

	P(VP)	**P(H)**	**P(VP)+P(H)**	**PD(VP)**	**PD(VP)+P(H)**
RMS error (cm)	28.31	7.28	11.24	29.36	37.31
Max error module (cm)	57.96	17.22	20.81	54.32	60.99

Chapter 5

Conclusions

In this dissertation, a single-camera vision guidance system was developed. This system achieves guidance by using the vanishing point estimation. The proposed controller uses monocular angle perception to estimate angular orientation in two different ways: First, by estimating the vanishing point and second, by averaging the position of the nearest trunks for each row. For the base trunk detection, Deep Learning object detection techniques were used. Four single-shot detectors (MobileNet-V1, MobileNet-V2, MobilDets and ResNet50) were trained using transfer learning and deployed on the Google coral dev mini using a built in-house dataset. The results were evaluated in terms of Precision, Recall, Average Precision and F1 Score. The best results were obtained from MobileNet-V1, that yielded an AP of 77.08% and an F1 Score of 0.85. However, Mobiledets must also be taken into account, as some False Positive were actually unlabeled vines. Moreover, this CNN showed the most detections, which is very beneficial to estimate the vanishing point. In terms of vanishing point estimation, all trained models had similar and good performance as the lowest average of absolute error was 0.48 degrees. On the other hand, the summer performance showed that progress still needs to be made as the lowest average absolute error was 1.67 degrees. In order to evaluate the created controller, a vineyard simulator was created. Two scenarios were designed, a simple straight vineyard and a curved vineyard inspired by the Douro region. The proposed joint controller was found to be more robust when using both monocular perception techniques in a straight line, resulting in a RMS error of 1.17 cm. In contrast, in the case of the curved vineyard, the vanishing point component of the joint controller proved to be disadvantageous. It can be concluded that for curved vineyards or for areas where the trunk is frequently occluded, the horizontal controller is the best choice, as it yielded a RMS error of 7.28 cm compared to the 29.36 cm of the vanishing point. Future work will extend the dataset to improve summer performance and investigate how to make the vanishing point more robust, with the goal of creating a more robust controller for curved vineyards, since, as seen with the straight line, combining the two controllers proved beneficial. In addition, fusion with other controllers for robustness will be explored.

References

Diego Aghi, Vittorio Mazzia, and Marcello Chiaberge. Local Motion Planner for Autonomous Navigation in Vineyards with a RGB-D Camera-Based Algorithm and Deep Learning Synergy. *arXiv*, pages 1–11, 2020. doi: 10.3390/machines8020027.

André Silva Aguiar, Filipe Baptista Neves dos Santos, Luís Carlos Feliz dos Santos, Vitor Manuel de Jesus Filipe, and Armando Jorge Miranda de Sousa. Vineyard trunk detection using deep learning – An experimental device benchmark. *Computers and Electronics in Agriculture*, 175 (March):105535, 2020. ISSN 01681699. doi: 10.1016/j.compag.2020.105535. URL https://doi.org/10.1016/j.compag.2020.105535.

Andre Silva Aguiar, Filipe Neves Dos Santos, Armando Jorge Miranda De Sousa, Paulo Moura Oliveira, and Luis Carlos Santos. Visual trunk detection using transfer learning and a deep learning-based coprocessor. *IEEE Access*, 8:77308–77320, 2020. ISSN 21693536. doi: 10.1109/ACCESS.2020.2989052.

André Silva Aguiar, Nuno Namora Monteiro, Filipe Neves Dos Santos, Eduardo J. Solteiro Pires, Daniel Silva, Armando Jorge Sousa, and José Boaventura-Cunha. Bringing semantics to the vineyard: An approach on deep learning-based vine trunk detection. *Agriculture (Switzerland)*, 11(2):1–20, 2021. ISSN 20770472. doi: 10.3390/agriculture11020131.

Aanis Ahmad, Dharmendra Saraswat, Varun Aggarwal, Aaron Etienne, and Benjamin Hancock. Performance of deep learning models for classifying and detecting common weeds in corn and soybean production systems. *Computers and Electronics in Agriculture*, 184(March), 2021. ISSN 01681699. doi: 10.1016/j.compag.2021.106081.

