POLYECHNIC OF TURIN

Master's degree: Mechatronic Engineering

Master's degree thesis

Convolutional Neural Network diseases detection of grapevines and UAV autonomous precision spray control

Supervisor

Prof. Elisa Capello

Co-supervisor

Ing. Nicoletta Bloise

Ing. Manuel Carreño Ruiz

Candidate

Marco Lecce

Academic year 2021-2022

Degree session: 04/2022

A mio padre

†A mia nonna

Abstract

Nowadays, Precision Agriculture (PA) and Digital Agriculture (DA) are becoming fundamental instruments to oppose and prevent the upcoming agricultural sector crisis due to fertile soils scarcity, climate change, famine, lack of water and demographic expansion. Conventional individuation of crops pandemic clusters and pathological status rely on manual inspection, affected by high subjectivity as well as being costly and time wasting. Furthermore, intensive spraying of Plant Protection Products (PPP) has been for decades the unique method to ensure large-scale productions, with dramatic consequences in terms of eutrophication, soil toxicity and resources wasting. The combination of automated health status detection and automated precision spray allows to increase the soil productivity, to use fertilizer and pesticides only where is needed and drastically cut down the costs. This thesis presents an Unmanned Aerial Vehicles (UAV) implementation in disease recognition and precision aerial spraying of grapevines. A diseases recognition algorithm has been proposed, based on leaves images in the visible spectrum. Moreover, this algorithm is based on the transfer learning of MobileNet2, a Convolutional Neural Network (CNN), that has been pre-trained on the big image database ImageNet to classify one-thousand objects. The CNN is proposed as features extractor, then Linear Support Vector Machine (LSVM) and Logistic Regression (LR) are used for the final classification of grapevine leaves presenting black rot, esca, and leaf blight (Isariopsis leaf spot) symptoms. The outcome of the algorithm is a cartesian map providing information about the individual request of PPP and the relative plant position, called waypoints. A path planning routine, aimed at maximizing the autonomy of the quadrotor, was designed to solve a modified Travelling Salesman Problem finding the optimal sequence of waypoints, using Artificial Intelligence (AI). The solver minimizes both the travel distance and the time-averaged carried payload, by means of a Genetic Algorithm (GA). A Linear Quadratic Regulator (LQR) has been proposed, with yaw-gain scheduling to control a 25 kg quadrotor with a tank of 10 L, and an Extended Kalman Filter (EKF) as state estimator. A sensor-free wind estimation algorithm was developed, allowing a local estimate of wind magnitude and direction. Furthermore, experimental relations between nozzle pressure, rotors downwash, air-UAV relative velocity and the drift entity of a hollow-cone nozzle were retrieved by image analysis. Intersecting wind data with drift experimental curves, was then developed an algorithm to correct the UAV path in presence of wind to maximize some irroration quality factors.

Contents

"If I have seen further, it is by standing on the shoulders of Giants"

[ISAAC NEWTON]

1 Introduction

Automation and mechanization of repetitive tasks has been the foundation of industrial and agricultural revolutions. Recent advances in artificial intelligence (AI), Internet of Things (IoT), Big data Analysis and robotics made possible to push forward the automation horizons enabling an increase of productivity of subjective and cognitive tasks, too. The so-called "Agriculture 4.0" introduced, in recent years, a novel way to organize, extract and process data allowing modern farmers gain enhancements in monitoring and treatment of the field. Climate changes are affecting, just now, different areas of the global agricultural sector and the scenario is forecasted to become worst. [1] discussed parasites diffusion, unpredictable and fast variation of the culture's productions, desertification and other cataclysmic events that will lead to a contraction of the agricultural supply chain. Furthermore, the world population growth was 1 % in 2020 and is predicted to reach 9 billion in 2043 [2].Food and Agriculture Organization (FAO), as well as the European Parliament's Committee on Agriculture and Rural Development, have estimated [3] an increase in food needs of around 60 %, compared to the numbers collected between 2005 and 2007. At the same time, estimates have shown a negligible increase in arable land in the coming decades. In addition, the increase in the middle class in emerging economies will increase the demand for varied products and will exert greater pressure on the issuance of guarantees of provenance, quality, and healthiness of production. The goal of agriculture in the coming decades will therefore be to produce more and more sustainably. To minimize the use of inputs and consequently increase farmers' incomes and reduce environmental impact, the agriculture of the future will have to increase yields compared to inputs such as water, fertilizers, plant protection products and manual labor. In this ever-changing context, precision agriculture comes into play. On 1 September 2015, the Italian Ministry of Agriculture appointed a special Working Group to draw up the National Guidelines for the development of precision agriculture. It was declared that it wanted to apply precision management to 10% of the national UAA (Utilized Agricultural Area) by 2021, compared to 1% in 2015, and the objectives have only been partially achieved, to date, especially in the implementation of autonomous driving systems in agricultural vehicles. Productivity increments are the unique response to the reduction of fertile soils and demographic expansion, and the agriculture 4.0 paradigm points in this direction. In particular, it is becoming prominent the utilization of Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) in crop monitoring, remote sensing, mapping and, most recently, in precision spraying[4]. There are single UAVs applications, swarm UAVs logics and UGVs-UAVs cooperation [5]. Exist different types of UAVs, classified by weight, type of take-off, number of rotors ad power source. In this thesis was considered a Vertical Take Off and Landing (VTOL) with 4 brushless DC motors UAV. Vineyards are of particular interest for the use of UAVs because of the rugged terrain that makes difficult the autonomous guidance of terrestrial robots, both for sensing and treatments purposes. UAVs are still strictly normed in urban areas and less in rural environment, where the human risk is lower because of less population density. However, as suggested in [6], there is a lack of normative codification for aerial spraying. UASs spray missions fall in the "Specific" category of the regulative text redacted by the European Aviation Safety Agency (EASA). It imposes to have a Maximum Take-Off Weight (MTOW) of 25 kg, so it is reduced the transportable Plant Production Product (PPP). This thesis focuses on the use of UAV for recognition of grapevines diseases and subsequent precision spray control of the vineyard. Although in this thesis only some of the argument are treated in depth, the entire UASs application in vineyard, thought in [6], consist of three stages. In the first and second phases small drones (under 5 kg MTOW) should perform an accurate 3D mapping of the field to collocate the rows and the plants relatively to the used reference frame. Then, the system should annotate information about the soil and the crops geometries. Subsequently, plant-by-plant, UAVs will capture images of leaves and trunks, and associate them with the previously stored map. A

description, filtering and classification of those images provides the pathological status of each plant, producing a prescription map in which PPP differentiate demands should be derived by image analysis itself. Once the map is produced, a larger quadrotor (25 Kg MTOW) should leave the charging station at the border of the field and start a precision spray mission. The rationale of the work, overall, was to obtain predictions on the performance of the spray system and the autonomy of the drone using a model-based approach. Based on the present literature, a simulation environment was developed on MATLAB and Simulink capable of producing predictive results on the final implementation of the project. The simulator takes in a 2D map of the vineyard consisting of a matrix whose elements correspond to the positions of the individual grapevines, which are associated with photographs of the foliage or a single leaf, and returns in output parameters of quality of spraying, the consumption of the mission and the spatial distribution of the sprayed product.

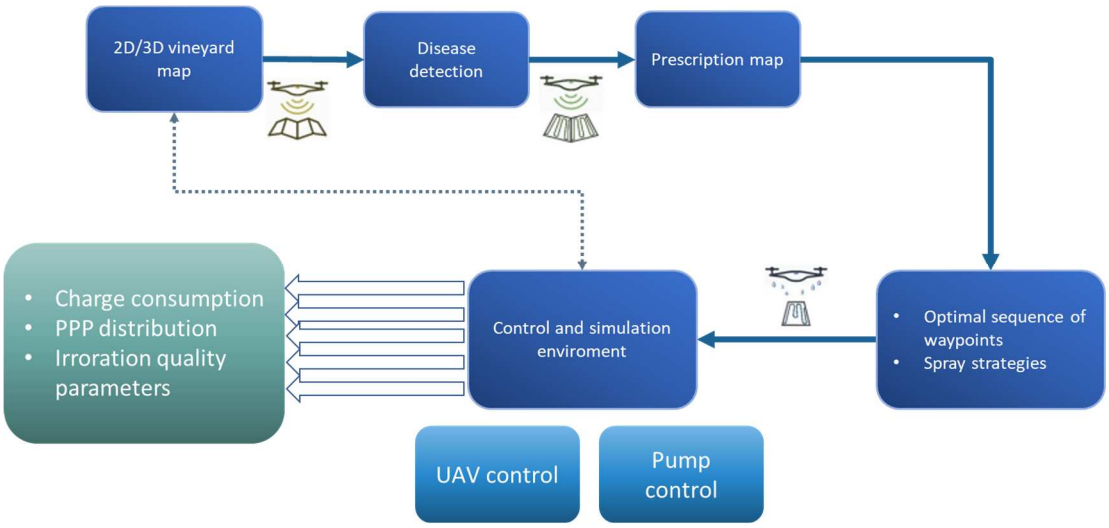


Figure 1 Simulation environment block scheme

Therefore, the application developed consist of several stages. In chapter 2 are revisited the most used Machine Learning (ML) techniques for characterizing, describing, and classifying plant diseases on the base of visible spectrum leaves symptoms. Taking inspiration by the work in [7], the pre-learned Convolutional Neural Network (CNN) MobileNet2 was used as a feature extractor of 4 different grapevines status (3 pathologic class and the healthy class), namely capable to provide a vectorized form of the image (the descriptor) containing all the relevant information linked to the specific disease. Then, L-SVM, Logisitc Regression and a re-trained MobileNet2 were used as tail-ender multinomial classifiers trained with a One-vs-All approach that permits to turn the network in a binary classifier, at the occurrence. Through the segmentation technique of Edge Boxes[8] was tried to push forward the performances of the classifier, providing it, in a test phase, a leaves detection of the generic input image whose acquisition scene could in general contain an entire plant and not only a first-plane of the leaf, as most of the available training dataset images. In chapters 3 and 4 we are studied the dynamical model of a 25 Kg quadrotor, the spray system, and their relative controls. The non-linear dynamics of the drone was linearized around the hovering conditions and a yaw gain-scheduling Linear Quadratic Optimal control was applied to the model in Simulink environment. The lack of directly measurable states with low-cost MEMS-based Inertial Measurement Units, the conflict between low updating frequency of conventional GPS signals and sensor drift problem caused by gyroscope and accelerometer dead-reckoning were discussed, and an Extended Kalman Filter was developed to get an estimate of the quadrotor attitude. Using experimental data provided by the study in [9] was retrieved geometrical properties of the spray flux in function of air-UAV relative velocity, nozzle pressure and rotors velocities, through image analysis. In chapter 4 is also presented an exhaustive discussion about different spray strategies and optimal path planning searching. A modified

Travelling Salesman Problem (TSP) was studied to get the UAV path planning whose tries to minimize the charge consumption of the system and increase the time horizon of the spray mission. The path scheduling was solved by means of Artificial Intelligence algorithms, such as the Genetic Algorithm (GA) and Nearest Neighborhood (NN). Merging together flux experimental curves and a sensor-free wind estimation mechanism[10], [11], an onboard path correction algorithm aimed at minimizing the spray drift problem and save PPP in different ways and scenarios was developed.

2 Grapevine leaf diseases detection

2.1 Vineyard pathologies

There are several pathologies and pathogens affecting grapevines in the world. Three of the most treated in literature and mostly present in the European region were used as specimens for the disease recognition algorithm. Esca disease, caused by attacks of *Phaeoacremonium aleophilum*, *Phaeomoniella chlamydospora* or *Fomitiporia mediterranea,* is one of the most widespread grapevine disease. Foliar symptoms, if present, form a peculiar shape consisting of leaf veins following strips, with yellow, red or green colors and intensities depending on species and other factors[7]. Until 2001, year in which was banned for its toxicity, the unique treatment was Sodium Arsenide based. There are recent alternative biocontrol products (as the Esquive WP), but the common, recent, way to contrast those fungi is prevention and manual inspection [7]. Black rot, another fungal disease (Guignardia bidwelli), appears in spring and summer. Its foliar symptoms appear as a cloud of black spots relatively very smaller than the leaf dimension. Also in this case, mainly manual and visual inspection are used for the identification. This is a time and cost-spending practice surely, but also affected by subjectiveness and low repeatability [12]. Pseudocercospora vitis causes the last analyzed disease, the leaf blight (Isariopsis leaf spot). As the black rot, all the leaf surface (if foliar symptoms appear) contains small black dots. Adjunctively a bleaching of the leaf occurs, becoming gradually a defoliation[13].

2.2 Image acquisition analysis and data augmentation

Two distinct datasets were available:

1. **Plant Village Dataset (PVD)**: Freely downloaded from Kaggle, it contains 4062 leaves images captured in a closed environment by the top of a grey desk. The dataset is divided in 1383 “Esca”, 423 “Healthy”, 1180 “Black Rot” and 1076 “Leaf Blight” images. All the images have the same resolution (256x256 pixels), and similar lighting, position, and leaf sizes.
2. **Università delle Marche Dataset**[14]**(UMD):** Freely available through the cited article, it contains 1768 images on-field. The dataset is divided in 887 “Esca” images and 881 “Healthy” images. Images have the same resolutions with grapevine crops captured in different sizes, number, and positions, furthermore backgrounds and confounding objects are present.

Figure 2 Plant Village Dataset images, from left to right: “Black Rot”, “Leaf Blight”, “Healthy”, “Esca”

Figure 3 Images of Esca leaf (left) and healthy leaf (Right), from Università di Ancona Dataset

Plant Village dataset takes advantage by its regularity and specificity with respect to leaves shape, color histograms and boundaries, clearly defined and insulated in each image. On the other hand, a classifier trained on this dataset may suffer of biasing knowledge, lowly capable to recognize the same features when an arbitrary on-field image is presented. On the contrary, a classifier trained on the second dataset difficultly could be able to learn about single leaf features, since many leaves are overlapped at different depths of the field. Many out-of-class objects should lead to associate wrong information to de final descriptors, so the result probably will be a variance suffering classifier, but with higher robustness facing real on field images.

2.2.1 Dataset 1, 2-classes "artificial grapevines" augmentation

A solution could be the fusion of the two datasets, but it is not completely possible. Unfortunately, UMD contains data only for "Esca" and "Healthy". The issue can be tackled with the artificial constructions of many images for "Black Rot" and "Leaf Blight" classes simulating a sort of artificial grapevine. This construction can be faced at different scales of sophistications, merging different overlapped leaves at different angular positions with a previous extraction from their backgrounds, but for our study a simple demonstration is done, picking four random leaves (previously augmented) and positioning them in the four corners of the new image. Starting by the images of PVD a data augmentation algorithm is used to multiply by 5 the data size, performing, for each image:

- Transposing of the image matrix
- Mirroring of the image matrix along horizontal and vertical axis
- Random changing of the image light intensity

Then, in "Esca" and "Healthy" classes are added the UMD images, instead a number of artificial grapevines, with a similar percentage with respect to the classes' sizes, are inserted in "Leaf Blight" and "Black Rot". The resulting sizes are shown below:

Table 1 Classes sizes of the Dataset 1

"Leaf Blight"	**"Black Rot"**	**"Esca"**	**"Healthy"**	**Entropy H**	**Total**
5729	6267	7807	3001	1,9273	22804

where the entropy was computed in this way

$$H = \sum_{i=1}^{K} -p_i log_2(p_i) \qquad (2.00)$$

And p_i is the frequency of the i-th class over the entire dataset, so upper bounding the sum of the probabilities to 1 the entropy results constrained between 0 (if the whole dataset is composed by only one class) and 2 (if each $p_i = 0.25$).

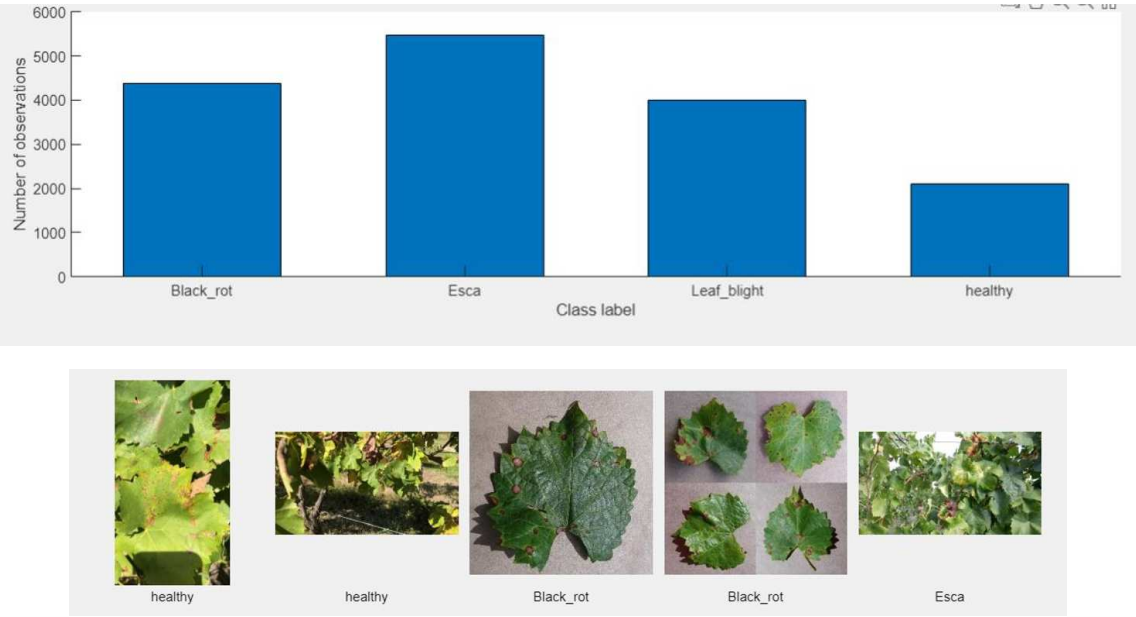


Figure 4 Dataset 1 images for the training set, obtained uniformly splitting dividing the original dataset in 70 % for training and 30% for testing

2.2.2 Dataset 2, 4-classes "artificial grapevines" augmentation with dataset balancing

Another possibility was to use PMD images only. Dataset 2 is proposed for two reasons:

- **Class balancing: a**lthough the entropy of the Dataset1 was acceptable, "Healthy" class, mainly, has a too low frequency that makes a future classifier prone to a high False Negative Rate (FNR) for this class, and/or high False Positive Rate (FPR) for the other. However, a class balancing passes through an augmentation of already existing images to compensate the class size difference, so, after the process, smaller classes will contain the highest percentage of augmented images, and, unavoidably, the smallest variance, since a consistent number of attributes of the original images propagates necessarily also in the augmented ones. Since these trade-offs was needed in Dataset2 a class balancing is proposed, contrary to Dataset1.
- **Uniform bias error of the artificial grapevines:** whatever the complexity and randomness of the artificial grapevine algorithm is, it always carries several attributes shared by all its output images, because strongly dependent by the algorithm structure itself (for the simplified version proposed, for example, all the artificial grapevines images contain a visible cross dividing the four corners). Dataset2 proposed an exclusion of the UMD dataset for the training, allocating for each class a set of artificial grapevine images. This dataset will lose the robustness with respect to on-field crop images, so a further leaves-detector algorithm will be necessary down-stream to restrict the classification step to an image as much as possible comparable with the single leaves used in training.

Starting from the PVD the same four geometrical augmentation steps presented above are performed, obtaining a x5 factor of the images' number.

Table 2 Classes size of the augmented Dataset2

"Leaf Blight"	"Black Rot"	"Esca"	"Healthy"	Entropy H	Total
5380	5900	6920	2115	1,8947	20315

Taking "Esca" class size as reference the dataset is balanced generating, for each class, a number of new images equal to the gap between the class size and the reference size. Those images are

constructed randomly extracting 2 already existing images in the class and doing a randomly weighted average of the two images.

$$Image_{new} = w_1 Image_1 + w_2 Image_2 \qquad w_1, w_2 \in [0,1] \qquad (2.01)$$

Figure 5 image constructed for dataset balancing

There are many other ways to do a balancing dataset, all image augmentation techniques. We could also balance them with the same image augmentation algorithm used initially, for example by rotating the leaves at different angles. The balancing described was chosen because the result is both geometrically and morphologically different from the originals. After the balancing is ultimate, 6920 images for each class are present. The final step is the composition of a further 20% of images as artificial grapevine, randomly picking images by the last augmented and balanced dataset. The result is a dataset with 33216 images and entropy exactly equals 2.

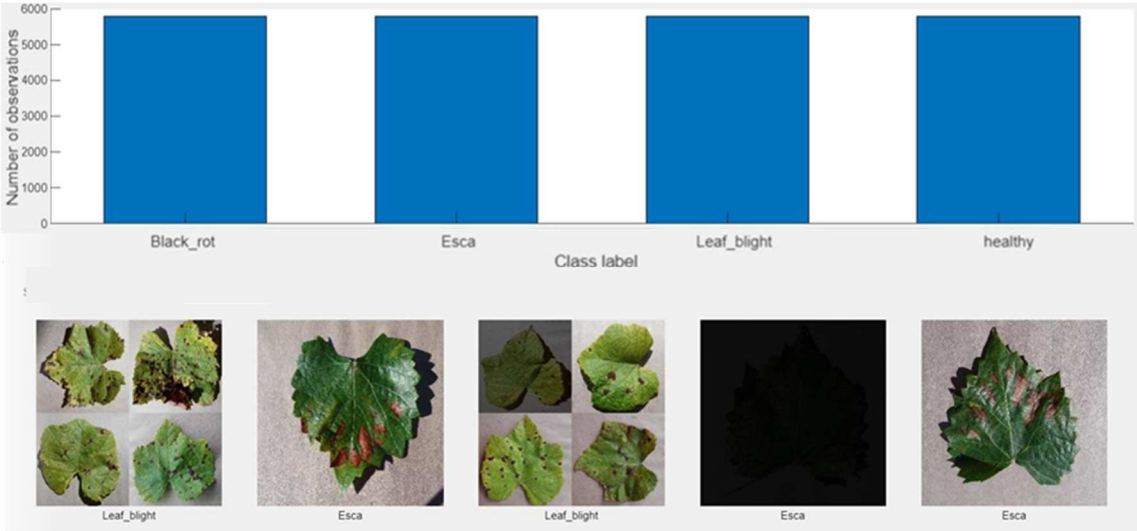


Figure 6 Dataset 2 images for the training set, obtained uniformly splitting dividing the original dataset in 70 % for training and 30% for testing

2.2.3 Principal Components Analysis

A principal component analysis is a dimension reduction technique capable to extract from the features matrix axes along which is seen the highest variance of its feature vectors. Employing a feature extractor explained below, an image is compressed in a 1D vector x_i of 1280 elements, so an image could be seen as a point in a 1280 dimensions vector space. Packing the vectors in columns a features matrix is obtained, from which a covariance matrix S can be extracted.

$$X = [x_1; \dots \dots x_i; \dots . . x_{N_{dataset}}] \qquad (2.02)$$

$$\hat{x} = \frac{1}{Ndataset} \sum_{i=1}^{Ndataset} x_i \qquad (2.03)$$

$$\hat{X} = X - \hat{x}\mathbf{1}^{\boldsymbol{T}} \quad (2.04)$$

$$S = \hat{X}\hat{X}^T \quad (2.05)$$

The i-th eigenvector of the covariance matrix S is the i-th principal component of the feature matrix X, in other words the i-th eigenvalue associated with the i-th eigenvector is the i-th greatest standard deviation of whole set of images features in the 1280-dimension space. Both for dataset 1 and dataset 2 the first 3 eigenvectors contain about the 25 % of the total variance. This result suggests that a graphical representation in 3 dimensions of the images features vector could be a suitable way to analyze them. For the two dataset is seen a significant separability for "Leaf Blight" images by the other classes, while a more complex superimposition is present for the other 3 classes. In the dataset 1, however, a well separable cluster of "Healthy" and "Esca" images is present. They are the features vectors belonging the UMD dataset, with very different attributes.

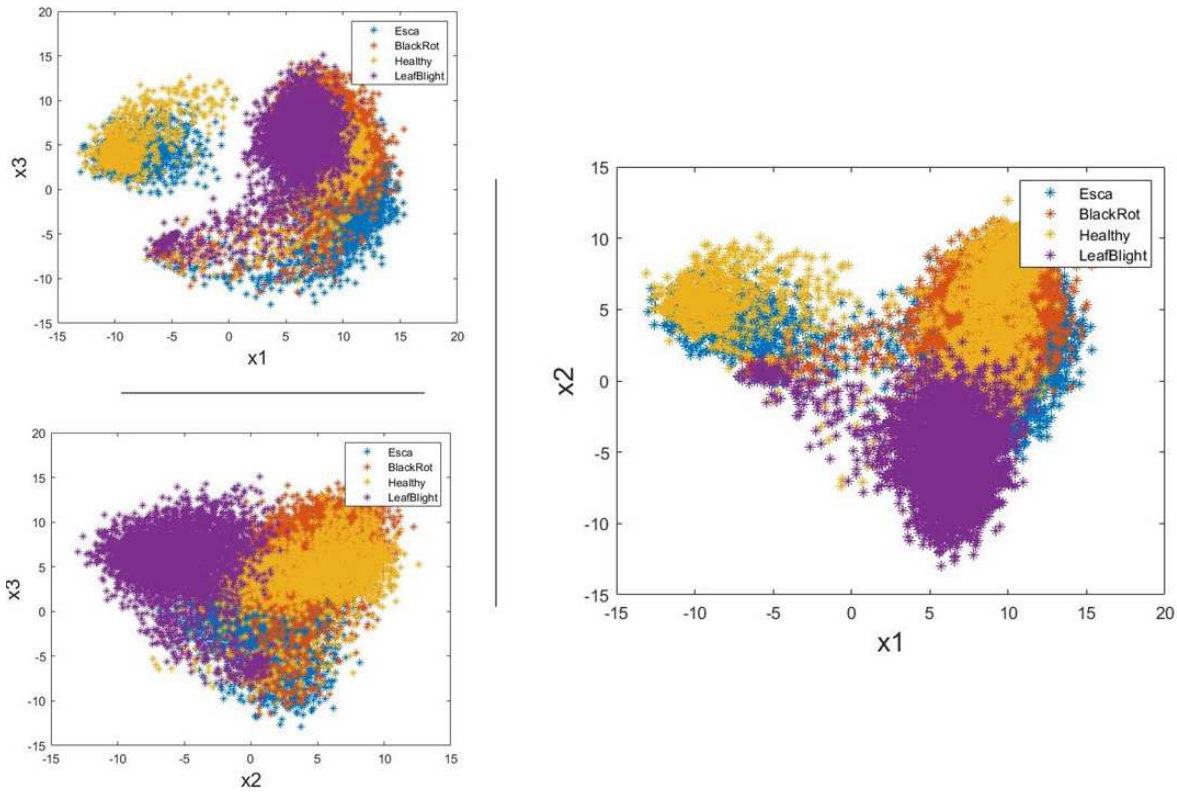


Figure 7 Dataset1 Principal Component Analysis

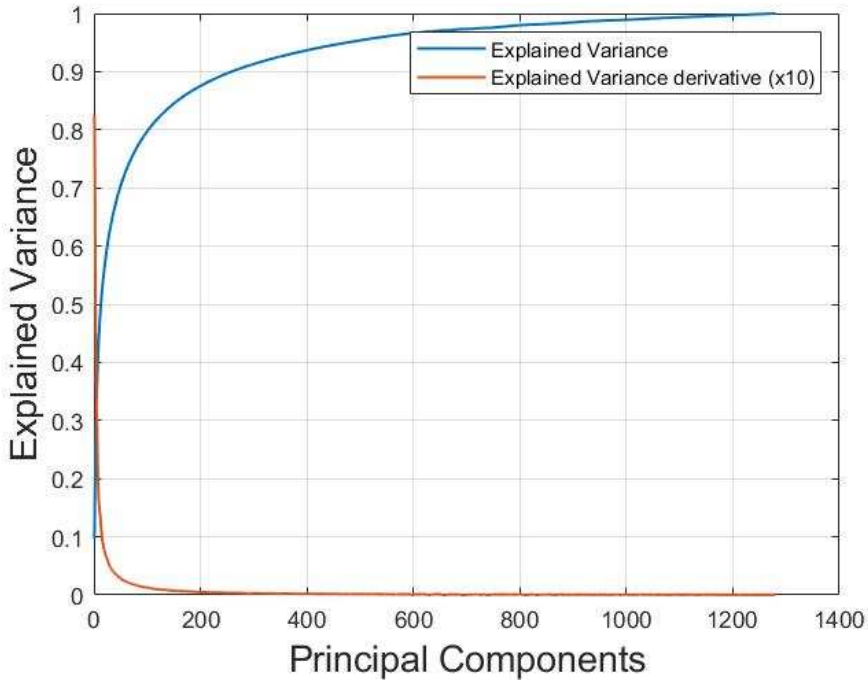


Figure 8 Explained variance of Dataset 1

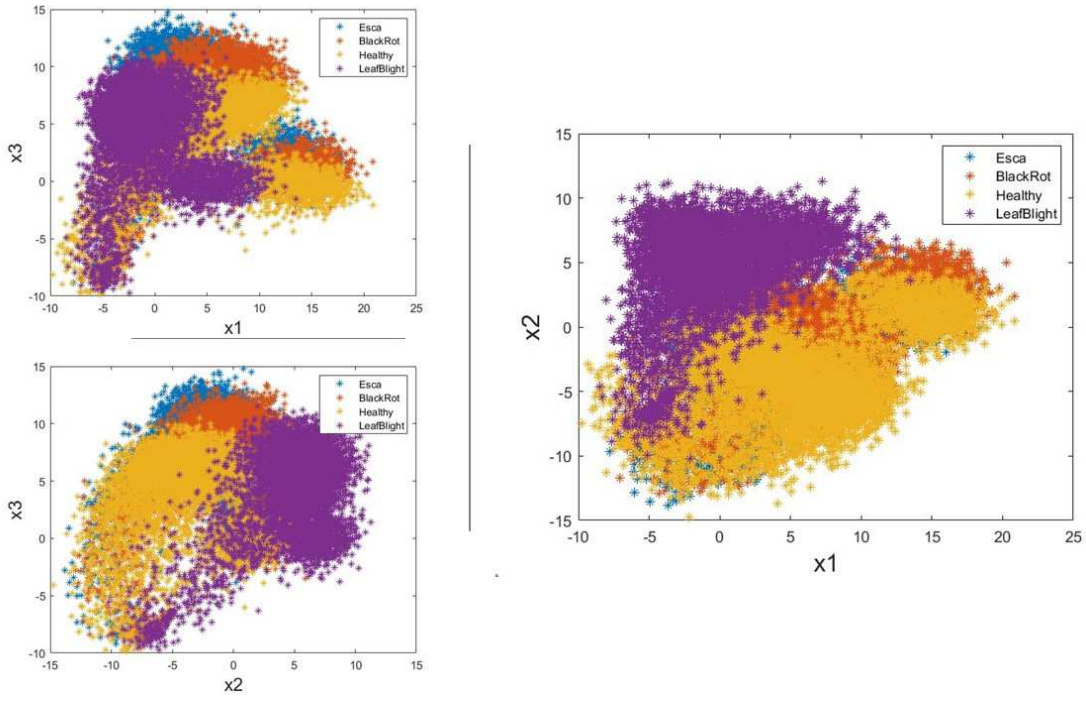


Figure 9 Dataset2 Principal Component Ananlysis

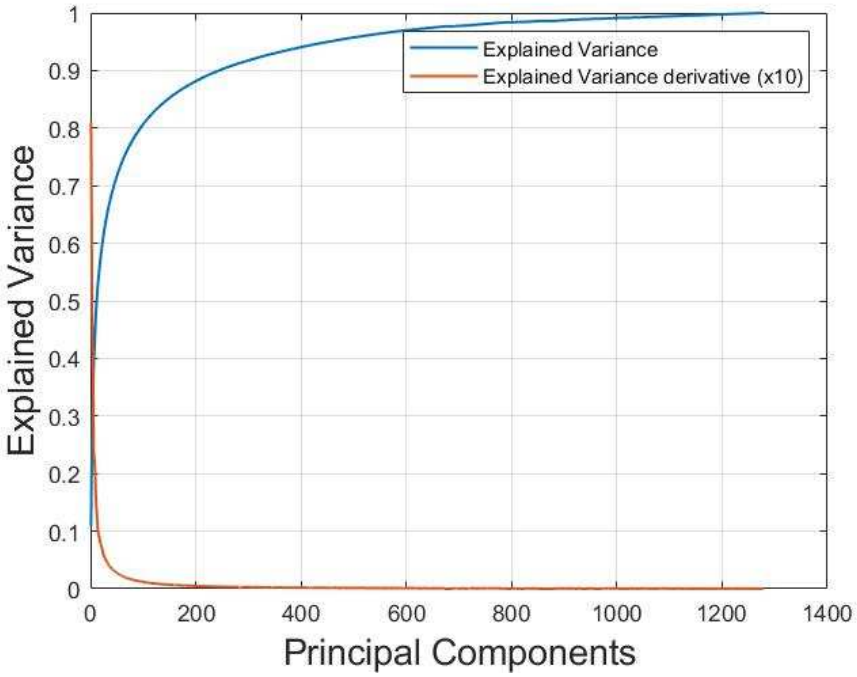


Figure 10 Explained variance of Dataset 2

2.3 Plant features extraction (Dog and SiFt/CNN)

Starting from 8 bit codified 3 channel image, a matrix can be obtain, where each element represents a vector of 3 values (Red,Green and Blue intensity) spacing between 0 and 255. *Description* phase of such data aims to reduce the number of elements to classify, leaving only containing relevant information being, simultaneously, mutually independent. Traditionally *Computer Vision Feature Detection Techniques (CVFDT)* are a group of engineered techniques used to extract features vectors from images based on their textures, colors, and shapes properties. The modern known CVFDT are:

- SIFT (Scale Invariant Features Transform)
- SURF (Speed-up Robust Features) and FAST (Features from Acelerated Segment Test), faster versions of SIFT
- ORB (Oriented FAST and Rotated Brief), is a free alternative of FAST and Brief descriptor algorithm developed by OpenCV labs
- CNN (convolutional Neural Networks) features extraction.[15]

There are also corner and edge detection algorithms (Harris detector, angular grids etc) that are, instead, not rotation and scale invariant. Since our application needs to recognize leaves, pictures taken from arbitrary distance and angular positions those techniques will be avoided.

2.3.1 Scale Invariant Features Transform (SIFT)

Sift methods accept as input images and returns a 1D descriptor vector, ensuring that such a descriptor will be scale invariant with the respect to object token with different angular and proximity positions. The image is scan and a variable size set of key-points coordinates in the image returns by the action of Difference of Gaussian (DoG) methods. Each key-point is surrounded by a grid of dimension dependent on the strategy chosen:

- Dense Sift: grids sizes are fixed and chosen based on preliminary optimal parameters searching.
- Phow Sift: as the Dense Sift fixed sizes are chosen but they are more than one.
- DoG Sift: grids size is outputted by the DoG algorithm.

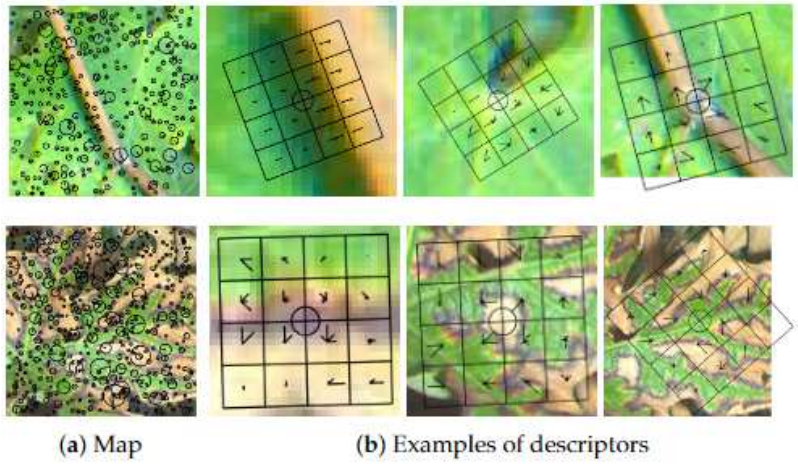


Figure 11 SIFT, image by [5]

For each key point is computed and stored the grid maximum increasing direction of the grid gradient and then for each element of the grid a local gradient is computed in 8 (e.g.) directions. The resultant descriptor is obtained forming a matrix whose columns are vector that enlist the modules of the resultant local gradient of the i-th key-point ordered in a fixed way using as reference the i-th key-point grid gradient maximum increasing direction. If the key points are *N*, the grids scales are 4, and the number of directions for local gradient is 8 then the descriptor will be a matrix *N x (4x4x8=128).* But the number *N* of key points differs, in general, among different images so at the end of the descriptor extraction a normalization of the final feature vector length must be performed, to make them comparable by a classifier[7]. In the learning dataset unsupervised learning approach (k-means) is applied in the features space: it means that all the *N* (variable) 128x1 vectors owned by each learning image are put together to evaluate the dominant *k* clusters, and so key-points, that better descript the images. Once the clustering is done, k new cluster centers are available and so new, compressed, and homogeneous features vectors can be obtained. For example, the feature vector can be obtained by an image computing the frequencies, for each cluster, of the original image key-points appearance in that specific cluster, the output (for a generic image) is a descriptor of size *k* (Bag of Words encoding). Otherwise exploiting VLAD (Vector of Locally Aggregated Descriptors) a more sophisticated descriptor could be generated computing, for each cluster, a 128-dimension vector resulted by averaging the distances between vectors belonging to the cluster and its center[7]. For VLAD the final feature vector has size *k x 128*.

2.3.2 Convolutional Neural Network (CNN)

Artificial Neural Network are used in machine learning for supervised classifiers and/or regressors. As many machines learning methods, they accept as input vectors or matrices then passes it throughout several perturbations compute as output binary, categorical or scalar predictors. The main difference with the respect to other machine learning mechanisms is that each element of the input feature vector can be partially or fully connected with neurons (elements) of the consecutive layer. Indeed, for example, in a traditional classifier (SVM, Logistic Regression, Naïve-Bayes, etc.) are learned a bias b and number of weights w_i equal to the size of the feature vector. The resultant hyperplane, described by the equation $b + \bar{w}^T \bar{x} = 0$, is the result of the minimization of certain loss function engaged to maximize the number of right-labeled

train feature vectors, then, often, it passes through an activation function (e.g. sigmoid) the returns the affiliation percentage of the corresponding data to a given class.

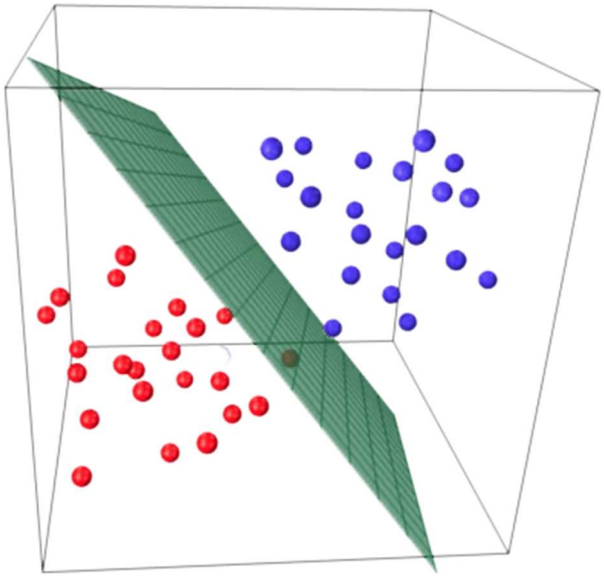


Figure 12 Boundary hyperplane in 3D case, taken from [16]

Instead, the peculiarity of neural networks is their complexity. A neural network is composed by a certain number N of layers, and in each of them there are n neurons. A single neuron can have its own bias and weights to learn and an activation function that outputs its specific confidence parameter. This structure results in a more sophisticated way to represent a generic Data Generation Mechanism than other machine learning methods, but the number of variables and non-linearities in the optimization problem causes the ANN to be also more prone to overfitting and higher computational costs.

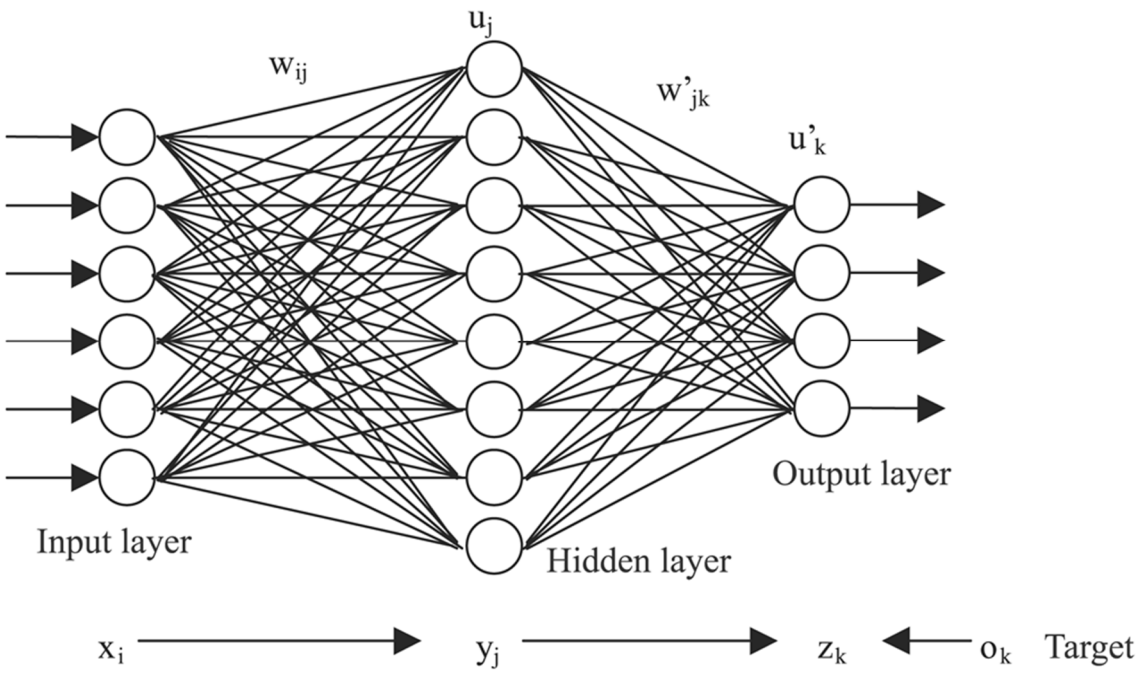


Figure 13 Non-deep Neural Network, image taken from [15]

Especially for image recognition tasks fully connected ANN are prohibitive by a computational point of view since the very high dimensionality of features in input. For these reasons recent development in ANN, and in particular Deep Neural Networks (ANN with more than one hidden layer), showed a dramatic increasing in the affordability-computation effort ratio in Deep Neural Networks constructed by means of several filtering convolution layers, ReLu activation functions and pooling layers aimed to construct a light-weight feature vector downwards fully or partially connected to the final classification layers[17]. The strength of CNN, as the name suggests, stays in the convolution matrix used to recursively filtering the original image matrix. Convolution matrix is a well-known strategy used in image recognition: once the matrix size and the step parameter are chosen the convolution acts as a scan that compute for each target element of the original input a strictly local property through a stencil that imposes a certain scalar transformation of the elements surrounding the target. Then the convolutional matrix moves toward other targets whose distances are codified in the step parameter. The result is a filtered version of the original image matrix. As biological analogies with optic nerve and visual cortex suggests [17], the most efficient way to connect the new convolutional layer to the next is not a fully connection, but a local connection. In other words, if we have an image of 32x32x3 and we want to fully connect it with the next layer will results 32x32x3 by the number of neurons of the next layer but applying a convolution filtering of scale 5 and step 1, supposing that the next layer is exactly a 31x31 grid (similar to the previous one). Each neuron is connected only with the corresponding previous neuron/features, will results a number of weights quals to only 31x31 by feature dimensionality. Often, for each neuron the bias and weights are equal, so the number of unknowns for the specific layer is only the feature dimensionality. This approach shows astonishing computation effort reduction mixed with very excellent performance in image recognition[17].

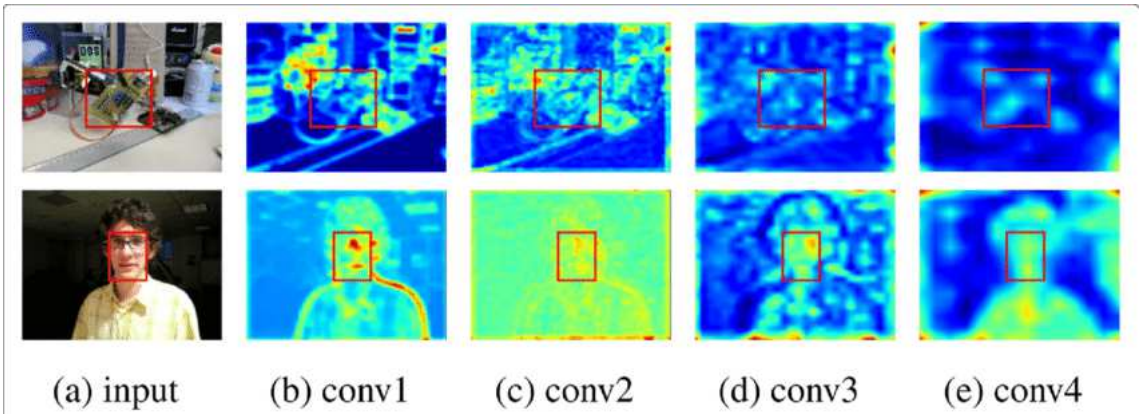


Figure 14 RGB adaptation of the input image taken from different convolutional layers, image from [17]

Convolutional layers can be used as filtering maintaining the same dimension of the input matrix, or reducing it. To maintain the same dimension of the input often is performed a padding operation adding an external frame of zero values, since the convolution always takes away at least 1 row and 1 column. When high dimension contraction is required, a pooling strategy is used: it may use maximum pooling, or statistic function pooling. After a suitable number of hidden convolutional, ReLu and Pooling layers a feature vector, hopefully containing the most relevant information of the image, is passed to few and dense fully connected layers to perform the final classification.

2.3.3 Comparison between CNN and SIFT

As mentioned in [7], CNN features extraction presents better performance both in terms of final accuracy on the test set and computational effort being MobileNet the network used for feature extraction. Furthermore, considering the number of phases in the Sift (DoG, Global Gradients,

Local sub-Gradients, row matrix features ordering, clustering, encoding) and the unique phase of the pre-trained CNN, the seconds seems the best also from a project time spent point of view.

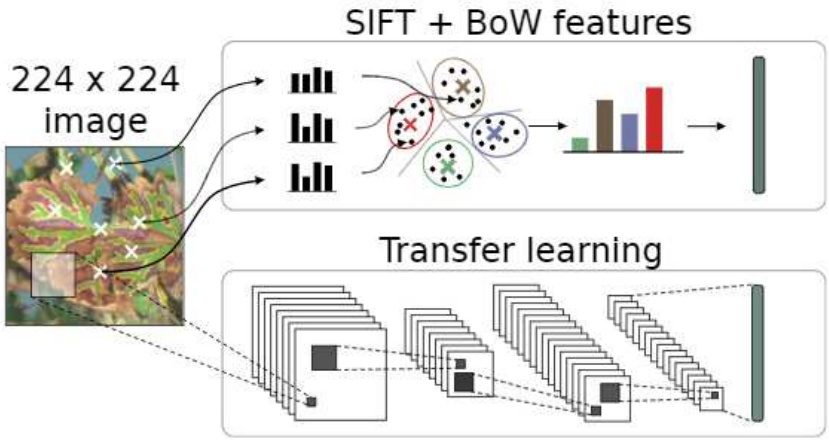


Figure 15 SIFT and transfer learning comparison, image from [5]

Considering that in the final classification layers are the densest in terms of weights content, for less complex classification tasks a pre-trained convolutional network can be used only to extract the feature vector, simply pruning the final classification block. Then a more lightweight classifier (Svm or Logistic Regression) will be used starting from the resulting descriptor.