Alireza Ahmadi, Lorenzo Nardi, Nived Chebrolu, and Cyrill Stachniss. Visual Servoing-based Navigation for Monitoring Row-Crop Fields. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4920–4926, 2020. ISSN 10504729. doi: 10.1109/ICRA40945.2020.9197114.

Marco Ammoniaci, Simon Paolo Kartsiotis, Rita Perria, and Paolo Storchi. State of the art of monitoring technologies and data processing for precision viticulture. *Agriculture (Switzerland)*, 11 (3):1–21, 2021. ISSN 20770472. doi: 10.3390/agriculture11030201.

Pablo Arbeláez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011. ISSN 01628828. doi: 10.1109/TPAMI.2010.161.

Marcel Bergerman, Silvio M. Maeta, Ji Zhang, Gustavo M. Freitas, Bradley Hamner, Sanjiv Singh, and George Kantor. Robot farmers: Autonomous orchard vehicles help tree fruit production. *IEEE Robotics and Automation Magazine*, 22(1):54–63, 2015. ISSN 10709932. doi: 10.1109/MRA.2014.2369292.

Michael Buzzy, Vaishnavi Thesma, Mohammadreza Davoodi, and Javad Mohammadpour Velni. Real-time plant leaf counting using deep object detection networks. *Sensors (Switzerland)*, 20 (23):1–14, 2020. ISSN 14248220. doi: 10.3390/s20236896.

Chin Kai Chang, Jiaping Zhao, and Laurent Itti. DeepVP: Deep Learning for Vanishing Point Detection on 1 Million Street View Images. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 4496–4503, 2018. ISSN 10504729. doi: 10.1109/ICRA.2018.8460499.

Ching Ju Chen, Ya Yu Huang, Yuan Shuo Li, Ying Cheng Chen, Chuan Yu Chang, and Yueh Min Huang. Identification of Fruit Tree Pests with Deep Learning on Embedded Drone to Achieve Accurate Pesticide Spraying. *IEEE Access*, 9:21986–21997, 2021. ISSN 21693536. doi: 10.1109/ACCESS.2021.3056082.

Luc Christiaensen, Zachariah Rutledge, and J. Edward Taylor. Viewpoint: The future of work in agri-food. *Food Policy*, 99(March 2020):101963, 2021. ISSN 03069192. doi: 10.1016/j.foodpol.2020.101963. URL https://doi.org/10.1016/j.foodpol.2020.101963.

Austin Costley and Randall Christensen. Landmark Aided GPS-Denied Navigation for Orchards and Vineyards. *2020 IEEE/ION Position, Location and Navigation Symposium, PLANS 2020*, pages 987–995, 2020. doi: 10.1109/PLANS46316.2020.9110130.

Álvaro García-Faura, Fernando Fernández-Martínez, Ricardo Kleinlein, Rubén San-Segundo, and Fernando Díaz-de María. A multi-threshold approach and a realistic error measure for vanishing point detection in natural landscapes. *Engineering Applications of Artificial Intelligence*, 85 (August):713–726, 2019. ISSN 09521976. doi: 10.1016/j.engappai.2019.08.001.

Google. Dev board, a. URL https://coral.ai/products/dev-board/.

Google. Dev board mini, b. URL https://coral.ai/products/dev-board-mini/.

Seung-Hun Han, Kyeong-Min Kang, Chang-Hyun Choi, Dae-Hyun Lee, et al. Deep learning-based path detection in citrus orchard. In *2020 ASABE Annual International Virtual Meeting*, page 1. American Society of Agricultural and Biological Engineers, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:770–778, 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.90.

Patrick J. Hennessy, Travis J. Esau, Aitazaz A. Farooque, Arnold W. Schumann, Qamar U. Zaman, and Kenny W. Corscadden. Hair fescue and sheep sorrel identification using deep learning in wild blueberry production. *Remote Sensing*, 13(5):1–18, 2021. ISSN 20724292. doi: 10.3390/rs13050943.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017. URL http://arxiv.org/abs/1704.04861.

Intel. Intel® neural compute stick 2. URL https://software.intel.com/content/www/us/en/develop/hardware/neural-compute-stick.html.

Jawad Iqbal, Rui Xu, Shangpeng Sun, and Changying Li. Simulation of an autonomous mobile robot for LiDAR-based in-field phenotyping and navigation. *Robotics*, 9(2):1–19, 2020. ISSN 22186581. doi: 10.3390/robotics9020046.

Y. Kondo, M. Numada, and H. Koshimizu. Fast and robust-vanishing point detection system using fast M-estimation method and regional division for in-vehicle camera. *Thirteenth International Conference on Quality Control by Artificial Vision 2017*, 10338:1033817, 2017. ISSN 1996756X. doi: 10.1117/12.2266854.

Yun Li. Li, Y. and Ang, K.H. and Chong, G.C.Y. (2006) PID control system analysis and design. *IEEE Control Systems Magazine*, 26(November):32–41, 2007.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9905 LNCS:21–37, 2016. ISSN 16113349. doi: 10.1007/978-3-319-46448-0_2.

Yin Bo Liu, Ming Zeng, and Qing Hao Meng. D-VPnet: A network for real-time dominant vanishing point detection in natural scenes. *Neurocomputing*, 417:432–440, 2020. ISSN 18728286. doi: 10.1016/j.neucom.2020.08.021. URL https://doi.org/10.1016/j.neucom.2020.08.021.

Yin-bo Liu, Ming Zeng, and Qing-hao Meng. Using Convolutional Neural Networks and Heatmap Regression. 70, 2021.

Hong Kun Lyu, Chi Ho Park, Dong Hee Han, Seong Woo Kwak, and Byeongdae Choi. Orchard free space and center line estimation using Naive Bayesian classifier for unmanned ground self-driving vehicle. *Symmetry*, 10(9), 2018. ISSN 20738994. doi: 10.3390/sym10090355.

Sandro Augusto Magalhães, Luís Castro, Germano Moreira, Filipe Neves, Mário Cunha, Jorge Dias, and António Paulo Moreira. Evaluating the Single-Shot MultiBox Detector and YOLO Deep Learning Models for the Detection of Tomatoes in a Greenhouse. pages 1–24, 2021.

Vasso Marinoudi, Claus G. Sørensen, Simon Pearson, and Dionysis Bochtis. Robotics and labour in agriculture. A context consideration. *Biosystems Engineering*, 184:111–121, 2019. ISSN 15375110. doi: 10.1016/j.biosystemseng.2019.06.013. URL https://doi.org/10.1016/j.biosystemseng.2019.06.013.

Dario Mengoli, Roberto Tazzari, and Lorenzo Marconi. Autonomous Robotic Platform for Precision Orchard Management: Architecture and Software Perspective. *2020 IEEE International Workshop on Metrology for Agriculture and Forestry, MetroAgriFor 2020 - Proceedings*, pages 303–308, 2020. doi: 10.1109/MetroAgriFor50201.2020.9277555.

Nvidia. Jetson nano developer kit, Apr 2021. URL https://developer.nvidia.com/embedded/jetson-nano-developer-kit.

Rafael Padilla, Sergio L. Netto, and Eduardo A.B. Da Silva. A Survey on Performance Metrics for Object-Detection Algorithms. *International Conference on Systems, Signals, and Image Processing*, 2020-July:237–242, 2020. ISSN 21578702. doi: 10.1109/IWSSIP48289.2020.9145130.

Josiah Radcliffe, Julie Cox, and Duke M. Bulanon. Machine vision for orchard navigation. *Computers in Industry*, 98:165–171, 2018. ISSN 01663615. doi: 10.1016/j.compind.2018.03.008. URL https://doi.org/10.1016/j.compind.2018.03.008.

Ade Ramdan, Ana Heryana, Andria Arisal, R. Budiarianto S. Kusumo, and Hilman F. Pardede. Transfer Learning and Fine-Tuning for Deep Learning-Based Tea Diseases Detection on Small Datasets. *Proceeding - 2020 International Conference on Radar, Antenna, Microwave, Electronics and Telecommunications, ICRAMET 2020*, pages 206–211, 2020. doi: 10.1109/ICRAMET51080.2020.9298575.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:779–788, 2016. ISSN 10636919. doi: 10.1109/CVPR.2016.91.