2.3.4 Pre-trained Convolutional Neural Networks: MobileNetV2

There are many CNNs freely available online interesting for this work. All of them can be used as pre-trained nets, since bias and weights of the entire net were trained over huge databases of images such as ImageNet in many cases being able to classify among one thousand objects. Pre-trained network offers a unseen before opportunity to design own machine learning project with a robustness achievable only with such a huge training stages, indeed, although their neurons are trained for a specific set of object recognition, the capability of features extraction remain enough constant across other object classes. These techniques are called Transfer Learning. As a matter of fact, for the proposed application, the choice of the suitable CNN must follow the requirement of real-time acquisition and classification activity on hardware supports and CPUs with characteristics of mobile devices. So, the better choice could be picking up the network with the less number of weights (excluding the fully connected layers) that traduces in less number of operations and less time per classification, compared to its declared performances. Is worth notice that the computational cost is not only linked to the amounts of weights, but it's also related to the specific structure of the convolutional layer[18], [19]. In the following table are enlisted some of the most interesting CNN available on MATLAB Deep Learning Toolbox studied for Mobile applications, and their whole and reduced numbers of weights.

Table 3 Structures of Convolutional Neural Networks available on the MATLAB Deep Learning Toolbox

Network Name	No. of total weights	No. of feature extractor weights	Depth of the Network	Input image resolution
AlexNet	61 M		8	227x227x3
VGG-16	138 M		16	224x224x3
VGG-19	144 M		19	224x224x3
ResNet-101	44,6 M		101	224x224x3

DenseNet-201	20 M		201	224x224x3
MobileNet-v2	3,5 M	2,219 M	53	224x224x3
Places365-GNet	7 M	6,625 M	22	224x224x3
NasNet Mobile	5,3 M	4,243 M	n/a	224x224x3
ShuffleNet	1,4 M	0,855 M	501	224x224x3
SqueezeNet	1,24 M	0,727 M	18	227x227x3

For the purpose MobileNet-v2, ShuffleNet and SqueezeNet shown the lowest number of parameters, and are also in the same order of magnitude. In this work MobileNetv2 is chosen for its proved accuracy on the ImageNet database, on which it reached Top-5 accuracy of about 87 %, and its CPU time spent of **75 ms** when run on a single large core of the Google Pixel 1 phone as feature extractor and classifier (using TF-Lite).[18]. Input image resized to 224x224x3 resolution enters MobileNet and a feature map can be extract at any depth of the CNN. Deeper extraction means higher level information such as shapes and space relationships but increasing in Multiple-Adds operations, while upward extraction returns lower-level information such as colors and intensities at a lower computational cost. Furthermore, at any extraction depth a clustering method (PCA, k-means, gaussian mixtures) can be applied in order to decrease the number of features and consequently training and test time.

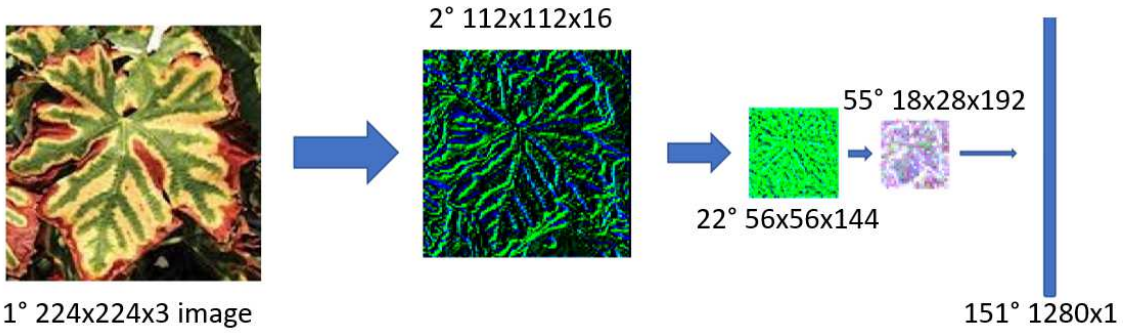


Figure 16 Visual representation of feature maps at different depths

2.4 Leaves level Classifier

For the final classification a comparison between two classical method and the full learning of MobileNet is proposed. The main differences between the three methods are:

- **Logistic Regression:** Only 5124 weights (1281 for each class) are learned, and the learning process is performed by a personally handwritten function based on Newton-Rapson gradient-descent algorithm since a closed form solution is not available (could be reached global minima).
- **Supported Vector Machine:** 5124 weights (1281 for each class) and 4-by-N_dataset relaxation factors ξ_i are learned, and the learning process is performed by the tool cvx (similar to QuadProg of MATLAB) based on the solution of a Convex Quadratic Program minimization with linear constraints (global minima could be reached).
- **CNN Mobile Net Full Learning:** 2,219 + 0,005124 millions of weights are learned (feature extraction block plus classification block), and the learning process is performed by an embedded algorithm based on Backpropagation, in this case the loss function is a Not-Convex function (ensured only local minima).

2.4.1 Logistic Regression and L-SVM probabilistic posterior models

Both Logistic Regression and L-SVM are linear classifiers based on the computation of the posterior probability of a feature vector $\bar{x}$ being in a certain class $y=k$, following the equation:

$$Prob(y = k|X = x) = \frac{e^{(b_k + \bar{w}_k^T \bar{x})}}{1 + \sum_{i=1}^{K-1} e^{(b_i + \bar{w}_i^T \bar{x})}}; \qquad Prob(y = K|X = x) = \frac{1}{1 + \sum_{i=1}^{K-1} e^{(b_i + \bar{w}_i^T \bar{x})}} \quad (2.06)$$

where $\bar{x}$ is the feature vector,while b_k and $\bar{w}_k^T$ are the bias and weights describing the *k-th* hyperplane dividing the class k features by all the others, and K is the total number of classes, in our case K=5, since we have to classify among "Esca", "Black Rot", "Leaf Blight", "Healthy" and confused classes, because the leaves can be naturally affected by other symptoms not encoded. From that follows that:

$$\sum_{i=1}^{K} Prob(y = k|X = x) = 1 \quad (2.07)$$

$$1 - Prob(y = k|X = x) = Prob(y \neq k|X = x) \quad (2.08)$$

$$k = K \; is \; the \; "confused" \; class \quad (2.09)$$

While the order of the other classes is not relevant. To simply employ the probabilistic model can be assumed a binary classification for which *y* can assume only the values 0 and 1, so *K=2*:

$$Prob(y = 1|X = x) = \left[1 + e^{-(b + \bar{w}^T \bar{x})}\right]^{-1} = sigm(b + \bar{w}^T \bar{x}) \quad (2.10)$$

$$Prob(y = 0|X = x) = 1 - Prob(y = 1|X = x) = \left[1 + e^{(b + \bar{w}^T \bar{x})}\right]^{-1} = 1 - sigm(b + \bar{w}^T \bar{x}) \quad (2.11)$$

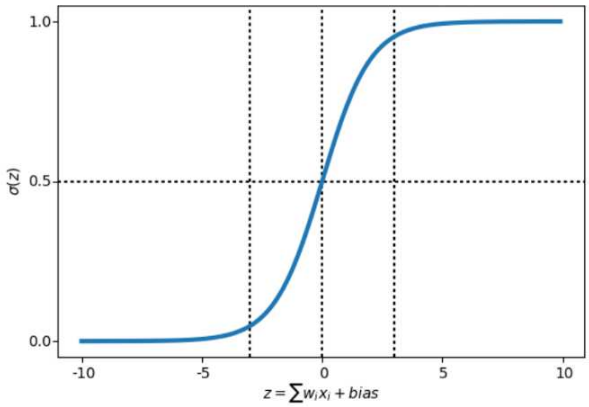


Figure 17 sigmoid function

This posterior probability model can be used for all 4 classes (3 diseases and "Healthy"), such that each class, independently from the others, can have a probability spacing in [0,1]:

$$\begin{cases} Prob(y = 1|X = x) = \dfrac{e^{(b_1 + \bar{w}_1^T \bar{x})}}{1 + e^{(b_1 + \bar{w}_1^T \bar{x})}} \\ Prob(y = 2|X = x) = \dfrac{e^{(b_2 + \bar{w}_2^T \bar{x})}}{1 + e^{(b_2 + \bar{w}_2^T \bar{x})}} \\ Prob(y = 3|X = x) = \dfrac{e^{(b_3 + \bar{w}_3^T \bar{x})}}{1 + e^{(b_3 + \bar{w}_3^T \bar{x})}} \\ Prob(y = 4|X = x) = \dfrac{e^{(b_4 + \bar{w}_4^T \bar{x})}}{1 + e^{(b_4 + \bar{w}_4^T \bar{x})}} \\ 1 = Healthy; \;\; 2 = Esca; \;\; 3 = Black \; rot; \;\; 4 = Leaf \; Blight \end{cases} \quad (2.12)$$

So the classification algorithm assesses the decision y as the class with the maximum posterior probability that, simultaneously, equals or exceed a threshold $T \in [0,1]$:

$$y = arg_k \max\{Prob(y = k|X = x)|Prob(y = k|X = x) \geq T\} \quad k = 1,2,3,4 \qquad (2.13)$$

And so, "Confused" class probability can be computed and y is set to Confused if there are no class posteriors exceeding the threshold:

$$y = 5 = Confused \qquad if \quad Prob(y = k|X = x) < T \quad \forall k = 1,2,3,4 \qquad (2.14)$$

This model will be useful in the test stage of both Logistic Regression and L-SVM, but in the training stage only for the Logistic Regression, since its loss function is based on the Bayesian approach, while the SVM one on geometric assumptions.

2.4.2 Logistic Regression Learning

Starting from (2.12), for each class, the Bernoulli model can be written:

$$Prob(y|X = x) = \left[1 + e^{-y\left(b+\bar{w}^T\bar{x}\right)}\right]^{-1} = \begin{cases} \dfrac{e^{\left(b+\bar{w}^T\bar{x}\right)}}{1 + e^{\left(b+\bar{w}^T\bar{x}\right)}}, & y = 1 \\ \dfrac{1}{1 + e^{\left(b+\bar{w}^T\bar{x}\right)}}, & y = -1 \end{cases} \qquad (2.15)$$

where, for simplicity, the negative label was assigned to -1 and not 0. Then considering the *i-th* training image producing the $\bar{x}_i$ feature vector and labeled with y_i. The couple $(\bar{x}_i\ ;\ y_i)$ is an observation, and the simultaneous appearance of its 2 element is:

$$Prob(\bar{x}_i\ ;\ y_i) = Prob(y_i|\bar{x}_i\ )Prob(\bar{x}_i) \qquad (2.16)$$

Assuming the observations (the images of the dataset) mutually independent and independently distributed, the cross probability of all of them, establishing the probability of the entire dataset is:

$$Prob(\bar{x}\ ;\ y) = \prod_{i=1}^{N} Prob(y_i|\bar{x}_i\ )Prob(\bar{x}_i) \qquad (2.17)$$

where N is the number of images of the considered class. But, since the model itself depends on the parameter $\theta = [b\ \bar{w}]$, the cross probability of the observations is nothing, but the likelihood of the observation conditioned to the parameter θ.

$$Prob(\bar{x}\ ;\ y|\theta\ ) = \prod_{i=1}^{N} Prob(y_i|\bar{x}_i;\ \theta)Prob(\bar{x}_i) = L \qquad (2.18)$$

Since we don't have any prior model on the weighting parameters $p(\theta)$, we can base our study only on the dataset, so applying Bayesian inference can be shown that a good point estimator $\hat{\theta}$ results by the maximization of the likelihood (2.17):

$$\hat{\theta} = arg_{\theta=b,w} \max Prob(\bar{x}\ ;\ y|\theta\ ) = arg_{\theta=b,w} \max \prod_{i=1}^{N} Prob(y_i|\bar{x}_i;\ \theta) \qquad (2.19)$$

where $Prob(\bar{x}_i)$ is neglected because does not depend on parameters unknown. Substituting the (2.15) in (2.18), we obtain:

$$\hat{\theta} = arg_{\theta=b,w} \max \prod_{i=1}^{N} \left[1 + e^{-y_i\left(b+\bar{w}^T\bar{x}_i\right)}\right]^{-1} = arg_{\theta=b,w} \max L \qquad (2.20)$$

To have not to deal with products, the maximization of the likelihood L equals the minimization of the negative log likelihood NLL.

$$\hat{\theta} = arg_{\theta=b,w} \min NLL = arg_{\theta=b,w} \min \sum_{i=1}^{N} \log\left(1 + e^{-y_i\left(b+\bar{w}^T\bar{x}_i\right)}\right) \qquad (2.21)$$

This is the Loss function to be minimized, a closed form solution does not exist, so a gradient descent algorithm must be set to find the global minima. Now, restoring the previous labeling, the (2.15) could be written as:

$$Prob(y|X = x) = \left[1 + e^{-(-1+2y)\left(b+\bar{w}^T\bar{x}\right)}\right]^{-1} = \begin{cases} \dfrac{e^{\left(b+\bar{w}^T\bar{x}\right)}}{1 + e^{\left(b+\bar{w}^T\bar{x}\right)}}, & y = 1 \\ \dfrac{1}{1 + e^{\left(b+\bar{w}^T\bar{x}\right)}}, & y = 0 \end{cases} \qquad (2.22)$$

And so the (2.21) becomes:

$$\hat{\theta} = arg_{\theta=b,w} \min NLL = arg_{\theta=b,w} \min \sum_{i=1}^{N} \log\left(1 + e^{-(-1+2y_i)\left(b+\bar{w}^T\bar{x}_i\right)}\right) \qquad (2.23)$$

The NLL gradient is:

$$\nabla_{b,w}\mathrm{NLL} = \sum_{i=1}^{N} \bar{x}_i \left(y_i - \frac{e^{\left(b+\bar{w}^T\bar{x}_i\right)}}{1 + e^{\left(b+\bar{w}^T\bar{x}_i\right)}}\right) \qquad (2.24)$$

While, the Hessian matrix is:

$$\nabla^2{}_{b,w}\mathrm{NLL} = -\sum_{i=1}^{N} \bar{x}_i\bar{x}_i{}^T \left(1 - \frac{e^{\left(b+\bar{w}^T\bar{x}_i\right)}}{1 + e^{\left(b+\bar{w}^T\bar{x}_i\right)}}\right)\left(\frac{e^{\left(b+\bar{w}^T\bar{x}_i\right)}}{1 + e^{\left(b+\bar{w}^T\bar{x}_i\right)}}\right) \qquad (2.25)$$

The hessian matrix is negative semidefinite, since the two-bracket expression are two probabilities greater or equal to zero, while the $\bar{x}_i\bar{x}_i{}^T$ factor is also positive semidefinite, then follows that the loss function is concave and a global minima can be found efficiently through Newton-Rapson numerical method. The algorithm is a gradient descent way to find the minima, and conists simply in the iteration of:

$$[b, \bar{w}]^{s+1} = \hat{\theta}^{s+1} = \hat{\theta}^s - \left(\nabla^2{}_{b,w}\mathrm{NLL}(\hat{\theta}^s)\right)^{-1}\nabla_{b,w}\mathrm{NLL}(\hat{\theta}^s) \qquad (2.26)$$

Dataset was randomly split in 2 subgroups, 70 % for the training set, and 30 % for the test set, without cross-validation. Setting a Tolerance and a starting point $\hat{\theta}^0$, the algorithm recursively performs (2.25), until the loss function NLL start to become an asymptote, so the stop condition is:

$$\left|NLL\left(\hat{\theta}^{s+1}\right) - NLL\left(\hat{\theta}^s\right)\right| \Big/ \left|NLL\left(\hat{\theta}^s\right)\right| < \mathrm{Tol} \qquad (2.27)$$

$\hat{\theta}^0$ was set to $\bar{0}$, and $Tol = 10^{-5}$.

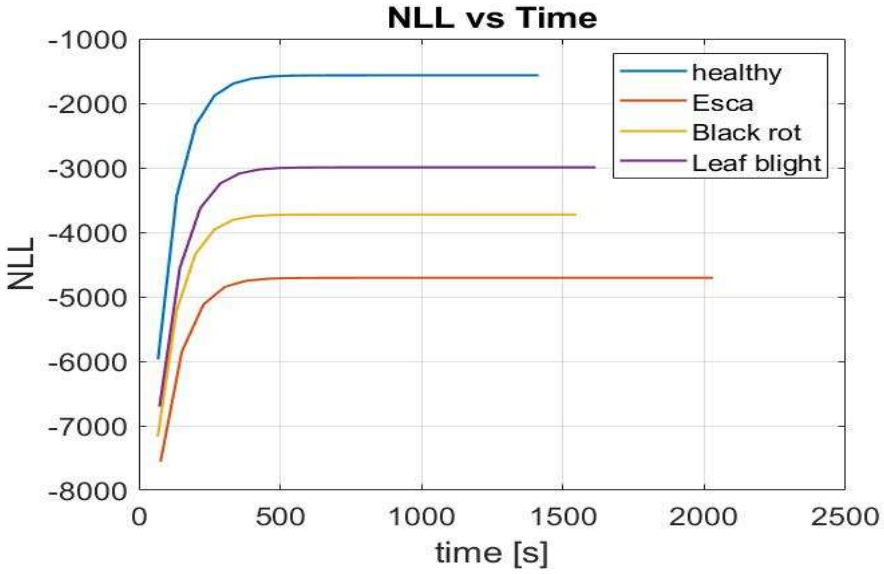


Figure 18 Logistic regression loss function optimization, dataset 1

The total time spent was 6608,7 seconds, so about 110 minutes to reach the optimum.

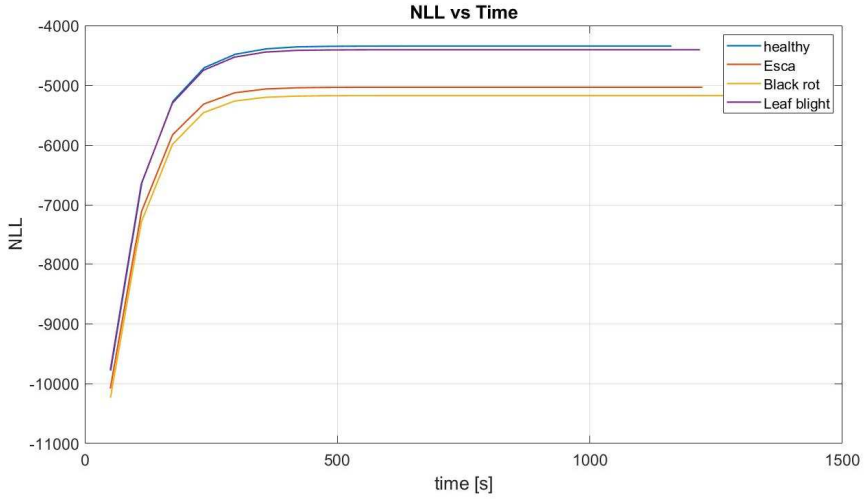


Figure 19 Logistic regression loss function optimization, dataset 2

The total time spent was 5066,4 seconds, so about 84 minutes to reach the optimum.

2.4.3 L-SVM learning

Linear Support Vector Machine follows the same fundamentals of Logistic Regression but with a geometrical model. The aim is, again, to find a boundary hyperplane who better approximates the n-dimensional surface exactly dividing two classes' points. There is also a Not Linear SVM capable to compute a not planar boundary surface for more intricate cases, however for this study is not reputed necessary.

$$\chi:\ b + \bar{w}^T\bar{x} = \bar{0} \quad \begin{cases} d = -\dfrac{b + \bar{w}^T\bar{x}}{\|\bar{w}\|} \\ d_0 = -\dfrac{b}{\|\bar{w}\|} \end{cases} \tag{2.28}$$

χ is the hyperplane definition, while d and d_0 are the signed distance of a generic point $\bar{x}$ and of the origin from the hyperplane. Conventionally the "positive" side of the space is described by the region toward which vector $\overline{w}$ is pointed. Another useful parameter is the labeled signed distance:

$$d^l = yd = -y\frac{b + \overline{w}^T\bar{x}}{\|\overline{w}\|}$$
$$\begin{cases} y = 1 \quad , & postive\ class \\ y = -1 \quad , & negative\ class \end{cases} \tag{2.29}$$

d^l is positive if and only if the features vector $\bar{x}$ is *right labeled,* negative otherwise. Target of the optimization could be to find χ maximizing each d^l in the training set obtaining a well-defined separation between classes:

$$\begin{cases} \max M \\ s.t.\ d_i^l = -y_i\dfrac{b + \overline{w}^T\bar{x}_i}{\|\overline{w}\|} \geq M \qquad i = 1, \ldots . N \\ M \geq 0 \end{cases} \tag{2.30}$$

This optimization problem could result unfeasible if the classes, in the training set, are not linearly separable because, in such a case, there will be always at least one point mislabeled which d_i^l is negative. To circumvent the problem a relaxation is applied:

$$\begin{cases} \min\dfrac{1}{2}\|\overline{w}\|^2 \\ s.t.\ d_i^l = -y_i(b + \overline{w}^T\bar{x}_i) \geq 1 - \xi_i \qquad i = 1, \ldots . N \\ \xi_i \geq 0 \end{cases} \tag{2.31}$$

where ξ_i is the relaxation factor, and, since the holonomic constraints, was assumed M=$1/\|\overline{w}\|$ so the maximization of the margin M became the minimization of $\|\overline{w}\|$, turned in the minimization of $0{,}5\|\overline{w}\|^2$ leading to the same result. (2.31) is the final form of the L-SVM model. It is a convex quadratic function with linear constraints and could be efficiently solve by means of "QuadProg" or "Cvx" on MATLAB. For the study "Cvx" toolbox was used, exploiting the penalty form of (2.31):

$$Loss_{svm} = \min\left(c\sum_{i=1}^{N}(max[0; 1 - y_i(b + \overline{w}^T\bar{x}_i)]) + 0{,}5\,\|\overline{w}\|^2\right); \qquad c = 10 \tag{2.32}$$

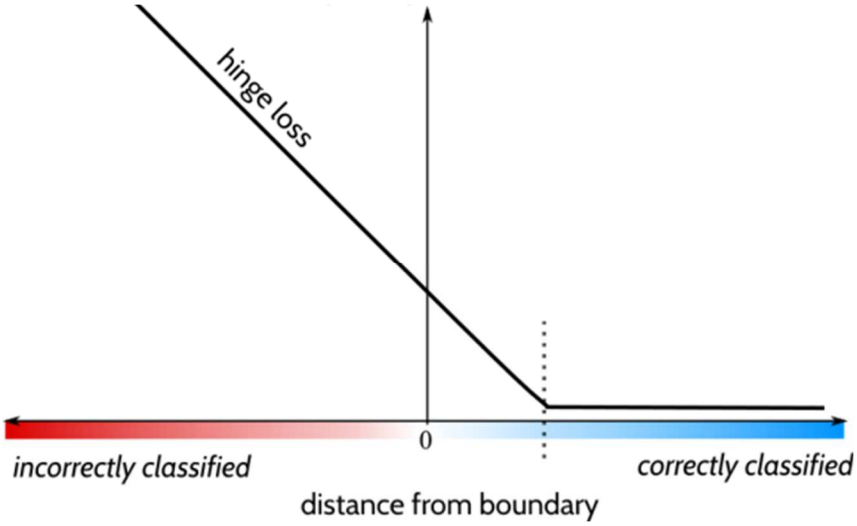


Figure 20 Hinge Loss, image from[20]

Dataset was randomly split in 2 subgroups, 70 % for the training set, and 30 % for the test set, without cross-validation. In the proposed L-SVM optimization, for each class, variables are not only the weights and biases but also a number *N* of relax parameters ξ_i, where *N* is the training dataset size, equals to 22804. So, there will be, in total, the usual 5124 variables plus 4 by N relaxation factors, resulting in 96340 unknowns. Differently by the logistic regression algorithm, real time loss computation, in training stage, is not available, but brief relevant information were extracted:

Table 4 L SVM training, dataset 1

Avg Optimal Loss Function	Avg CPU time (min)	Total CPU time (min)	Loss variation Tolerance	CPU time per iteration (sec)
10,67	330,8	1323,53 min	$1{,}49 \times 10^{-8}$	296,33

Can be noticed an astonishing 12x factor CPU time spent between L-SVM and logistic regression algorithm, against the 18x factor between SVM and logistic regression variables number.

2.4.4 MobileNetv2 learning

For the full learning of the CNN, all the 151 layers used in the feature extractor was reunified with the tail layers:

- **Fully connected layer**: takes in input the feature vector (1280 x 1 dimension), each element is connected with four neurons. The layer has 4 activations, simply equals to:

$$a_k = b_k + \bar{w}_k^T \bar{x} \quad k = 1,2,3,4 \tag{2.33}$$

- **Soft max layer**: takes the four a_k as input and outputs 4 softmax functions:

$$Prob(y = k|X = x) = \frac{e^{(b_k + \bar{w}_k^T \bar{x})}}{1 + \sum_{i=1}^{K-1} e^{(b_i + \bar{w}_i^T \bar{x})}} \qquad k = 1,2,3,4 \tag{2.34}$$

- **Classification layer**: takes the soft-max of each class and outputs a final classification "k" based on an optimal thresholding.

Also, in this case the dataset was randomly split in two subgroups, 70 % for the training set, and 30 % for the test set, without cross-validation. The back-propagation algorithm, with step of mini-batch size fixed to 124, updates all the weights, and when all the dataset is explored a next Epoch is started. Each 50 iteration was updated also the validation Loss and validation accuracy, and the

process was stopped after 411 minutes due to the constantans of the loss function for dataset1 and after 190 minutes for dataset2. For the training following parameters was set:

- Learn Rate=0,01 with Drop Factor of 0,1 each 10 iterations
- Solver: Sgdm
- Gradient threshold Method= l2norm
- Loss function: cross entropy

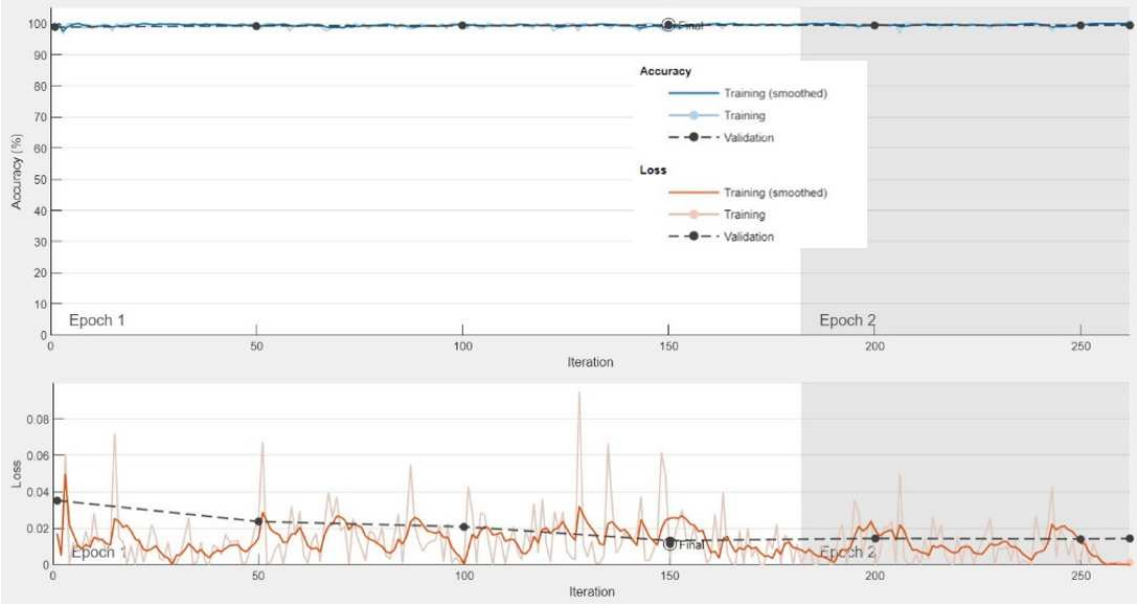


Figure 21 Training Mobilenetv2 database 2

2.4.5 Results and performances

Performances were evaluated on the augmented test set of 6841 images for the dataset1 and 9964 for the dataset2. Being the dataset1 quite unbalanced, in the performances was added also the balanced accuracy that considers the average between the specificity (True negative rate) and the recall (True positive rate) and the Roc curve. All the performances parameters were obtained in relation to the threshold *T* used to impose a percentage, over which, in the specific classifier, the sample is considered positive *(y=1)* or negative *(y=0).* Doing this the optimal value of *T* can be used as good trade-off for the specific task objective. For each classifier was computed the total number of:

- TN: True Negatives
- TP: True Positives
- FP: False Positives
- FN: False Negatives

If is supposed that ∂ is the *right label* for the generic sample in the test set, and being *y* the *assigned label*, then can be computed, for each class:

$$Prob(\partial = k | y = k) = precision = r \, {Prob(\partial = k)}\Big/{Prob(y = k)} \quad (2.35)$$

$$Prob(y = k | \partial = k) = recall\ (r) = True\ Positive\ Rate\ (TPR) = \frac{TP}{FN + TP} \quad (2.36)$$

$$Prob(y \neq k | \partial \neq k) = Specificity(s) = True\ Negative\ Rate\ (TNR) = \frac{TN}{FP + TN} \quad (2.37)$$

And so:

$$\begin{Bmatrix} False\ Negative\ Rate\ (FNR) = \dfrac{FP}{TN+FP} = 1 - TNR \\ False\ Positive\ Rate\ (FPR) = \dfrac{FN}{FN+TP} = 1 - TPR \end{Bmatrix} \quad (2.38)$$

ROC curve is made as function of TPR and FPR leaving varying the parameter *T*, the smaller is *T* the easier is the appearance of False Positive and True Positive, the higher it is the more specific the classifier becomes leaving only True positive, but then also more false negatives. The right choice of *T* is then the best trade-off between TPR and FPR, while the Area Under ROC (AUROC) is a good performance parameter because is strictly linked on how the classifier structure is able to divide the clusters in the probabilistic space (since if it tends to one means that the roc is a perfect square and so there is the possibility that there is *FPR=0* when *TPR=1*, this means that 1-AUROC is related to the overlapping area in the aliasing of the two pdf below). The balanced accuracy is:

$$mean(TPR + TNR) = \left(\frac{TP}{FN+TP} + \frac{TN}{FP+TN}\right)/2 \quad (2.39)$$

2.4.5.2 Logistic Regression Dataset1

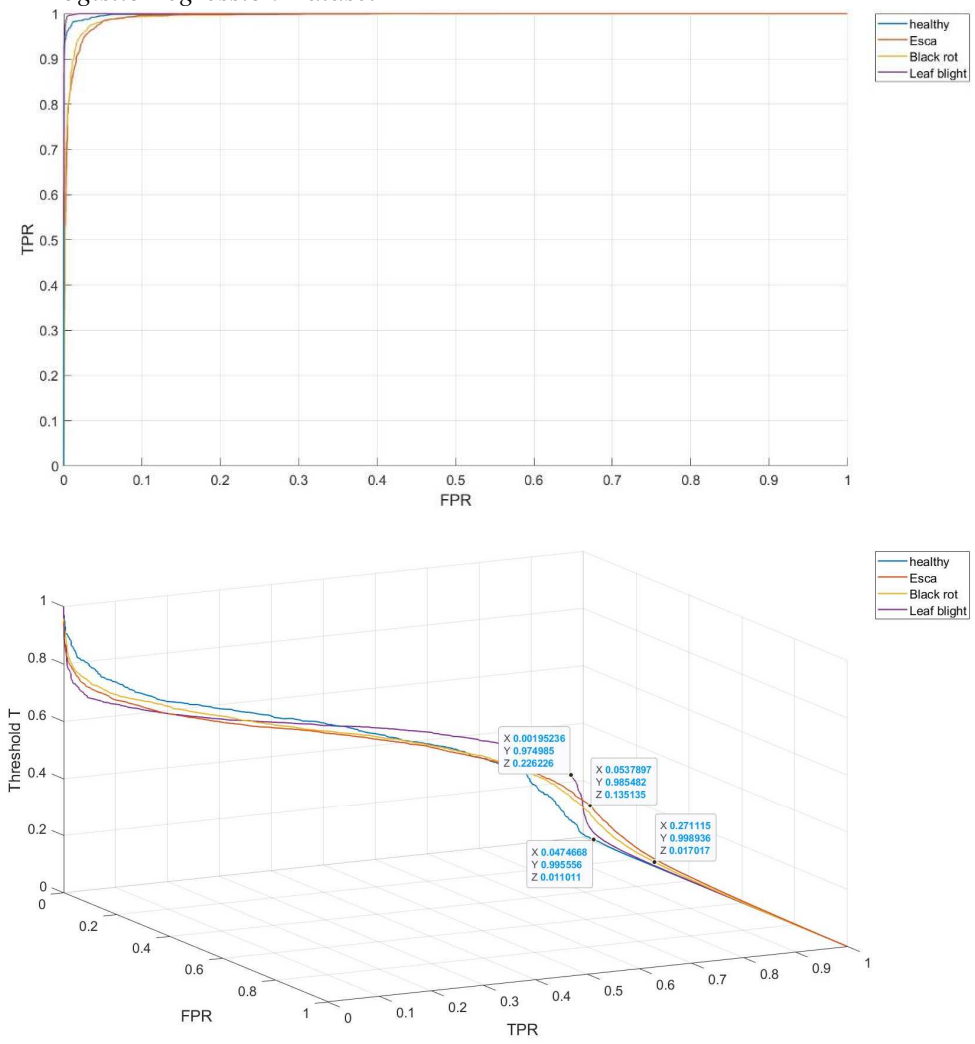


Figure 22 ROC curve, logistic regression, Dataset 1

Table 5 Logistic regression performances, Dataset 1

Avg AUROC	"Esca" AUROC	"Leaf Blight" AUROC	"Healthy" AUROC	"Black Rot" AUROC
0,9965	0,9936	0,9997	0,9987	0,9941

As the graph suggests (figure 22) a good compromise between FPR and TPR can be obtained picking up a T value in the up-left elbow. This is confirmed by the 3D view, since when the threshold increases its slope leaving constant TPR and FPR means that T is varying in a range where probability density functions of the two classes are quite negligible. Since the mentioned elbow is preceded and, also, succeeded by two threshold plateaus, it means that that range contains the optima threshold to split the classes. This argument will be implicitly used also in the other classifiers analysis. The Logistic Regression presents very small variance in the 4 classifiers, and the above-mentioned range is roughly comprised in $0{,}17 \leq T \leq 0{,}22$.

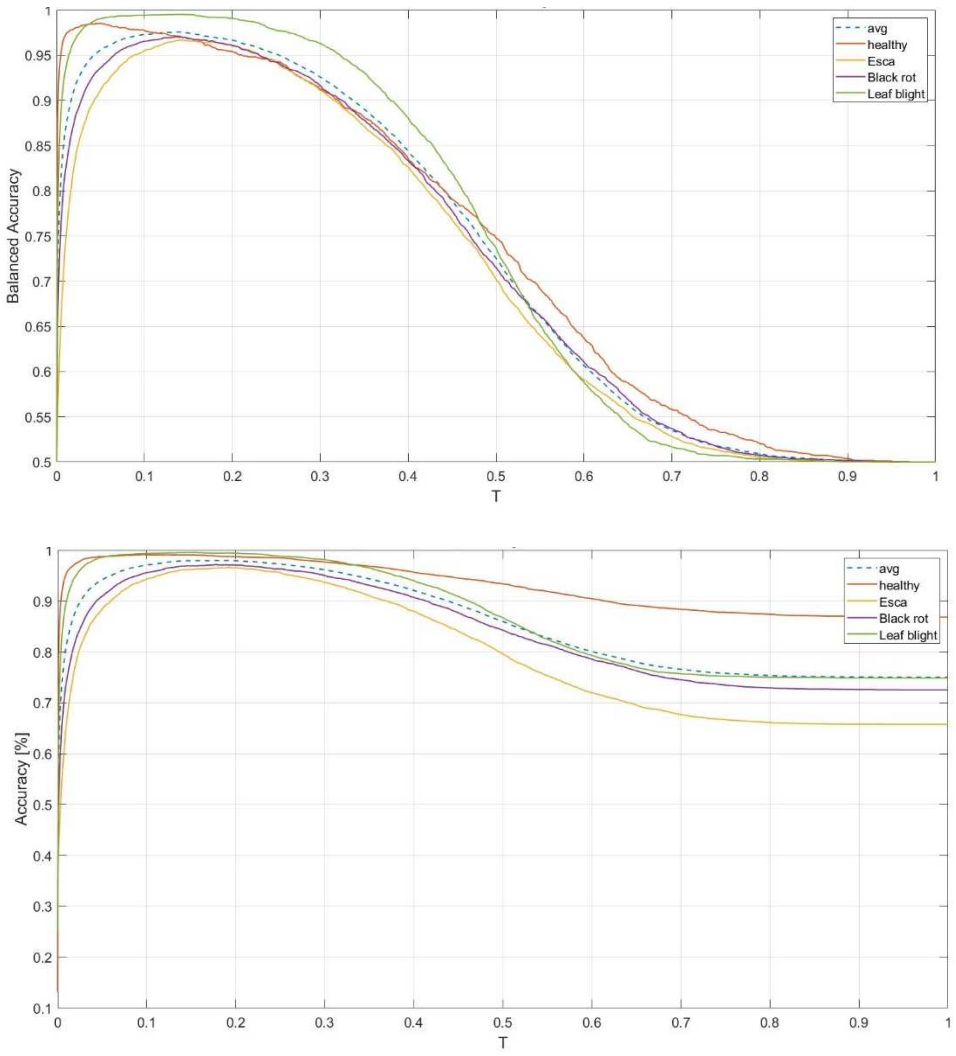


Figure 23 Balanced accuracy (Top) and accuracy (bottom), Logistic regression, dataset 1

That is predictable that, in an unbalanced dataset, the lowest appearing class ("Healthy") tends to have the highest accuracy (equals the frequency of its absence over the entire dataset) when the classifier becomes more specific and the lowest when it turns in more sensible (equals the frequency of its presence over the entire dataset), and vice versa for the highest appearing one ("Esca"). For those reasons the balanced accuracy tackles the issue. Generally, for the interesting range of T, "Leaf Blight" presents the highest accuracies, while "Esca" and "Healthy" the lowest

(figure 23). Averaging the 4 classifiers performances, in the logistic regression the maximum balanced accuracy reached is 97,57 % at a common threshold of 13,51%.

2.4.5.3 L-SVM Dataset1

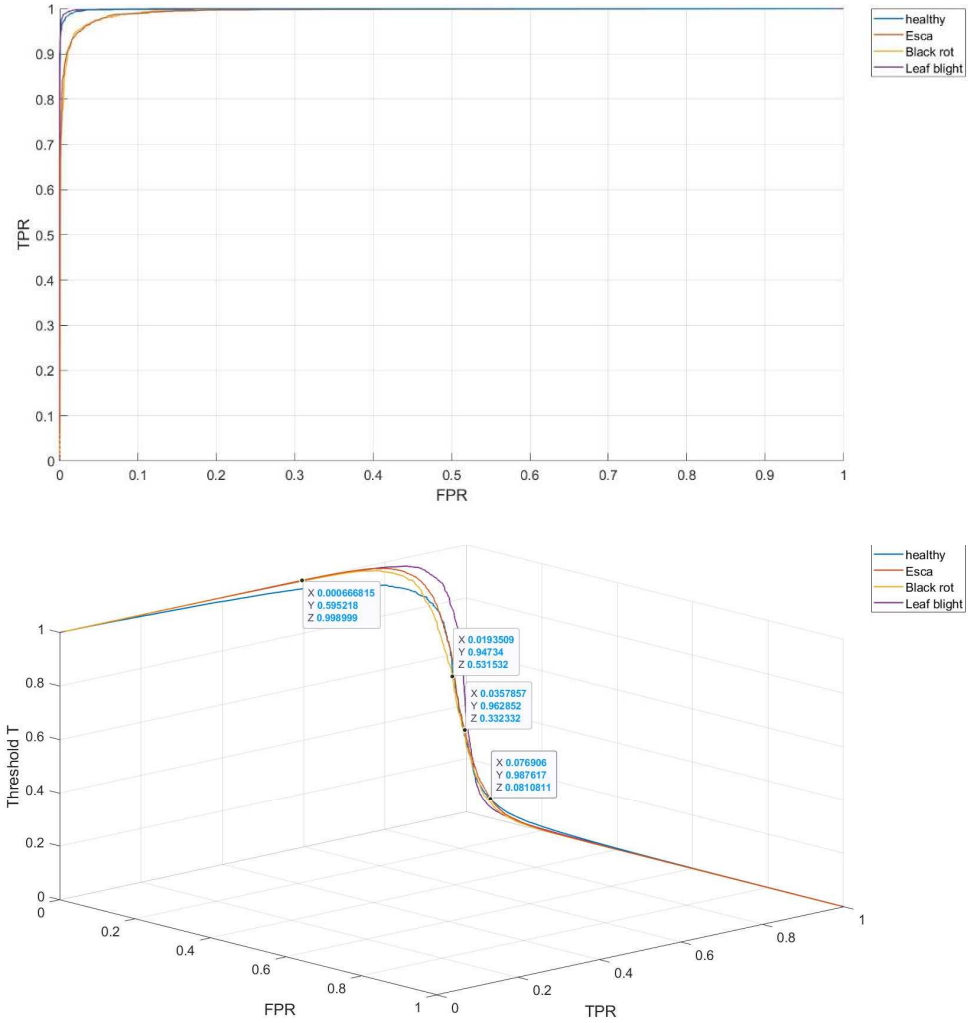


Figure 24 ROC curves, L-SVM, dataset 1

Table 6 LSVM performances, Dataset 1

Avg AUROC	**"Esca" AUROC**	**"Leaf Blight" AUROC**	**"Healthy" AUROC**	**"Black Rot" AUROC**
0,9966	0,9935	0,9996	0,9992	0,9940

Also, in this case the variance between classifiers is negligible, but the optimum threshold range is wider than the logistic regression, with $0{,}08 \leq T \leq 0{,}99$. This homogeneous behavior is caused by the geometric fashion employed to develop the model (figure 24).

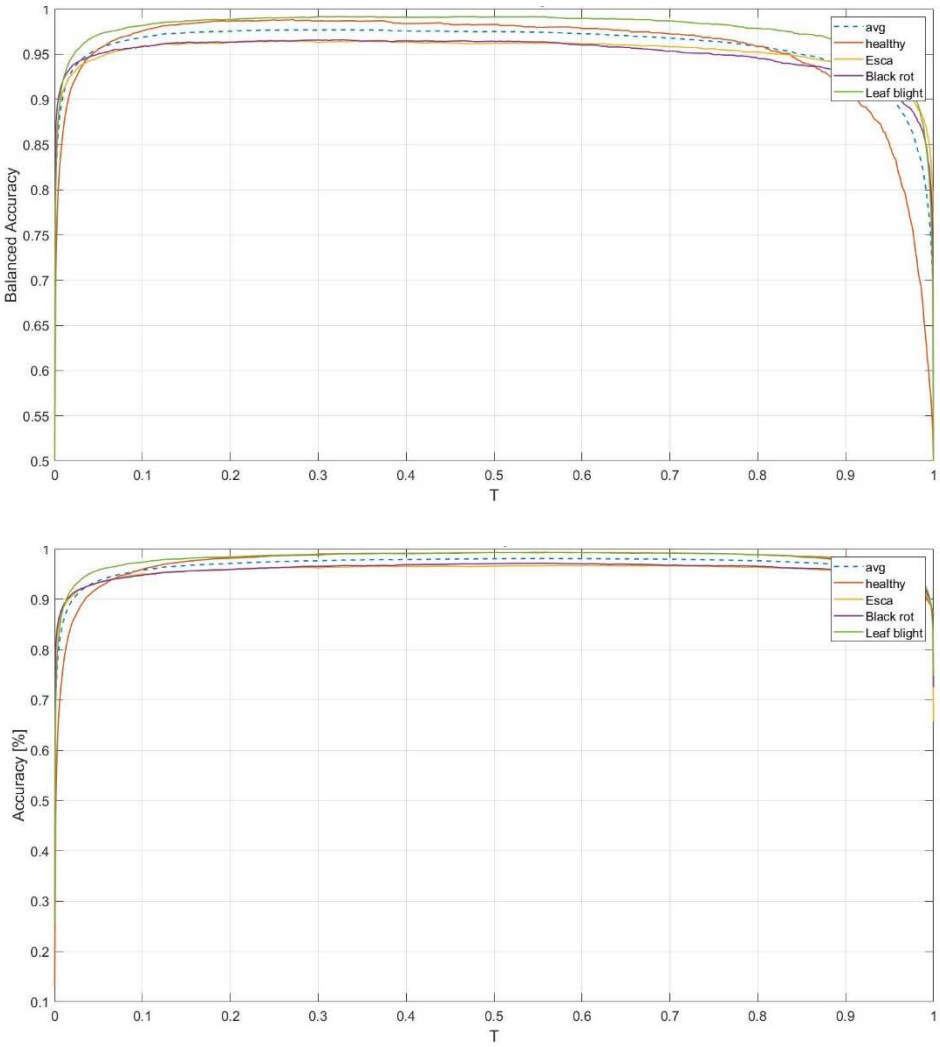


Figure 25 Balanced accuracy (Top) and accuracy (bottom), L-SVM, dataset 1

The general behaviors are like those of logistic regression, although in this case the lowest accuracies are for "Esca" and "Black Rot" classes across all the thresholding. Averaging the 4 classifiers performances, in the Linear Supported Vector Machine the maximum balanced accuracy reached is 97,71 % at a common threshold of 32,63% (figure 25).

2.4.5.4 *MobileNet-v2 Dataset 1*

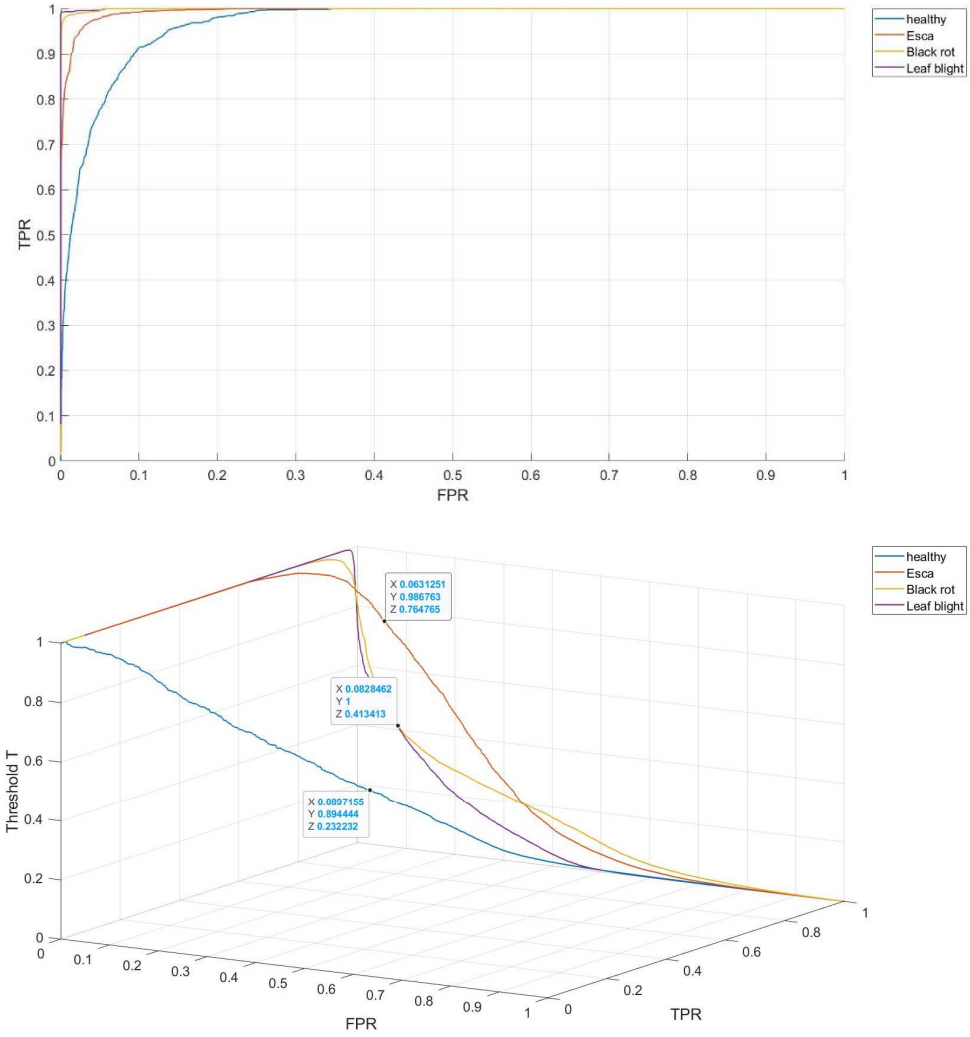


Figure 26 ROC, fully trained MobNet2, dataset 1

Table 7 MobNet2 performances, Dataset 1

Avg AUROC	**"Esca" AUROC**	**"Leaf Blight" AUROC**	**"Healthy" AUROC**	**"Black Rot" AUROC**
0,9901	0,9947	0,9997	0,9664	0,9993

For the full training of MobileNetv2 the results are quite different with the respect to the other two. First of all, can be noticed a high variance between the 4 classifiers, in particular "Esca" and "Healthy" curves slightly enlarge the average behavior (figure 26). Then is clear that for "Healthy" classifier the elbow point has not the fast slope seen until now, this means that was result difficult, in the training stage, to find a hyperplane able to strongly distinguish the positive and negative labels for this class, leading to a strong overlapping between the two probability density functions. This last anomaly results in the lowest AUROC index seen until now: 0,9664 for "Healthy".

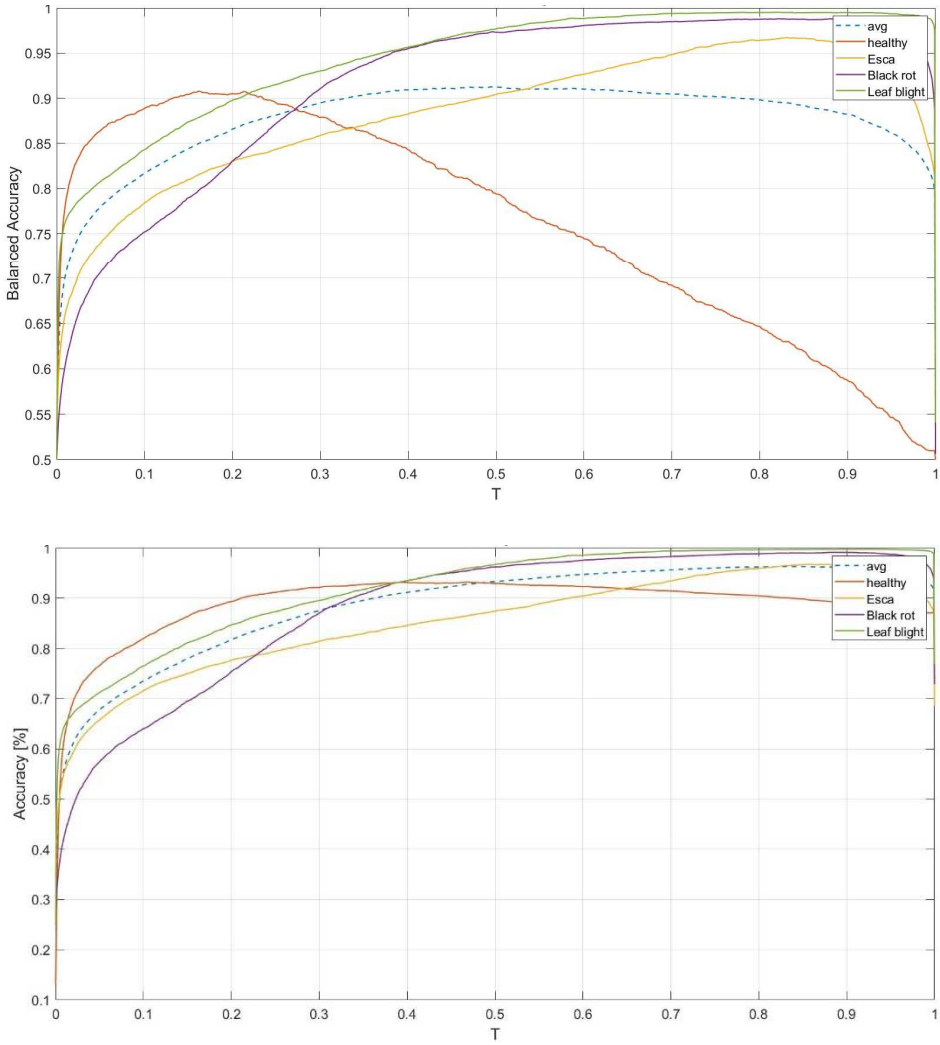


Figure 27 Balanced accuracy (Top) and accuracy (bottom), fully trained MobNet2, dataset 1

In this case the choice of threshold is not immediate because, although the average balanced accuracy has a constant behavior, it's not true for the individual classes. Indeed, can be seen that "Healthy" balanced accuracy has a dramatical falling around T=0,2, but the highest balanced accuracy, equals to 91,27 %, is for T=49,45 % (figure 27). For this reason, in this case, and, less urgently, in the other classifiers, the choice for the threshold could be better explained not only seeing the averaged balanced accuracy parameter, but also the individuals' balanced accuracies. Furthermore, the choice of T can be done individually for the 4 classes in order to tune the classifier for the specific needs. For example, is obvious that, in economic terms, false diseases detection is less dispendious than false "Healthy" detection in some cases. For the "Esca" disease a false negative means that a plant is left without future human prevention that has a fixed cost but taking the risk of the disease spreading over the neighborhood whose costs are arbitrary. In the case of the curable diseases "Black Rot" and "Leaf Blight" is not so obvious, because other factors, such as drugs consumptions for unit surface, and in that case an optimization function, dependent by the threshold T, can be minimized in order to achieve the wanted trade-off.

2.4.5.5 *Logistic Regression Dataset2*

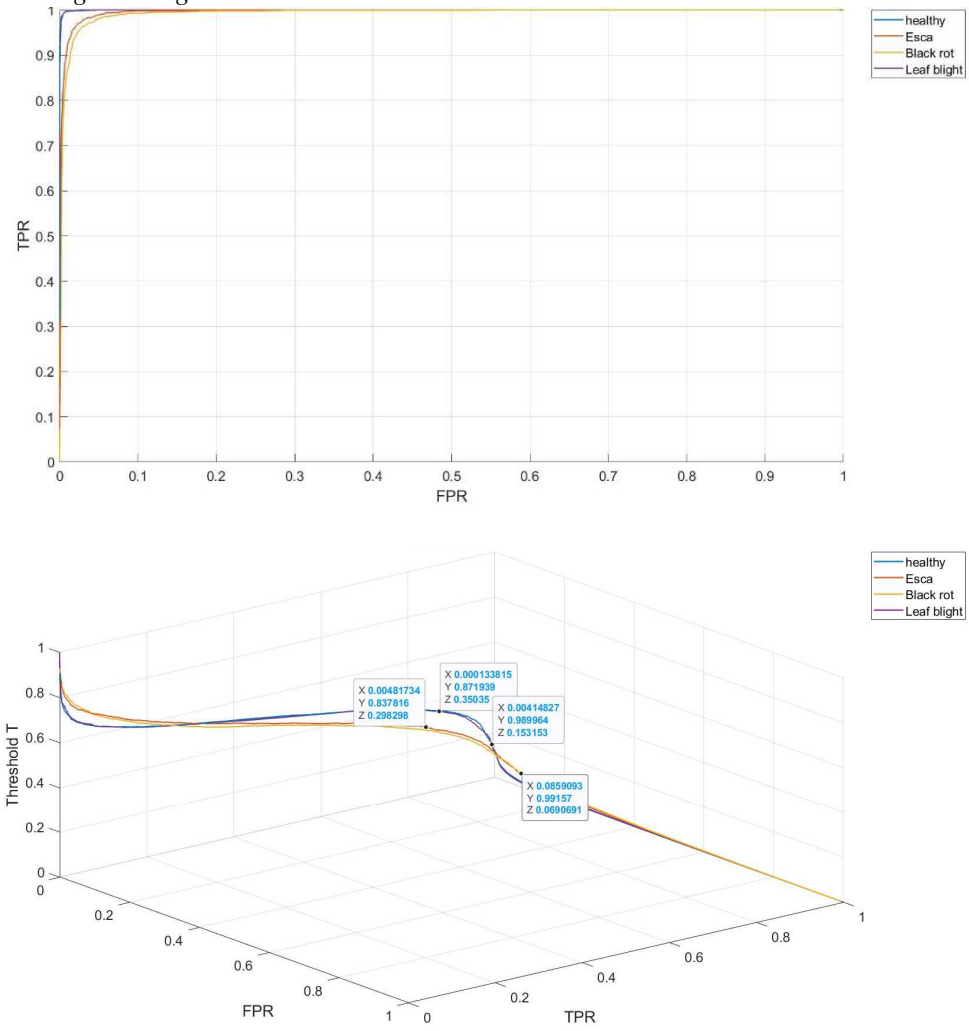
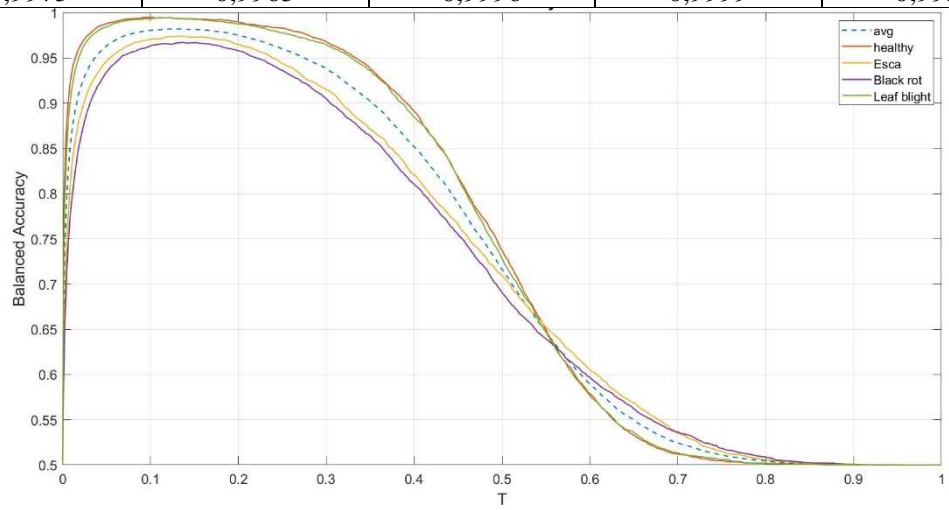


Figure 28 ROC, Logistic regression, dataset 2

Table 8 Logistic Regression performances, Dataset 2

Avg AUROC	"Esca" AUROC	"Leaf Blight" AUROC	"Healthy" AUROC	"Black Rot" AUROC
0,9973	0,9963	0,9996	0,9999	0,9935



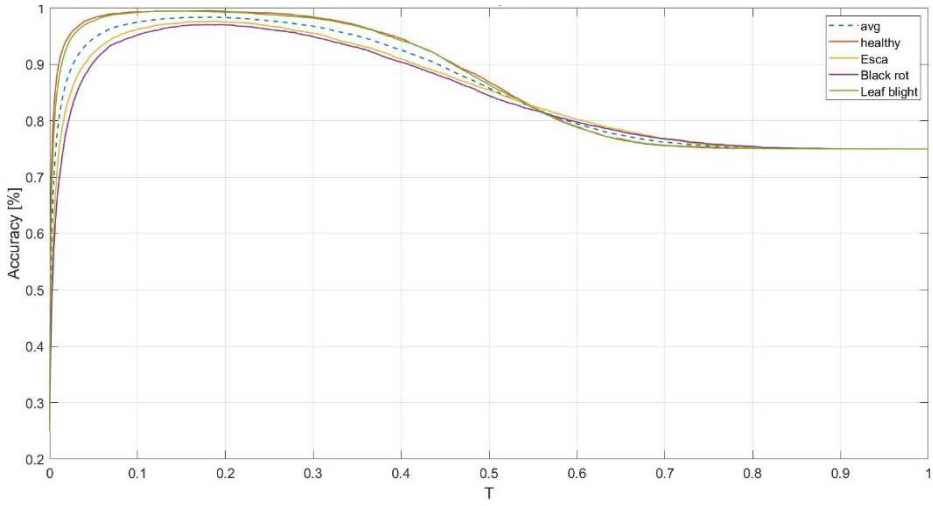


Figure 29 Balanced accuracy (Top) and accuracy (bottom), Logistic regression, dataset 2

Differently by the unbalanced case of Dataset1, now the Accuracy is comprised between 0,25 (for an extremely sensible classifier the accuracy is *nk/N*) and 0,75 (for an extremely sensitive classifier the accuracy is *(N-nk)/N*), as seen in figure 29.

2.4.5.6 L-SVM Dataset 2

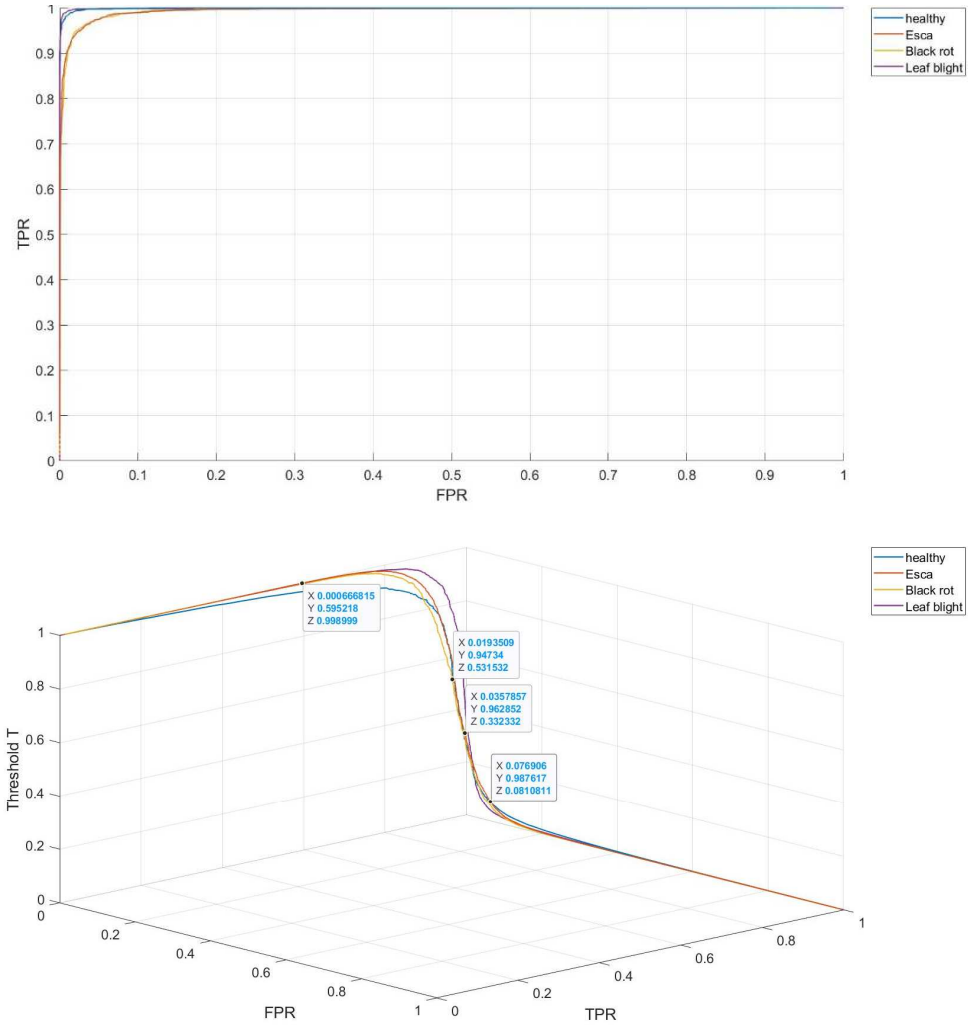


Figure 30 ROC, LSVM, dataset 2

Table 9 LSVM performances, Dataset 2

Avg AUROC	"Esca" AUROC	"Leaf Blight" AUROC	"Healthy" AUROC	"Black Rot" AUROC
0,9961	0,9958	0,9987	0,9998	0,9903

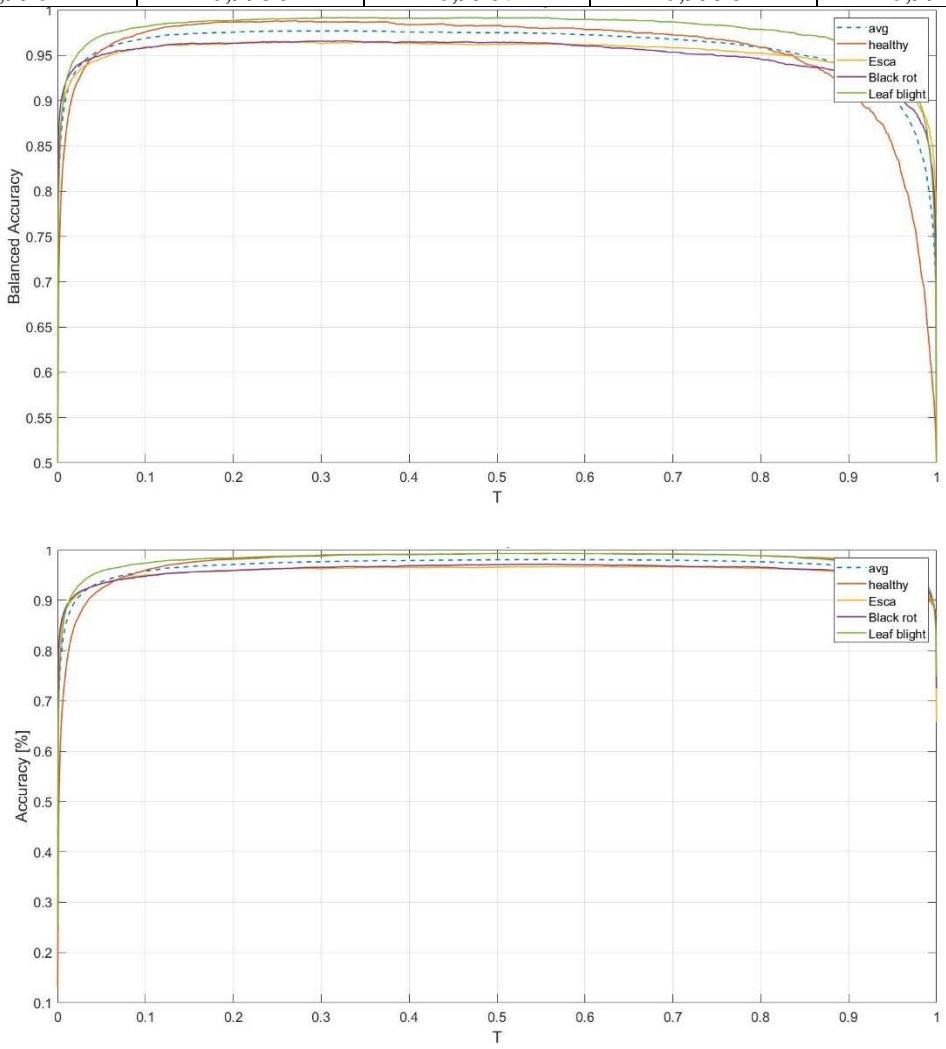


Figure 31 Balanced accuracy (Top) and accuracy (bottom), LSVM, dataset 2

2.4.5.7 *MobileNet-v2 Dataset 2*

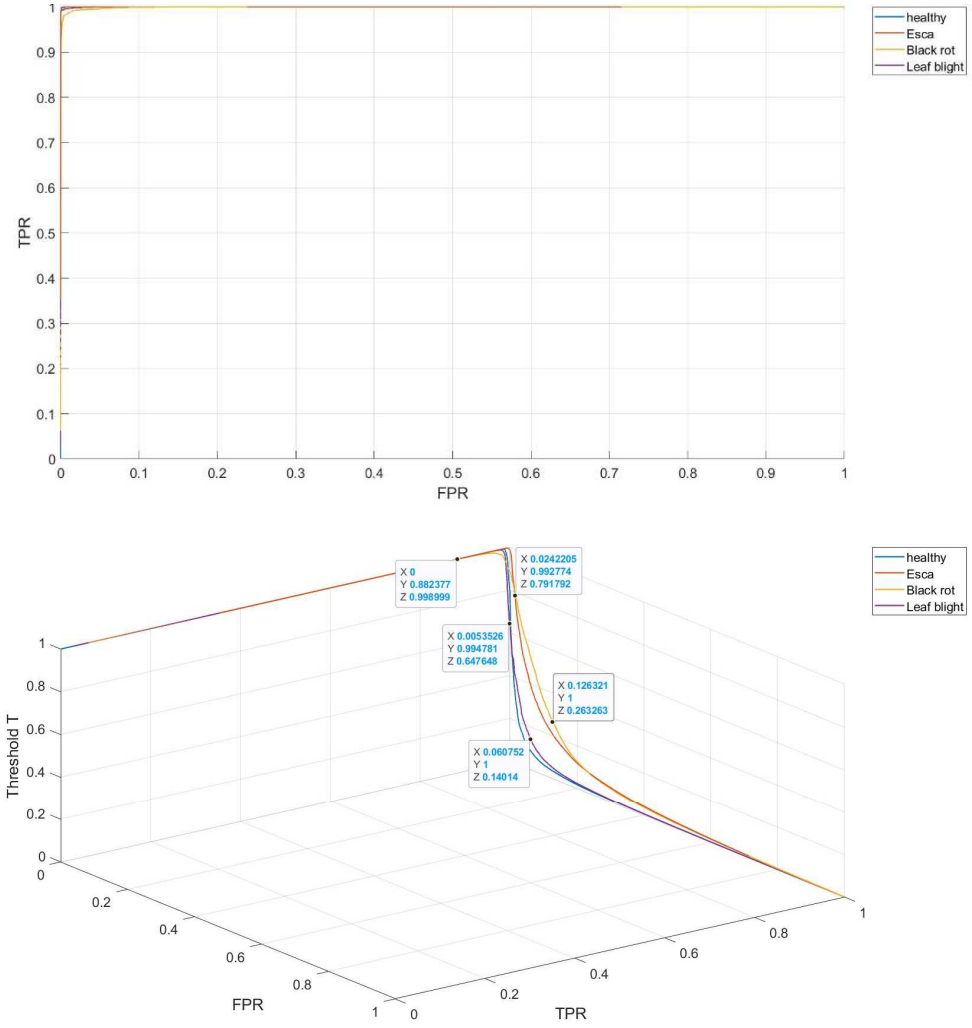


Figure 32 ROC, fully trained MobNet2, dataset 2

Table 10 MobNet2 performances, Dataset 2

Avg AUROC	**"Esca" AUROC**	**"Leaf Blight" AUROC**	**"Healthy" AUROC**	**"Black Rot" AUROC**
0,9998	0,9998	0,9999	0,9999	0,9994

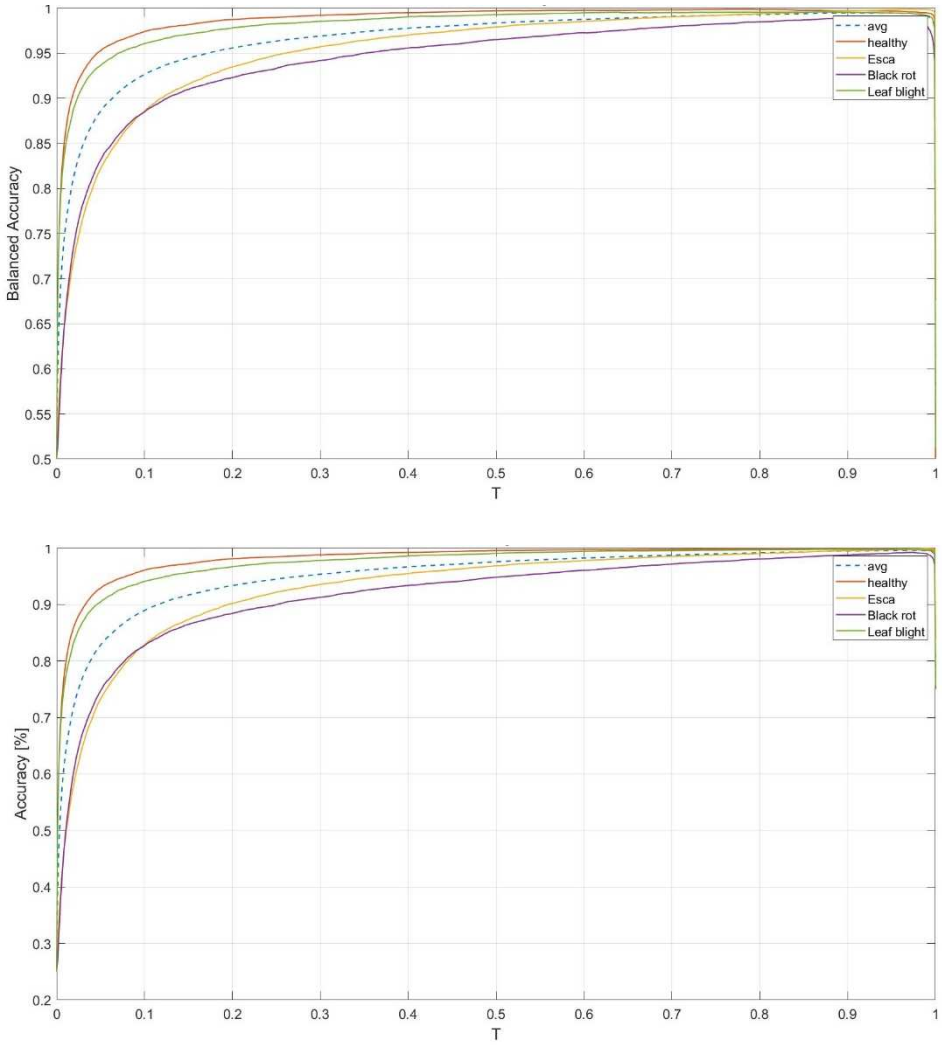


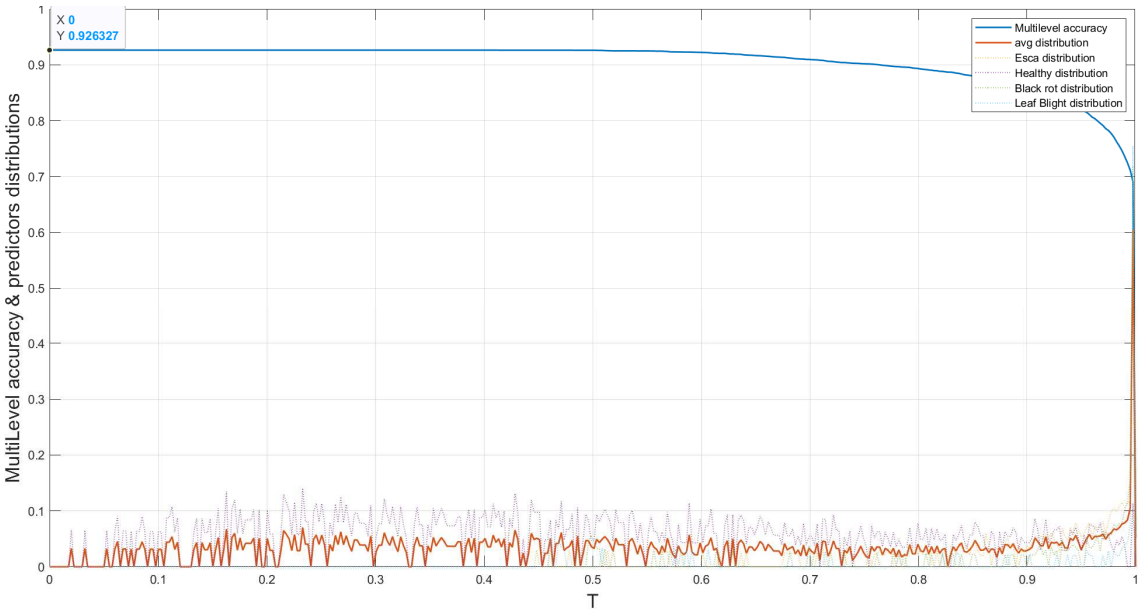
Figure 33 Balanced accuracy (Top) and accuracy (bottom), fully trained MobNet2, dataset 2

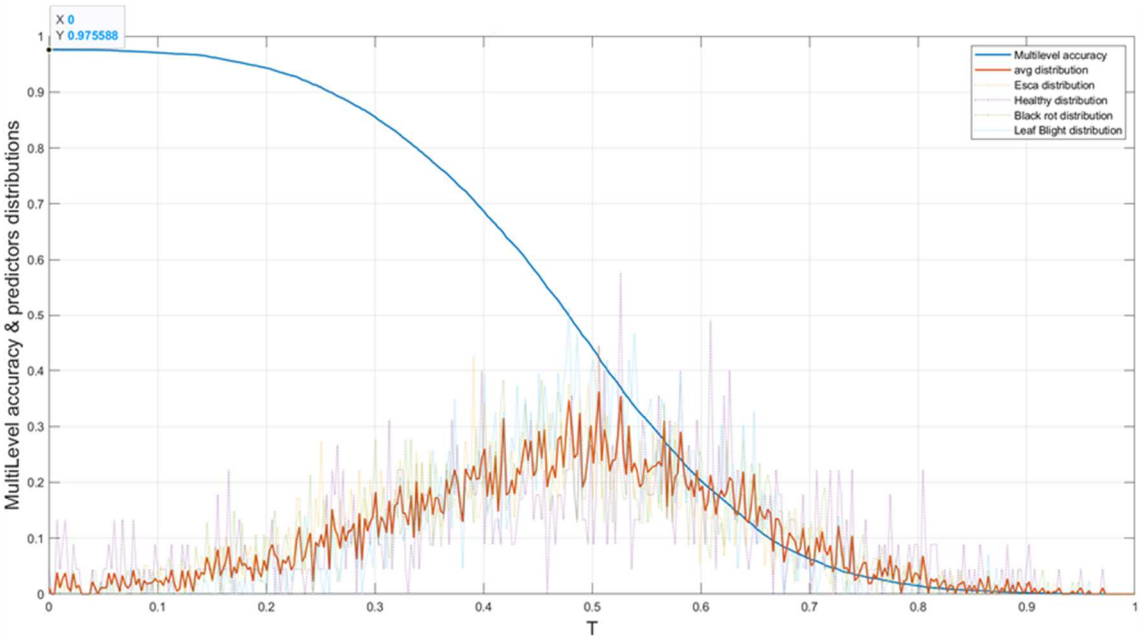
MobileNet2 for Dataset2 shows the best performances ever seen in this study, both in terms of accuracies and variances between classes. Indeed, it significatively overcomes the performances of the previous dataset, showing a quiet constant behavior among the classes, without anomalies.

2.4.5.8 Multi-Level accuracy

Previous analyses on single binary classifiers performances are very useful to visualize optimal thresholding in cases for which only one or few diseases are interesting to be recognized and insulated from other confounding classes. The fully exploiting of a **one-vs-all** multinomial classifier, instead, consist in the application of the algorithm discussed above in chapter 2.4.1. An image enters as input of the classifier, 4 distances are computed, one from each class hyperplane, then if none of them is higher than a threshold *T* (converted in percentage) the classification outputs "Confused", otherwise outputs the class for which the distance is the highest. With this approach multinomial thresholding assumes a new meaning. threshold must be set with a trial-and-error fashion based on some on field experiment, since its scope is to reject or allows potential other out-of-class diseases to increasing or decreasing specificity of the whole classifier. In our case there is a closed set on which analyses are done, without labeled confounding classes, so increasing the threshold has the only effect of start decreasing the Multilevel accuracy at some point (figures 34,35). That threshold corresponds with the minimum image posterior computed as sigmoid of the distance from the correct class of that image. In other words when the general threshold increases all the hyperplanes are shift forward (to positive regions), so the hyperplane "k" with the nearest image/point *belonging to its class k* starts to approach the clouds of points of

its class then the accuracy starts to fall. Before that critical threshold the multilevel accuracies are always the same, and the maxima, equal about the maximum average balanced accuracies computed before and critical thresholds about equal thresholds $\boldsymbol{T^*}$**,as experimentally seen**. The Multilevel accuracy is computed dividing the correct classifications $n_{cor}(T)$ over the entire dataset size N and can assume all values between zero (all Confused prediction) and 1.





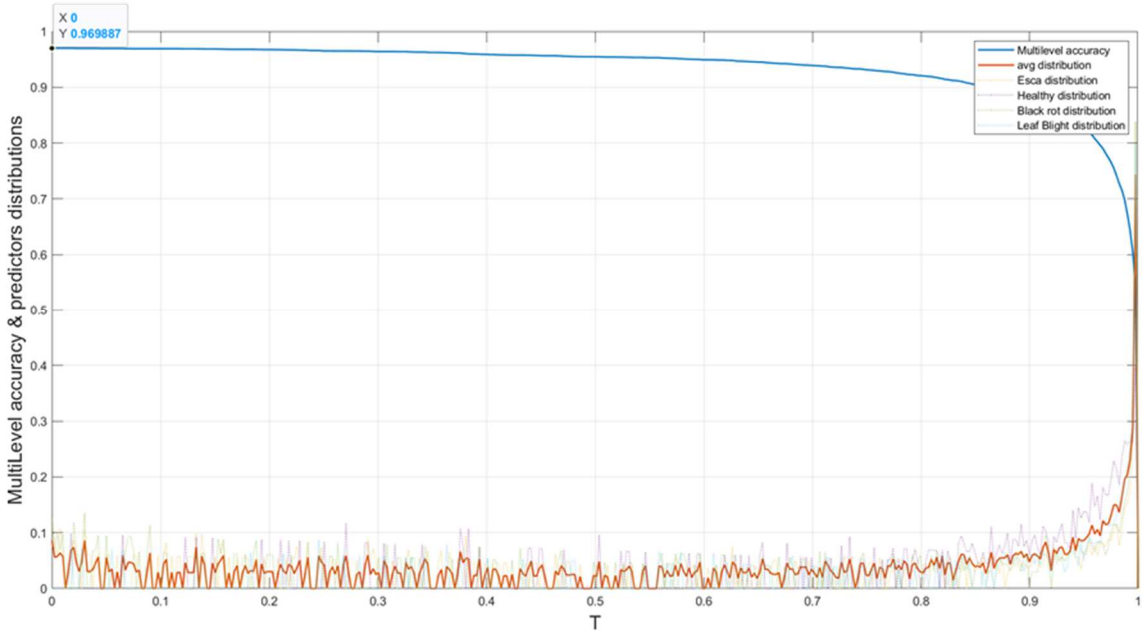
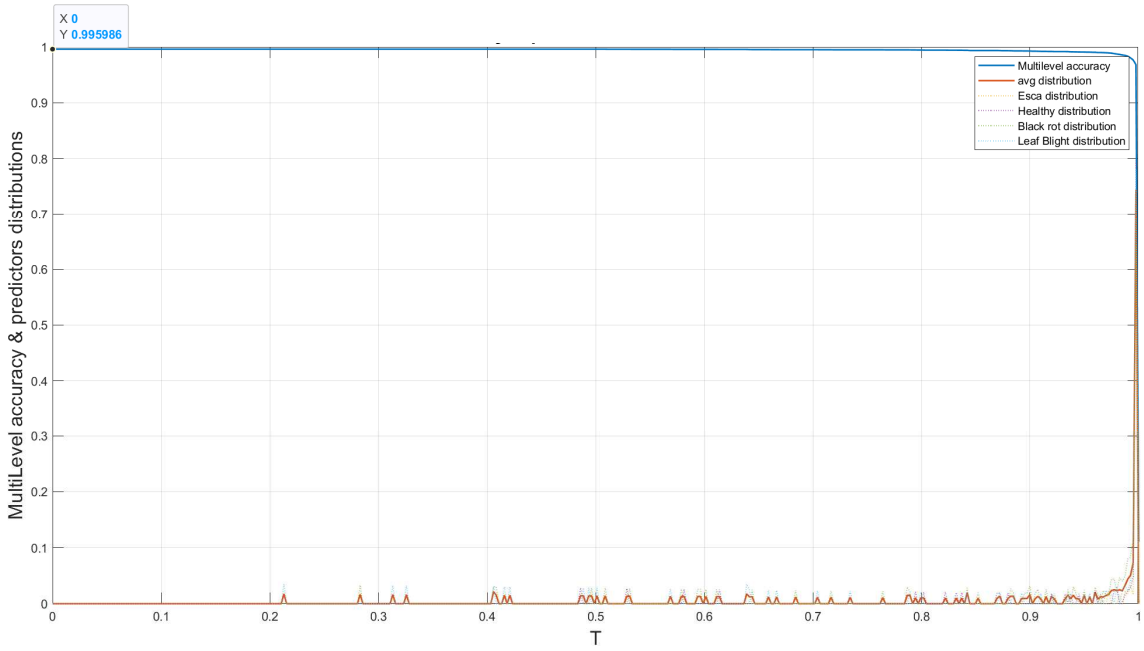


Figure 34 Multilevel accuracy and predictor density distributions of Test set of Mobilenetv2(top), Logistic Regression (central), LSvm (bottom) trained on Dataset1



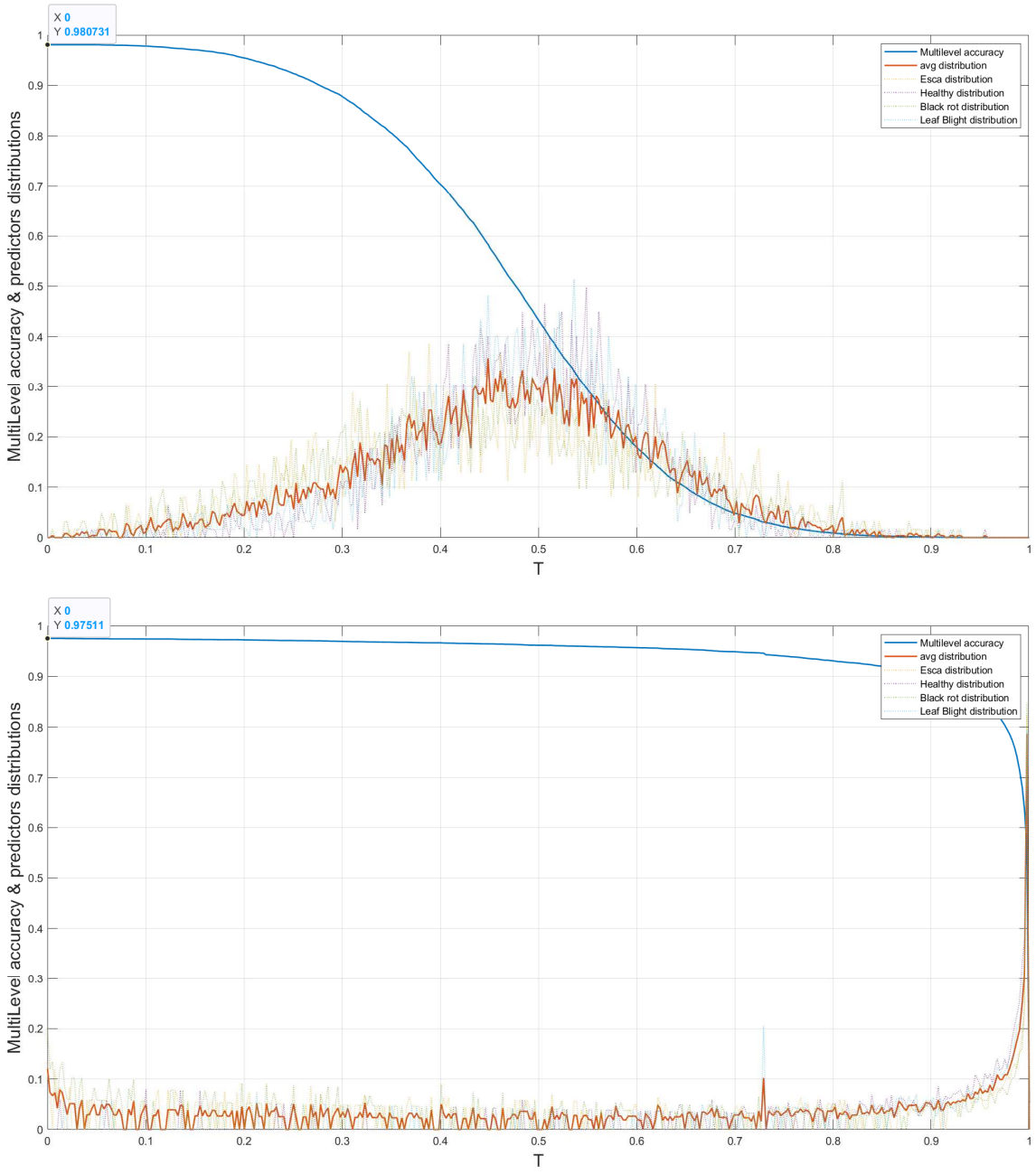


Figure 35 Multilevel accuracy and predictor density distributions of Test set of Mobilenetv2(top), Logistic Regression (central), LSvm (bottom) trained on Dataset2

Distributions are obtained computing, for each hyperplane, distances of images labeled with the class of that plane, then such distances are converted in percentage passing by a sigmoid and are plotted their frequencies over the sub-dataset, with size N_k, containing only their class in intervals of 0,0025 between 0 and 1.

$$\begin{cases} Accuracy_{multi}(T) = {}^{n_{cor}(T)}/_{N} & ;n_{cor}(T) \equiv correct\ classifications\ with\ threshold\ T \\ Distribution_k(T) = {}^{n_k(T)}/_{N_k} & ;n_k(T) \equiv images\ i \in k_{subset}\ with:\ sigm\left(b_k + \bar{w}_k^T \bar{x}_i\right) = T \end{cases} \quad (2.40)$$

For Logistic regression distributions a 40x factor was applies to highlight the curve, showing a large hell centered in 0,5. LSVM and MobileNet2 instead shows a dramatic increasing number of correctly classified images toward 95-100%, indeed a nonlinear vertical stretching of the density distribution was applied in order to empathize the behavior. All the original density function had integral equal to for T between 0 and 1. This confirms that logistic regression hyperplane is over

its right position of about 30-40%, while LSVM and mobile net are shifted forward, with respect to this test set.

2.4.5.9 Results summary

The following table summarizes performances in terms of averaged Auroc, Auroc variance, maximum averaged balanced accuracy and balanced accuracy standard deviation at that optimum, threshold of maximum averaged balanced accuracy, total training time and single classification test time.

Table 11 Training and Validation summary, Dataset 1

Dataset1	**Avg AUROC**	**AUROC variance**	**Max avg balanced Accuracy %($at\ T^*$)**	**Balanced Accuracy std dev %($at\ T^*$)**	**Optimal threshold T^* %**	**Training time (mins)**	**Avg Test time (secs)**
Logistic Regression	0,9965	$9{,}7\ 10^{-6}$	97,57	1,31	13,51	110,14	0,3
L-SVM	0,9966	$1{,}07\ 10^{-5}$	97,71	1,41	32,63	1323,53	0,3
MobileNetv2 Full Training	0,9901	$2{,}52\ 10^{-4}$	91,27	8,36	49,45	412	0,3

Table 12 Training and Validation summary, Dataset 2

Dataset2	**Avg AUROC**	**AUROC variance**	**Max avg balanced Accuracy %($at\ T^*$)**	**Balanced Accuracy std dev %($at\ T^*$)**	**Optimal threshold T^* %**	**Training time (mins)**	**Avg Test time (secs)**
Logistic Regression	0,9973	$9{,}17\ 10^{-6}$	98,21	0,3	13,81	84	0,3
L-SVM	0,9961	$1{,}79\ 10^{-5}$	97,72	1,7	43,34	5760	0,3
MobileNetv2 Full Training	0,9998	$7{,}1\ 10^{-8}$	99,45	0,4	93,39	190	0,3

Both training and test time referrer to a MATLAB script timing ran on an Acer Nitro AN515-51 with 16 GB Ram, intel core i7-7700HQ with 2,81 GHz of clock and 8Mb cache, furthermore, test time concerns only the feature extraction and the prediction phase, being the Network weights yet pre-allocated in memory. Is worth noticing that, comparing the 3 methods in a new test stage, Logistic Regression was also the more robust to recognize the Confused class both for a randomized out-of-classes leaf and for a completely random image, but, on the other hand, seems also to be too much specific and prone to false "confused". However, a numeric evaluation was not performed since a control dataset of out-of-classes leaves was not enough for a reasonable analysis.

2.5 Plant level leaves detection

Pictures shot by the UAV acquisition system are not necessarily leaves in foreground, but there is also the possibility to capture whole vinegars comprising trunks, sky and other objects. In order to make the above classifier robust with respect to so complex images, a leaves detector algorithm is necessary. Main possibilities are:

- Segmentation of the image, using computer vision features
- Convolutional Neural Networks for object detection

CNN can be used as regressors to output a set of boxes that are likely to contain an object or taking in input a set of Region of Interest (RoI) from another block. There are many networks implementing these approaches.

- **Sliding-Window** is the slower approach since each candidate box is sent to the feature extractor and then classified.
- **Two-Step Detectors** is used a mechanism that infers a certain number of boxes, then they are sent to the backbone CNN (R-CNN, Fast and Faster R-CNN).
- **One-step Detectors**, the above two steps are done simultaneously (RetinaNet, YOLO)[7].

An implementation of Fast/Faster RCNN can be easily performed altering MobileNet and inserting a Region Proposal network at an empirically chosen feature extraction layer. Then, splitting the final subnetworks in classification and regression, the output should represent the extracted boxes and their relative classification scores. This technique can be very efficient because many layers of the CNN are shared by the classifier and the regressor. The main problem of this technique is that it needed a training on the specific dataset used. Do such a training meant prepare the dataset of thousands of images with a labeler software, marking, for each image, a reasonable number of leaves in a semi-automatic way and then fed all the result in the final training of the network. Being the classifier learned on datasets where appears both single leaves that natural and artificial set of leaves for each class, the priority is to filter the on-field image leaving to pass only boxes that contains a reasonable high number of leaves with respect to other confounding objects and backgrounds. Such an aim can be obtained also with an algorithm that is agnostic on the dataset and objects type. Many segmentation algorithms are based on an edge detection phase and then on local or global object researching phase. Edge detection is performed applying Gradient and/or Laplacian to the image (or Canny methods etc..), then thresholding it a good representation of low-level information of the original image is obtained[8]. This resulting image goes into the detection stage, where edges are grouped following certain affinity functions (Hough transformation, thresholding etc..). Then, often, a sliding window approach is performed, scanning all the image with a sub-sized matrix and scoring each of them on the base of their objectiveness. The objects detection stage must increase the efficiency of the classification stage, so to have a detector slower than the backward classifier has no meaning. Many segmentation techniques are far to be suitable for real time applications, while the Edge Boxes segmentation technique shown a possible real time application, reaching speed of one detection in a quarter of a second with the right parameters[21]. This technique uses the online available Structured Forest for Edge detection to construct in a cost-effective way the initial edges map (figure 36).