Ricardo Reis, Jorge Mendes, Filipe Neves Dos Santos, Raul Morais, Nuno Ferraz, Luis Santos, and Armando Sousa. Redundant robot localization system based in wireless sensor network. *18th IEEE International Conference on Autonomous Robot Systems and Competitions, ICARSC 2018*, pages 154–159, 2018. doi: 10.1109/ICARSC.2018.8374176.

D. Reiser, D. S. Paraforos, M. T. Khan, H. W. Griepentrog, and M. Vázquez-Arellano. Autonomous field navigation, data acquisition and node location in wireless sensor networks. *Precision Agriculture*, 18(3):279–292, jun 2017. ISSN 15731618. doi: 10.1007/s11119-016-9477-2.

Giuseppe Riggio, Cesare Fantuzzi, and Cristian Secchi. A Low-Cost Navigation Strategy for Yield Estimation in Vineyards. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2200–2205, 2018. ISSN 10504729. doi: 10.1109/ICRA.2018.8462839.

Francisco Rovira-Más, Christophe Millot, and Veronica Sáiz-Rubio. Navigation strategies for a vineyard robot. *American Society of Agricultural and Biological Engineers Annual International Meeting 2015*, 5:3936–3944, 2015. doi: 10.13031/aim.20152189750.

Francisco Rovira-Mas, Veronica Saiz-Rubio, and Andres Cuenca-Cuenca. Augmented Perception for Agricultural Robots Navigation. *IEEE Sensors Journal*, 21(10):11712–11727, 2021. ISSN 15581748. doi: 10.1109/JSEN.2020.3016081.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. ISSN 10636919. doi: 10.1109/CVPR.2018.00474.

Luís C. Santos, André S. Aguiar, Filipe N. Santos, António Valente, José Boa Ventura, and Armando J. Sousa. Navigation Stack for Robots Working in Steep Slope Vineyard. *Advances in Intelligent Systems and Computing*, 1250 AISC(September 2020):264–285, 2021. ISSN 21945365. doi: 10.1007/978-3-030-55180-3_21.

Luís Santos, Filipe Santos, Jorge Mendes, Pedro Costa, José Lima, Ricardo Reis, and Pranjali Shinde. Path planning aware of robot's center of mass for steep slope vineyards. *Robotica*, 38 (4):684–698, 2020. doi: 10.1017/S0263574719000961.

Mostafa Sharifi and Xiaoqi Chen. A novel vision based row guidance approach for navigation of agricultural mobile robots in orchards. *ICARA 2015 - Proceedings of the 2015 6th International Conference on Automation, Robotics and Applications*, pages 251–255, 2015. doi: 10.1109/ICARA.2015.7081155.

J. P. Vasconez, J. Delpiano, S. Vougioukas, and F. Auat Cheein. Comparison of convolutional neural networks in fruit detection and counting: A comprehensive evaluation. *Computers and Electronics in Agriculture*, 173(November 2019):105348, 2020. ISSN 01681699. doi: 10.1016/j.compag.2020.105348. URL https://doi.org/10.1016/j.compag.2020.105348.

Xuewei Wang and Jun Liu. Multiscale Parallel Algorithm for Early Detection of Tomato Gray Mold in a Complex Natural Environment. *Frontiers in Plant Science*, 12(May):1–12, 2021. ISSN 1664462X. doi: 10.3389/fpls.2021.620273.

Yunyang Xiong, Hanxiao Liu, Suyog Gupta, Berkin Akin, Gabriel Bender, Yongzhe Wang, Pieter-Jan Kindermans, Mingxing Tan, Vikas Singh, and Bo Chen. MobileDets: Searching for Object Detection Architectures for Mobile Accelerators. 2020. URL http://arxiv.org/abs/2004.14525.

Zihan Zhou, Farshid Farhat, and James Z. Wang. Detecting Dominant Vanishing Points in Natural Scenes with Application to Composition-Sensitive Image Retrieval. *IEEE Transactions on Multimedia*, 19(12):2651–2665, 2017. ISSN 15209210. doi: 10.1109/TMM.2017.2703954.

J. G. Ziegler and N. B. Nichols. Optimum settings for automatic controllers. *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, 115(2B):220–222, 1993. ISSN 15289028. doi: 10.1115/1.2899060.