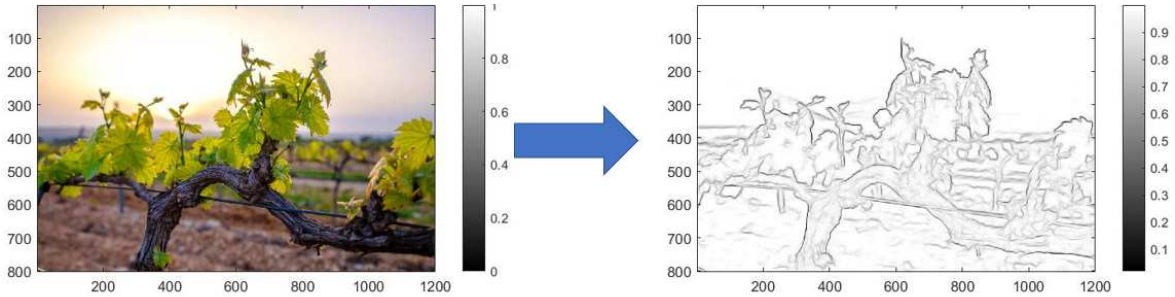


Figure 36 Edge Boxes segmentation

Then a set of boxes are proposed with different sizes, aspect ratios and positions. The worth observation of the authors is the correlation between the number of edges wholly enclosed in a certain box and the likelihood of that box to contain a complete object.

Figure 37 Edge Boxes implementation, object detection

To ensure real time computation is advice a set of parameters, also used in this study. They are:

- α= 0,65 is the step size of sliding window search
- β= 0,75; is the Non-Maximal Suppression threshold for object proposals
- Minimum Score = 0,01; is the minimum score of boxes to detect
- Max number of Boxes proposed = 10^6

With this set of parameters, the algorithm is able to perform object detection in about 0,04 seconds with a tested image of 224x224 resolution and about 0,8 seconds with a 640 x 960. Among the outputted boxes (i.e., with a score greater than the imposed threshold) a manual extraction of firsts k scored boxes is performed in order to reduce the number of final images to be classified, furthermore is inserted the parameter s who excludes, in the set of the k elements, the firsts s. The last parameter was inserted after having noticed that, for a typical on-field image, greatest scores were given to boxes of size comparable to the original image. The correct trade-off chosen is:

- s=4
- k=6

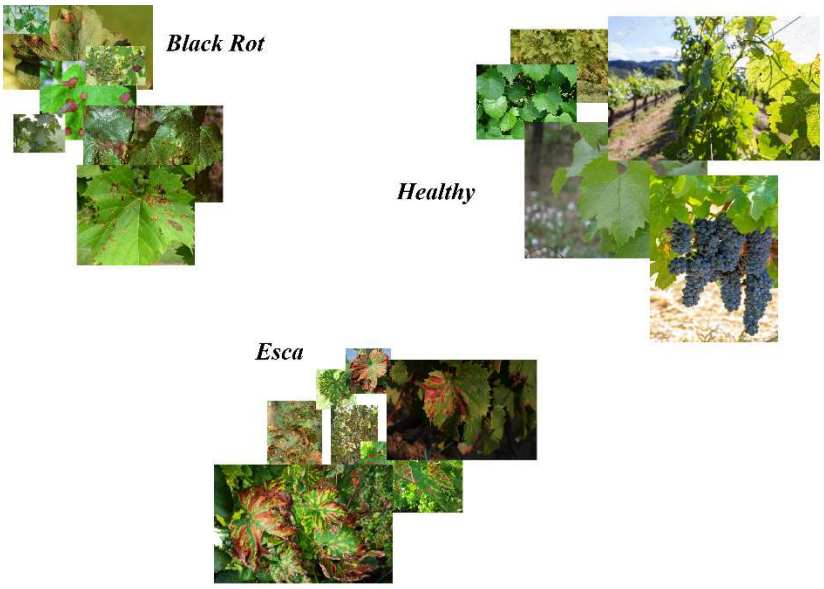
So, for each image, the whole algorithm spends a time equal to:

$$time_{total} = (k - s + 1)\, time_{classifier} + time_{detector} \qquad (2.41)$$

For example, to classify a 640x960 input image a total time of 1,73 seconds is needed (on the machine used). Structured edge detector and EdgeBoxes algorithms are available on GitHub.

2.5.1 Validation on an external dataset

Both on dataset1 and dataset2 classifiers a further validation on on-field images coming form an external dataset was required, in order to evaluate the robustness and weakness points. The external dataset contains online available images from Google with different sizes and sources. "Black Rot", "Esca" and "Healthy" class contains 49 images each one, while resulted very difficult to find Isariopsis Leaf Spot (the specific "Leaf Blight" case examined here) images for grapevine leaves. So was decided to exclude "Leaf Blight" class by the 2nd Validation stage. This choice can be considered negligible in Dataset2 classifier, since is probable that with same conditions the behavior is homogenous for the 4 classes, but not in Dataset1 classifiers where the unique class trained on similar condition of "Leaf Blight" is "Black Rot". Obliged by available online images, properties of images were not very homogeneous among the 3 classes. "Black Rot" and "Esca" folders contain images nearer to crown than "Healthy" class, and this could be the reason for which "Healthy" class performances dramatically drop for some conditions.



The validation was performed with and without the Object Detector.

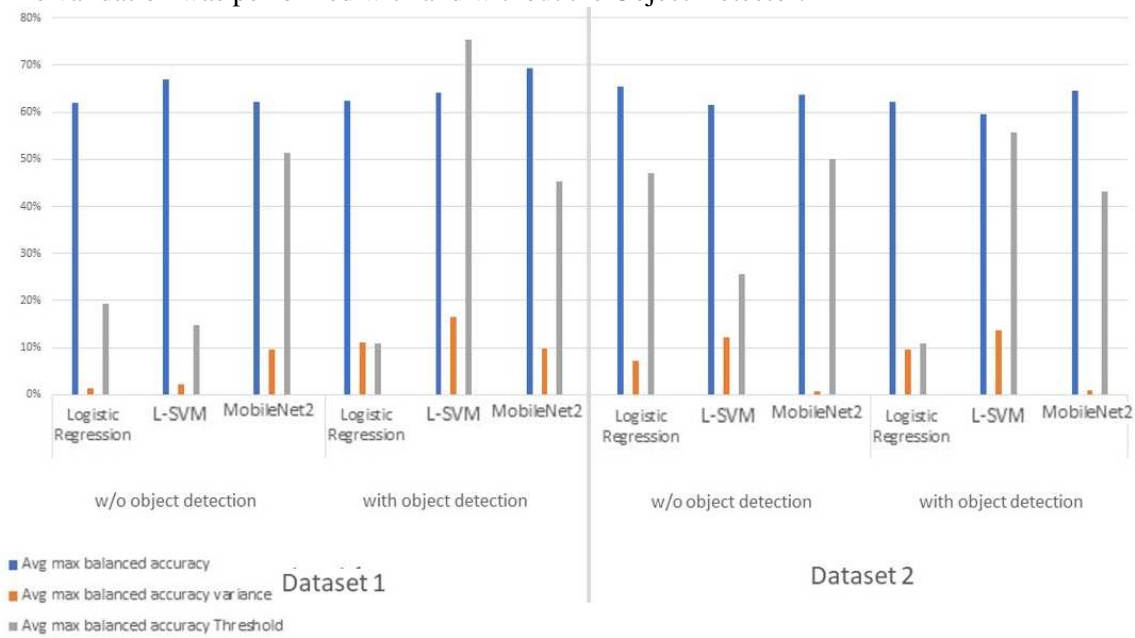


Figure 38 Maximum Averaged Balanced Accuracy; Maximum Averaged Balanced accuracy variance; Maximum Averaged balanced accuracy threshold

Logistic Regression and MobileNet2 classifiers improve their balanced accuracy in Dataset2 with respect Dataset1 without object detection, instead the implementation of the object detector in Dataset1 substantially does not affect Logistic Regression and strongly improves MobileNet2 performances of about 7%, while, surprisingly, in Dataset2 the object Detector reduces of 4% balanced accuracy of Logistic regression, leaving unvaried the MobileNet2 one. L-Svm classifier is in general more accurate in Dataset1 and reduces its performances with object detector in both datasets.

In Dataset2 the variance of the maximum balanced accuracy is much higher than in Dataset1 touching about the 10% for all the classes but MobileNet2 for which is observed the opposite behavior. Furthermore, for all the datasets, introducing the object detector L-Svm and Logistic Regression deeply increase their variance, while MobileNet2 variance seems unresponsive to object detection.

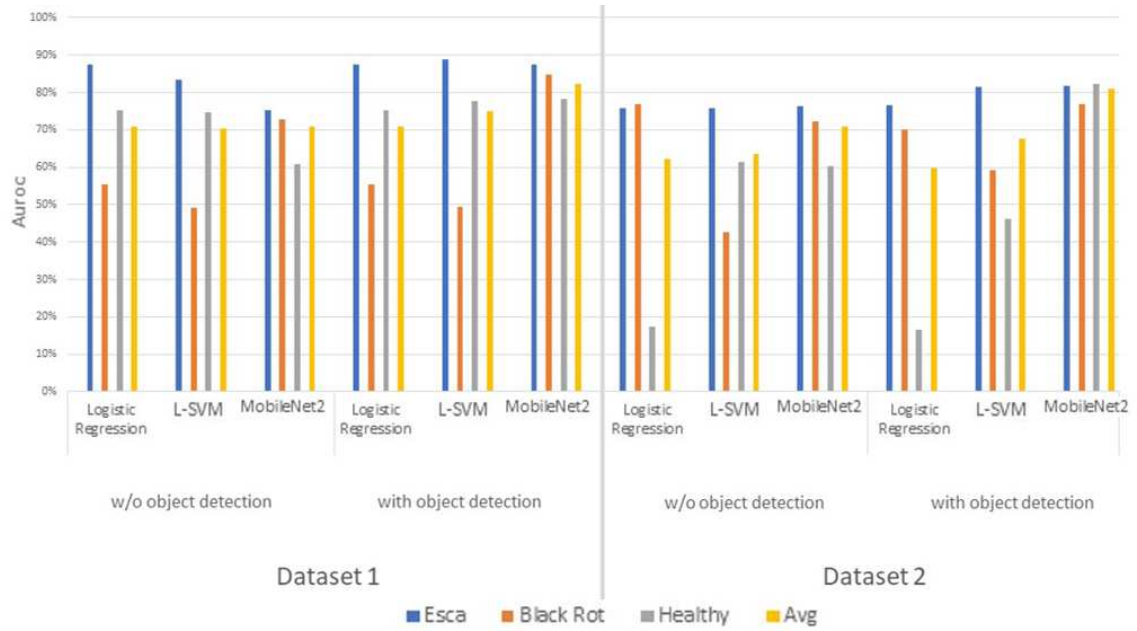


Figure 39 Area Under the ROC for each class and averages

The strong effect of object detection can be shown through Auroc coefficients. The introduction of Object detection does not affect Average Auroc of Logistic Regression but increases it mildly (5-6%) for L-Svm and highly (10-11 %) for MobileNet2, showing that object detectors not only have importance for a correct classification, but also for the good tradeoff between sensitivity and specificity, allowing to better recognize Confused prediction also. Auroc in Dataset1 are generally significatively higher than in Dataset2, but not for MobileNet2, who seems to be invariant with respect the dataset employed for its training. "Esca" is the most accurately classified class and, excluding wholly MobileNet2 and L-Svm without object detection in Datset2, is followed by "Healthy" and "Black Rot" in Dataset1 or by "Black Rot" and "Healthy" in Dataset2. "Healthy" Auroc dramatically drop in Dataset2 for Logistic Regression reaching so low values to become useless. For the exception cited the behavior of 2nd and 3rd positions are inverted.

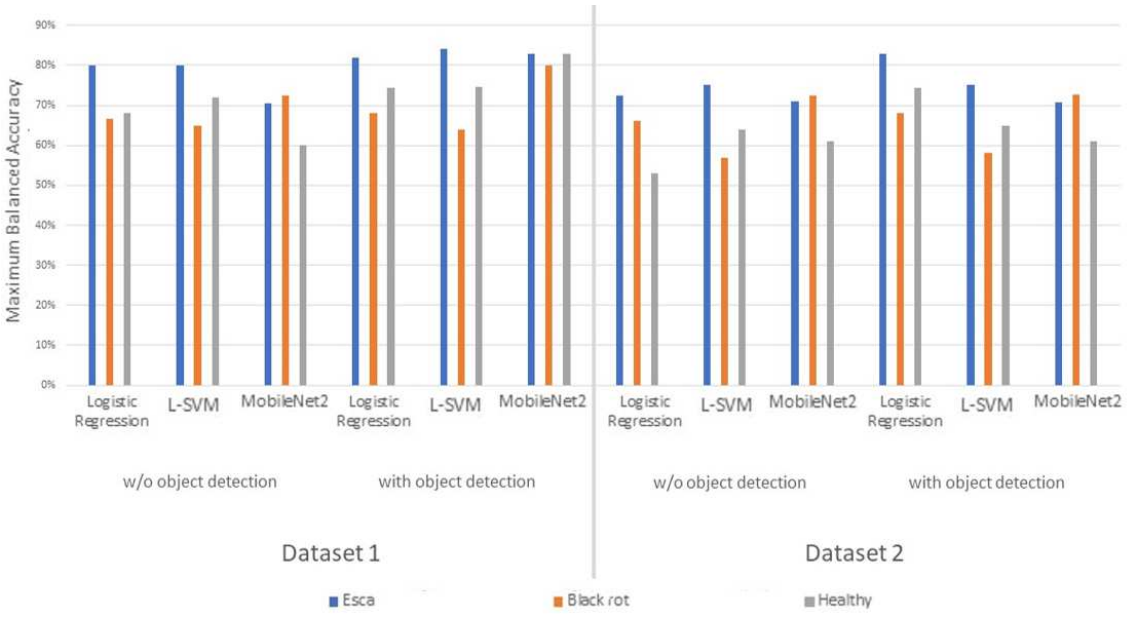


Figure 40 Maximum balanced accuracy reached by each class, given a classifier, with and without object detection

In figure 40 are represented the maximum balanced accuracies for each class, independently by the average balanced accuracies. That implies different optimal thresholds for each class in the

generic classifier. This threshold values are quite different by case, but in general “Esca” threshold are higher than “Black Rot” and “Healthy” thresholds. For Logistic Regression “Esca” optimal thresholds are always between 20% and 30%, while “Black Rot” and “Healthy” thresholds between 1% and 10 %. For L-Svm “Esca” optimal thresholds are between 30% and 60% and between 1% and 20% the other two. Finally, in MobileNet2 “Esca” optimal thresholds are around 80-90%, while between 15% and 50% the other two. For what concerns relationships with object detection and Datasets this graph suggests the same seen for Auroc graph.

2.6 Conclusions and suggestions

There is not a unique best choice among all the possible configurations explored. Fully trained MobileNet2 classifier shows the better performances in terms of balanced accuracy, Auroc and variance when an object detector is set upped in both datasets, but if the choice falls on a faster algorithm then, without object detector, the three classifiers becomes affordable about in the same way (in particular best overall performances are seen in dataset 1 although the high false positives linked to “Esca” for the unbalanced training) and at that point the right choice could be done based on the specific weight assigned to different diseases depending on the application. “Esca” and “Healthy” classification are often the bests. This behavior seems to be not related with the richer information provided by UMD Dataset for those classes, because the same, at least for “Esca”, happens in the second Dataset where on field images were not employed for the training. A reasonable argument could be the manifest detectable symptoms of “Esca” disease, clearer and more distinguishable also by human eyes. Moreover, often exceeding the 80% balanced accuracy “Esca” classification can be already thought for an implementation, while the others are less affordable and need a more accurate dataset to be improved. In that sense, this work is flexible. Starting by a not-binary categorical classification training, the recognition phase can be focused only on a binary fashion, simply deactivating the recognition of other classes and classifying them as “Confused” class depending on the scope. These are not only possibilities, but also necessity in some cases.

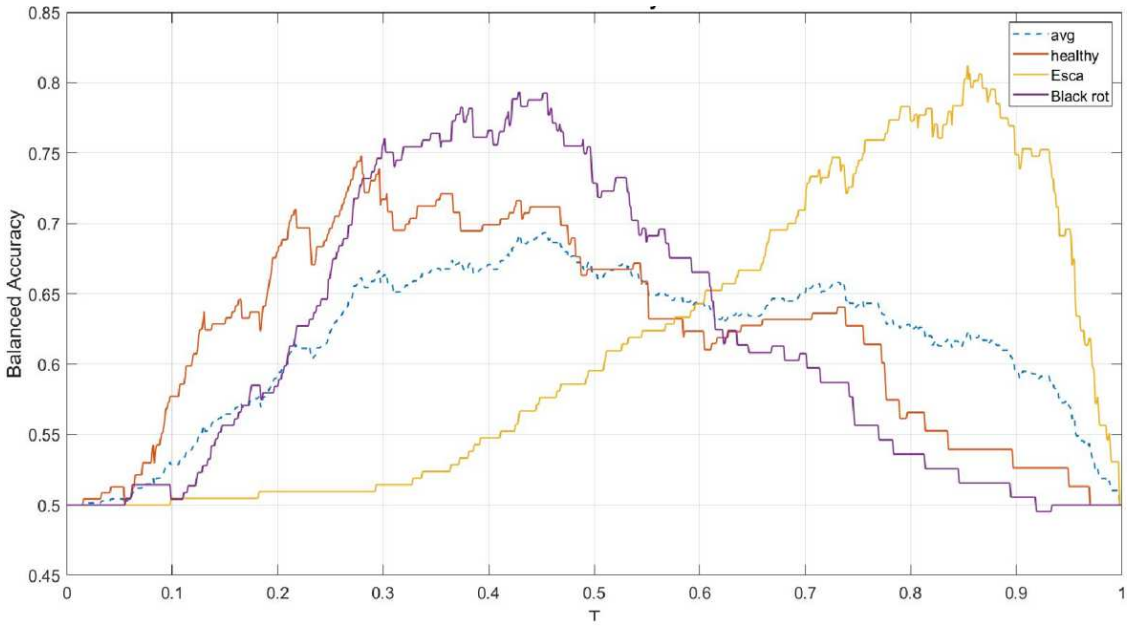


Figure 41 MobileNet2 Dataset1 with object detection balanced accuracy vs Threshold

Supposing the choice falls on a classifier where all the classes reach high balanced accuracies, for example MobileNet2 with object detection in Dataset1 (figure 41). If the aim is the maximum robustness in recognizing “Esca” by other generic things, the threshold must be set to about 86% where the balanced accuracy reaches 82% for “Esca”, but for “Black Rot” and “Healthy” at the same threshold the balanced accuracy is about 52%, making their classifier substantially a useless

toss of coins if used as binary classifiers. Vice versa exploiting the maximum robustness for "Black Rot" and "Healthy" means moving the threshold around the 30-40% where "Esca" balanced accuracy drastically drop. This asymmetric behavior between "Esca" and other classes is constant among all the classifier studied and it is simply due to the dataset-specific behavior putting "Esca" images points far by the hyper plane. So, if the aim is to give the maximum priority to highlight specifically one class versus all the other the right choice is to move the threshold toward its maximum balanced accuracy and classify eventual non-Positive as "confused". However, if one wants to fully exploit the classifier, assigning a specific name to possibly all the images in input, must leaves some specific class accuracy to reduce the variance among all class accuracies using a Multinomial thresholding obtained by other reasonings. In each classifier the maximum theoretical multilevel accuracy is substantially the maximum averaged balanced accuracy, as discussed in 1.4.5.8. With no prior knowledge about the dataset the prediction threshold must be tuned necessarily with trial-and-error in order to ensure recall/specificity wanted. If, as in the second validation example, a sample of the grapes, on which the classifier will work, is present then some adjustments can be done. For example, again in the MobileNet2 Dataset1 with object detection, can be noticed that "Black Rot" and "Esca" images have larger *signed* distances (higher posteriors) from all the hyperplanes than "Healthy" images, although the separation quality is similar. This fact well explains the different peaks of balanced accuracy for the classes (peaks positions are strictly dependent on the dataset chosen). In the algorithm proposed for prediction this behavior will result often in high false negatives for "Healthy" class, since often "Healthy" points are more distant from "Black Rot" and "Esca" planes than their own plane. A solution is to move "Healthy" and "Esca" planes toward their cluster's centers, increasing the relative biases. A fast way to do so is simply normalizing all the posteriors computed for each image to the highest medium posterior. In this case, for example, after computing 3 posteriors for all the images are obtained 3 posterior vectors $p_{black\ rot}$, p_{Esca} and p_{Helthy} of 147 elements. Computing the averages for each posterior vector 3 values are obtained:

$$\frac{1}{147}\sum_{i=1}^{147} sigm\left(b_{Black\ rot} + \bar{w}_{Black\ rot}^{T}\bar{x}^{i}\right) = \widehat{p_{black\ rot}} = 0{,}53 \qquad (2.42)$$

$$\frac{1}{147}\sum_{1}^{147} sigm\left(b_{Esca} + \bar{w}_{Esca}^{T}\bar{x}^{i}\right) = \widehat{p_{Esca}} = 0{,}35 \qquad (2.43)$$

$$\frac{1}{147}\sum_{1}^{147} sigm\left(b_{Healthy} + \bar{w}_{Healthy}^{T}\bar{x}^{i}\right) = \widehat{p_{Healthy}} = 0{,}1 \qquad (2.44)$$

Is not surprising that "Esca" medium posterior, computed in this way, is not greater than "Black Rot" medium posterior as expected by the balanced accuracies curves. The magnitudes order of max balanced accuracies thresholds indices not the medium threshold (and so distance) of *all* the images from the class hyperplane, as in the previous calculations, but represent the medium threshold of the only real "Esca" images of the dataset, approaching to which the FPR decreases and overcoming which the FNR increases. In this reinforcing learning, instead, is important to include all the images distances/posteriors in order to have an idea of how much images of other classes are distant from a hyperplane they do not belong to, in order to compensate the effect over all classes. Normalizing for $\widehat{p_{black\ rot}}$ the other two posteriors are obtained 2 multiplication factors. Then multiplying them with all "Esca" and "Healthy" image posteriors in the classification process the Multilevel accuracy is increased of about 15 %, moving toward the maximum theoretical of about 69%.

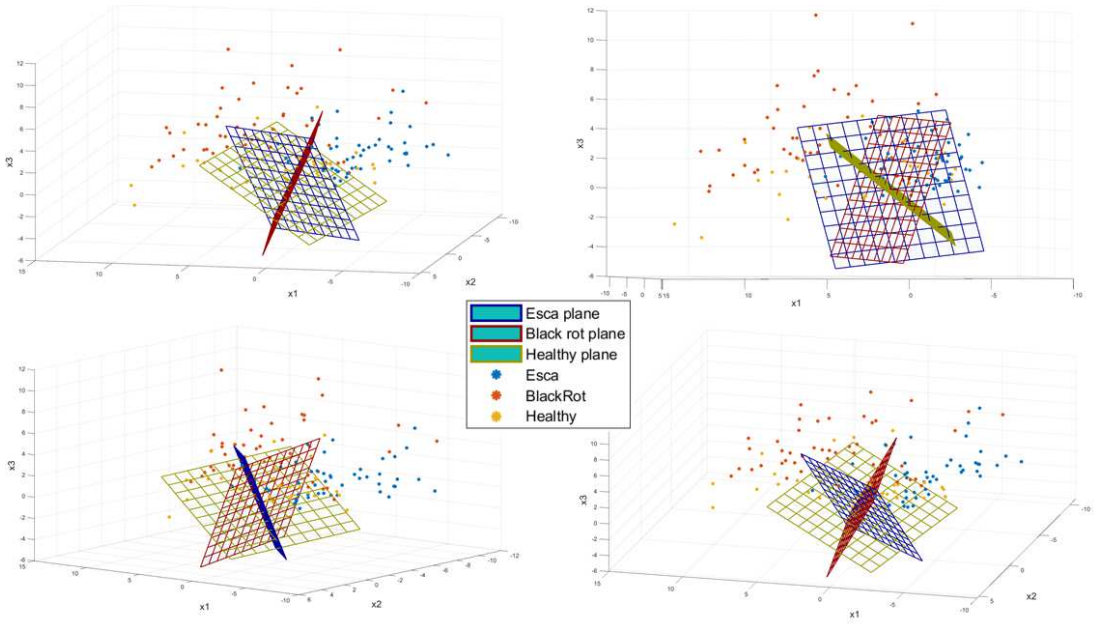


Figure 42 MobileNet2 classifier Hyperplanes in firsts 3 Principal components; Dataset2

$$\widehat{p_{black\ rot}} / \widehat{p_{Esca}} = 1{,}514 \qquad (2.45)$$

$$\widehat{p_{black\ rot}} / \widehat{p_{Healthy}} = 5{,}3 \qquad (2.46)$$

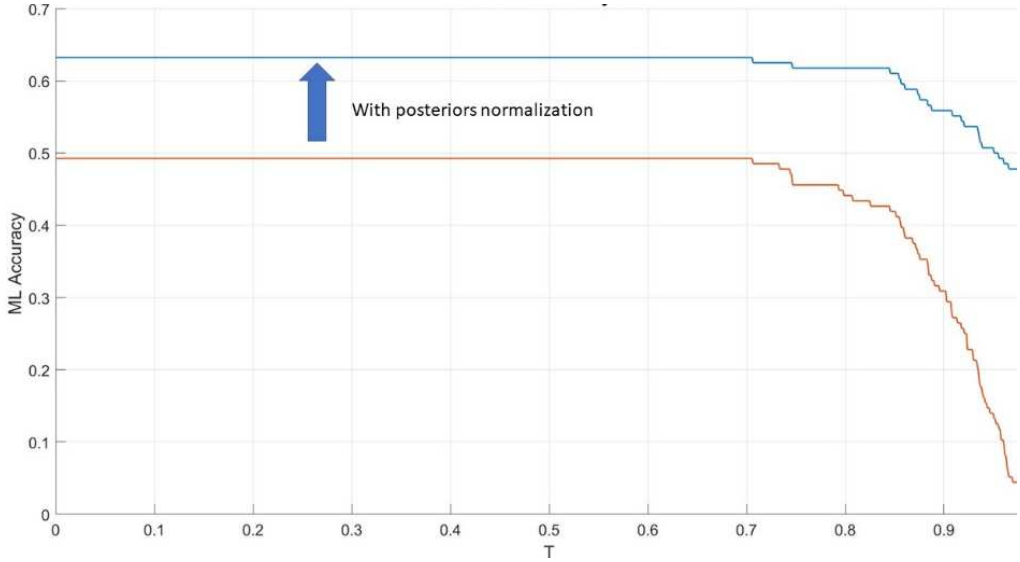


*Figure 43 MobileNet2 in Dataset1 with object detection Multilevel accuracy versus multilevel threshold **

* although "Healthy" and "Esca" planes have been moved forward the critical threshold always remains the same at 0.705, this is because probably the point, correctly classified, closest to a hyperplane at the beginning was a black rot point, and continues to be so

This type of *assisted* learning can be applied only if a reasonable sized sample of the grapevines on which the classifier will work permanently is available. The main assumption making consistent these tricks is the strictly dependance of the medium distances from hyperplanes on the site's grapevine whom, in general, is different by the medium distance trained in the training dataset. Many other ways can be followed to power the algorithm. Some of them insist to do reinforcing learning, pushing the algorithm toward its maximum theoretical possibilities. For example, fast color and/or shape filters can be used downwards the object recognition phase to priorly reject all the images, or sub-images, with characteristics strongly different by any known

leaf. Additionally, each posterior can be weighted with the base-risk related to the registered occurrence of the linked disease. Other, stronger, reinforcements unavoidably pass by the acquisition of a more suitable dataset overcomes difficulties and limits encountered here. All classes should contain an equal number of real grapevines captured in same conditions of lightening, angular position and distances, possibly by the vinegar specie, in the same period of the year. Furthermore, for an excellent tuning of the thresholds a consistent dataset of out-of-class leaves must be captured, including insect diseases, hydric stresses and other confounding symptoms. In general, a cross-linking of this study with specific agronomical studies is necessary.

3 UAV dynamics and control system

The second stage of the mission consist in the autonomous spray of the crops by means of a 25 Kg quadrotor, with 10 Kg of PPP payload. In the first part of this chapter the quadrotor and actuators structures, kinematics and dynamic models will be discussed. At the resulting dynamics a set of control strategies in continuous and discrete time will be applied evaluating the performances of the whole control loop supposing known and perfectly measurable states. Then will be discussed the lack of fully measurable states and presented the problem of observability of the nonlinear UAV system and the Data Fusion approach through the Extended Kalman Filter. An experimental test will play the role of validator of the Simulink sensors and EKF model providing a locus of control parameters that makes the model coherent with reality.

3.1 Dynamic Model

3.1.1 Kinematics

Was considered a right-handed inertial frame attached to the earth surface and a right-handed body frame fixed in the quadrotor CoG with y axis passing from the rotor CoG and x axis passing from rotor 4 CoG. Using the *z-y'-x''* Euler angle triad was then extracted the relative rotation matrices between the two reference frames and between angle rates and twist. Rotation matrix from body frame to Inertial frame following the euler triad *z-y'-x''* (from inertial to body frame):

$$R_{Ib} = [Rx(\phi)Ry(\gamma)Rz(\varphi)]^T = R_{bI}^T \quad (3.001)$$

$$= \begin{matrix} c(\varphi)*c(\gamma) & s(\dot{\phi})*s(\gamma)*c(\varphi) - c(\dot{\phi})*s(\varphi) & c(\dot{\phi})*s(\gamma)*c(\varphi) + s(\dot{\phi})*s(\varphi) \\ c(\gamma)*s(\varphi) & s(\dot{\phi})*s(\gamma)*s(\varphi) + c(\dot{\phi})*c(\varphi) & c(\dot{\phi})*s(\gamma)*s(\varphi) - s(\dot{\phi})*c(\varphi) \\ -s(\gamma) & s(\dot{\phi})*c(\gamma) & c(\dot{\phi})*c(\gamma) \end{matrix}$$

Transfotmation matrix from analitical angular rates to phisycal angular velocities in *z-y'-x''* triad:

$$T_{YRP} = \begin{matrix} 1 & 0 & -s(\gamma) \\ 0 & c(\dot{\phi}) & s(\dot{\phi})*c(\gamma) \\ 0 & -s(\dot{\phi}) & c(\dot{\phi})*c(\gamma) \end{matrix} \quad (3.002)$$

(Singular for γ=pi/2)

Transformation matrices from analytical angular accelerations to physical angular accelerations in *z-y'-x''* triad:

$$\bar{\alpha} = [\dot{\phi}, \gamma, \varphi]$$

$$\bar{\omega} = T_{YRP}\,\dot{\bar{\alpha}} \quad (3.003)$$

$$\dot{\bar{\omega}} = T_{YRP}\,\ddot{\bar{\alpha}} + \dot{T}_{YRP}\,\dot{\bar{\alpha}} \quad (3.004)$$

$$\dot{T}_{YRP} = \begin{matrix} 0 & 0 & -\dot{\gamma}c(\gamma) \\ 0 & -\dot{\phi}s(\dot{\phi}) & -\dot{\gamma}s\left(\dot{\phi}\right) * s(\gamma) + \dot{\phi}c\left(\dot{\phi}\right) * c(\gamma) \\ 0 & -\dot{\phi}c(\dot{\phi}) & -\dot{\gamma}c\left(\dot{\phi}\right) * s(\gamma) + \dot{\phi}s\left(\dot{\phi}\right) * c(\gamma) \end{matrix} \quad (3.005)$$

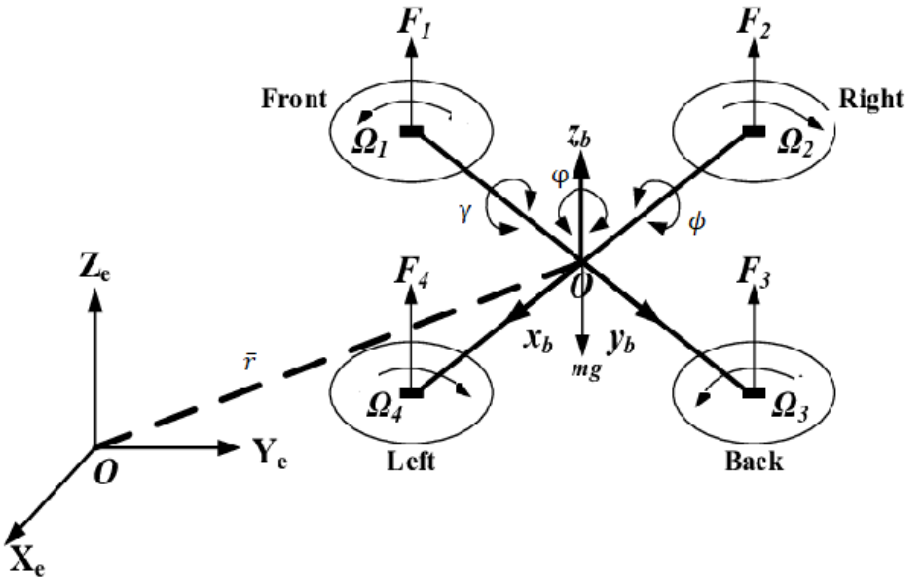


Figure 44 Quadrotor free-body diagram

3.1.2 UAV dynamics equations

1° Newton Law for translational and angular degrees of freedom (d.o.f.s.) in Inertial frame:

$$\begin{cases} f_{In} = \frac{\partial}{\partial t}\overline{P} = m\,[\ddot{x}, \ddot{y}, \ddot{z}] \\ M_B = \frac{\partial \bar{L}}{\partial t} = I\left(\dot{\bar{\omega}}\right) + \bar{\omega} \times (I\bar{\omega}) \end{cases} \quad (3.006)$$

The inertial frame, attached to the ground, was considered ideally inertial with negligible contribution of centrifugal force due to the earth rotation and negligible Coriolis effect since for the application the latitude variation of the drone will be negligible. Inertia matrix *I* is considered to be diagonal since the body reference frame is oriented along the 3 principal axes of inertia. Substituting angular velocities with YPR angular rates one obtains:

$$\begin{cases} f_B = m\,[\ddot{x}, \ddot{y}, \ddot{z}] \\ M_B = I\,\dot{T}_{YRP}\ddot{\bar{\alpha}} + I\,\dot{T}_{YRP}\dot{\bar{\alpha}} + T_{YRP}\,\dot{\bar{\alpha}} \times \left(I\,T_{YRP}\,\dot{\bar{\alpha}}\right) \end{cases} \quad (3.007)$$

To allow a manageable non-linear state space representation the gyroscopic term can be rewritten as:

$$T_{YRP}\,\dot{\bar{\alpha}} \times \left(I\,T_{YRP}\,\dot{\bar{\alpha}}\right) = \boldsymbol{SGyro_D}\left(\bar{\boldsymbol{\alpha}}, \dot{\bar{\boldsymbol{\alpha}}}\right)\dot{\bar{\alpha}} \quad (3.008)$$

With $\boldsymbol{SGyro_D}\left(\bar{\boldsymbol{\alpha}}, \dot{\bar{\boldsymbol{\alpha}}}\right)$ equal to:

$$\begin{bmatrix} (Iy-Iz)\left(\frac{\dot{\gamma}\cos(\varphi)\sin(\gamma)}{2}-\frac{\dot{\varphi}\cos(\gamma)\sin(\varphi)}{2}+\dot{\phi}\cos(\gamma)\sin(\gamma)\sin(\varphi)\right) & -(Iy-Iz)\left(\frac{\dot{\varphi}\cos(\varphi)}{2}-\frac{\dot{\phi}\cos(\varphi)\sin(\gamma)}{2}\right) & -(Iy-Iz)\left(\frac{\dot{\gamma}\cos(\varphi)}{2}+\frac{\dot{\phi}\cos(\gamma)\sin(\varphi)}{2}\right) \\ (Ix-Iz)\left(\frac{\dot{\varphi}\cos(\gamma)\cos(\varphi)}{2}+\frac{\dot{\gamma}\sin(\gamma)\sin(\varphi)}{2}-\dot{\phi}\cos(\gamma)\cos(\varphi)\sin(\gamma)\right) & -(Ix-Iz)\left(\frac{\dot{\varphi}\sin(\varphi)}{2}-\frac{\dot{\phi}\sin(\gamma)\sin(\varphi)}{2}\right) & -(Ix-Iz)\left(\frac{\dot{\gamma}\sin(\varphi)}{2}-\frac{\dot{\phi}\cos(\gamma)\cos(\varphi)}{2}\right) \\ \left(\frac{\dot{\gamma}\cos(\gamma)(\cos(\varphi))^2-\sin(\varphi)^2)}{2}-\dot{\phi}\cos(\gamma)^2\cos(\varphi)\sin(\varphi)\right)(Ix-Iy) & (Ix-Iy)\left(\dot{\gamma}\cos(\varphi)\sin(\varphi)+\frac{\dot{\phi}\cos(\gamma)(\cos(\varphi))^2-\sin(\varphi)^2)}{2}\right) & 0 \end{bmatrix}$$

And the model becomes

$$\begin{cases} f_B = m\,[\ddot{x},\ddot{y},\ddot{z}] \\ (I\,T_{YRP})^{-1}M_B - (I\,T_{YRP})^{-1}(SGyro_D(\bar{\alpha},\dot{\bar{\alpha}}) + I\,\dot{T}_{YRP})\dot{\bar{\alpha}} = \ddot{\bar{\alpha}} \end{cases} \quad (3.009)$$

Forces applied to the system are in general the sum of the controlled forces, disturbances, and viscous air-body friction.

$$\overline{f_{In}} = \overline{f_{control\_In}} + \overline{f_{Dist\_In}} + \overline{f_{d\_in}} \quad (3.010)$$

where the control force is the thrust vector, always aligned with the z axis of body frame rotated in the Inertial frame. Here, for each rotor a static correlation *KF* is considered between the thrust force exerted and the rotors' squared speeds.

$$\overline{f_{control\_In}} = R_{Ib}\,\overline{f_{control\_b}} = R_{Ib}\,[0\ 0\ thrust]^T \quad (3.011)$$

$$thrust = KF\,(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \quad (3.012)$$

The disturbances are in general the sum of the deterministic gravity force and other aleatory disturbances (e.g wind) in Inertial coordinates. The last term expresses the eventual viscous air friction with a diagonal uncoupled matrix.

$$\overline{f_{Dist\_In}} = mg[0\ 0\ 1]^T + \overline{dist_{In}} \qquad \overline{f_{d\_in}} = \begin{bmatrix} Kdx & 0 & 0 \\ 0 & Kdy & 0 \\ 0 & 0 & Kdz \end{bmatrix}\dot{\bar{x}} = Kd\,\dot{\bar{x}} \quad (3.013)$$

The torque can be expressed as the sum of the control torques, eventual disturbance torques and the gyroscopic torque in *x_body* and *y_body* frame due to the rotors and propellers, all expressed in the body frame:

$$\overline{M_b} = \overline{M_{control\_b}} + \overline{M_{Dist\_b}} + \overline{M_{gyroR\_b}} \quad (3.014)$$

$$\overline{M_{control\_b}} = \begin{matrix} KF\,(\Omega_3^2 - \Omega_1^2)l \\ KF\,(\Omega_2^2 - \Omega_4^2)l \\ KQ\,(\Omega_1^2 - \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{matrix} \quad (3.015)$$

where the static correlation between propeller torques and propelled squared speeds is expressed by *KQ* and l is the distance between the center of gravity of the rotors and the UAV center of gravity.

$$\overline{M_{gyroR\_b}} = \begin{matrix} (J_{rz} - J_{ry})\,\omega_y(\Omega_1 - \Omega_2 + \Omega_3 - \Omega_4) \\ (J_{rx} - J_{rz})\,\omega_x(\Omega_1 - \Omega_2 + \Omega_3 - \Omega_4) \\ 0 \end{matrix} \quad (3.016)$$

Jrx, Jry and Jrz are the time-averaged inertias of the whole rotor/propeller block seen by the axis x,y,z in the body frame. Since their difference is supposed to be negligible this torque component is considered as a simple disturbance.

3.1.3 Actuator's dynamics

Excluding the motor/propeller dynamics (supposed to have a characteristic frequency larger than those of the mechanical plant) remains to be defined the transformation between $\overline{u_{ctrl}}$ and the four propeller velocities. Rewriting the equations of control input and torques in matrix form one obtains:

$$\boldsymbol{QQ} = \begin{bmatrix} KF & KF & KF & KF \\ -KF\,l & 0 & KF\,l & 0 \\ 0 & KF\,l & 0 & -KF\,l \\ KQ & -KQ & KQ & -KQ \end{bmatrix} \tag{3.017}$$

$$\boldsymbol{QQ}\begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} = \begin{bmatrix} thrust \\ Cx \\ Cy \\ Cz \end{bmatrix} = \overline{u_{ctrl}} \qquad \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix} = \boldsymbol{QQ^{-1}}\,\overline{u_{ctrl}} \tag{3.018}$$

$With\ \det(\boldsymbol{QQ}) = 8\,KQ\,KF^3 l^2 \neq 0$

So, the matrix is always invertible, but a low/up saturation is inserted to take into account the nonlinear behavior of the actuators.

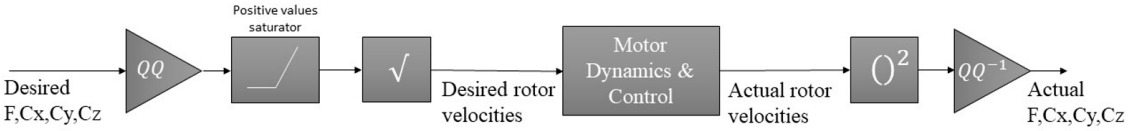


Figure 45 Rotor velocities command generation block scheme

In between command input desired rotors speed and actual rotor speed a simplified model of the motor is added. The motor chosen is a brushless P80 T-Motor with characteristics reported below.

Test Report			
Test Item	P80 KV100	Report NO.	P.00003
Specifications			
Internal Resistance	45-51mΩ	Configuration	36N42P
Shaft Diameter	15mm	Motor Dimensions	Φ91.6×43mm
AWG	14#	Cable Length	90mm
Weight Including Cables	650g	Weight Excluding Cables	/
No.of Cells(Lipo)	6-12S	Idle Current@10v	1.3-1.6A
Max Continuous Power 180S	2800W	Max Continuous current 180S	60A

Figure 46 T-motor technical datasheet P80

The manufacturer specified a maximum thrust of 17 kg and considering 4 rotors the maximum thrust is 68 kg, enough higher than the maximum takeoff weight to be ensured of 25 kg (15 kg structure and a maximum of 10 kg payload PPP). Generally, an electronic controlled brushless motor consists of an applied single phase DC voltage, a DC/AC inverter that provides 120° delayed three phase square wave voltages to the windings of the stator and a permanent magnet rotor. Three phase switching ensures orthogonality between the rotor magnetic field and the stator magnetic field. Finally, an Electronic Speed Controller (ESC) acts as speed/torque controller variating the average voltage applied to the windings though PWM signal, increasing the power

when a load is applied. In the brushless only two phases have simultaneously non-zero voltage applied, and the third is often used to sense the back electromotive voltage to derive the rotor velocity. However, for modeling purposes, a single phase only and the Pulse width modulation of the applied voltage could be considered, remember to scale the applied voltage by $\frac{\sqrt{3}}{3}$ to extract the specific phase voltage. Equations governing the simplified brushless motor in their general form includes the internal friction between rotor and stator, rotor inertia, generated torque and load torque. For the electrical domain a Kirchhoff equilibrium equation is performed.

$$\begin{cases} I\dot{\omega} + \nu\omega = C_m - C_{load} \\ V_a = L\dfrac{di}{dt} + Ri + K_v\omega \end{cases} \quad (3.019)$$

where ν is the viscous friction coefficient in $N \cdot m \cdot rad^{-1}s$, I is the rotor inertia moment in $Kg \cdot m^2$, L the inductance of the windings in H, R the resistance in Ω and K_v the time averaged electromotive force constant in $V \cdot rad^{-1}s$. *T*. The traction torque $C_m = K_t\, i$, where K_t is expressed in $N \cdot m \cdot A^{-1}$ and can be posed numerically equal to K_v for an ideal motor. The datasheet reports the idle current when is applied an equivalent V_a=10 V in steady state conditions with no load. Furthermore, the manufacturer provides the "KV" constant equal to 100 rpm/V that relates the rotor velocity and the applied voltage in steady state condition with no load. Therefore, supposing the parameters independent by motor state and temperature and neglecting the transient, results:

$$\begin{cases} \nu\omega = K_t\, i_0 \\ V_a = Ri_0 + K_v\omega \\ \dfrac{\omega}{V_a} = KV \cdot {}^{2\pi}/_{60} \;\; rad\, s^{-1}\, V^{-1} \end{cases} \quad (3.020)$$

So that

$$\begin{cases} \nu = \dfrac{K_t\, i_0}{V_a \cdot KV \cdot {}^{2\pi}/_{60}} \\ K_v = \dfrac{V_a - Ri_0}{V_a \cdot KV \cdot {}^{2\pi}/_{60}} \end{cases} \quad (3.021)$$

Substituting all the values an estimate of ν, K_v and K_t is obtained:

$$\begin{cases} \nu = 0{,}0013 & N \cdot m \cdot rad^{-1}s \\ K_v = 0{,}0948 & V \cdot rad^{-1}s \\ K_t = 0{,}0948 & N \cdot m \cdot A^{-1} \end{cases} \quad (3.022)$$

These values will be considered constant for simplicity. The inductance value is not reported by the manufacturer, so for simplicity, the input power will be considered coincident with the active input power since for a detailed characterization of the phase factor an experimentation with a power analyzer is required [22]. Furthermore, the electric time constant is often several order of magnitude smaller than the mechanical time constant of the rotor, so also in the dynamics the inductive effect can be neglect. Are also neglected the power leakages due to the MOSFETs of the ESC, by the way their efficiency is often higher than 90% [23].

$$\begin{cases} P_{in} = V_a \cdot i \\ P_{out} = C_m \cdot \omega \end{cases} \quad (3.023)$$

Starting from these definitions a model of the motor efficiency in time domain is obtained. It will be important to justify some control strategies with respect to other, considering that the motor is

prone to have lower efficiency in transient phases. In condition of no load, and neglecting the inductance, the development the above system of equations in Laplace domain is:

$$\begin{cases} sI\omega + \nu\omega = K_t\, i \\ V_a = Ri + K_v\omega \end{cases} \tag{3.024}$$

where V_a is the medium PWM voltage imposed and can be seen as a step function $V_a = V_{a,0}/s$. Follows that:

$$\begin{cases} i(s) = \dfrac{V_{a,0}}{R}\dfrac{s + {\nu}/{I}}{s\left(s + \dfrac{\nu + {K_v}^2/R}{I}\right)} \\ \omega(s) = \dfrac{i(s)}{sI + \nu} \end{cases} \tag{3.025}$$

And in time domain results:

$$\begin{cases} i(t) = \dfrac{V_{a,0}}{R}e^{-t\left(\frac{\nu + {K_v}^2/R}{I}\right)} + \dfrac{\nu\, V_{a,0}}{\nu R + {K_v}^2}\left(1 - e^{-t\left(\frac{\nu + {K_v}^2/R}{I}\right)}\right) \\ \omega(t) = \dfrac{K_v\, V_{a,0}}{\nu R + {K_v}^2}\left(1 - e^{-t\left(\frac{\nu + {K_v}^2/R}{I}\right)}\right) \end{cases} \tag{3.026}$$

Considering that $C_m = K_t\, i = K_v\, i$ the time variant efficiency results:

$$eff(t) = \frac{P_{out}}{P_{in}} = \frac{C_m \cdot \omega}{V_{a,0} \cdot i} = \frac{K_v \cdot i \cdot \omega}{V_{a,0} \cdot i} = \frac{{K_v}^2}{\nu R + {K_v}^2}\left(1 - e^{-t\left(\frac{\nu + {K_v}^2/R}{I}\right)}\right) \tag{3.027}$$

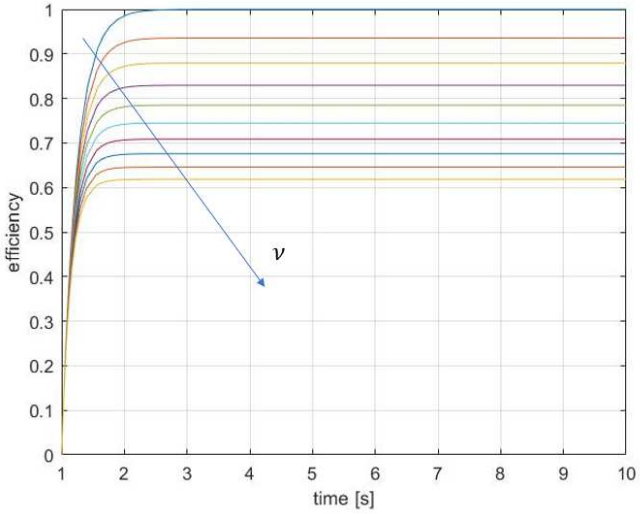


Figure 47 motor efficiency for different viscous coefficient, R=50 mOhms

The time constant of the motor is about 260 ms, but with ESC controller inserted in the dynamics it is further reduced at the order of milliseconds. Was chosen G28"x9,2" propellers produced by

T-Motor with a reference thrust level of about 5 Kg (with 50-60 % of throttle and 12S voltage) for which was available a table containing the current, rpm and thrust obtained in function of the P80 motor throttle that spaces between 50 and 100%. Considering a quadratic relation between the propeller velocity and thrust produced and considering this relation independent from the temperature and pressure condition, an estimate of this coefficient is extracted by experimental data:

$$C_m = KQ \cdot \omega^2 \tag{3.028}$$

$$Thrust = KF \cdot \omega^2 \tag{3.029}$$

Data was collected with 17 °C ambient temperature, air density $\rho = 1{,}225\ \frac{kg}{m^3}$, and 52 °C on recorded on the external motor surface:

$$\begin{cases} i = (6 \quad 7.9 \quad 9.8 \quad 12 \quad 17.2 \quad 23.3 \quad 35)\ A \\ s.t.\ C_m = K_t\, i\ \ then \\ C_m = (0.56 \quad 0.75 \quad 0.93 \quad 1.15 \quad 1.63 \quad 2.21 \quad 3.29)\ Nm \\ Thrust = (31 \quad 39 \quad 45 \quad 51 \quad 66 \quad 82 \quad 106)\ N \\ \omega = (138 \ \ 153 \ \ 165 \ \ 188 \ \ 199 \ \ 221 \ \ 253)\ rad/s \end{cases} \tag{3.030}$$

Results a KQ/KF vector for the seven data point:

$$\text{KQ} = (0.2980 \quad 0.3189 \quad 0.3394 \quad 0.3235 \quad 0.4097 \quad 0.4505 \quad 0.5128) \cdot 10^{-4} = c \tag{3.031}$$

$$KF = (0.0017 \quad 0.0017 \quad 0.0017 \quad 0.0014 \quad 0.0017 \quad 0.0017 \quad 0.0017) = d \tag{3.032}$$

For the motor models the fitting curves were used to recreate closely realistic conditions, so that

$$C_m = KQ(\omega) \cdot \omega^2 \tag{3.033}$$

$$Thrust = KF(\omega) \cdot \omega^2 \tag{3.034}$$

While, in the actuation matrix $\boldsymbol{QQ}$ their mean value is inserted:

$$\text{KQ} = 3.7898 \cdot 10^{-5} \quad KF = 0{,}0016 \tag{3.035}$$

Is worth noticed that the rotors velocities under propulsion load condition are always lower than those one obtainable in steady state condition and related to the voltage by the KV constant:

$$\frac{\omega}{V_{a,0}} < KV \cdot {}^{2\pi}/_{60} = \frac{K_v}{\nu R + {K_v}^2} \tag{3.036}$$

For this reason, the ESC act as a speed controller reading the rotor velocity through back-EMF and increases the applied voltage accordingly. In the model in the speed feedback control loop was inserted a transfer function that ensures a rising time of few milliseconds.

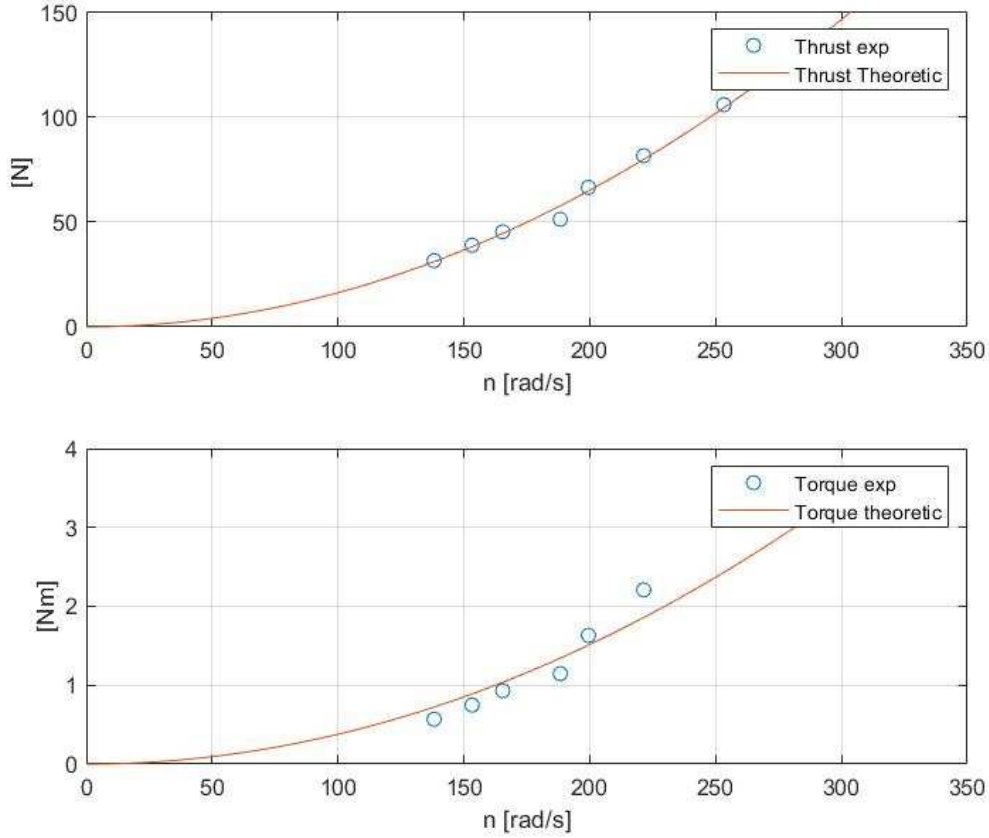


Figure 48 Curve fitting of the 7 datasheet points

Rotor and propeller inertia was modelled as a combination of a cylinder with height h_r and radius r_r and a slab of length l and width w (the propeller), considering the measures reported by the manufacturer drawings.

$$I_{rotor,z} = 0{,}5 \cdot m_r \cdot {r_r}^2 + \frac{1}{12} \cdot m_{propeller} \cdot \left(l^2 + w^2\right) = 0{,}0472\ kgm^2 \qquad (3.037)$$

With

$$\begin{cases} m_r = 0{,}6\ kg & motor\ mass, excluding\ cables \\ m_{propeller} = 0{,}082\ kg & propeller\ couple\ mass \\ r_r = 0{,}045\ m & \\ l = 0{,}48\ m & \\ w = 0{,}14\ m & \end{cases}$$

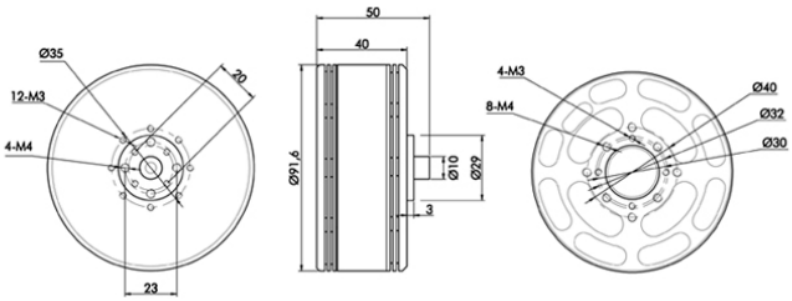


Figure 49 P80 motor

Finally, the battery used was the same batteries used in the DJI matrix 600 [24], namely TB47S battery with 22,2 V output, a capacity of 7500 mAh, a weight of 595 g. Are used in a configuration of three parallel of 2 batteries in series, i.e., with a 44,4 V output voltage and a total capacity of 22500 mAh. The resultant total weight is of 3,57 kg, that is the 23,8 % of the structural weight of the drone. The ESC controller a reference velocity input from the MCU controller, and after having computed the error with the actual motor velocity produces a control voltage PWM with 0-5V scale that is amplified by the internal MOSFETs to the battery output voltage of 44,4 V. The final Simulink model was the following:

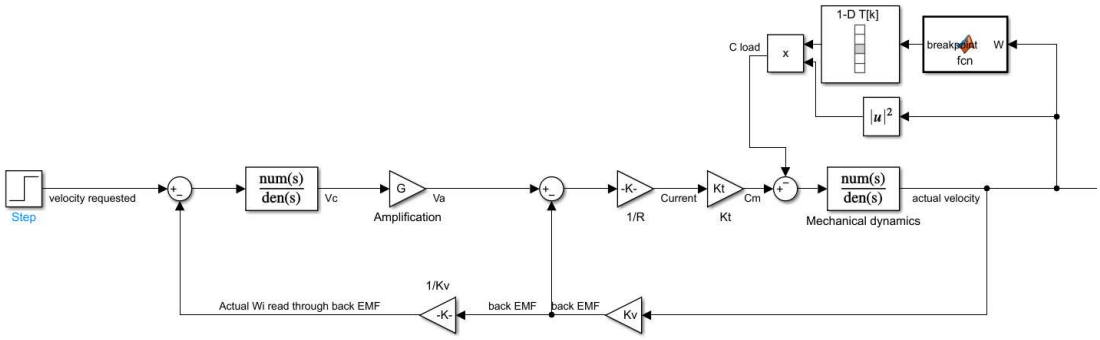


Figure 50 Single motor dynamic model and control, with propeller torque applied, SIMULINK model

At the above scheme (figure 50) two saturation blocks are applied. The first one limits the maximum $V_{a,0}$ to +/- 44,4 V (assuming ideally that the battery voltage remains constant with charge consumption) and the maximum current peaks to 3 times the maximum nominal continuous current of the ESC-motor of 60 A, so 180 Amperes. 180 Amperes can be maintained for few seconds, then the Esc or the motor could be damaged. Using the maximum continuous current can be also imposed the maximum roll and pitch angle over which the thrust vertical component of the propulsion system cannot compensate the gravity force. To the maximum current corresponds a maximum torque and a maximum rotor speed:

$$\boldsymbol{C_{m,max}} = K_t \cdot i_{max} = \boldsymbol{5,688\ Nm} \tag{3.038}$$

$$\boldsymbol{\omega_{max}} = \sqrt{C_{m,max}/KQ} = \boldsymbol{387\ rad/s} \tag{3.039}$$

$$\boldsymbol{Thrust_{max,total}} = 4 \cdot KF \cdot {\omega_{max}}^2 = \boldsymbol{976\ N} \tag{3.040}$$

Considering the rotation matrix between body frame and inertial body frame the vertical component of the maximum thrust in the inertial z axis is:

$$Thrust_{max} \cdot c(\phi) \cdot c(\gamma) \tag{3.041}$$

And in the most conservative situation, when the payload is maximum, it must satisfy:

$$Thrust_{max} \cdot c(\phi) \cdot c(\gamma) \geq 15 \cdot 9{,}81\ N = 245{,}25\ N = MW \tag{3.042}$$

where MW is the maximum weight. Results:

$$\begin{cases} \phi < \pm \operatorname{acos} \dfrac{\mathrm{MW}}{\mathrm{Thrustmax} \cdot\ c(\gamma)} \\ \gamma < \pm \operatorname{acos} \dfrac{\mathrm{MW}}{\mathrm{Thrustmax} \cdot\ c(\phi)} \end{cases} \tag{3.043}$$

If one of the two angle is zero, then the condition on the maximum tilt angle is:

$$(\boldsymbol{\gamma}\ \mathbf{or}\ \boldsymbol{\phi})_{\boldsymbol{max}} = \mathbf{76}\,^{\circ} \tag{3.044}$$

Well distant from the linear control variables contour. However, this maximum angle is further decreased if are considered also the maximum thrust that the propellers can exert dealing with deeper fluid dynamic analysis.

3.1.4 State space model

The state chosen is the pose and its derivative, so of dimension 12

$$X = \begin{bmatrix} x & y & z & \dot{\phi} & \gamma & \varphi & \dot{x} & \dot{y} & \dot{z} & \dot{\phi} & \dot{\gamma} & \dot{\varphi} \end{bmatrix} \tag{3.045}$$

$$A = \begin{bmatrix} \mathbf{0_{6X6}} & \boldsymbol{eye_{6X6}} \\ \mathbf{0_{3X6}} & \boldsymbol{Kd}\,;\mathbf{0_{3X3}} \\ \mathbf{0_{3X9}} & (\boldsymbol{I\,T_{YRP}})^{-1}(\boldsymbol{SGyro_D}(\bar{\alpha},\dot{\bar{\alpha}}) + \boldsymbol{I\,\dot{T}_{YRP}}) \end{bmatrix} \tag{3.046}$$

$$B_{act} = \begin{bmatrix} \mathbf{0_{6X4}} \\ \dfrac{\boldsymbol{R_{Ib}}(:,\mathbf{3})}{m}\ ; \quad \mathbf{0_{3X4}} \\ \mathbf{0_{3X1}} \quad ; \quad (\boldsymbol{I\,T_{YRP}})^{-1} \end{bmatrix} \tag{3.047}$$

$$B_{dist} = \begin{bmatrix} \mathbf{0_{6X6}} \\ \dfrac{\boldsymbol{eye_{3X3}}}{m}\ ; \quad \mathbf{0_{3X3}} \\ \mathbf{0_{3X3}} \quad ; \quad (\boldsymbol{I\,T_{YRP}})^{-1} \end{bmatrix} \tag{3.048}$$

The state space representation is:

$$\dot{X} = \boldsymbol{A}X + \boldsymbol{B_{act}}\,\overline{u_{\boldsymbol{ctrl}}} + \boldsymbol{B_{dist}}\,\overline{u_{\boldsymbol{dist}}} = f(X,\ \overline{u_{\boldsymbol{ctrl}}}) \tag{3.049}$$

With:

$$\overline{u_{\boldsymbol{ctrl}}} = [\left\|\overline{f_{control_b}}\right\|\ ,\ \overline{M_{control_b}}]^T \tag{3.050}$$

$$\overline{u_{\boldsymbol{dist}}} = [\overline{f_{Dist\_In}}\ ,\ \overline{M_{Dist\_b}} + \overline{M_{gyroR\_b}}]^T \tag{3.051}$$

3.2 Control architecture

From now the UAV will be considered inspired to the Matrix 600 DJI used in experimental phases by our research group to make spray analysis in a vineyard situated at the Agronomical center of the Turin University. Starting from the manufacturer datasheet several component choices will be inspired by it in the next sessions. Concerning about the mechanical quantities and dimensions the following one will be used:

$$\begin{cases} m = 25\ Kg\ \ (total\ mass) \\ m_{no\ payload} = 15\ Kg = m_{central\ body} + m_{spray\ system} + 4 \cdot m_{single\ arm} + m_{rotors} \\ m_{payload} = 10\ Kg\ (10\ L\ of\ PPP) \\ m_{spray\ system} = 3\ kg \\ m_{central\ body} = 6\ Kg\ \ (including\ battery\ pack) \\ m_{single\ arm} = 1{,}5\ Kg \\ m_{rotors} = 4 * 0{,}6\ Kg = 2{,}4\ Kg \\ m_{spray\ system} = 0{,}6\ Kg \\ I_{rotor,z} = 0{,}0472\ Kgm^2 \\ l = 0{,}65\ m \\ Maximum\ Flow\ rate = 2{,}4\ L/min \\ h = 0{,}3\ \ (body\ heigth) \end{cases} \tag{3.052}$$

Motors and propellers masses and dimension derive from the actuator dynamics choices in the

next session. Starting from these masses and considering the simplified geometry of the quadrotors with 90° angular distance between each arm, the other useful quantities are computed. Approximating the UAV shape as 4 cylindric arms connected with the cylindric rotor at one side and with the cylindric central body at the other side, the first approximation of the 3 components of the diagonal inertia matrix, where the payload is not considered, is:

$$\begin{cases} Ix = 1{,}6068\ Kgm^2 \\ Iy = 1{,}6068\ Kgm^2 \\ Iz = 2{,}9984\ Kgm^2 \end{cases} \quad (3.053)$$

where Ix and Iy was computed approximating the inertia moment of the motors equals in each axis.

$$Ix, Iy = \frac{1}{4} m_{central\ body} \left(\frac{l}{2}\right)^2 + \frac{1}{12} m_{central\ body} \left(\frac{l}{4}\right)^2 + 2I_{rotor,z} + 2(I_{rotor,z} + m_{rotor} \cdot l^2 + \\ + \frac{1}{12} m_{central\ body} \left(\frac{l}{2}\right)^2 + m_{single\ arm} \left(\frac{3l}{4}\right)^2) \quad (3.054)$$

$$aIz = 0{,}5 m_{central\ body} \left(\frac{l}{2}\right)^2 + \\ + \left(I_{rotor,z} + m_{rotor} \cdot l^2 + \frac{1}{12} m_{single\ arm} \left(\frac{l}{2}\right)^2 + m_{single\ arm} \left(\frac{3l}{4}\right)^2 \right) \cdot 4 \quad (3.055)$$

Here the central body cylinder is considered to have radius equals the half of the rotor-CoG distance l.

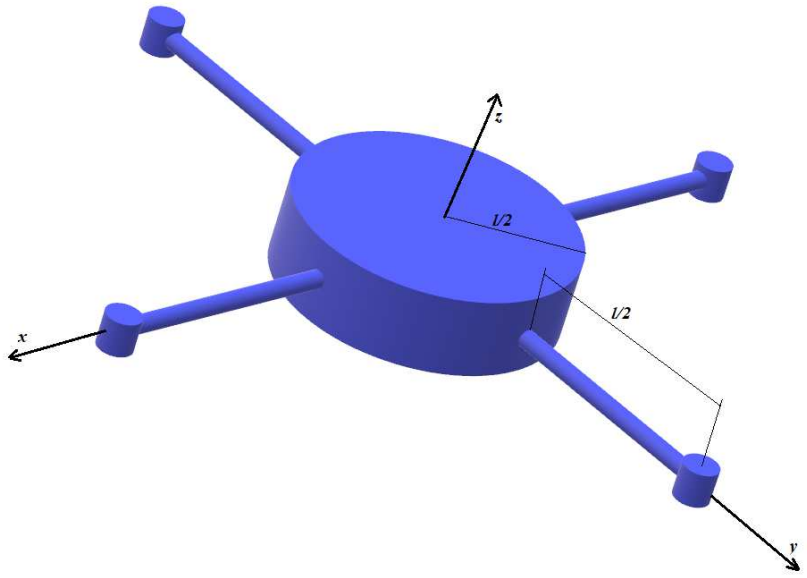


Figure 51 UAV cylindrical bodies approximation

Inertias computed including the tank payload of 10 L, supposing it as a rigid body totally included in the central body, are bigger of about the 14% than the inertia computed with no load, while the mass variation is of 40% of the initial weight. Since the maximum mass variation, after spraying, is slightly bigger than the inertia variation and the non-full tank inertia estimation cannot be reduced to a cylindrical body, because of the oscillating behavior of the contained liquid, in this

thesis the UAV will be considered with constant inertia and variable mass. Furthermore, these inertia values, computed before neglecting the payload mass, are the minimum that the UAV should encounter. From a system stability point of view under estimations generate a bigger B matrix in the linearized system, at which follows a smaller control gain matrix, so a slower response. Therefore, since in our application stability of the UAV is preferable to its agility, an underestimation of the inertia matrix is better than an overestimation. Various control strategies have been applied to the quadrotor. In literature, as the best as I know, nonlinear controls, such as sliding mode, appears to have reasonable robustness and wider operativity ranges than linear techniques because of the last one keeps reliability only around some equilibrium points. However, for non-aggressive maneuvers and/or without severe environmental turbulences linear techniques have shown reasonable results, considering the easier mathematical background required to theorize and implement the control algorithm. Some studies have compared PID control and LQR control [25] , other studies have implemented more sophisticated modern linear techniques to ensure stability margin and robustness of the control, such has Hinf control. In general, excluding robustness and optimization problems, differences and equivalences between linear control strategies are mainly dependent on the gain tuning techniques. In this thesis a modified LQR control will be implemented.

3.2.1 Linearized model, LQR with yaw gain-scheduling

To perform a linear control the UAV model is linearized around the hovering operating locus. The motor model was neglected by the control construction since its bandwidth can be considered several orders of magnitude larger than the mechanical one. That is posing $\widetilde{\overline{u_{ctrl}}} = [mg\ 0\ 0\ 0\ ]^T$ having only the gravity disturbance $\overline{u_{dist}} = [0\ 0 - mg\ 0\ 0\ 0]^T$. Imposing the stationary condition:

$$f\left(X, \widetilde{\overline{u_{ctrl}}}\right) = 0 \tag{3.056}$$

According to the state space definition above, immediately results:

$$\begin{cases} [\dot{x}\ \dot{y}\ \ \dot{z}\ \dot{\phi}\ \ \dot{\gamma}\ \ \dot{\varphi}] = 0_{1X6} \\ so \\ \boldsymbol{B_{dist}}\,[0\ 0 - mg\ 0\ 0\ 0]^T + \boldsymbol{B_{act}}[mg\ 0\ 0\ 0\ ]^T = 0_{12X1} \end{cases} \tag{3.057}$$

Substituting the matrices of the second equations and leaving only the non-null values implies that the vertical component of the thrust, in inertial frame, must equate the gravity force:

$$\boldsymbol{R_{Ib}}(:, \boldsymbol{3})\, mg = [0\ 0 - mg\ ]^T \tag{3.058}$$

i.e.

$$\begin{cases} (\cos(\phi)\cos(\varphi)\sin(\gamma) + \sin(\phi)\sin(\varphi))\, mg = 0 \\ (\cos(\phi)\sin(\varphi)\sin(\gamma) - \sin(\phi)\cos(\varphi))mg = 0 \\ \cos(\phi)\cos(\gamma)\, mg = mg \end{cases} \tag{3.059}$$

And the solution is:

$$\phi, \gamma = 0\ ; \quad \forall\varphi \tag{3.060}$$

A first clear effect of the linearization is the complete neglection of the gyroscopic torques, because in hovering condition the angle rates are null. This means that

This leads to the operating surface:

$$\widetilde{X} = \left[x\ \ y\ \ z\ \ \dot{0}\ \ 0\ \ \varphi\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\right] \tag{3.061}$$

This means that the UAV keeps its hovering equilibrium for each value of *x,y,z,* and yaw.

Anyway, in general the 1° order Taylor series expansion of the Nonlinear system depends on all the states equilibrium points:

$$\dot{X} = f(X, \overline{u_{ctrl}}) \sim f(\widetilde{X}, \widetilde{\overline{u_{ctrl}}}) + \left.\frac{\partial f}{\partial X}\right|_{\tilde{x},\tilde{u}} (X - \widetilde{X}) + \left.\frac{\partial f}{\partial u}\right|_{\tilde{x},\tilde{u}} (u - \tilde{u}) \quad (3.062)$$

With:

$$\left.\frac{\partial f}{\partial X}\right|_{\tilde{x},\tilde{u}} = A_{jac} \qquad \left.\frac{\partial f}{\partial u}\right|_{\tilde{x},\tilde{u}} = B_{actjac} \quad (3.063)$$

$$A_{jac} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & g\sin(\varphi) & g\cos(\varphi) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -g\cos(\varphi) & g\sin(\varphi) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.064)$$

$$B_{actjac} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & \frac{1}{\mathrm{Ix}} & 0 & 0 \\ 0 & 0 & \frac{1}{\mathrm{Iy}} & 0 \\ 0 & 0 & 0 & \frac{1}{\mathrm{Iz}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.04 & 0 & 0 & 0 \\ 0 & 0.6223 & 0 & 0 \\ 0 & 0 & 0.6223 & 0 \\ 0 & 0 & 0 & 0.3335 \end{bmatrix} \quad (3.065)$$

The controllability property of a linear system is the ability to directly, or indirectly, impact the time evolution of the whole set of states. For large order systems this can be check easily computing the controllability matrix:

$$C = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix} \quad (3.066)$$

where "n" is the state dimension. The system is controllable if:

$$rank(C) = n = 12 \quad (3.067)$$

In this case the above equivalence is valid for all the φ values. The controllability of the linearized system ensures a small-time local controllability of the non-linear system around the equilibrium points, namely a linear control around them is possible [26]. The important result obtained here is *the independence of the system controllability by the cartesian position and the yaw angle of the drone*. The linearized system can be seen equivalently as the dynamics of the UAV seen by a fixed frame rotating with the same attitude of the body, where the x and y accelerations are proportional respectively to pitch and roll. Furthermore, in the linearized system angular rates

coupling, due to the gyroscopic effect, is completely neglected. Turns out, also, that the jacobians \ depend on the position *x,y,z* ,but still depends on the yaw. Since is kept the overall hovering equilibrium for each yaw value, and we need to control the heading of the UAV in a wide range, the optimal solution is to use a real time Gain Scheduling in order to adjust (i.e. rotated around *z*) the control gain matrix to follow the actual yaw value[27]. Full state feedback is applied to an augmented system containing also the tracking errors of position and yaw. Can be shown that the UAV dynamic model obeys to the property of differential flatness of x,y,z,φ in the studied equilibrium point, that is the whole UAV could be theoretically controlled only controlling x,y,z,φ [28]. Full state feedback control is, anyway, a reasonable choice because creates redundancy that absorbs eventual imprecisions in the various sensors. Furthermore x,y,z,φ sensing is often the less accurate, so a control based exclusive on their tracking error is not suitable. Considering as output x,y,z,φ the augmented system is written as:

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{eye_{3X3}} & \boldsymbol{0_{3X9}} \\ \boldsymbol{0_{1X3}} & 1\,,\ \boldsymbol{0_{3X8}} \end{bmatrix} \tag{3.068}$$

$$\boldsymbol{A_{aug}} = \begin{bmatrix} \boldsymbol{Ajac} & \boldsymbol{0_{12X4}} \\ \boldsymbol{-C} & \boldsymbol{0_{4X4}} \end{bmatrix} \tag{3.069}$$

$$\boldsymbol{B_{aug}} = \begin{bmatrix} \boldsymbol{B\_actjac} \\ \boldsymbol{0_{4X4}} \end{bmatrix} \tag{3.070}$$

$$X_{aug} = \begin{bmatrix} x & y & z & \dot{\phi} & \gamma & \varphi & \dot{x} & \dot{y} & \dot{z} & \dot{\phi} & \dot{\gamma} & \dot{\varphi} & q_x & q_y & q_z & q_\varphi \end{bmatrix} \tag{3.071}$$

$$\bar{q} = \int \begin{bmatrix} (x_r - x) & (y_r - y) & (z_r - z) & (\varphi_r - \varphi) \end{bmatrix} dt \tag{3.072}$$

Final gain matrix $\boldsymbol{K_{aug}}$ is chosen in order to place the wanted eigenvalues to the matrix:

$$[\boldsymbol{A_{aug}} - \boldsymbol{B_{aug}}\boldsymbol{K_{aug}}(\varphi = 0)] \tag{3.073}$$

Gain scheduling of the control matrix will consist in the yaw (counterclockwise) rotation of the submatrices involved in the control of *Cx* and *Cy* inputs given the sensing of $x, y, \dot{x}, \dot{y}, q_x, q_y$ where the matrix to be rotated is that one computed for zero yaw rotation. That is:

$$K_{aug}(\varphi) = \begin{bmatrix} k11 & k12 & k13 & k14 & k15 & k16 & k17 & k18 & k19 & k110 & k111 & k112 & k113 & k114 & k115 & k116 \\ -s(\varphi)k22 & c(\varphi)k22 & k23 & k24 & k25 & k26 & -s(\varphi)k28 & c(\varphi)k28 & k29 & k210 & k211 & k212 & -s(\varphi)k214 & c(\varphi)k214 & k215 & k216 \\ c(\varphi)k31 & s(\varphi)k31 & k33 & k34 & k35 & k36 & c(\varphi)k37 & s(\varphi)k37 & k39 & k310 & k311 & k312 & c(\varphi)k313 & s(\varphi)k313 & k315 & k316 \\ k41 & k42 & k43 & k44 & k45 & k46 & k47 & k48 & k49 & k410 & k411 & k412 & k413 & k414 & k415 & k416 \end{bmatrix} \tag{3.017}$$

where kij are the element of the zero yaw Gain matrix $K_{aug}(0)$. $K_{aug}(0)$ matrix is computed minimizing the cost function, that leads to solve the static Ricatti equation in continuous time:

$$u^* = \arg min_u \int_0^t (X_{aug}^T\, Q\, X_{aug} + u^T R\, u) \tag{3.074}$$

The controller is in discrete time, so the above optimization problem can be seen equivalently in discrete time:

$$u^* = \arg min_u \sum_0^{t=k\cdot T_s} (X_{aug}^T(t)\, Q\, X_{aug}(t) + u(t)^T R\, u(t)) \tag{3.075}$$

$$u^*(t) = -\, K_{aug}(\varphi = 0)\, X_{aug}(t) \tag{3.076}$$

where *k* is number of samples when the sampling time is T_s. K_{aug} is:

$$K_{aug} = R^{-1} \cdot {B_{aug}}^T \cdot P \tag{3.077}$$

And P the solution of the Ricatti equation in discrete time:

$$P = {A_{aug}}^T \cdot P \cdot A_{aug} - \left({A_{aug}}^T \cdot P \cdot B_{aug}\right)\left(R + {B_{aug}}^T \cdot P \cdot B_{aug}\right)^{-1} \cdot {B_{aug}}^T \cdot P \cdot A_{aug} + Q \quad (3.078)$$

Where the state space matrices are considered in their discretized fashion, considering the first order approximation of the Taylor expansion with respect to the time exponential they are:

$$\begin{cases} A_{aug} \rightarrow (Identity_{12\times12} + T_s) \cdot A_{aug} \\ B_{aug} \rightarrow T_s \cdot B_{aug} \end{cases} \quad (3.079)$$

3.2.2 Analysis of the feedback control loop

The model of UAV and motors was then inserted in the closed loop with LQR gain matrices divided in K_x and K_q, respectively the output tracking error and state gain matrices.

$$\begin{cases} K_x(\varphi) = K_{aug}(\varphi)(1{:}4\,, 1{:}12) \\ K_q(\varphi) = K_{aug}(\varphi)(1{:}4, 13{:}16) \end{cases} \quad (3.080)$$

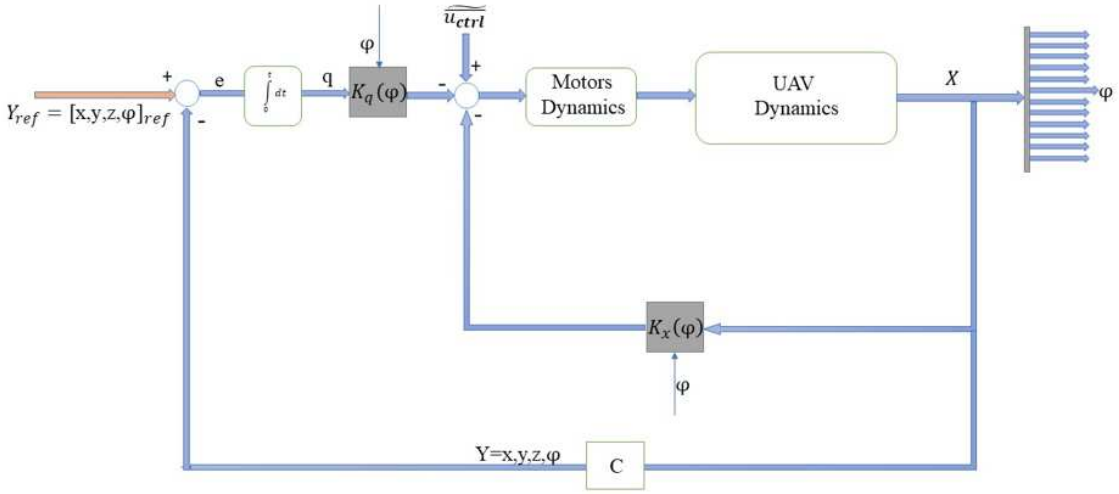


Figure 52 continuous time control loop block scheme

Since the cartesian position equilibrium point does not affect the linearized matrices values, and the yaw feeds the gain matrices, but does not influence its own dynamics was chosen to put the equilibrium state vector equaling zero, so that:

$$\left(X - \widetilde{X}\right) = X \quad ; \quad \left(Y - \widetilde{Y}\right) = Y \quad (3.081)$$

The control output produced is the command input variation, indeed the constant equilibrium control input is added $\widetilde{u_{ctrl}} = [mg\ 0\ 0\ 0\ ]^T$, although, in order to avoid too fast rotors acceleration, the hovering thrust reference passes from zero to mg smoothly, once requested. In order to evaluate the feedback control loop in Laplace domain the output control and the stabilized plant was converted into transfer function matrices, neglecting actuator dynamics, in the following way:

$$\begin{cases} C(s) = {K_q(0)}/{s} \\ P(s) = C\left(eye_{12\times12} \cdot s - A_{jac} + B_{actjac}K_x(0)\right)^{-1}B_{actjac} \end{cases} \quad (3.082)$$

And the loop function is:

$$L(s) = P(s)C(s) \quad (3.083)$$

where:

$$\begin{cases} L_{11} = {}^{x}/_{(x_{ref})} = \dfrac{-7{,}7981}{s\,(s^2 + 3{,}814s + 4{,}131)\,(s^2 + 1{,}813s + 4{,}916)} \\ L_{22} = {}^{y}/_{(y_{ref})} = \dfrac{-9.6125}{s\,(s^2 + 3{,}938s + 4{,}182)\,(s^2 + 2{,}126s + 5{,}989)} \\ L_{33} = {}^{z}/_{(z_{ref})} = \dfrac{-0{,}072061}{s\,(s^2 + 0{,}8334s + 0{,}3468)} \\ L_{44} = {}^{\varphi}/_{(\varphi_{ref})} = \dfrac{-0{,}54627}{s\,(s^2 + 1{,}636s + 1{,}341)} \end{cases} \quad (3.084)$$

The loop function is a diagonal matrix. L_{ij} is the feedforward transfer function between the j-th output reference and the i-th output. Exploiting the Nyquist Theorem $L(s)$ can be used to evaluate the stability of the whole feedback loop transfer function, since all the Input-Output relationships in the system have the common denominator:

$$eye_{4\times4}/(eye_{4\times4} + L(s)) \quad (3.085)$$

In this phase, the research of Q and R matrices consisted of leaving $R = eye_{4\times4} \cdot 110$ while Q was differentially tuned in its own elements in order to reach a good enough time domain response and a suitable gain and phase margin in frequency domain. After a trial and error resulted:

$$Q = diag(10,\ 10\ ,\ 10\ ,\ 1\ ,\ 1\ ,\ 10\ ,\ 400\ ,760\ ,120\ ,\ 0.1\ ,\ 0.1\ ,\ 0.1\ ,\ 190\ ,\ 290\ ,\ 360\ ,\ 300) \quad (3.086)$$

$$R = diag(110\ ,\ 110\ ,\ 110\ ,\ 110) \quad (3.087)$$

Note: Although the translational control (in $x_{Inertial}$ and $y_{Inertial}$) should be theoretically identical, is not the case for our application because the geometry of the vineyard imposes more fast and aggressive tracking along the rows than in the orthogonal direction. Conventionally will be always considered x aligned with the rows and y orthogonal to them. For those reasons the elements of the Q matrix associated with $x_{Inertial}$, $y_{Inertial}$ and their integral or derivatives are in general different.

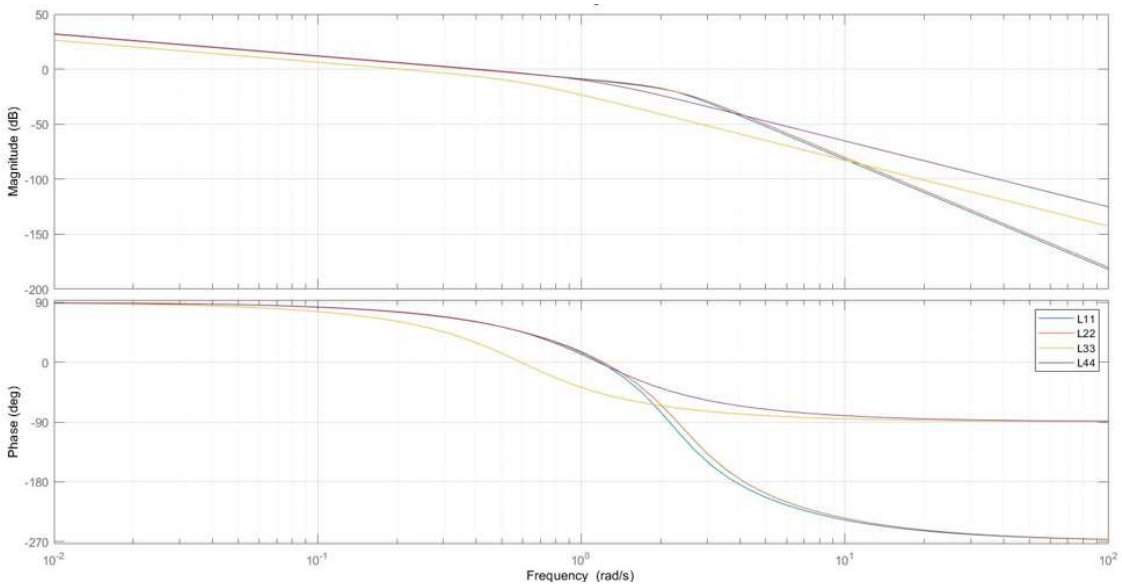


Figure 53 Bode diagram of L(s)

Gain margin is defined as the ratio of the unity and the loop transfer function magnitude at the frequency where the phase is the crossover phase of 180°. Dually the phase margin is defined as the difference between the phase at the frequency where the magnitude is 1 and -180°. In the table below are summarized the two quantities and the crossover frequency ω_c for which the magnitude equals 1:

Table 13 Loop functions characteristics

	L_{11}	L_{22}	L_{33}	L_{44}
Phase margin [°]	240	240	242	240
Gain Margin [dB]	41	41	/	/
ω_c **[Hz]**	0,06	0,06	0,02	0,06

Always supposing a linear system, form the table can be noticed that the z position and the yaw don't have a computable gain margin because their phases never reach -180°. This guarantees the stability of z and yaw whatever are the control constant gain values. Another important transfer function to evaluate is the whole feedback loop transfer function which correlates the reference and the output:

$$F(s) = L(s)/(eye_{4\times4} + L(s)) \quad (3.088)$$

Significative frequencies of this transfer function could be a first estimate of the loop bandwidth, useful to give an idea of the digital control sampling time order of magnitude.

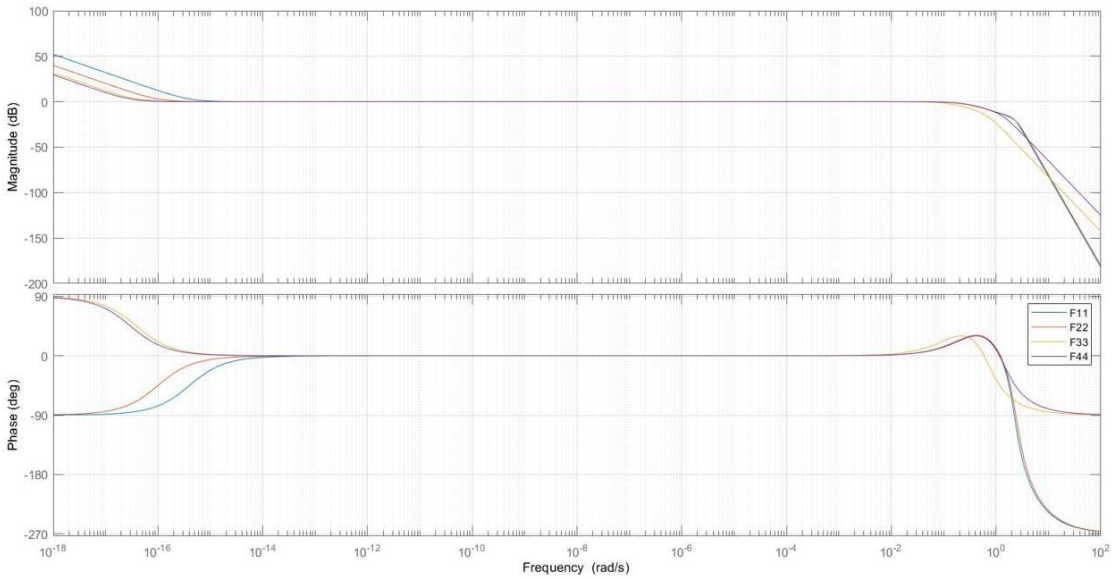


Figure 54 F(s) bode diagram

For each one of the outputs signals the cut-off frequency is in the range of 3-30 Hz, and this can be seen as the bandwidth of the whole continuous time feedback control loop, namely the minimum frequency the control system must be able to operate. The control was also tested in 4 principal movements:

- Latitudinal translation along x
- Longitudinal translation along y
- Vertical take off
- Yawing

These 4 principal operations are tested in a decoupled way, namely ensuring that when the pitch angle is increased to move latitudinally roll, yaw, roll rate and yaw rate are null and the same applies for roll and yaw tests. Step input is the most aggressive reference input since at the step

starting time the derivative is infinite, and this traduces in an infinite velocity reference. For this reason, the time response of the four output has large overshoots. However, in the path planning the variables indirectly controlled are the velocity of the outputs. They will be controlled changing the slope of the reference time history for each controlled variable and, at that point, overshoot becomes less evident or absent. In the z response can be seen a larger time delay between the step activation (at time zero) and the first output variation (about 7 seconds) than other 3 outputs. This happens because in the model was inserted a realistic UAV-ground-gravity interaction that causes the drone to be subjected to ground reaction and gravity when $z=0$, and only gravity when takes off, so that the takeoff starts when the rotors reach a velocity that produce enough thrust to lift the body (this happens not immediately because of the limitation on the equilibrium command input discussed before).

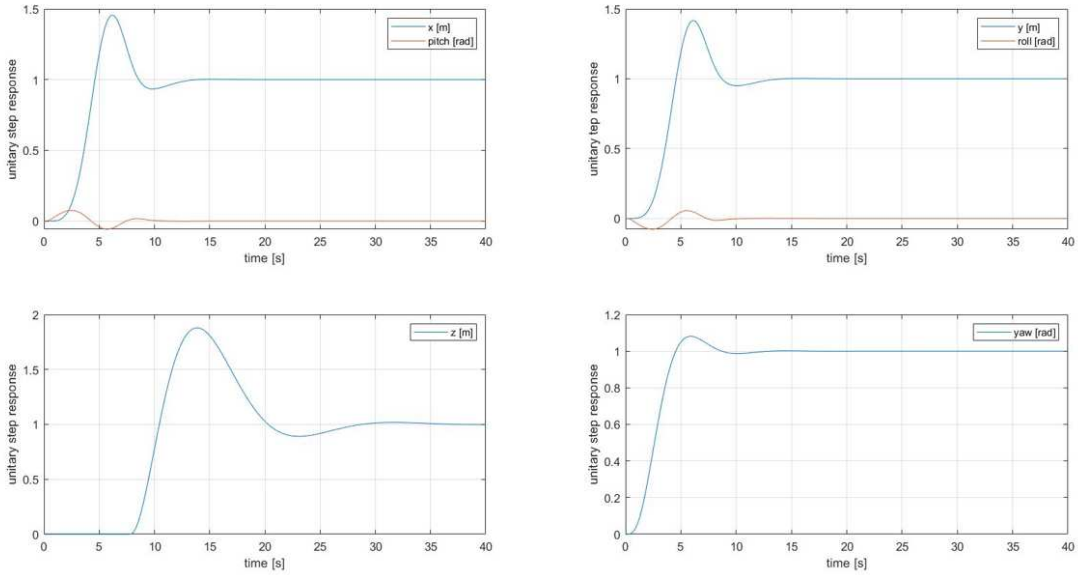


Figure 55 Unitary step responses

3.2.3 Rows-following simulation

Controller was tested on about 60 seconds path planning in which a series of distances, trajectories and velocities tries to simulate the future operations of the drone in this specific application. Was simulated a 20 meters long vineyard with 2 meters in between each row. The UAV take off and reach 5 meters altitude, while starts to follow the row with 2 m/s and zero yaw. Once overflights the whole row turns with a 1-meter radius and simultaneously inverts the heading reaching 180° of yaw. Once the half of each row is reached the UAV stopped, keeps hovering condition for 2 seconds and then restarts. The same is repeated 3 times in a time window of about 56 second. The same path planning will be used in the rest of this chapter.

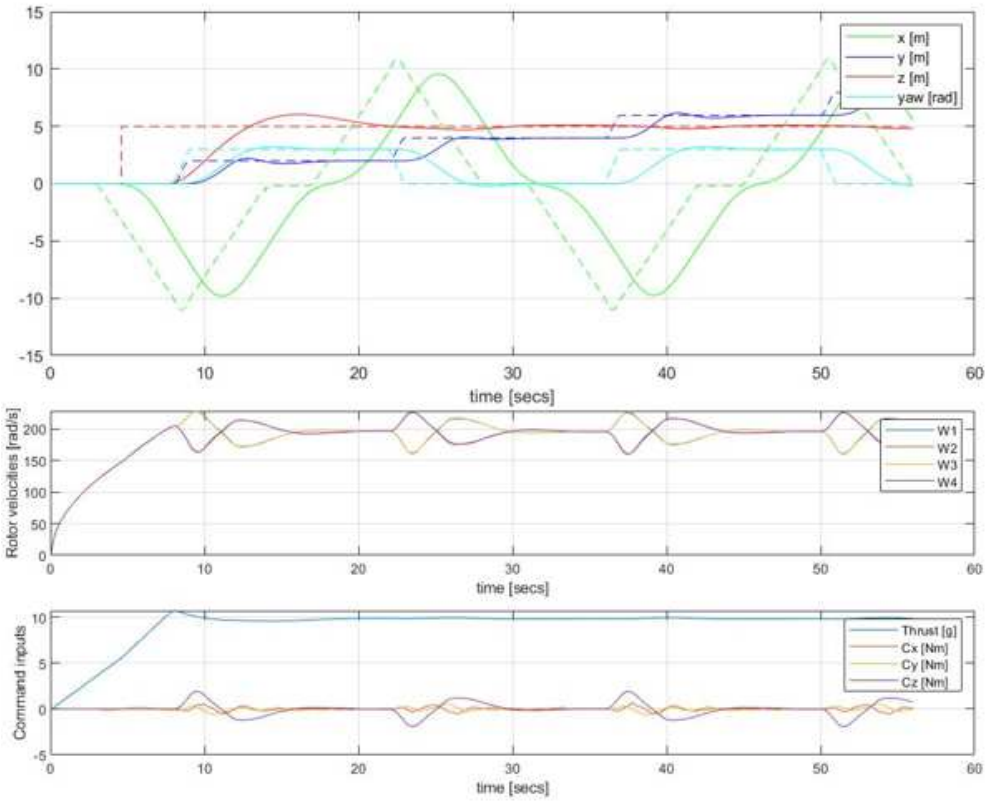


Figure 56 State tracking and command inputs in time

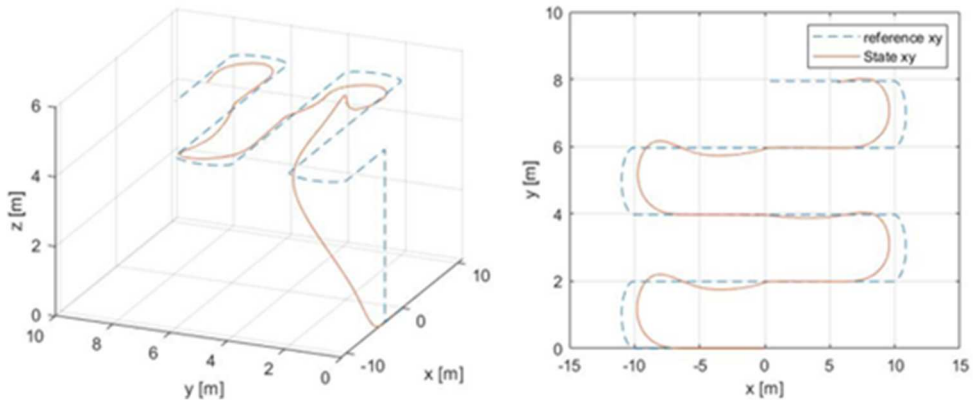


Figure 57 State tracking in space

Two error sources are present:

- The single integration action does not allow a zero steady state error when the input reference is more than linear, so a constant error due to a delay between ramp inputs and actual state is always present. However, this can be corrected simply anticipating the reference of a complementary amount of time if necessary.
- In the turning phase the x deceleration is about 5 m/s^2 against a 2 m/s^2 acceleration in *y*. This asymmetry causes the not perfect tracking of the semicircle but is only due to the aggressive deceleration imposed in *x* direction, thoughted to test the control. In the real application there will be no reasons to adopt such high accelerations.

The sampling time of the controller was 1 millisecond. However, a control parameter finer tuning will be done once the state estimation block and other nonlinearities will be introduced. In the rest

of the chapter will be supposed that the whole mass of the drone does not change with time, then the problem of the payload tank emptying will be discussed.

3.3 State estimation

State estimation in aerial vehicles is crucial because some states are not directly measurable, or their direct measure has a too low Signal to Noise Ratio (SNR). Linear state observer, such as the Luenberger observer or the Kalman Filter can be extended to the nonlinear dynamics of aerial vehicles or nonlinear techniques as the sliding mode, particle filter or Unscendent Kalman FIlter could be adopted if there are too severe nonlinearities. Linear techniques are well known, computational lighter and of easier implementation, however if their convergence to the actual state is mathematical guaranteed for linear systems this is not true for nonlinear systems. The commonly used sensors equipment in aerospace industry includes an inertial unit, composed by an accelerometer and a gyroscope, a magnetometer and a RTK-GPS or a ground triangulation system, such as Ultra-Wide Band (UWB). With these sensors the quantities directly measured are the body frame accelerations with the accelerometer, the body frame twist with the gyroscope, the inertial attitude with the magnetometer and the inertial position with GPS or other equivalent solutions. However, the attitude measured by a commercial magnetometer is affected by severe electromagnetic interference coming from the environment and the on-board electronic itself and the position accuracy ensured by common GPS systems is in the order of meters (10-30 cm if is added an RTK system) and the GPs signal reliability depends on the climatic conditions. These factors make unappropriated the solely use of magnetometer and GPs for autonomous guidance. On the other hand accelerometer and gyroscope gives a short-term accurate measure of velocity, position and attitude simply integrating acceleration and angle rates sensed (Dead Reckoning), but this method suffer the drift of the estimated quantities due to biases and noises in the original signals[29]. Ground positioning systems such as the UWB have demonstrated accuracy in the order of centimeters. It exploits nanosecond time window radiofrequency pulses sent by ground fixed stations and an on-board receiver that receives these signals. With a ToF (time of flight) technique the system retrieves the receiver position [30]. UWB technology seems to overcome some of the main problems linked to the GPS system (signal leakage, low accuracy range ecc..), but will be not covered in this thesis.

3.3.1 State observer applied to the sensor models: AHRS approach

Classical state observers use the dynamic model of the controlled system to give an estimate of the actual unmeasurable states. If the system is linear, it can be written in state space form

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases} \quad (3.089)$$

Exploiting this model is designed an observer dynamic that emulates the original system taking in input the (known) system command input with the same B matrix and the difference between the measured and the estimate output, multiplied by a tunable gain:

$$\begin{cases} \dot{\hat{x}} = A\hat{x} + Bu - L(y - \hat{y}) = A\hat{x} + Bu + L(Cx - C\hat{x}) \\ \hat{y} = C\hat{x} \end{cases} \quad (3.090)$$

So that the estimation error becomes

$$\dot{x} - \dot{\hat{x}} = (A - LC)(\dot{x} - \dot{\hat{x}}) \quad (3.091)$$

If the system is observable the L matrix can be computed through pole placement (Luenberger observer) or as result of a cost function identical to that one used for the LQR, but where Q and R are the covariance matrices of noise applied to the output measures y and to the input measures u respectively (Kalman Filter, Linear Quadratic Gaussian). However, in our case, there are many problems to apply these strategies:

- The UAV model is not linear. Extended Luenberger observer or Extended Kalman filter could circumvent the problem computing the L gain in real time for all the equilibrium points, but the system is in general not observable for all the equilibrium points.
- Disturbances and model uncertainties could lead to a failure of this approach.
- Measuring the command input u means to evaluate in real time thrusts of each rotor trusting only on their experimental relation with propellers velocities.

To overcome the problem the same approach can, however, be applied to the sensors dynamical models and it leads to the Data Fusion solved by of the AHRS (attitude-heading reference systems) to estimate the attitude of the drone and the fusion of INS (inertial navigation system) and GPS to estimate the cartesian coordinates. In this chapter is studied the AHRS only, since the IMU+GPS data fusion follows the same principal logics[29].

3.3.2 Sensors models

Since the aim is the estimation of the attitude the only dynamics is the relation between the body frame twist $\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}$ directly sensed by the gyroscope and the Euler angles rates $\begin{bmatrix}\dot{\phi}\\ \dot{\gamma}\\ \dot{\varphi}\end{bmatrix}$. The kinematic relationship is expressed by the inverse of the transformation matrix $\boldsymbol{T_{YRP}(\alpha)^{-1}}$. At the twist must be subtracted the, supposed, white gaussian noise vector and the constant bias vector of the sensor. As output are selected the body frame acceleration and magnetic field components computed rotating the inertial gravity vector and the inertial magnetic field components through $\boldsymbol{R_{Ib}^{T}(\alpha)}$. Also, to them must be added a vector that includes white gaussian noises and biases of the accelerometer and magnetometer signals. Is worth to notice that the outputs of this dynamical system are the body frame accelerations and magnetic field components computed based on the sensor model attitude. The magnetometer is fundamental to correct the yaw prior estimate because the rotation of the gravity vector from inertial to body frame depends only by pitch and roll. Supposing this model for the sensor, an observer must compute the time evolution of the attitude based on the model and subtract to it the difference between the prior estimate of the output, computed linearizing $h(\alpha, \bar{v} + \overline{b^{a,m}})$ through the assumed attitude, and the actual measured output derived from accelerometer and magnetometer. Then, similarly to the linear case, the *K* matrix will derive from the choice of Q and R matrices, solving the Ricatti equation.

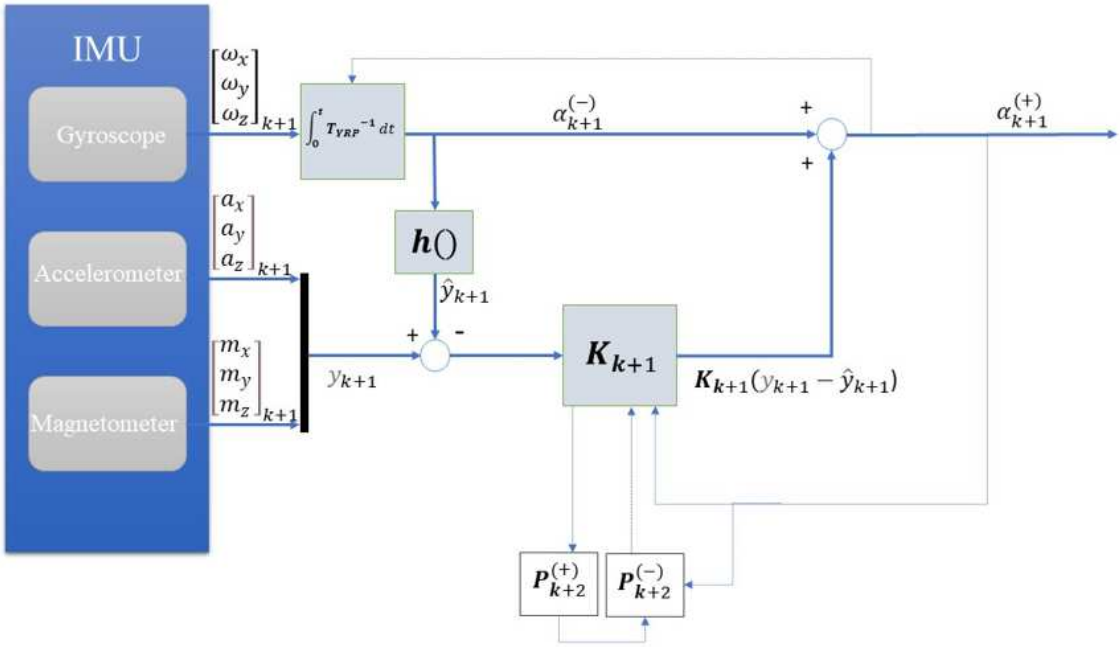


Figure 58 AHRS Extended Kalman filter blocks scheme

3.3.3 Extended Kalman Filter algorithm

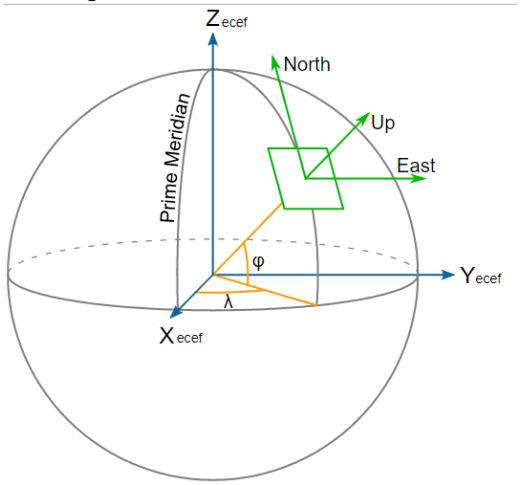


Figure 59 NEU reference frame

Magnetometer, accelerometer and gyroscope models[31]:

$$\dot{\alpha} = \begin{bmatrix} \dot{\phi} \\ \dot{\gamma} \\ \dot{\varphi} \end{bmatrix} = \boldsymbol{T_{YRP}(\alpha)^{-1}} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} - \boldsymbol{T_{YRP}(\alpha)^{-1}} \left(\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} + \begin{bmatrix} b_x^G \\ b_y^G \\ b_z^G \end{bmatrix} \right) \quad (3.092)$$

$$\begin{bmatrix} a_x \\ a_y \\ a_z \\ m_x \\ m_y \\ m_z \end{bmatrix}_b = \begin{bmatrix} \boldsymbol{R_{Ib}^T(\alpha)} & \boldsymbol{0_{3X3}} \\ \boldsymbol{0_{3X3}} & \boldsymbol{R_{Ib}^T(\alpha)} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -g \\ M_x \\ M_y \\ M_z \end{bmatrix}_{In} - \begin{bmatrix} \boldsymbol{R_{Ib}^T(\alpha)} & \boldsymbol{0_{3X3}} \\ \boldsymbol{0_{3X3}} & \boldsymbol{R_{Ib}^T(\alpha)} \end{bmatrix} \left(\begin{bmatrix} v_x^a \\ v_y^a \\ v_z^a \\ v_x^m \\ v_y^m \\ v_z^m \end{bmatrix} + \begin{bmatrix} b_x^a \\ b_y^a \\ b_z^a \\ b_x^m \\ b_y^m \\ b_z^m \end{bmatrix} \right) \quad (3.093)$$

Mx, My and *Mz* are the earth magnetic field components along the inertial axes, if *y* Inertial is aligned with north the magnetometer measures about (N-E-Up config):

$$\mathrm{Mx} = -1\ \mathrm{uT} \quad \mathrm{My} = 26\ \mathrm{uT} \quad \mathrm{Mz} = -44\ \mathrm{uT} \tag{3.094}$$

The measure was effectuated in a closed environment in Turin (To), Italy, without filtering and compensation of environment (inertial) electromagnetic interference and body frame (MCU an electronics) electromagnetic interference.

where $\overline{w} \sim N(0, Q)$ $\bar{v} \sim N(0, R)$ white gaussian noises and b_i^j constant biases.

Expressing the system as nonlinear state space model:

$$\dot{\alpha} = f(\alpha, \overline{w} + \overline{b^G}) \tag{3.095}$$

$$y = h(\alpha, \overline{v} + \overline{b^{a,m}}) \tag{3.096}$$

Given a current state $\tilde{\alpha}$, with:

$$A = \left.\frac{\partial f}{\partial \alpha}\right|_{\alpha=\tilde{\alpha},\ \overline{w}=\overline{b^G}=0} \quad ; B = \left.\frac{\partial f}{\partial(\overline{w}+\overline{b^G})}\right|_{\alpha=\tilde{\alpha},\ \overline{w}=\overline{b^G}=0} ; C = \left.\frac{\partial h}{\partial \alpha}\right|_{\alpha=\tilde{\alpha},\ \overline{v}=\overline{b^{a,m}}=0} \quad ; D = \left.\frac{\partial f}{\partial(\overline{v}+\overline{b^{a,m}})}\right|_{\alpha=\tilde{\alpha},\ \overline{v}=\overline{b^{a,m}}=0} \tag{3.097}$$

Results:

$$B = -\boldsymbol{T_{YRP}}(\alpha)^{-1} \qquad D = -\begin{bmatrix} \boldsymbol{R_{Ib}^T}(\alpha) & \boldsymbol{0_{3X3}} \\ \boldsymbol{0_{3X3}} & \boldsymbol{R_{Ib}^T}(\alpha) \end{bmatrix} \tag{3.098}$$

The discretized sensor model is:

$$\alpha_{k+1} = \alpha_k + Ts\boldsymbol{f}\left(\boldsymbol{\alpha}, \overline{\boldsymbol{w}} + \overline{\boldsymbol{b^G}}\right) \quad \cong \quad \left(\boldsymbol{Id} + Ts\,\boldsymbol{A}(\boldsymbol{\alpha_k})\right)\alpha_k + Ts\,\boldsymbol{B}(\boldsymbol{\alpha_k})\,(\overline{w} + \overline{b^G}) \tag{3.099}$$

$$\hat{y}_{k+1} = \boldsymbol{h}\left(\boldsymbol{\alpha}, \overline{\boldsymbol{v}} + \overline{\boldsymbol{b^{a,m}}}\right) \quad \cong \quad \boldsymbol{C}(\boldsymbol{\alpha_k})\alpha_k + \boldsymbol{D}(\boldsymbol{\alpha_k})\,(\overline{v} + \overline{b^{a,m}}) \tag{3.100}$$

The discrete time Extended Kalman Filter algorithm (EKF) is the following:

Initialization:

- Set Q, R analyzing the sensors covariance
- Set α_0 knowing the initial *YPR* downstream a calibration
- Set P_0 (state covariance matrix) as larger as the uncertainty on the initial state

Prediction:

Use only the model to predict the a priori knowledge about state dynamics and solve the difference Ricatti equation to update the covariance matrix.

$$\alpha_{k+1}^{(-)} = \alpha_k^{(+)} + Ts\boldsymbol{f}(\boldsymbol{\alpha}, \omega_x, \omega_y, \omega_z) \tag{3.101}$$

$$\boldsymbol{P_{k+1}^{(-)}} = \boldsymbol{P_k^{(+)}} + Ts\,[\boldsymbol{A}(\boldsymbol{\alpha_k})\boldsymbol{P_k^{(+)}}\boldsymbol{A}(\boldsymbol{\alpha_k})' + \boldsymbol{B}(\boldsymbol{\alpha_k})\boldsymbol{Q}\boldsymbol{B}(\boldsymbol{\alpha_k})'] \tag{3.102}$$

Update: Compute the Kalman observer gain solving the static Ricatti equation at current time step, i.e., scaling the prior state covariance matrix and the observation covariance matrix by the weight matrices of the state space model. Then obtain a posteriori prediction of the state adjusting it to the current measurement, and finally reupdate the a posteriori state covariance matrix:

$$\boldsymbol{K_{k+1}} = \boldsymbol{P_{k+1}^{(-)}}\,\boldsymbol{C}(\boldsymbol{\alpha_k})'\,[\boldsymbol{C}(\boldsymbol{\alpha_k})\boldsymbol{P_{k+1}^{(-)}}\boldsymbol{C}(\boldsymbol{\alpha_k})' + \boldsymbol{D}(\boldsymbol{\alpha_k})\boldsymbol{R}\,\boldsymbol{D}(\boldsymbol{\alpha_k})']^{-1} \tag{3.103}$$

$$\alpha_{k+1}^{(+)} = \alpha_{k+1}^{(-)} + \boldsymbol{K_{k+1}}([a_x\ a_y\ a_z\ m_x\ m_y\ m_z]^T - \boldsymbol{C}(\boldsymbol{\alpha_k})\alpha_{k+1}^{(-)}) \tag{3.104}$$

$$\boldsymbol{P_{k+1}^{(+)}} = \boldsymbol{P_{k+1}^{(-)}} + \boldsymbol{K_{k+1}}\boldsymbol{C}(\boldsymbol{\alpha_k})\boldsymbol{P_{k+1}^{(-)}} \tag{3.105}$$

The algorithm generates a Kalman gain of dimension *3x6* since 3 magnetometer and 3 accelerometer measures produce the so-called *innovation factor* to correct the 3 angles. Is well-known that [29]:

- Magnetometers suffers wider sources of interference than the inertial sensors because of lower Signal-to-Noise ratio caused by the small magnitude of the earth magnetic field (micro-Tesla) and potentially high field distortion caused by external electromagnetic fields or ferromagnetic materials.
- At least MEMS magnetometer, have higher internal noise levels than gyroscope and accelerometer.
- Pitch and Roll angle estimation precision needs to be relatively higher than yaw estimation since also small errors could cause the system to become unstable.

For those reasons is decided to nullify the submatrix of K responsible of Pitch and roll a posteriori correction through magnetometers reading, namely:

$$K([1,2],[4,5,6]) = \mathbf{0_{2X3}} \quad (3.106)$$

However, pitch and roll correction by means of accelerometer only is theoretically correct in translational stationary condition of the UAV, but, if not the case, any acceleration is recorded and added to the gravity vector. Anyway, for small accelerations this issue can be neglected, as seen later.

3.3.4 Validation of the sensor model through real implementation

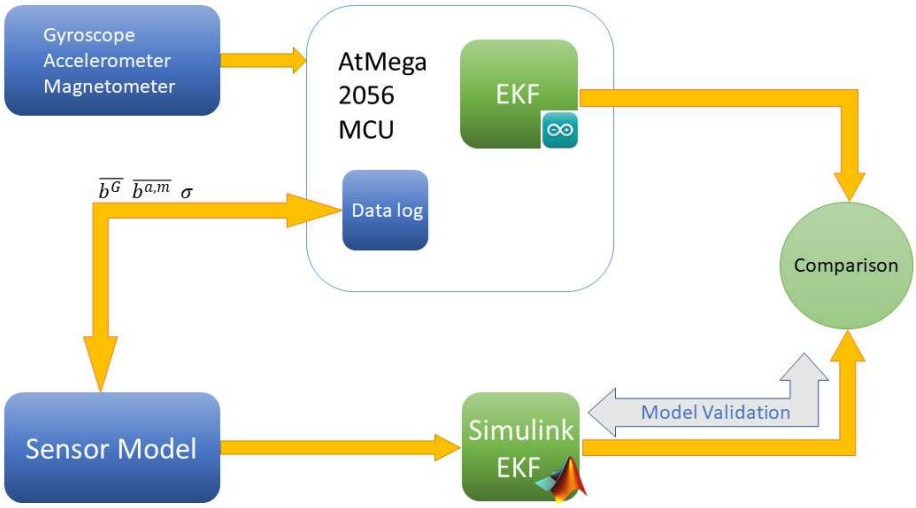


Figure 60 Sensor models validation conceptual scheme

For the experimentation was used the MPU-9250 Inertial Measurement Unit by InveSense. It is a 9-dof IMU containing 3 axis gyroscope, 3 axis magnetometer and 3 axis accelerometer and a temperature sensor in addition. Under the assumption of uncorrelated gaussian noises with constant biases and assuming negligible the influence of temperature on the sensor electronics, values of interest was extracted experimentally reading the sensor values over time. The manufacturer provided the noise amplitude spectral density of the gyroscope noise (squared root of the power spectral density (PSD)), expressed in °/s/√Hz, and an estimate of its rms value when the sampling time is 94 Hertz. In the same way noise spectral densities are provided also for gyroscope and magnetometer. These values are obtained:

Table 14 MPU-9250 IMU tecnical sheet

	Noise Rms @92Hz	**Noise ASD**	**Full Scale Range**	**Sensitivity**
Gyroscope	1,7301e-03 rad/s	1,7301e-04 rad/s/√Hz	±8,6505 rad/s	3.7859e+03 LSB/rad/s
Accelerometer	8 mg	300 µg/√Hz	±8 g	4,096 LSB/g
Magnetometer	/	/	±4800 µT	1,7 LSB/ µT

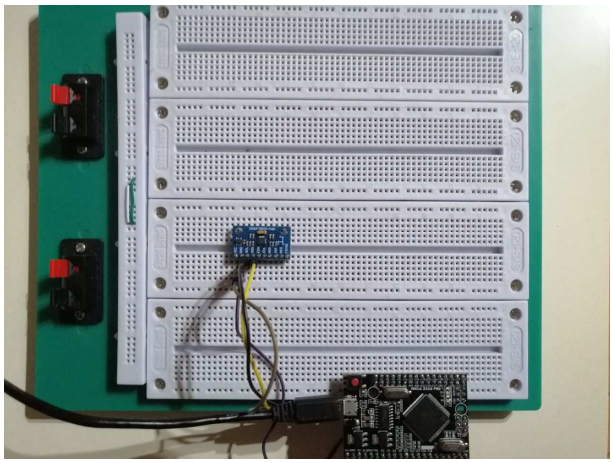
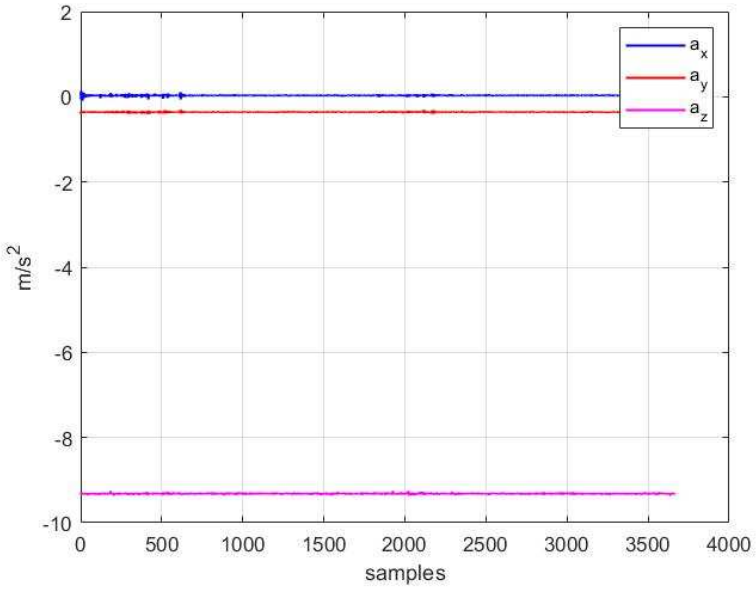


Figure 61 MPU 9050 Imu and AtMega 2056 in I2C interface

For sake of precision the estimation of noise variance and bias was extracted reading directly the sensors, then use them in the simulation to model the sensors with realistic quantities. To simulate the zero-mean noise, a band-limited white gaussian noise block is inserted with the chosen sample time and the reported *PSD* for accelerometer and gyroscope. Biases and variances are estimated living stationary and flat the IMU and measuring the offsets. The update frequency of the MCU used (AT Mega 2056) with the running routine was about 20 Hz against a maximum frequency update of 250 Hz for the IMU. Routine was written in Arduino Ide environment using a precompiled library for MPU 9050 with already incorporated scale factors and signal/axles correlation. The communication protocol used was I2C. The records span over about 5 minutes:



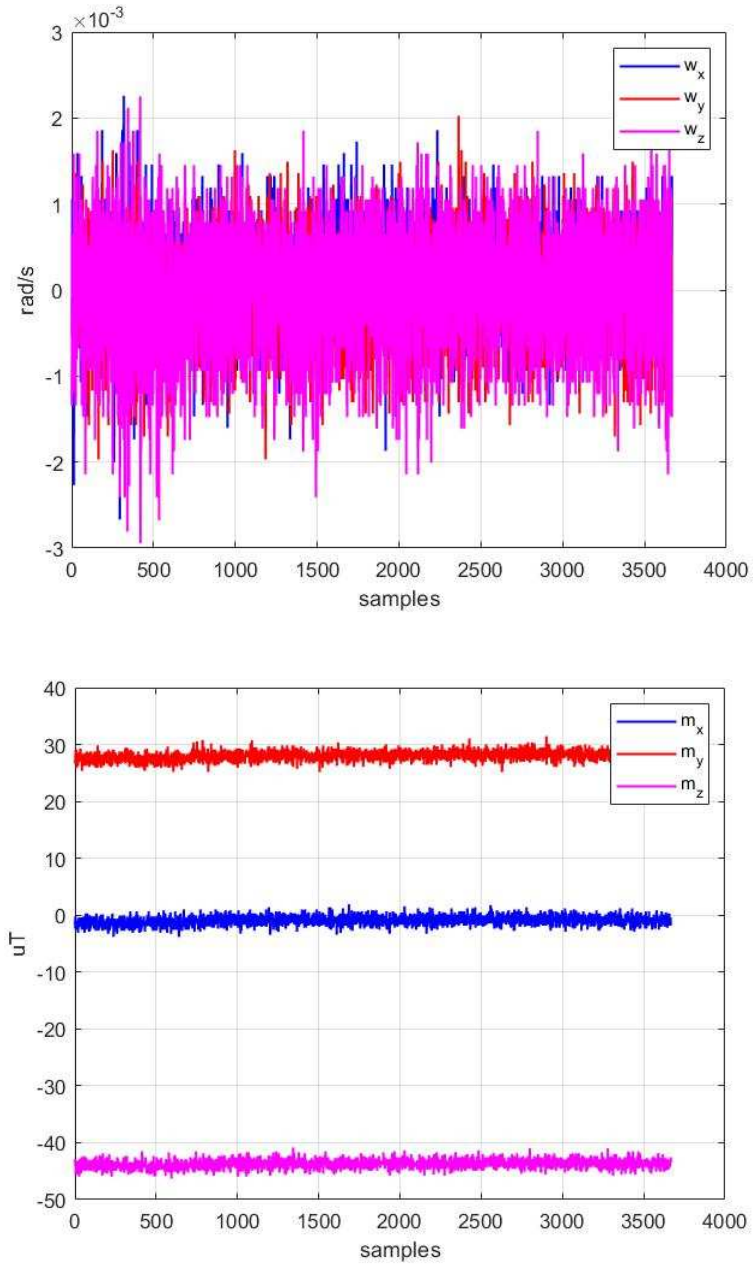


Figure 62 Accelerometer (top), gyroscope (center) and magnetometer (bottom) records

The gyroscope is the only sensor for which the measurements in stationary conditions always give zero values since it measures angular rates. Accelerometer and magnetometer measure the absolute orientation of the IMU relative to the gravitational field and the magnetic field. Thus, although the IMU is oriented with y-axis pointing North, x-axis pointing East and Z pointing up there will be always a tolerance in orientation shared by the three components. This means that, for the gyroscope, biases can be obtained individually for the 3 components directly by the mean of their samples, instead was assumed equal the 3 biases component for accelerometer and for magnetometer obtained computing a single vector magnitude bias and retrieving by it the 3 equal components. Is worth noticing that differently for the gyroscope, whose biases will be estimated and compensated by the filter, a good calibration of magnetometer and accelerometer is essential to compute a coherent a posteriori estimate.

For the gyroscope results:

$$\begin{cases} {\sigma^2}_G = [0{,}2524 \quad 0{,}2909 \quad 0{,}4700]\, 10^{-6} \quad [{rad^2}/{s^2}] \\ b_G \quad = [0{,}1146 \quad -0{,}1002 \quad 0{,}0138]\, 10^{-3} \quad [{rad}/{s}] \end{cases} \tag{3.107}$$

For the accelerometer and magnetometer:

$$\begin{cases} {\sigma^2}_A = [0{,}1539 \quad 0{,}0943 \quad 0{,}1788]\, 10^{-3} \quad \left[{m^2}/{s^4}\right] \\ b_A \quad = -[0{,}283 \quad 0{,}283 \quad 0{,}283] \quad \left[{m}/{s^2}\right] \\ {\sigma^2}_M = \quad [0{,}6801 \quad 0{,}6979 \quad 0{,}5942] \quad [\mu T^2] \\ b_M \quad = \quad [0{,}3304 \quad 0{,}3304 \quad 0{,}3304] \quad [\mu T] \end{cases} \tag{3.108}$$

The computations shows that noises standard deviations are generally an order of magnitude lower than RMS values specified by the manufacturer, so the sensor was properly working. Magnetometer and accelerometer biases were computed decomposing the differences between the measured magnetic vector and gravity vector magnitudes and the supposed magnetic vector and gravity vector magnitude at the current location. The bias components are then multiplied by the sign of the averages of their corresponding measurements. (in this way if the bias is positive it goes to increase the modulus of the component, if negative it reduces it).

$$b_A = \frac{\sqrt{3}}{3}\left(\sum_{k=1}^{N_{sample}} \frac{\sqrt{\left(a_{x,k}^2 + a_{y,k}^2 + a_{z,k}^2\right)}}{N_{sample}} - G\right)[sign(\widehat{a_x}) \quad sign(\widehat{a_y}) \quad sign(\widehat{a_z})] \quad \left[{m}/{s^2}\right] \tag{3.109}$$

$$b_M = \frac{\sqrt{3}}{3}\left(\sum_{k=1}^{N_{sample}} \frac{\sqrt{\left(m_{x,k}^2 + m_{y,k}^2 + m_{z,k}^2\right)}}{N_{sample}} - M\right)[sign(\widehat{m_x}) \quad sign(\widehat{m_y}) \quad sign(\widehat{m_z})] \quad [\mu T] \tag{3.110}$$

where:

$$G = 9{,}81 \quad \left[{m}/{s^2}\right] \qquad ; \qquad M = 47{,}26 \quad [\mu T] \tag{3.111}$$

These last 2 computations can be used both to further calibrate the sensors and increase the consistence of the a posteriori estimation and to validate an eventual EKF the estimate the cartesian position fusing IMU and GPS signals [29]. Furthermore, all those values can be implemented in the simulation, adding a bandwidth limited white noise block running at the same sampling time of the microcontroller with a noise power $P_{noise} = \sigma^2\, T_{sampling}$. Adding, also, relative biases and a 12 bits quantization, as for the used real MCU, the sensor model is complete. The EKF algorithm was developed directly in a MATLAB function block in the Simulink environment containing all the models developed until now, that are UAV dynamic model, motors model and sensors model. The next step is to compare the behavior of the standalone EKF in simulation with the experimental test bench, in static conditions. After that, more complicated state tracking of the drone will be proposed in the simulation environment. For the comparison a $T_{sampling} = 35\, ms$ was chosen to reduce the computational burden on the MCU. Both in the simulation and in the reality the initial condition of the Kalman filter are set aleatory to 1 radiant (57.8 °) in order to evaluate the convergence speed. Instead, the real values to be tracked are perfectly set to zero in the simulation and near to zero in the experimental phase since was impossible to calibrate the IMU on a perfectly horizontal position and to orient the y axis toward north, taking also into account that each time the routine is started the magnetometer read sensibly different magnetic field components, both for the electronic noise and for the unavoidable and unpredictable indoor interference. For those reasons the value to be tracked in the experimental EKF was chosen as the steady state value reached by each angle after several minutes of recording. Since the purpose is to evaluate the convergence time in the transient the above approximation is sufficient. Each of the next graphs shows an angle estimation in Simulink and

its respective in the test bench experimentation. The algorithms were identical as also the main EKF parameters:

$$R = diag(1,1,1,100,100,100)10^{-3} \quad (3.112)$$

$$Q = diag(1,1,1)10^{-4} \quad (3.113)$$

$$P_0 = diag(1,1,40)10^{-5} \quad (3.114)$$

The test was repeated 3 times, alternating the 1 radiant initial condition to each angle.

$$RPY_{initial\ conditions} = \begin{matrix}[1\ 0\ 0] \\ [0\ 1\ 0] \\ [0\ 0\ 1]\end{matrix}\ rad \quad (3.115)$$

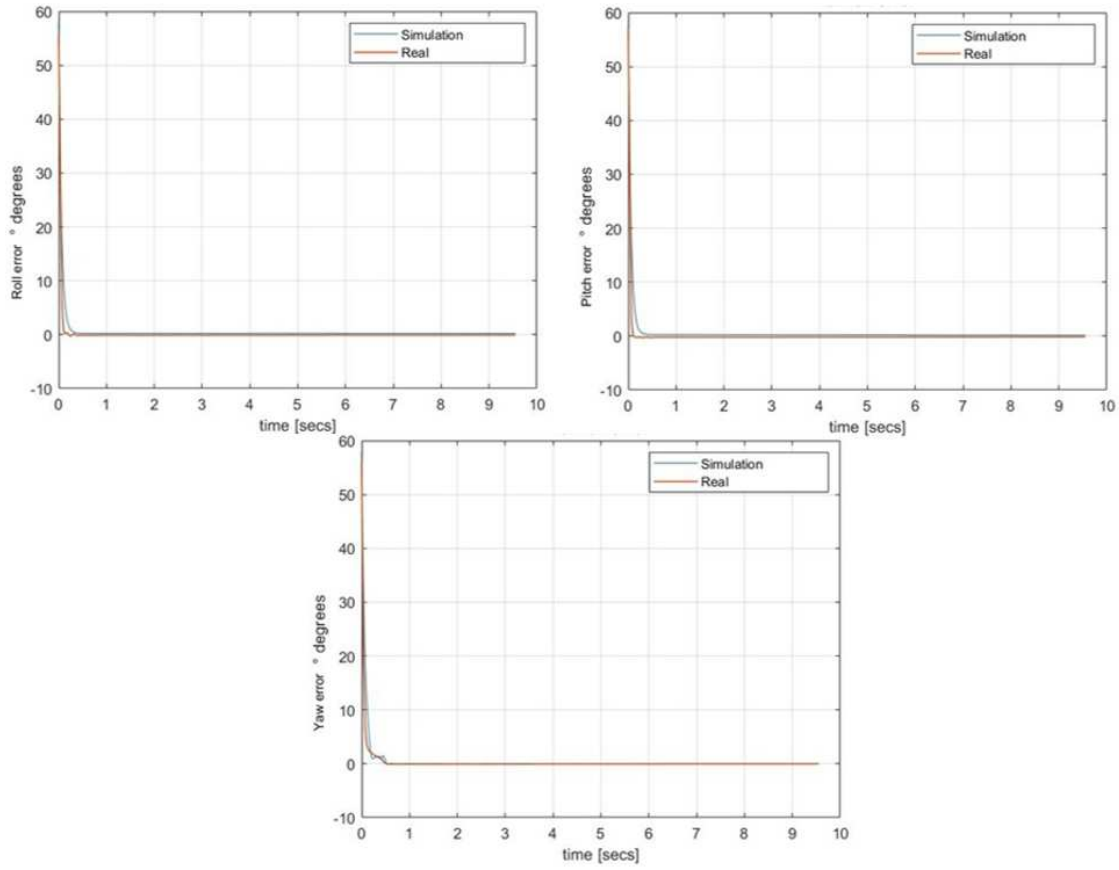


Figure 63 Kalman error in simulation and experimetation

The graphs are practically coincident, measuring a convergence speed of about 110°/s. Is worth noticing that the parameters chosen was obtained by a trial-and-error approach trying to go as close as possible to the instability region of the EKF. Speed up the convergence means increasing the initial covariance P_0 and the gyroscope noise covariance matrix Q or, equivalently, reducing the accelerometer and magnetometer covariance matrix R. Follows that doing some of such operations leads to an unstable EKF, for the sampling time chosen. Was also analyzed, in a time window of 2000 seconds, the comparison between the EKF estimate of the three angles and the Dead Reckoning estimate of the same, namely the simple discrete integration of the gyroscope crude measurements (after being rotated though the T_{YPR} matrix).

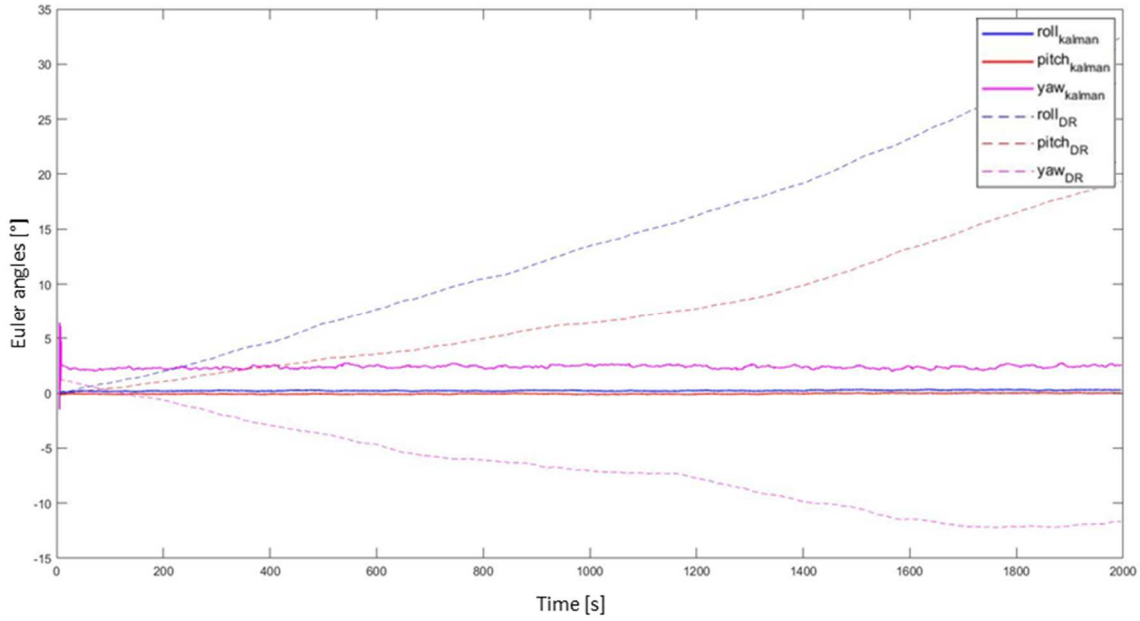


Figure 64 Experimental Dead Reckoning vs EKF

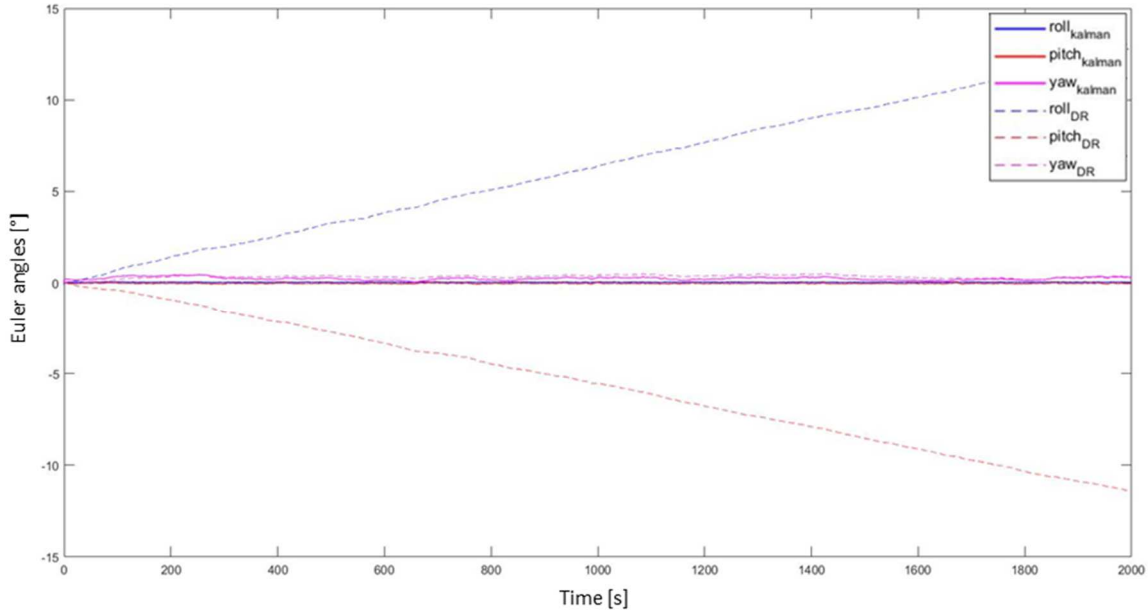


Figure 65 Simulative Dead Reckoning vs EKF

The initial condition of the two EKF are now elaborated outside the loop in the setup phase. Collecting the average of the firsts 1000 measurements (gyroscope, accelerometer and magnetometer) and inverting the trigonometric relationships between the 9 measurements and the RPY angles an initial condition estimate is performed. Both for simulation and experimentation the algorithm shows a good rejection of the *Gyroscope drift* (dashed lines), but some observations must be done:

- The drift of the simulation expresses the propagation of the integration of the previously computed *constant* bias through the transformation from body twist to *rpy* angular rates, and in the random walk resulted by the integration of a white noise.
- The drift of the experimentation results by the integration of the real noise and its bias, that, respectively, are non-Gaussian and non-constant and maybe cross-correlated.

Considering these two observations is clear that in the real Dead-Reckoning the drifts are non-linear because of the non-linearity of the real biases. Can be seen the roll angle drifting more rapidly than in the simulation, the pitch angle in the same magnitude but with an opposite sign, while the yaw angle gain about -10° in the real test bench when is quite constant in the simulation. Although these evident differences the overall behavior of the two EKF is quite similar, and, mostly, the principal EKF aim for data fusion in navigation is the estimate and compensation of the gyroscope bias, and the experimentation shows this robustness also against real and time variant biases. This makes the simulation model enough accurate to be applied in the control loop of the UAV. Since the EKF convergence is not theoretically guaranteed, due to the non-linearities of the sensor models, it was tested also directly to track the states under Path-Planning-Transverse reference imposed to evaluate the performances in about one minute time window in a rapidly angles variation situation. The aim of this simulation is to show the reliability of the model developed when is inserted in the loop control under the already presented path planning reference. The following simulation was ran using the optimal parameters found in the next session, namely a sampling frequency of 100 Hz and an R scale factor equal to 3. As discussed before also the EKF parameters was relaxed when it is inserted in the control loop simply reducing the third component (related to the yaw rate noise variance) of the zero-covariance matrix and setting the angles initial conditions to zero.

$$P_0 = diag(1,1,1)10^{-5} \qquad (3.116)$$

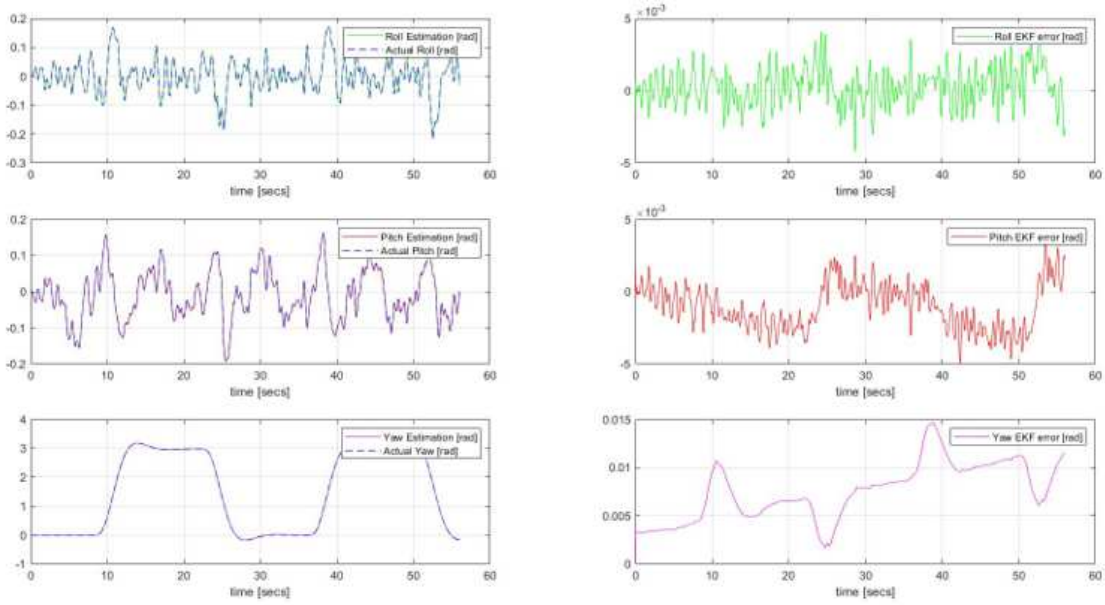


Figure 66 Extended Kalman filter attitude estimation and errors

The path planning time window is about of 1 minute. In the graphs can be seen a good superposition of the estimated states and actual states. The error for pitch and roll converges rapidly to zero or oscillates around it, while for the yaw a small drift of about 0,5° is present because its convergence was set to be slower both in term of R covariance matrix and P_0 since the magnetometer high noise caused the filter to be unstable if too much reliability was given to it. Pitch and roll errors are attributed to the approximation of discrete time process, noises of the sensors and, mainly, translational accelerations. For sake of simulation was also inserted a position error source in the form of white gaussian noise with a standard deviation in the order of some centimeters. Was supposed to use a Real Time Kinematic (RTK) system in combination with GPS to estimate the cartesian position of the UAV in the space. RTK is a novel technology aided to the common GPS system, it computes the relative distance between a body and a fixed station whom coordinates are known. The RTK-GPS system improves the common GPS accuracy

by meters to centimeters level [32], but some problems related to temporary such as signal absence and ionosphere interference remain, so a data fusion with the Inertial Navigation System is required. In the following session is presented an integration of the EKF in the control loop studied above, are tested the performances of the whole system for certain parameters setting and then the same performances behaviors are related to the variation of the main control settings, namely sampling time, LQR matrices and sensor noise.

3.4 Simulation Results

3.4.1 Exploitation of the UAS model in scenario analysis

Nonlinearities in the whole model developed make difficult to extract analytically the quantity of interest such as charge consumption, tracking errors, estimation errors and safety regions to ensure UAV stability as function of the principal control parameters. The following study shows relations between such quantities of interest and sampling time. Theoretically the sampling frequency should be chosen at least 2 times the bandwidth of the feedback control loop, in order to well describe the transient phase and avoiding aliasing effect. However, this frequency can be extracted only for the linearized system that does not considers gyroscope effect and rotation matrices and so the coupling between states and between inputs. On the other hand, is not possible to increase too much the sampling frequency because it

- Improves destabilizing effects due to conversion accuracy
- Increases the negative quantization effects
- Increases the cost of A/D D/A devices, more than linearly

For those reasons a tradeoff is convenient. The most significant frequency of the ideal control loop in continuous domain developed in (3.2) is about 500 Hz and starting from it was chosen a range of sampling time spacing between $0{,}002\ s \leq T_s \leq 0{,}1\ s$. Performing a sweep, on the interval, 300 simulations data on the simulative environment developed until now was collected and elaborated. All the quantities plotted are MAE (mean absolute error), averages or maximum values *across the timeseries*.

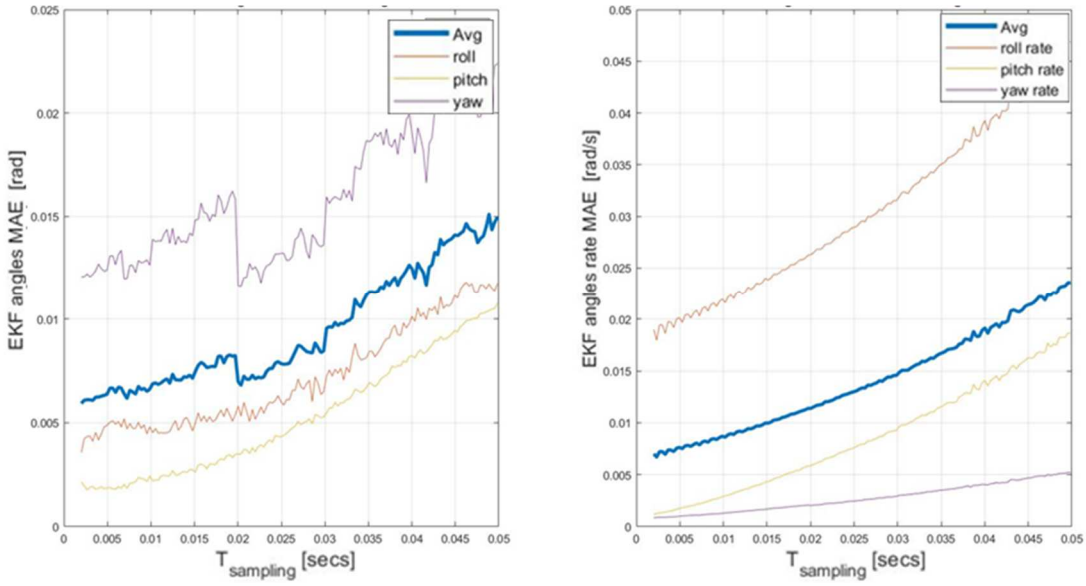


Figure 67 Estimated angles errors (Left) and estimated angles rates errors in function of the sampling time

As predictable, the estimation error increases with the sampling time. However, is noticeable that grows more than linearly, and a rapidly decreasing yaw error in $T_s = 0{,}02\ s$. Exceeding $\boldsymbol{T_s = 0{,}048\ s}$ the system became unstable, so this last T_s value can be used also to give an idea of what combination of states make the UAV unstable. In this case a plot of the *maximum* roll, pitch, roll rate and pitch rate reached is presented.

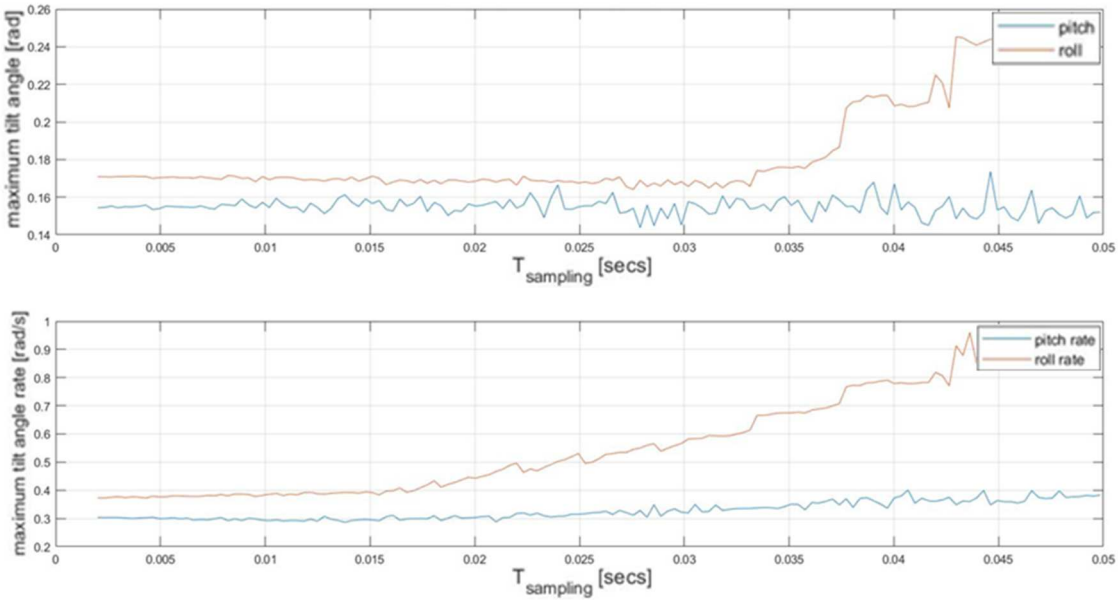


Figure 68 Stability evaluation based on tilting angles (Top) and tilting angle rates (bottom) in function of the sampling time

Given a certain reference velocity imposed to the ESC of the motors, the more time is spent in transient phase the less the efficiency is, since it reaches its theoretical maximum in steady state. When the sampling time is increased, the control, in combination with EKF, start to have difficulties to follow fast transients leading the motor a delayed steady state rotor velocity. This could be one of the main reasons for which the charge consumption goes from the 1440 *mAh (6,4 % of the total charge) for Ts=2 ms to about 1550 mAh for the maximum stable Ts of 48 ms.*

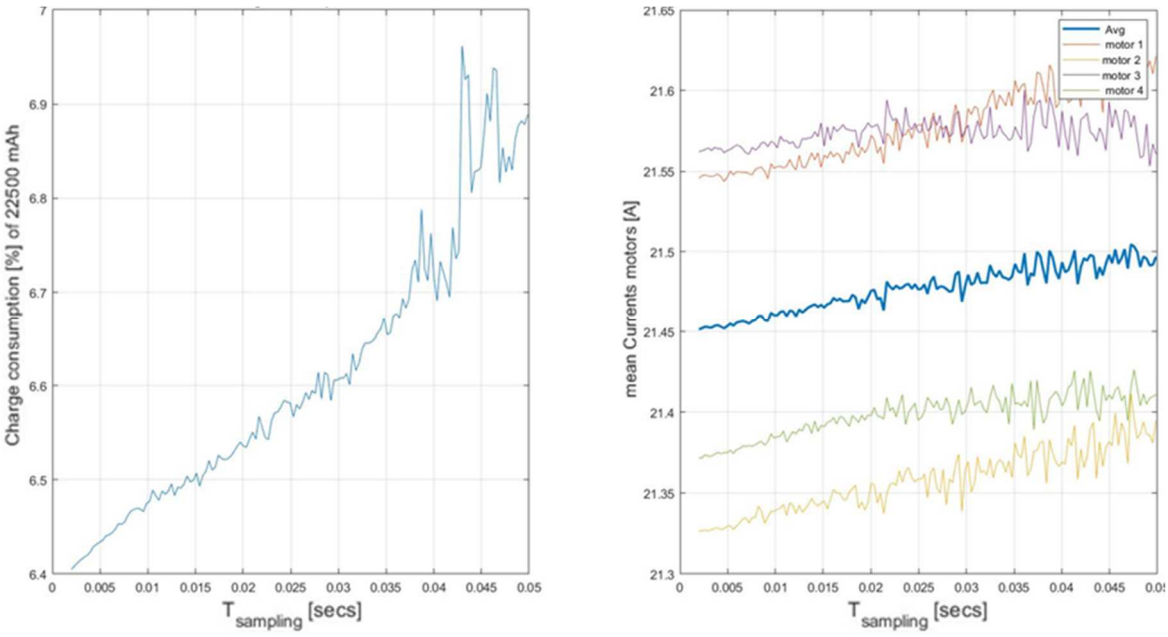


Figure 69 Charge consumption (left) and mean currents (right) in function of the sampling time

For what concerns the tracking errors behavior, there is a growth related to the increasing sampling time, but it is has a small slope compared with the minimum error level (for Ts=2 ms). This may be misleading, because the errors reported in the graph are the averages over time and consider the error due to time delay the dynamic response against an abrupt ramp reference, but the corresponding way point are always reached. Indeed, as can be seen in the next sub-section,

plotting the output states and comparing it with the reference in the 3D space the errors in space are highly less.

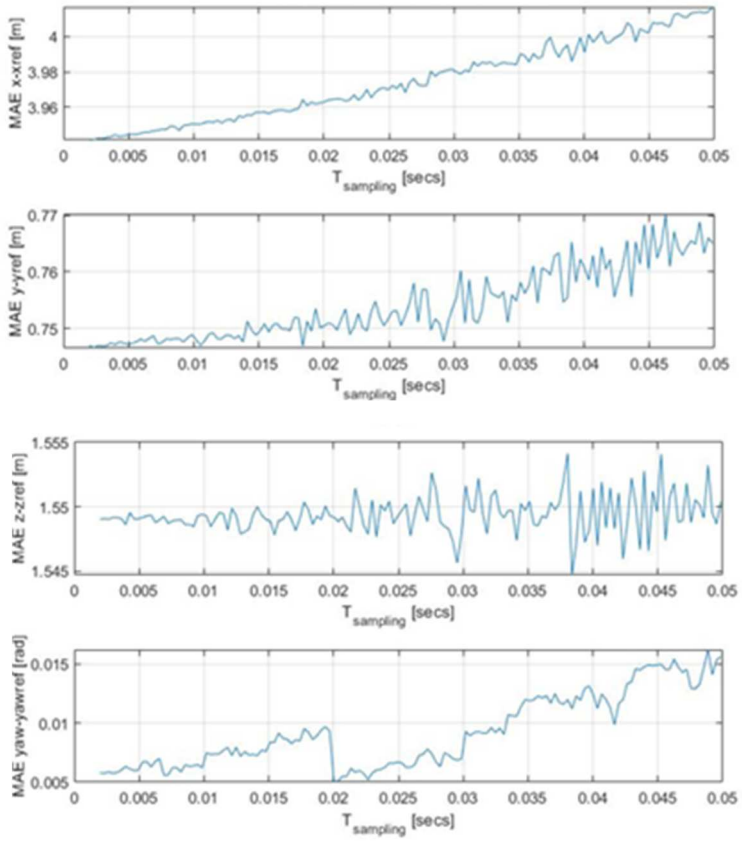


Figure 70 Output errors in function of the sampling time

Unifying the results, the close-optimal sampling time choice was $\boldsymbol{T_s = 0,01\ s}$, since it permits a low/medium cost MCU system simultaneously intercepting minimum charge consumptions, currents, estimation and tracking errors and it keeps angles and angles rate enough distant by the unstable region. Is also important to quantify the response of the system in relation to the aggressivity of the control command inputs. In the presentation of the LQR control, after a tuning, the relative weights in the *Q* matrix were chosen in order to give the right emphasis to the generic controlled state with respect to another. The *R* matrix, that complementary adjust the integral in time of the squared command input was left equals to an identity matrix, multiplied by 110. In the following, leaving invariant *Q,* the same *R* matrix is multiplied by a scalar factor range of 300 points and, again, is computed a sweep of the parameter in the simulation to evaluate the system related behavior. The sampling time is set to 0,01 seconds.

$$\begin{cases} R = Scale_{factor} \cdot \boldsymbol{Identity}_{\mathbf{4\times4}} \cdot 110 \\ 0{,}01 \leq Scale_{factor} \leq 100 \end{cases} \quad (3.117)$$

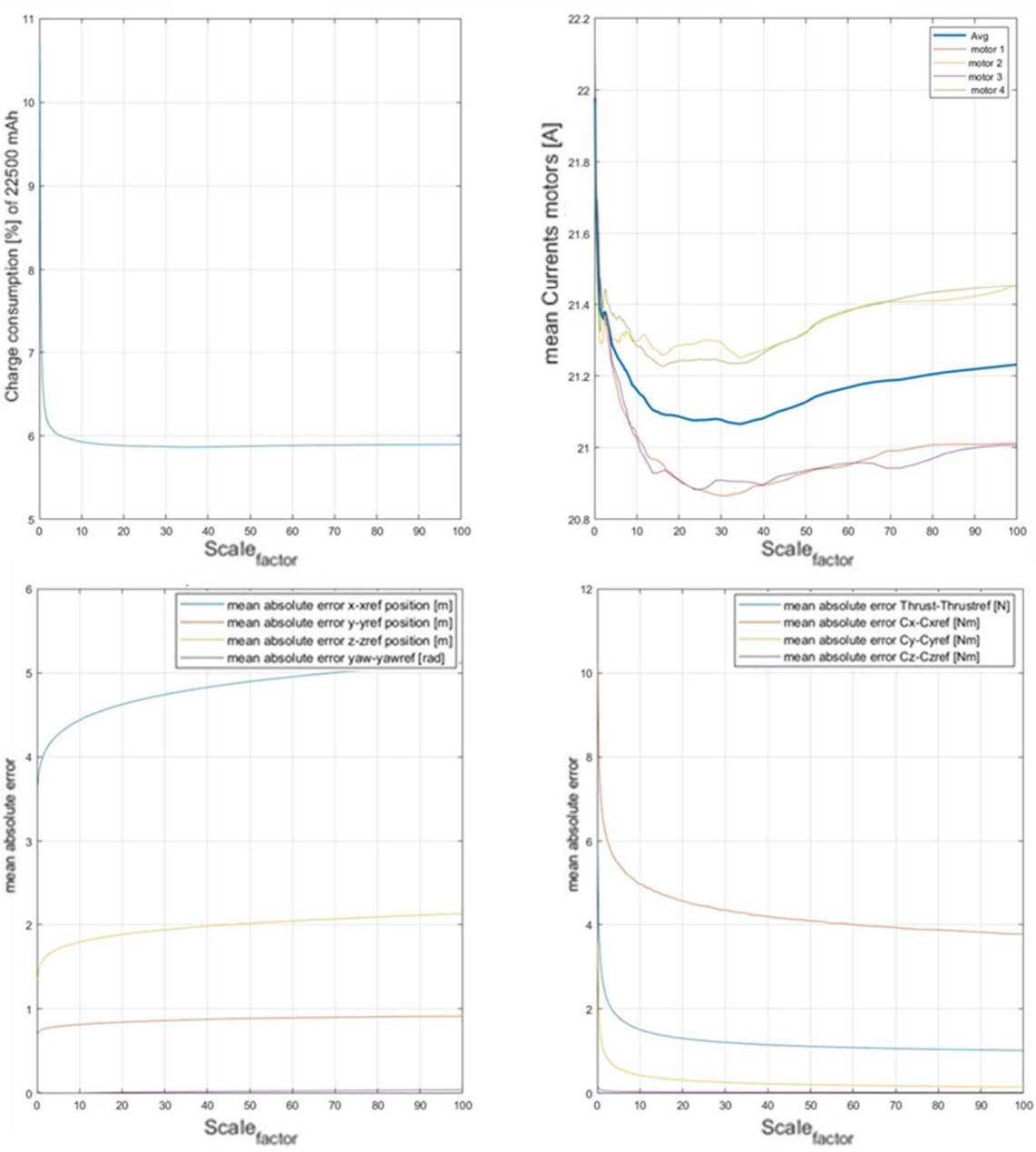


Figure 71 Charge consumption (top-left), mean currents (top-right), output tracking errors (bottom-left) and control input tracking error (bottom-right) in function of the scale factor

It's evident that a decrement of the factor enhances the reference tracking while dramatically increases the energy cost (about 2100 mAh are discharged in one minute) and mean currents flows, as the theory suggests. A less intuitive consequence of the scale factor decrement is the augmenting difference between the instantaneous command input request and the actual, delayed, actuation that, for more aggressive maneuvers, encounters more difficulties in following the control reference of torques and thrust due to motor dynamics characteristic time. Another interesting consequence found analyzing the *R* matrix sweeping was the gyroscopic torque/angles rate characteristic curve. The gyroscopic torque is one of the principals non linearities in the whole system and cause instabilities; indeed, catastrophic failures often appears when there are simultaneous high body frame angular velocities. In the graph are shown the maximum angular rates versus the 3 maximum gyroscopic torques reached for different values of the scale factor. Is interesting that for a large spectrum of the scale factor (about between 1 and 100) angles rates are under 1 rad/s and the gyroscopic torques under 0,5 Nm, while the last simulation before the instability region shows angular rates between 1 and 1,5 rad/s, 1,4 Nm *x-gyroscopic torque* and

1,8 Nm *y-gyroscopic torque*. The gyroscopic torque around the z body frame axis is always zero because in the model were supposed ideally equal inertial momentum components around *x* and *y* axle.

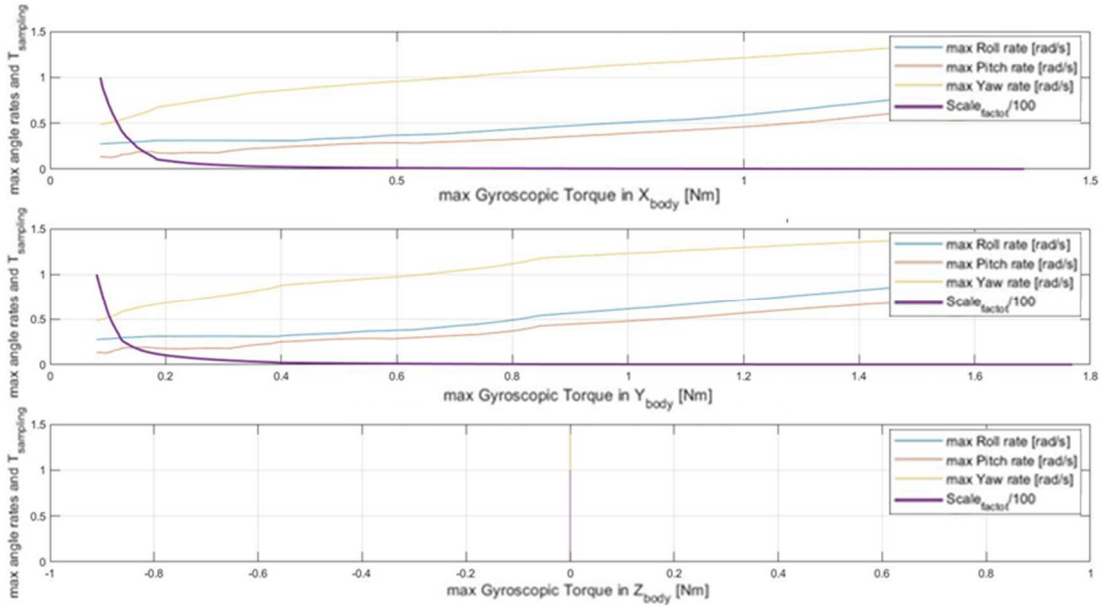


Figure 72 Stability evaluation through gyroscopic X torque (top), Y torque (central) and Z torque (bottom) and angle rates relationships

A heuristic optimal value of the scale factor was $\boldsymbol{Scale_{factor} = 3.}$ The following table resumes the parameters chosen for the control model that will be used from now on:

Table 15 Discrete controller optimal parameters

$T_{sampling}$	Q_{LQR}	R_{LQR}
10 ms	Diagonal (10, 10, 10, 1, 1, 10, 400, 760, 120, 0.1, 0.1, 0.1, 190, 290, 360, 300)	Diagonal (330, 330, 330, 330)

3.4.2 Rows-following with optimal tuning, ideal scenario

Starting from the optimal control parameters chosen, a simulation of all the merged models was performed. The path planning was always the same (path_planning_transverse) chosen for this chapter. The discretization of the control algorithm, the motors models and the online state estimation introduce delays and noises in the control loop.

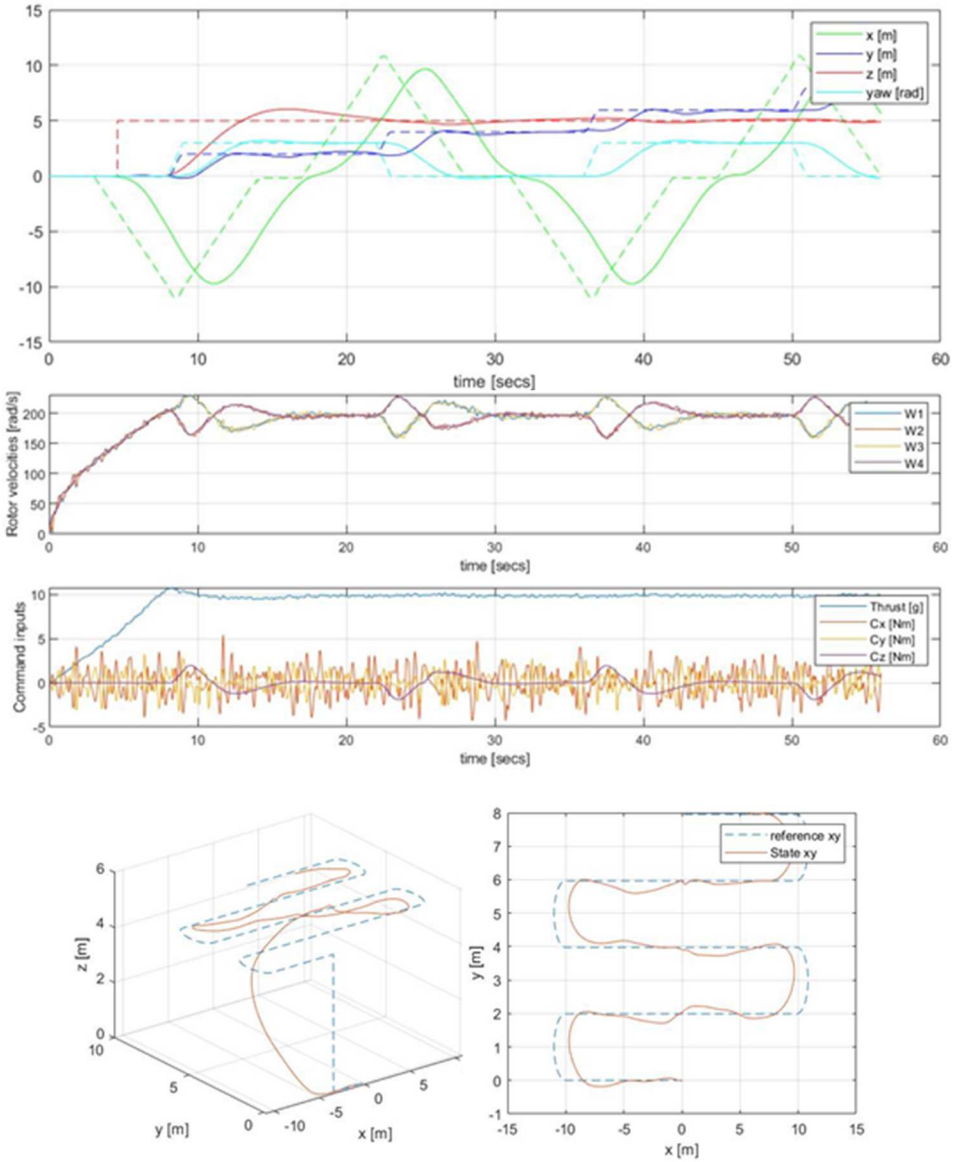


Figure 73 State tracking in time (top), in space (bottom) and commands in time (central)

" g" is the thrust-to-weight ratio

As expected, the steady state tracking error of ramp or parabolic reference input cannot be zero due to the choice of a single integrator in the control. This leads, as seen before, to a delay of 1-2 seconds between the reference input and state behavior. However, delays are not cumulative so in the tridimensional space the UAV follows the reference in a manner precise enough to approach the way points that will be send by the prescription map presented in the next chapters. As can be seen the rotors velocities is well confined in the highest efficiency region specified by the propellers manufacturer explained in 3.1.3. In the table below are specified the main quantities of interest related to this simulation.

Table 16 Simulation, quantities of interest 1

Avg mean RPY angles Kalman errors [rad]	**Avg of the 4 mean currents [A]**	**Charge consumption [mAh]**
0,0024	21,4	1438

Table 17 Simulation, quantities of interest 2

max Gyroscopic Torque X [Nm]	max Gyroscopic Torque Y [Nm]	max Pitch angle [rad]	max Roll angle [rad]
0.564	0.630	0.154	0.172

In this simulation nozzles flux and ground footprint modification due to the UAV velocity, altitude and wind disturbance is not considered, so in the next paragraph is discussed the addition of a wind model, a wind estimation algorithm, and a real-time path planning refinement.

4 Spray System

One of the strategies thoughted to overflight the vineyard rows is the one depicted in figure 74. It was called cross configuration and consists in following the row directly on the top utilizing the nozzles mounted under rotor 1 and 3 to continuously spraying the crops indicated by the prescription map and switch off them when the UAV overflights "Healthy" crops. A more exhaustive explanation of the spraying system will be presented in this chapters.

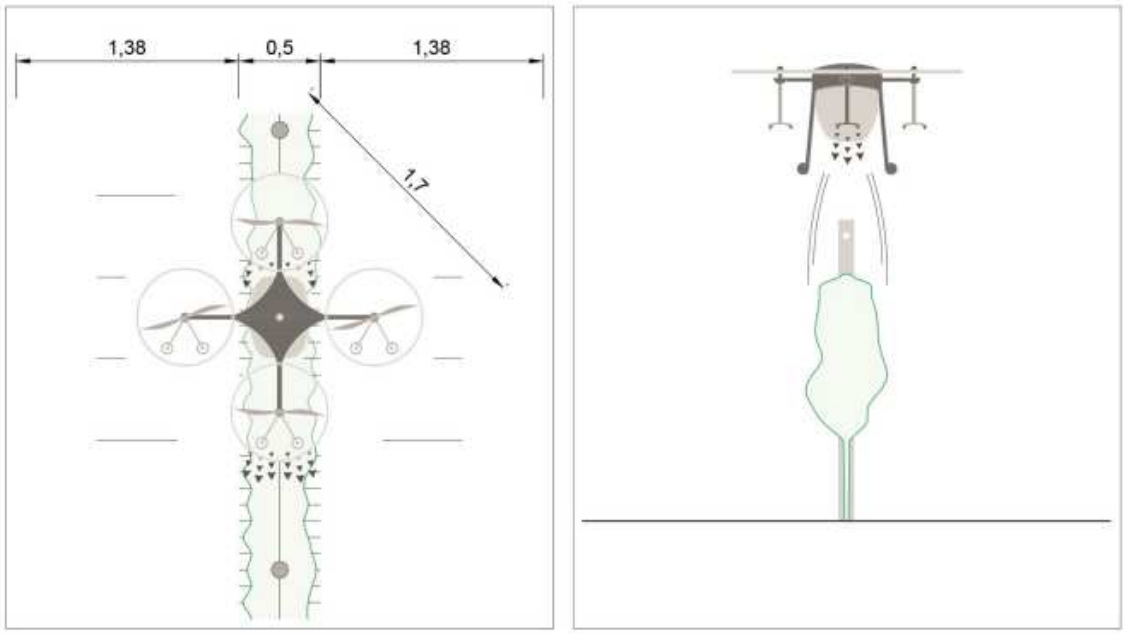


Figure 74 Cross configuration, image from[9]

One challenge associated with the spraying control system is its robustness to guarantee the irroration of the grapevines against wind disturbance and, in general, air-UAV relative velocity (velocity of air seen by the UAV). In cross configuration, in absence of wind, the drone follows the row, and the spray flux axle is deflected in the same plane of the row and UAV velocity, but in opposite direction so that there is only a time delay between the moment in which the vehicle reaches a point and that one in which it is reached by the spray. In presence of wind instead, the plane of deflection becomes another and without some sort of wind estimation is impossible to ensure the correct irroration. In this section we discussed path corrections and spray system mechanisms act to maximize the effective volume delivered to the plants. From now on the path planning used is a modified version of that one used until now, then in chapter 5 a more exhaustive description of the possible spray strategy is present.

4.1 Ceramic hollowcone 80° nozzle model

The study in [9] consist of an experimental validation of CFD analysis thoughted to predict droplets distribution and sizes knowing the 3D profile of the blades, rotors velocity, environmental conditions and impact air speed. The system UAV/spray was tested in a wind tunnel in different conditions of rotors velocity, nozzles upward pressures, nozzle-rotors relative positions and air-nozzles relative velocities. Were utilized 2 types of nozzles. The first one is an anti-derive fan nozzle with 80-110° wingspan, the other a hollow-cone nozzle with 80° wingspan. Nozzles was mounted under six-rotors Matrix 600 drone by DJI.

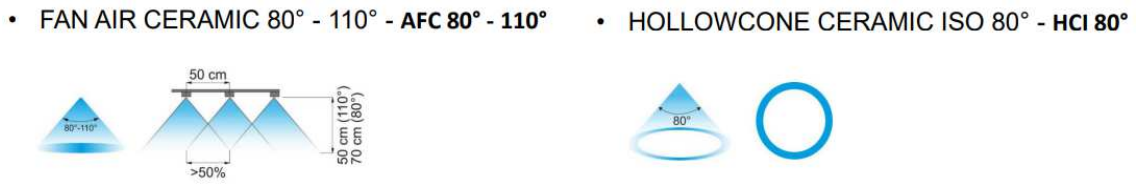


Figure 75 From ARAG datasheet

From now on the nozzle, in the simulation, a model of the hollow-cone will be used. The frontal and lateral profile of the nozzle flux was backlighting and gray scale images (figure 76) for different conditions was collected [9]. The resulting data was used to extract, by image processing, analytic relations between the flux shape and the above-mentioned quantities. The experience was done in SEASTAR (Sustainable Energy Applied Sciences, Technology, and Advanced Research) Wind Tunnel at the Environment Park in Turin, Italy. The drone was positioned in a test section of 6.4x2.4 meters that takes air from the outside of the building through a convergent intake and expels it through ten fans toward a divergent outlet. A high-speed camera (6016x4016 pixels) was used to extract lateral (wind direction from left to right) and frontal (wind direction from behind to forward) images of the nozzle stream illuminated by 532 nm laser (1st methodology) and backlighting by LED (2nd methodology) from a 1.5x1 meters scene.

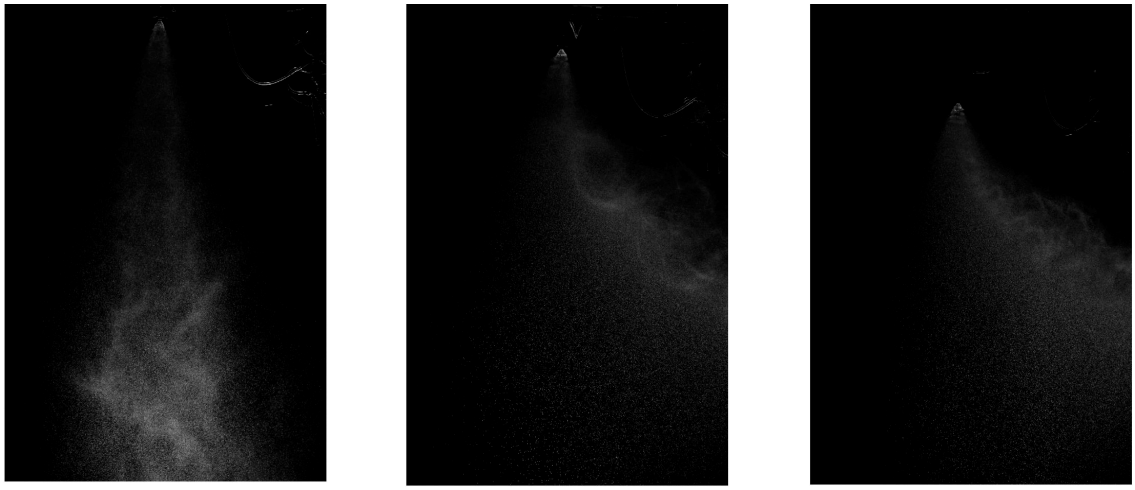


Figure 76 Hollow Cone flux for increasing air relative velocity (Wrotors= 2000 rpm)

Matrix600 was fixed to the structure. The mounted blade was a T-motor 15”x5” of which a 3d scan was generated by an Optical Precision Measuring Machine (OPMM) to be insert in the CFD analysis.

Figure 77 Matrix600 fixed to the structure, image from[9]

The spray system used consists of a DC 12V membrane pump with a maximum flow rate of 6 L/min, TB48S batteries of the drone, an ON/Off switch, a pressure control system composed of a pressure regulator and a pressure gauge aimed at optimizing the pressure values to those optimal specified by the nozzle manufacturer. In the analysis were used four nozzle positioning. The first two was 10 cm and 20 cm far by the blades horizontal plane (under the drone body) in the vertical direction and at a horizontal distance of 0% and 50% of the blade radius by the rotor centers in a direction aligned with the wind velocity. The wind velocities tested was 0, 2 and 3 m/s. No rotors correspond to 0 rpm, Idle to 1000 rpm and Throttle to 5100 rpm. In general, 72 measures were extracted and in the following table only a fraction of those is collected.

TEST I.D.		Wind speed [m/s]			Nozzle position [cm]				Nozzle type		Motor speed		
Name	#	0	2	3	Ant. = Post.				Hollowcone ISO 02	Fan air anti-drift ISO 02	0	Idle	Max throttle
					Radial pos.		Vertical pos.						
					0	50	10	20					
HC_v0_2	01			X	X		X		X		X		
HC_v0_2	02			X	X		X		X			X	
HC_v0_2	03			X	X		X		X				X
HC_v0_1	04			X		X	X		X		X		
HC_v0_1	05			X		X	X		X			X	
HC_v0_1	06			X		X	X		X				X
HC_v0_4	07			X	X			X	X		X		
HC_v0_4	08			X	X			X	X			X	
HC_v0_4	09			X	X			X	X				X
HC_v0_3	10			X		X		X	X		X		
HC_v0_3	11			X		X		X	X			X	
HC_v0_3	12			X		X		X	X				X
FAN_v0_2	13			X	X		X			X	X		
FAN_v0_2	14			X	X		X			X		X	
FAN_v0_2	15			X	X		X			X			X
FAN_v0_1	16			X		X	X			X	X		
FAN_v0_1	17			X		X	X			X		X	
FAN_v0_1	18			X		X	X			X			X
FAN_v0_4	19			X	X			X		X	X		
FAN_v0_4	20			X	X			X		X		X	
FAN_v0_4	21			X	X			X		X			X
FAN_v0_3	22			X		X		X		X	X		
FAN_v0_3	23			X		X		X		X		X	
FAN_v0_3	24			X		X		X		X			

Figure 78 measurements combinations of the wind tunnel test, [9]

From the study emerged that the position 1 (50% radius radial position and 10 cm vertical) was the best choice to minimize the drift, then only measurement in this position will be evaluated.

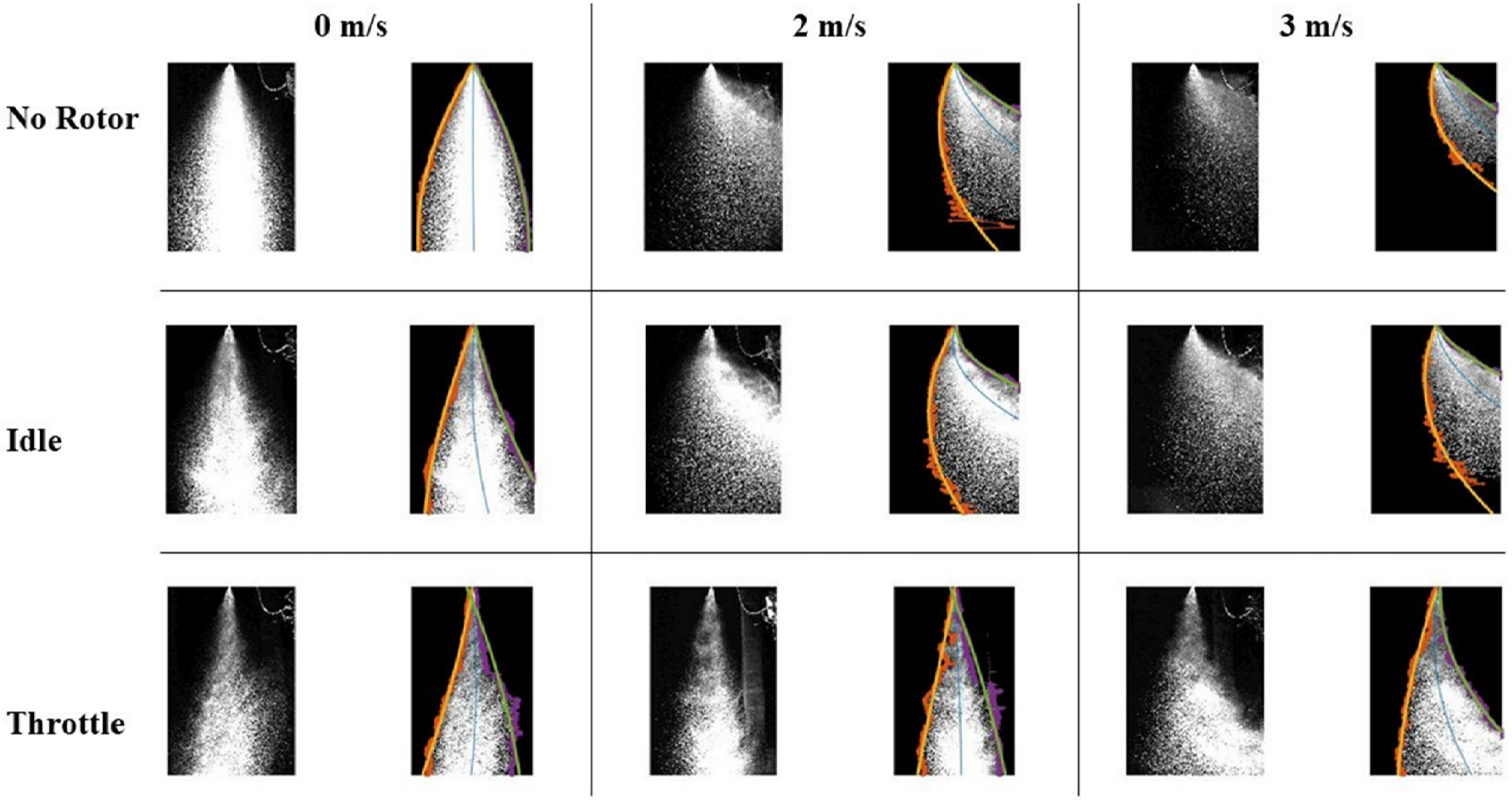

0 m/s
2 m/s
3 m/s
No Rotor
Idle
Throttle

Figure 79 Lateral view, Nozzle in position 1 subjected to 3 different air velocities and 3 different rotor velocities (p=2 bar)

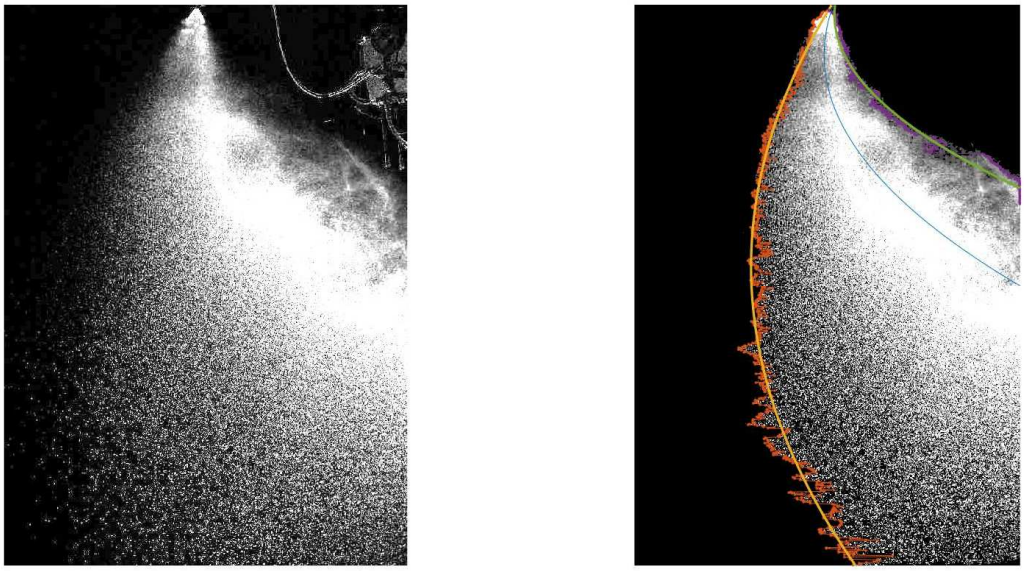


Figure 80 Idle, v=2 m/s

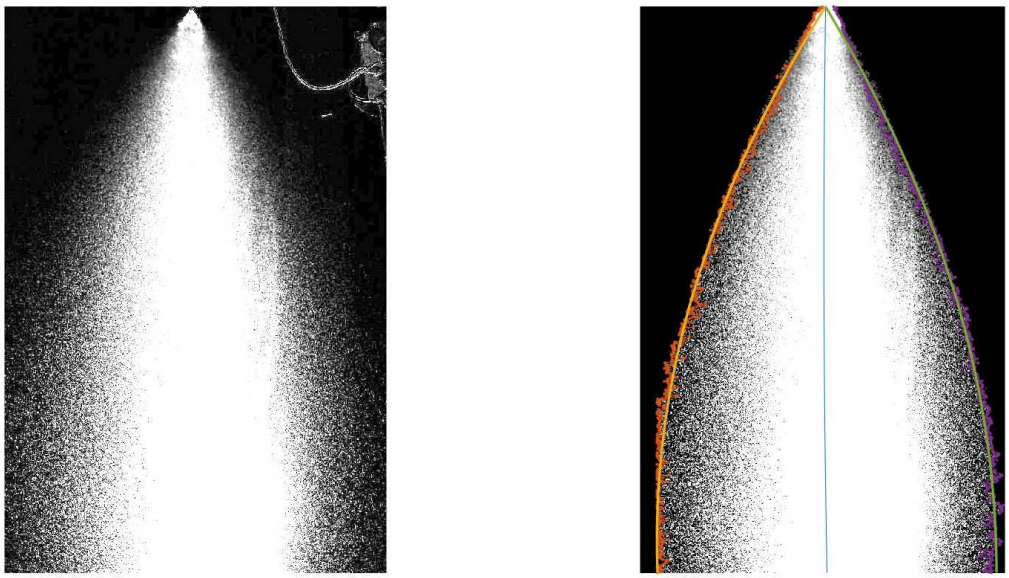


Figure 81 No Rotor, v=0 m/s

Concerning the lateral images (the most important from a wind-caused drift point of view) were collected the 9 cases, reported in the figure 79. Images was 3 channels codified in 8 bit, so 256 possible values for each pixel per-channel. Each RGB image matrix was subjected to sequential filtering intended to ponder outliers and integrated droplets/pixels and exclude intrusive objects or disturbing illuminations. First, after having increased uniformly the image lighting by factor 10, all the pixels smaller than 70 was put to zero and was inserted a triangular mask to exclude the intruder object (top-right). To reduce the number of outliers pixels a 5x60 (h x w) grid was left analyze each image pixel, computing the mean of all the elements belonging to the grid and comparing the value with a threshold. The lower is the mean of the pixel contour, the higher the probability of ling in an insulated area. The threshold was set, after several trial and errors, to 60 and all the elements with a grid mean lower than this threshold were set equal to the minimum value of the entire grid, invariant otherwise. Finally, the edge recognition consisted in picking up, for each horizontal line, the first and the last pixel greater than 100 representing y_{sx} and y_{dx} edge

points of the i[th] row, respectively. Other, smaller, refinement were necessary to ensure the edge lines follow the real flux boundaries (e.g. in No Rotor and Idle with 2 and 3 m/s air velocity the flux shape is cut by the right frame border, so to avoid the curve to fit the artifact vertical line of the border only a limited number of y_{dx} points which position was greater than the 95% of the image width were sent to the regression solver). Doing so, were available 2 edge analytical quadratic curves for each of the 9-case study. Passing by pixels to meters (with the conversion of 940 pixels each 1,5 meters) was then collected 9 couples of the type:

$$\begin{aligned} y_{dx} &= a_{dx}z^2 + b_{dx}z \\ y_{sx} &= a_{sx}z^2 + b_{sx}z \end{aligned} \tag{4.01}$$

Then, each case study flux "k" is represented by the coefficients vector:

$$[a_{dx}\,, b_{dx}\,, a_{sx}\,, b_{sx}\,]_k \tag{4.02}$$

And by the UAV variables:

$$[\|\overrightarrow{v_r}\| \quad W_{rotor}]_k \tag{4.03}$$

The problem is to find a relation $a_{dx}\,, b_{dx}\,, a_{sx}\,, b_{sx}\,(\|\overrightarrow{v_r}\|, W_{rotor})$, hence a simple Ridge regression could be enough. Was supposed that the relation is affine in $\|\overrightarrow{v_r}\|, W_{rotor}$. The problem can be solved simply in matrixial form. **AB** is a 9x4 matrix containing in each row "i" the coefficients of the case study "i", in the specified order. **Vrpm** is a 9x3 matrix which "i" row contains the air velocity and rotor velocity of the case study "i" (the first 3 are 0 rpm and 0/2/3 m/s of air velocity, the second 1000 rpm and 0/2/3 m/s and the third 5100 rpm and 0/2/3 m/s) and 1 to consider the constant term of the relation. **W** is the unknown matrix. It is a 3x4 matrix containing for each column the 3 weight values to be used to extract one of the four edge curve coefficients, knowing $\|\overrightarrow{v_r}\|, W_{rotor}$.

$$\boldsymbol{AB} = \begin{bmatrix} a_{dx,1} & b_{dx,1} & a_{sx,1} & b_{sx,1} \\ a_{dx,2} & b_{dx,2} & a_{sx,2} & b_{sx,2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{dx,9} & b_{dx,9} & a_{sx,9} & b_{sx,9} \end{bmatrix} \tag{4.04}$$

$$\boldsymbol{Vrpm} = \begin{bmatrix} 1 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 1 & 3 & 5100 \end{bmatrix} \tag{4.05}$$

$$\boldsymbol{W} = \begin{bmatrix} w_{0,a_{dx}} & w_{0,b_{dx}} & w_{0,a_{sx}} & w_{0,b_{sx}} \\ w_{V,a_{dx}} & w_{V,b_{dx}} & w_{V,a_{sx}} & w_{V,b_{sx}} \\ w_{rpm,a_{dx}} & w_{rpm,b_{dx}} & w_{rpm,a_{sx}} & w_{rpm,s_{sx}} \end{bmatrix} \tag{4.06}$$

The problem is then formulated as:

$$\boldsymbol{AB} = \boldsymbol{Vrpm} \cdot \boldsymbol{W} \tag{4.07}$$

So, the solution can be found computing the Moore-Penrose pseudo-inverse of $\boldsymbol{Vrpm}$**:**

$$\boldsymbol{W} = \left(\boldsymbol{Vrpm}^{\boldsymbol{T}} \cdot \boldsymbol{Vrpm}\right)^{-1} \cdot \boldsymbol{Vrpm}^{\boldsymbol{T}} \cdot \boldsymbol{AB} \tag{4.08}$$

From which the 2 curves with the complete dependency on nozzle distance, air velocity and rotor velocity were extracted:

$$\boldsymbol{a_{dx}}(W_{rotor}, \|\overrightarrow{v_r}\|) = -W_{rotor} \cdot 1{,}56 \cdot 10^{-4} + \|\overrightarrow{v_r}\| \cdot 3{,}67 \cdot 10^{-1} + 4{,}05 \cdot 10^{-1} \tag{4.09}$$

$$\boldsymbol{b_{dx}}(W_{rotor}, p, \|\overrightarrow{v_r}\|) = -W_{rotor} \cdot 8{,}53 \cdot 10^{-5} + \|\overrightarrow{v_r}\| \cdot 4{,}60 \cdot 10^{-2} + 5{,}54 \cdot 10^{-1} \tag{4.11}$$

$$\boldsymbol{a_{sx}}(W_{rotor}, \|\overrightarrow{v_r}\|) = -W_{rotor} \cdot 10^{-4} + \|\overrightarrow{v_r}\| \cdot 1{,}36 \cdot 10^{-1} + 2{,}93 \cdot 10^{-1} \tag{4.12}$$

$$\boldsymbol{b_{sx}}(W_{rotor}, p, \|\overrightarrow{v_r}\|) = -W_{rotor} \cdot 5{,}19 \cdot 10^{-5} + \|\overrightarrow{v_r}\| \cdot 5 \cdot 10^{-3} - 5{,}25 \cdot 10^{-1} \tag{4.13}$$

$$\begin{Bmatrix} W_{rotor}\,[rpm] : velocity\ of\ the\ rotors\ on\ top\ the\ considered\ nozzle \\ p\,[bar] : internal\ pressure\ of\ the\ nozzle \\ \|\overrightarrow{v_r}\|\ \left[\frac{m}{s}\right] : air - UAV\ relative\ velocity\ module \end{Bmatrix} \quad (4.14)$$

From which derives the flux axis deflection curve, used in next sections:

$$r = \frac{(y_{dx} + y_{sx})}{2} = \frac{\boldsymbol{a_{dx}} + \boldsymbol{a_{sx}}}{2} z^2 + \frac{\boldsymbol{b_{dx}} + \boldsymbol{b_{sx}}}{2} z \quad (4.15)$$

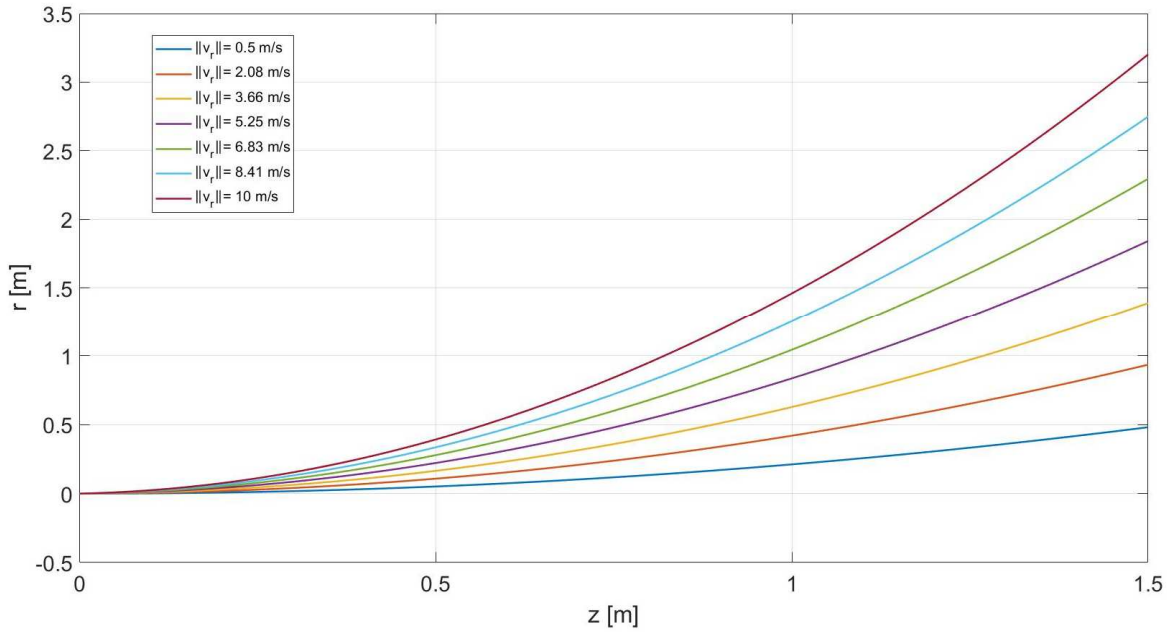


Figure 82 deflection curve for Wrotor=2000 rpm

Analyzing this case can be seen that the wind velocity increases significatively the drift, but simultaneously the downwash effect of the rotors dominates the air velocity reducing the drift and reduces the opening angle of the nozzle flux.

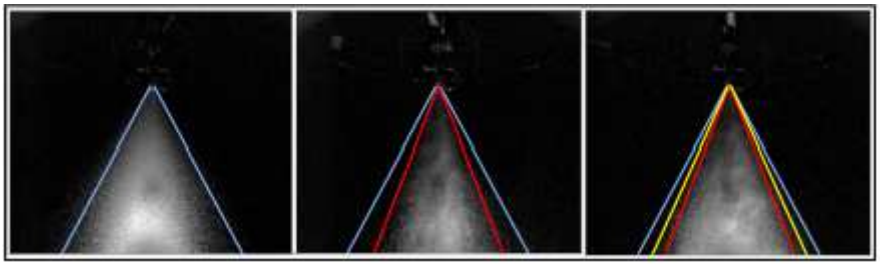


Figure 83 Frontal view of spray angle with no rotor - 2 bar (Blue), full throttle - 2 bar (Red) and full throttle - 3 bar from left to right (Yellow)

The first 4 delimiting curves can be used to extract the two semi-axes of an, supposed, elliptical shape of the flux, subtracting $y_{sx} - y_{dx}$. In particular, the frontal axis was obtained identically as done for the lateral, namely extracting the flux edges and fit them with a variable coefficient quadratic function, although for the frontal axes the coefficients depended solely on pressure and rotor velocity. Frontal axis computation was useful only to get an estimate of the opening angle of the nozzle with respect to the rotor's velocities and pressure.

$$\boldsymbol{saxis_{Frontal}}(W_{rotor}, p, z) = \frac{1}{2}\left(\boldsymbol{z}^2(0.0000052\ p + 0.00000026\ W_{rotor} - 0.000843) + \boldsymbol{z}(1.39\ + 0.0732\ p\ - 0.000153\ W_{rotor})\right) \quad (4.16)$$

Is noticed that, in the acquisition phase, frontal images were considered invariant with respect to wind velocity, while lateral images invariant with respect to nozzle pressure. From the frontal

semiaxis was then extracted the opening angle of the nozzle by trigonometric relationship between the nozzle height "z" and the frontal semiaxis at distance "z" from the nozzle. The opening angle was computed at $z = 10\ cm$.

$$\boldsymbol{Opening}(W_{rotor}, \|\overrightarrow{v_r}\|, z = 0{,}1) = \operatorname{atan}\left(\boldsymbol{saxis_{Frontal}}(W_{rotor}, \|\overrightarrow{v_r}\|, z = 0{,}1)/_{0{,}1}\right) \quad (4.17)$$

Observing the hovering rotors velocity of about 200 rad/s the opening angle can be considered oscillating in the range 30-35°, depending on the applied pressure.

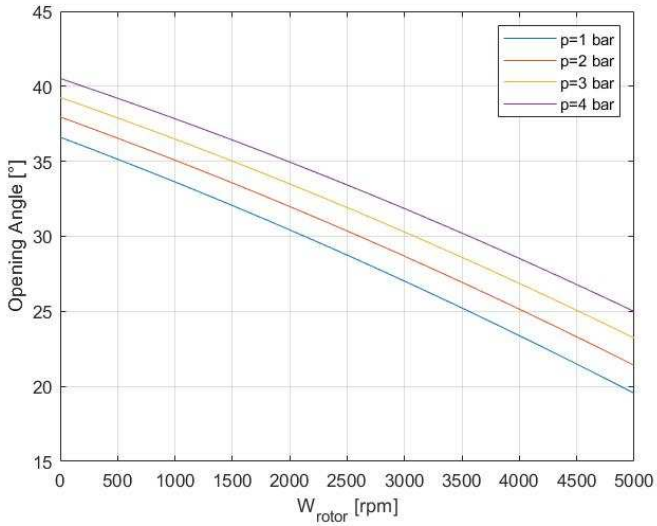


Figure 84 Frontal nozzle opening angle

The blade supposed to be used in this thesis have bigger length and width than those used in the experimentation, therefore the effects of the rotor downwash described by the above relations should be considered underestimated. To evaluate the PPP distribution in liters and validate all the model and algorithms performed until now a 3D model of the flow rate distribution of the hollow-cone nozzle was studied. To not complicated too much the formulation, the distribution model was thought to fit a scenario in which air velocity due to wind, nozzle-ground relative velocity and rotor velocities are null, pressure and flow rate have reached a stable value and therefore the atomization process and the downward averaged droplets trajectories are stable. In this condition is reasonable consider the geometry of the hollow cone as a conic wall with constant and a center-symmetric geometries. The cone produced by the nozzle has a lateral wall thickness in the order of centimeters in proximity of the nozzle outlet, and it is supposed to gradually increase the thickness moving away from the nozzle [33]. The thickness is, here, considered as the thickness of the circular crown containing all the droplets at a given nozzle distance. Air viscosity and gravity are neglected so the direction of the generic droplet remains the same, once exits the nozzle. Following these assumptions, for a given distance from the nozzle, the droplets (and so the flow rate) could be imagined as deposited in great quantities along the circumference with radius equal to the mean of the maximum and minimum radius of the crown, and with lower quantities for higher and lower radius.

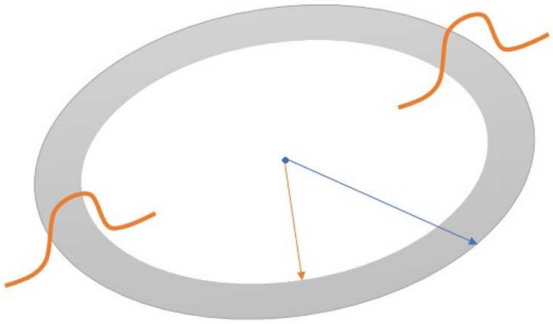


Figure 85 Schematic representation of the droplet spatial distribution 2D cross section

This behavior is also presented in the datasheet of the used hollow cone nozzle as reported in figure 86.

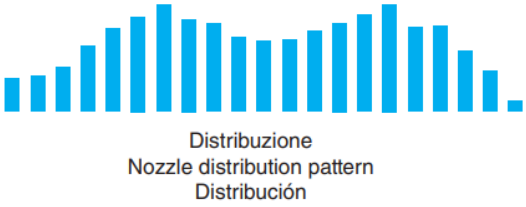


Figure 86 From ARAG website

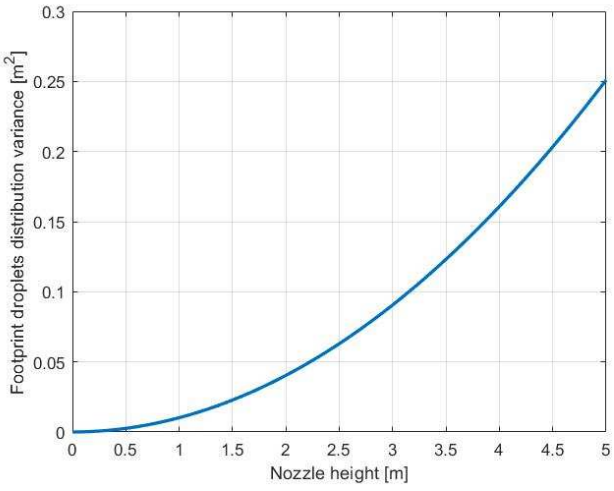
This behavior was modeled as a revolute gaussian with a variance quadratically increasing with the distance from the nozzle and a mean equal to the mean radius of the crown, i.e.:

$$\sigma^2 = (\sigma_0 + \sigma_v \cdot z)^2 \tag{4.18}$$

$$\begin{cases} \sigma_0 = 0{,}001 \\ \sigma_v = 0{,}1 \end{cases} \tag{4.19}$$

$$\mu = z \cdot \tan(Opening) \tag{4.20}$$

σ_0 and σ_v was chosen to have a wall thickness of about 3 cm close to the nozzle and 50 cm at 5 meters.



The radial gaussian was then formulated:

$$N(\mu,\sigma^2)=\left.\frac{1}{\sqrt{2\pi}\sigma}\cdot e^{-\frac{1}{2}\frac{(r-\mu)^2}{\sigma^2}}\right/_{2\pi} \quad (4.21)$$

The expression is normalized by 2π to have:

$$\int_{-\infty}^{+\infty}\int_{0}^{2\pi} N(\mu,\sigma^2)\cdot dr\cdot d\theta = 1 \quad (4.22)$$

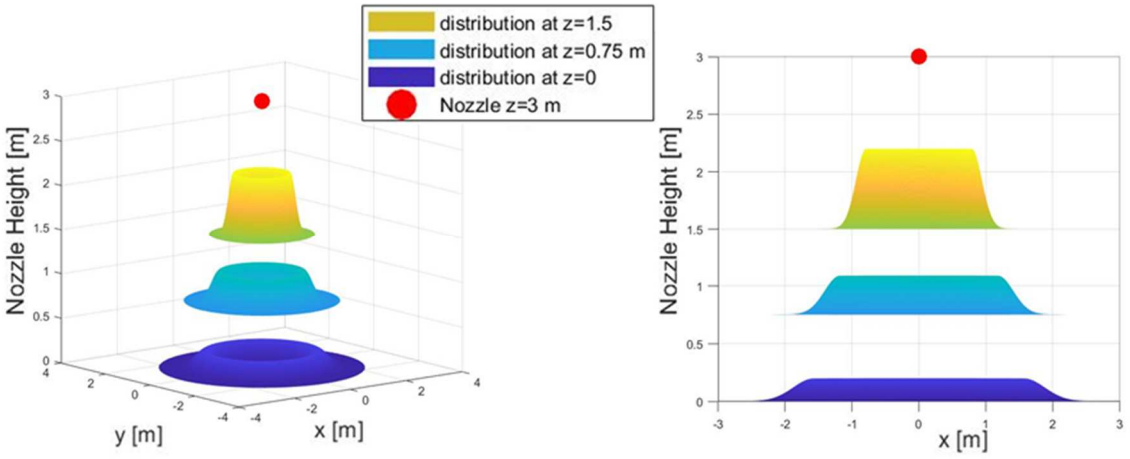


Figure 87 Flow Rate distribution in function of nozzle distance, Wrotors=2000 rpm, p=2 bar (PDF values are qualitatively represented)

From the code point of view the model is inserted as an appendix to the simulation environment to compute the actual amount of PPP released by the quadrotor when overflies a region. A MATLAB function block accepts as input the footprint center position at time step k $(x,y)_{swat,k}$, in general different by the UAV current position, and uses it as the center of the footprint circular crown. Also knowing the real time nozzle height, left equal to the drone height, z_k it computes the gaussian expected value μ_k and the variance ${\sigma^2}_k$. Passing by polar to cartesian coordinates the bi-dimensional Probability Density Function (PDF) is discretely integrated into the plane grid and multiplied by the instantaneous flow rate and the simulation sampling time to obtain finally the released liters for each grid element. The process is repeated for the whole simulation time resulting in a complete map of the PPP sprayed in the virtual vineyard.

- The simulation floor is coded as a n-dimensional 2D grid $(X,Y)_{nxn}$.
- The time series of interest enters the code; they have sampling time T_s^{sim} with length N_{time}:

$$\begin{cases}\overline{(x,y)_{swat}}\\ \overline{FR}\\ \bar{z}\\ \overline{W_{rotors}}\\ \overline{Pressure}\end{cases} \quad (4.23)$$

- Variance and mean values are computed each time-step as:

$$\begin{cases}{\sigma^2}_k=(\sigma_0+\sigma_v\cdot z)^2\\ \mu_k=z_k\cdot\tan\left(Opening(W_{rotors_k},Pressure_k)\right)\end{cases} \quad (4.24)$$

- For each k, each point in the grid is associated with a radius, expressing the distance from $(x,y)_{swat_k}$:

$$r_k^{i,j}=\sqrt{(x_i-x_{swat,k})^2+(y_j-y_{swat,k})^2} \quad (4.25)$$

- The radius integration steps is defined as the length of the diagonal of a single element in the grid, while the angle integration as a fraction of 360° that increases if the $r_k^{i,j}$ decreases, in order to take into account the discrete domain of the angular integration:

$$\begin{cases} dr = \sqrt{\left(\frac{X_{max} - X_{min}}{n}\right)^2 + \left(\frac{Y_{max} - Y_{min}}{n}\right)^2} \\ d\theta_k^{i,j} = \frac{\pi/2}{\left(1 + \frac{r_k^{i,j}}{dr}\sqrt{2}\right)} \end{cases} \quad (4.26)$$

- Finally, the PPP deliver map is computed as:

$$Liters_{i,j} = \cdot \sum_{k=1}^{N_{time}} \left(\frac{1}{2\pi\sqrt{2\pi\sigma^2{}_k}} \cdot \left(e^{-\frac{1}{2}\frac{\left(r_k^{i,j} - \mu_k\right)^2}{\sigma^2{}_k}} + e^{-\frac{1}{2}\frac{\left(r_k^{i,j} + \mu_k\right)^2}{\sigma^2{}_k}} \right) \cdot dr \cdot d\theta_k^{i,j} \right) \cdot FR_k \cdot T_s^{sim} \quad (4.27)$$

where the second exponential is inserted to sum up the tail of the diametrically opposite gaussian. To consider the height of the grapevine (1,5 meters) the height of the nozzle with respect to the target was considered as the UAV height subtract by the grapevine height. This assumption will be further developed later. An example of the PPP distribution is in figure 88. Here, the simple path planning tracking without wind disturbances is considered. The flow rate is 0.8 L/min when the drone is on the rows and 0 L/min during the inversion; this last feature was added manually, but in the next paragraphs will be presented an automized version of it. The altitude of the drone is kept to 2 meters.

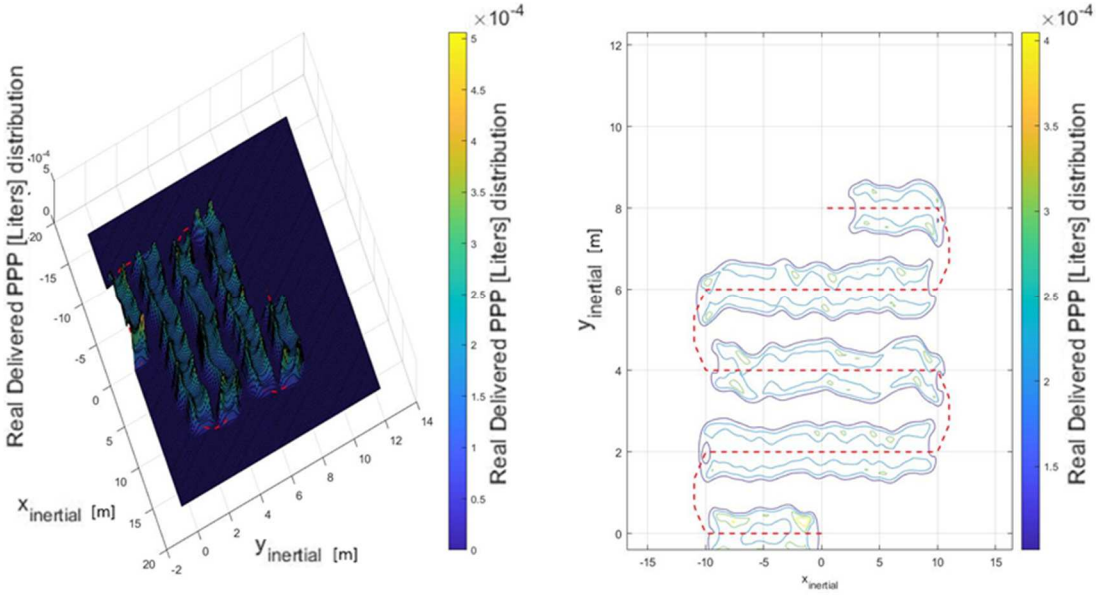


Figure 88 PPP volume distribution, grid thick of 5 cm

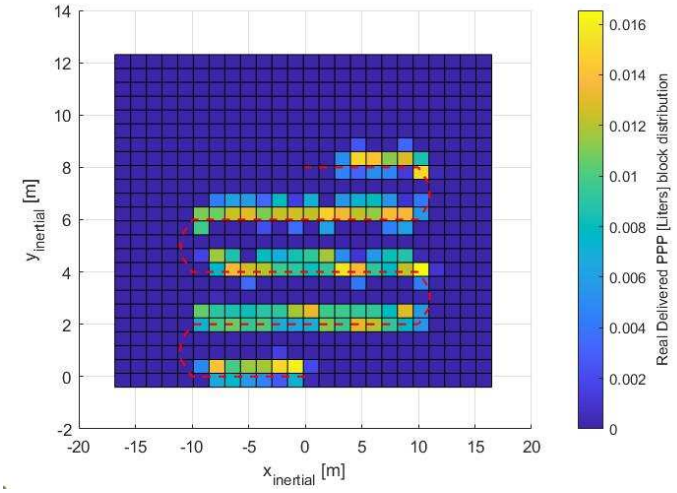


Figure 89 PPP volume distribution, grid thick of 69 cm

In a squared control surface with base of about 140 centimeters are mediumly released 0,045 Liters. This is coherent with the formulation of the volume density provided by the nozzle manufacturer, that gives about 0,048 Liters in the same control surface inserting our operation parameters. The computation was done considering one single nozzle (*n=1*), a flow rate of 0,8 L/min, an inter-row distance of 2 meters and a velocity of 3,6 km/h. The result is a per-hectare released product of 67 Liters. Considering the surface of the simulation path containing 4 rows in between $1\ m \leq y \leq 9\ m$ and $-10\ m \leq x \leq 10\ m$ (8x20=160 m^2), in order to take into account the half inter-row distance as offset, and knowing the total released PPP volume of the simulation of 1,06 L the proportion leads to a per-hectare released product of $\frac{10000\ m^2}{160\ m^2} \cdot 1{,}06\ L = 66{,}25\ L$, which substantially matches the manufacturer formulation.

Figure 90 ARAG HOLLOWCONE CERAMIC ISO 80° - HCI 80° datasheet

Since the nozzle is studied for in-motion spraying the empty surface of the hollow cone surface is filled with the shift of the footprint released each time step. However, trusting on the model, the central zone of the shifted footprint is depressed relatively to the lateral distribution that reaches higher densities. Considering the opening angle of the hollow-cone nozzle the UAV flight altitude was seen optimal at 50 centimeters distant from the plant crown. The more this distance increases the more PPP waste due to drift and opening angle increases, while the more the distance decreases the more increases the risk of quadrotor impact with the plants. Further studies could integrate the study in [34] with the autonomous guidance in order to locally extrapolate plants geometry data in-flight allowing to fly at optimal distance with respect to their crown. In the study [34] the average crop surface was estimated to be 0,2 m^2 so with a square side of about 45 cm. Since in our study all the PPP sprayed outside the plant region will be considered wasted, we have increased the square side to 90 cm to include a wider area in which the product could realistically enters in contact with the leaves. To extract a first approximated performance index of the delivered PPP, along the reference path, in correspondence of the rows only, was inserted an "active grid" with thickness of about 2 times the supposed plant radius, of 45 cm. This grid

represents the plant projected surface in the horizontal plane positioned at 1,5 meters height. Liters will be considered delivered to the plant if lie in this area, wasted otherwise.

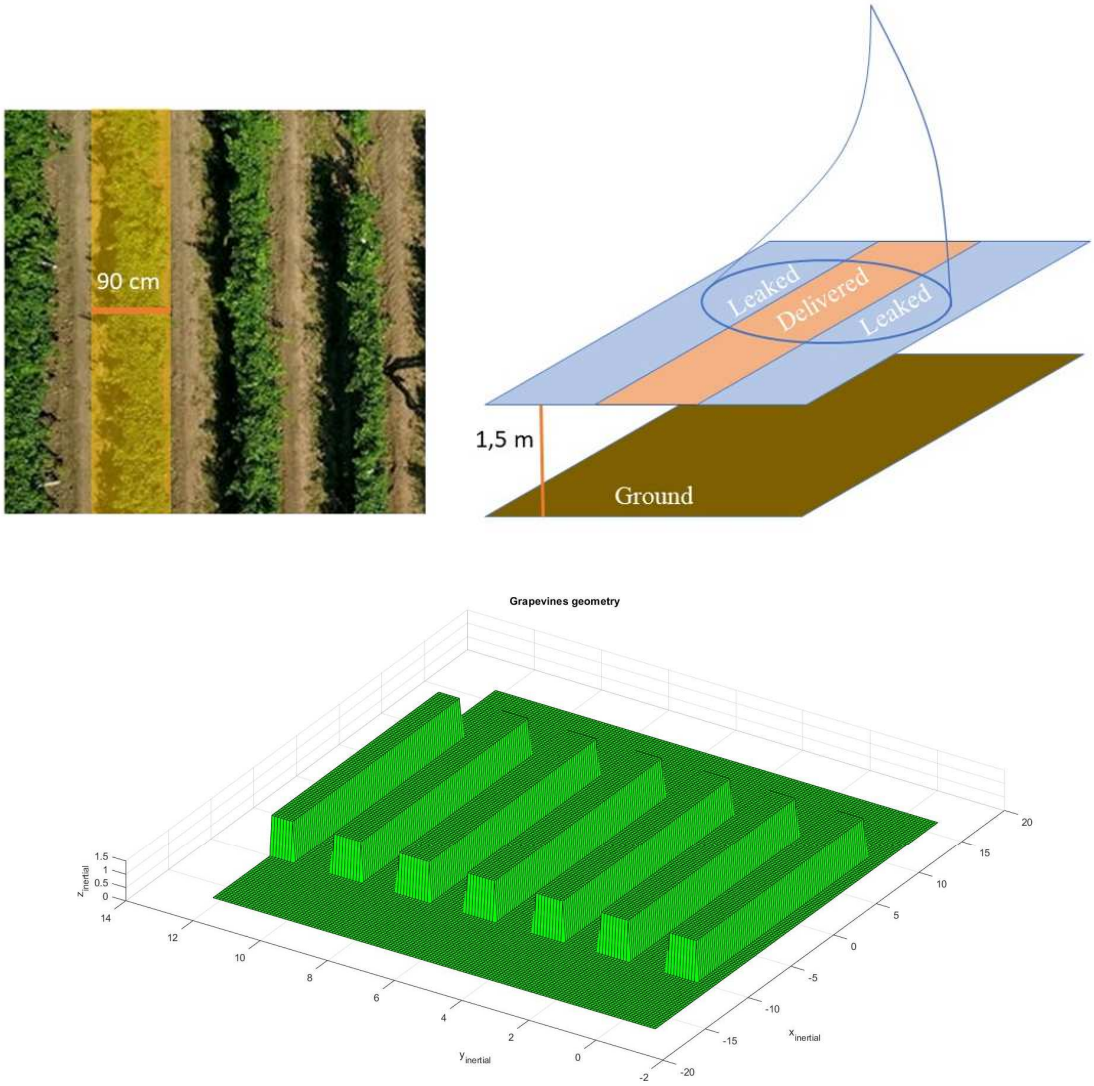


Figure 91 Grapevine, active surface

The Delivered/Sprayed ratio expresses how much the delivering PPP process is efficient, because expresses how much of the total sprayed product contacts the plant. The Delivered/Requested Volume ratio expresses how much of the total volume required in the operating area, namely the sum of all the prescription map elements, has been released on the plants, so expresses how much the spray system is sensible to the presence of target plants. Both ratios can be expressed as global ratios (i.e. the three volumes comprise the sum of the single plants volumes) or as per-plant ratios. Sprayed and Requested volumes can be different:

- If is introduced a safety factor that increases the commanded Requested volume release with respect to the original requested volume (conservative approach).
- If there is not a closed loop control of the sprayed volume (per-plant or global) such as in intensive continuous spray.
- If the spray circuit presents leakages (also with a closed loop control of the sprayed volume)

Delivered/Sprayed (D/S) ratio expresses how much of the total sprayed product contacts the plant.

Delivered/Requested (D/R) ratio expresses how much of the total volume required in the operating area, has been released on the plants.

If commanded volume release **equals Requested Volume** (not conservative):

- (D/R) = (D/S) in any PtP strategy.
- (D/R) = (D/S) in Continuous strategy if proximity check is deactivated. (idealmente, ma In realtà è >= considerando lo 0,015)
- (D/R) < (D/S) in Continuous strategy with proximity check enabled (in general).

If commanded volume release is **greater than Requested Volume** (conservative):

- (D/R) ≥ (D/S) (DR/DS=const.) in any PtP strategy.
- (D/R) ≥ (D/S) in Continuous strategy if proximity check is disabled.
- (D/R) <= (D/S) in Continuous strategy with proximity check enabled.

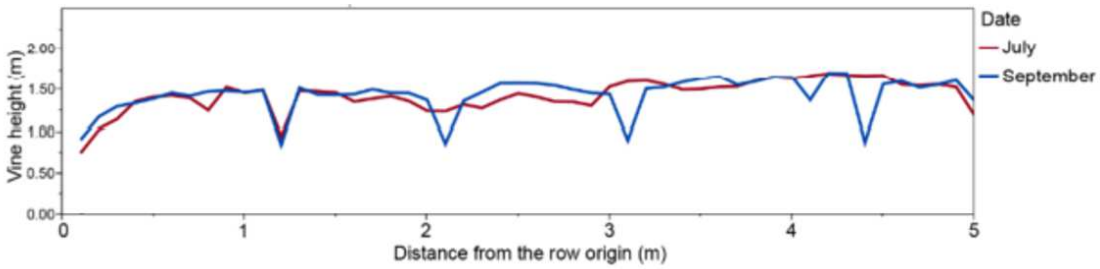


Figure 92 Height of the single plants in function of the distance from the row origin. Data collected in two different period in[34] province of Lleida, Northeastern Spain, from Merlot, Albarino and Chardonnay fields [34]

4.2 Pump model and relative control

The pump used in the experimentation was a Cybernova 12V DC 131 PSI (figure 93). Data reported by the manufacturer are limited to maximum pressure and flow rate values and relative maximum voltage and current. Furthermore, no experimental data of the time response of the pump was recorded, so a simplified model will be developed and controlled in open loop, although having a precise dynamic characteristic of the pump would allow to insert a PID control law. As specified by the manufacturer, the pump can be controlled though an external valve or modulating the input voltage. Our choice has been to use a PWM signal of duty cicle β to control the pump.

$$V = \beta \cdot 12\ [V] \tag{4.28}$$

The PWM signal is obtained after an amplification of the 5V dynamic of the MCU though a MOSFET, plugged the system directly to the drone battery pack with a voltage regulator.

Voltage Source	12 V DC
Max current	6 A
Max Power	70 W
Max Flow Rate	4-6 L/min
Max Pressure	9 bar
Weight	600 g

Figure 93 Diaphragm pump, Cybernova 12V DC 131 PSI

To develop the model, the maximum flow rate is considered of 4 L/min and maximum pressure of 9 bar (figure 94). These two values are considered as the open circuit flow rate (when no pressure load is inserted) and the stall-pressure (pressure produced when the flow rate is zero) respectively, when the source voltage is amplified at 100% of 12 V. The hypotheses needed are:

1. $Q_{no\ load}$ and p_{stall} are proportional, at steady state, to the voltage input (in Volts) with a slope computed by the provided data:

$$Q_{no\ load}(V) = \frac{1}{3}V \quad [\frac{L}{min}] \tag{4.29}$$

$$p_{stall}\,(V) = \frac{3}{4}V \quad [bar] \tag{4.30}$$

2. The pressure, in each operating point, is proportional to the current flow, expressed in Amperes:

$$p\,(i) = \frac{3}{2}i \quad [bar] \tag{4.31}$$

3. At a given voltage, the characteristic curve of the pump, expressed as $p_V(Q)$ is an affine function of the type $Qw + w_0$ with $w \neq w(V)$, namely for each voltage V the characteristics are all parallel to each other.

$$p\,(Q,V) = -\frac{9}{4}Q + \frac{3}{4}V \tag{4.32}$$

4. Following the nozzle manufacturer indications, the ideal pressure for a good droplet diameters distribution is in between 3 and 6 bar, so that a saturation block will play the role of pressure gauge and regulator.
5. The pressure required by the nozzle to reach a given flow rate is a linear function in the operating region. Using the ARAG datasheet values the relationship is:

$$p_{Nozzle}(Q_{Nozzle}) \cong 9 \cdot Q_{Nozzle} - 4.2\ [bar] \tag{4.33}$$

COD. CODE CÓD.	(l/min)									
	3 bar	4 bar	5 bar	6 bar	7 bar	8 bar	9 bar	10 bar	11 bar	12 bar
422HCI8002	0.80	0.92	1.03	1.13	1.22	1.31	1.39	1.46	1.53	1.60

Figure 94 80° hollow cone nozzle Q-p characteristics, from ARAG datasheet

With Q expressed in L/min.

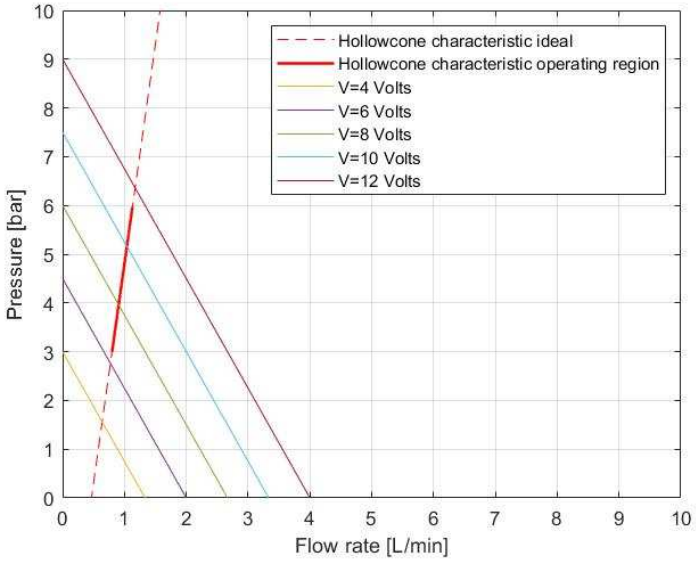


Figure 95 Thoric Q-p characteristic curves of pump and hollow cone 80° nozzle

Equaling $p\,(Q,V)$ and $p_{Nozzle}(Q_{Nozzle})$ the emerges the relation between the requested flow rate and the duty cycle:

$$Q_{Nozzle}(\beta) = \frac{3}{45} \cdot 12 \cdot \beta + \frac{4}{45} \cdot 4{,}2 \cong 0{,}8 \cdot \beta + 0{,}37 \quad (4.34)$$

And in the same way is found the relation between the duty cycle and the current:

$$i(\beta) = \frac{2}{3} p_{Nozzle}(\beta) = \frac{2}{3}(9 \cdot Q_{Nozzle}(\beta) - 4.2) \quad (4.35)$$

So, results:

$$i(\beta) = \frac{2}{3}\left(\frac{3}{5} \cdot 12 \cdot \beta - \frac{1}{5} \cdot 4{,}2\right) \cong 4.8 \cdot \beta - 0{,}56 \quad (4.36)$$

To keep the optimal pressure region, the duty cycle is saturated by the ideal pressure regulator:

$$\mathbf{53{,}75\ \%} \leq \boldsymbol{\beta} \leq \mathbf{95\ \%} \quad (4.37)$$

If the requested flowrate is exactly zero, then the duty cycle is not computed anymore and the current equals zero too. So, the current:

$$\begin{cases} 2{,}02\ A \leq i \leq 4\ A & if\ Q_{Nozzle\_requested} > 0 \\ i = 0 & if\ Q_{Nozzle\_requested} = 0 \end{cases} \quad (4.38)$$

And the actual flow rate:

$$\begin{cases} 0{,}8 \frac{L}{min} \leq Q_{Nozzle} \leq 1{,}13\ \frac{L}{min} & if\ Q_{Nozzle\_requested} > 0 \\ Q_{Nozzle} = 0 & if\ Q_{Nozzle\_requested} = 0 \end{cases} \quad (4.39)$$

This is a conservative strategy, because ensures that also the plants requiring very low PPP volume are sprayed.

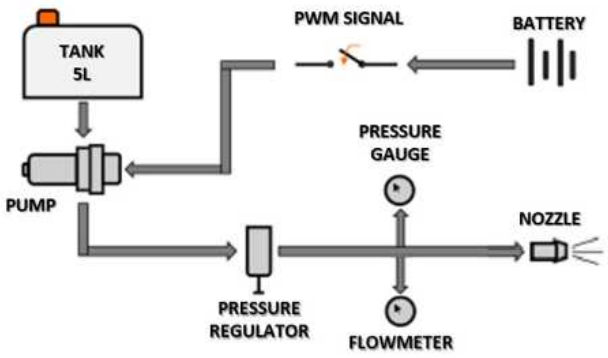


Figure 96 Spray system scheme, modified by [9]

An experimental test bench should be use to retrieve a relationship between the duty ratio and steady state flow rate of the pump-nozzle system, as done in [35]. An open loop control of the pump has many limitations. First, the pump flow rate response to an applied voltage encounters delays due to the inertia of the mechanical components of the pump and of the fluid. Furthermore, trusting on the look up table of the pump characteristics becomes reasonable only if an experimental data collection is performed. A closed loop control of the pump should consist in the real time sensing of the flow rate with a relative adjustment of the PWM frequency through a PID signal of the error. To add some dynamics the pump was modeled as a 1st order transfer function with a delay. The time constants values t1 and t2 was retrieved by experimental results in [35], where a PLD1206 diaphragm pump with similar characteristics of Cybernova 12V DC 131 PSI connected to a Lechler 110-01 fan-shaped nozzle were tested.

$$Q_{Nozzle}(t) = L^{-1}\left\{\frac{\left(\frac{1}{t2}\right)}{s+\frac{1}{t2}}\cdot\left(e^{-s\cdot t1}0{,}8\cdot\frac{V(s)}{12}+\frac{0{,}37}{s}\right)\right\} \qquad if\ L^{-1}\{\beta\cdot e^{-s\cdot t1}\}>0 \qquad (4.40)$$

$$Q_{Nozzle}(t) = L^{-1}\left\{\frac{\left(\frac{1}{t2}\right)}{s+\frac{1}{t2}}\cdot\left(e^{-s\cdot t1}0{,}8\cdot\frac{V(s)}{12}\right)\right\} \qquad if\ L^{-1}\{\beta\cdot e^{-s\cdot t1}\}=0 \qquad (4.41)$$

where *t1=0,53* represents the stabilization time of the pump and t2=0,16 the startup/shutdown time delay. The mechanical inertia of the diaphragm pump determines the delay characteristics of the spray startup and shutdown, while the mechanical structure and manufacturing process of the nozzle determine the spray stabilization time[35]. In the Simulink model this dynamic was added considering the non-linear and discontinuous characteristic deduced before:

$$Q_{Nozzle}(\beta) = 0{,}8\cdot\beta + 0{,}37 \qquad if\ \beta > 0 \qquad (4.42)$$

$$Q_{Nozzle}(\beta) = 0{,}8\cdot\beta \qquad if\ \beta = 0 \qquad (4.43)$$

From the results below is clearly seen that using the chosen MCU sampling time of 10 milliseconds the shattered behavior of the steady state response has a magnitude of about 0,02 L/min. The response of the pump is the composition of pump delay and nozzle dynamic.

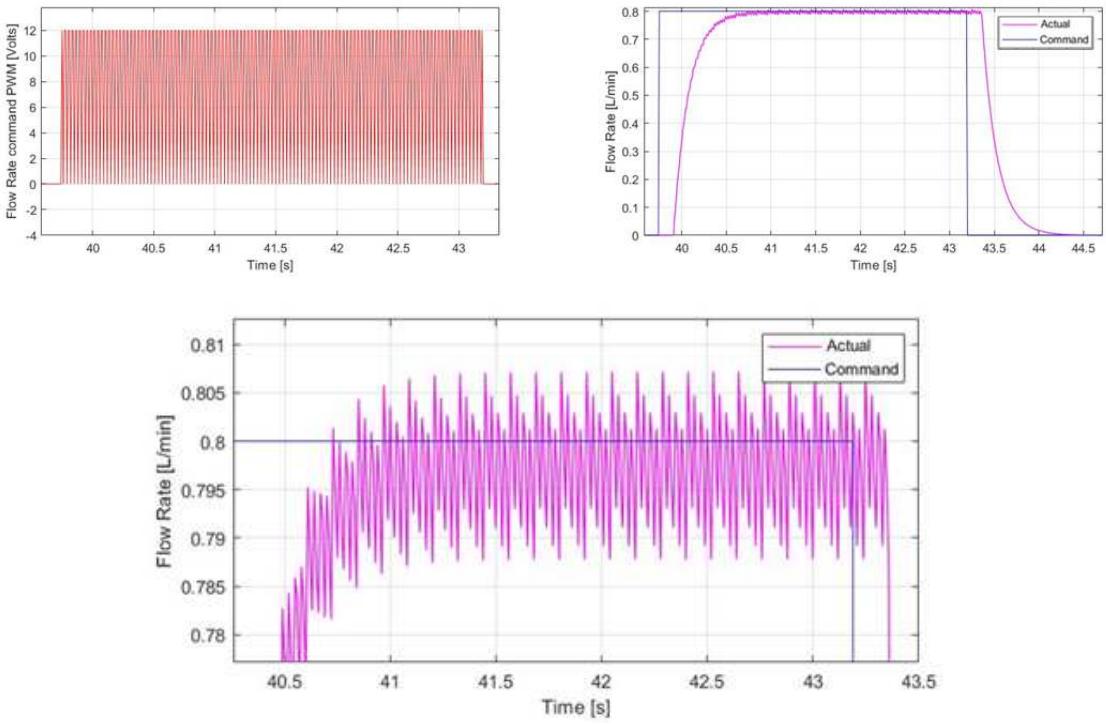


Figure 97 Pump step response, 53% Duty Ratio, 10 ms PWM sampling time

5 Spray strategies and adaptive path planning

In chapter 2 have been discussed some Convolutional Neural Network approaches to recognize diseases at a plant and leaf level and simultaneously distinguish “Healthy” plants; in chapter 3 has been developed a mechanical model of a 25 Kg quadrotor and designed an LQR control on the linearized system dynamics and, finally, in chapter 4 has been presented the spray system, including hollow-cone nozzle flux profile and deviation experimental relationships with nozzle pressure, air speed and rotors speed, an analytical model of the footprint distribution and its interaction with the grapevines geometries. The objective of this chapter is to merge all together the previous achievements, proposing a realistic scenario in which the disease recognition phase

provides a 2D coordinates prescription map containing the PPP needy plants positions and relative volume demands. This prescription map, in general, is a points cloud, then, considering some case-specific constraints, an algorithm to search for an optimal sequence of waypoints is needed. The resulting optimal sequence is then converted offline in a path planning. Two different spray strategies are studied, one suitable for intensive spray purpose and another more focused to precision spray missions. The offline computed path planning can be handled with one of the two strategies, following the application requirements, and a wind estimation algorithm is studied to obtain information about the flux deposition area and distribution in function of UAV-air relative velocity, through which an online correction of the path planning is possible. The footprint coordinates estimation has been exploited both to compensate the spray drift (path correction mechanism) and to preserve PPP if the footprint is deviated enough, from the plant barycenter, to justify the pump shutdown (proximity check).

5.1 Spray strategies

5.1.1 Continuous spray

With continuous spraying is intended the spray strategy in which the UAV releases PPP without stop in hovering over the target plant. The immediate implementation of this strategy is in intensive spraying of all the plants in a certain area. To have a so large coverage the UAV should follows the rows in a parallel way (Cross-Configuration) or orthogonally (X-configuration, not considered in this thesis) setting offline a constant nominal flow rate that ensure to be in 60/120 L/hectare, as explain in 3.1. However, if needed, the strategy can be also adapted to precision spray intents, regulating flow rate and drone velocity to indirect control the released volume per plant. Under the hypothesis that the single vineyard can be contained in a 90 cm side square, the leaves crown are comprised in between 1-1.5 meters height [34] , with n number of nozzles, and the UAV flies at 2 meters altitude, the PPP distribution covers a reasonable portion of the plant crown and released a *maximum* quantity of product, per plant, that follows this approximated formula:

$$PPP_{plant} = 0{,}015 \cdot \frac{FR \cdot n}{v_{UAV}} \tag{5.01}$$

$$\begin{cases} FR: \quad Total\ flow\ rate\ of\ the\ Hollowcones\ system\ \left[\frac{L}{min}\right] \\ v_{UAV}: \ drone\ velocity\ over\ the\ row\ \left[\frac{m}{s}\right] \\ with\ the\ hypotses: \\ Single\ grapevine\ comprised\ in\ a\ square\ with\ side\ of\ 90\ cm \\ Grapevine\ crown\ leaves\ of\ 1{,}5\ m\ height \\ UAV\ flight\ height\ of\ 2\ meters \end{cases} \tag{5.02}$$

*The formula comes simply from considering how long it takes the drone to travel the plant (t = 0.9/v secs = 0.9/60/v = 0.015/v). Multiplying this time by the flow rate [L/min] gives the volume deposited on the plant (PPP_plant = 0.015*FR/v).This formula considers the minimum travel time of the plant, that is when the trajectory of the drone is aligned to the row; in all other cases the distance to be covered on the plant tends to 0.9*sqrt(2), consequently the above formula offers the condition that maximizes the flow rate so it is to be considered conservative because it maximizes the liter per plant).

In continuous spraying, while drone overflights the plants crown the flow rate is adjusted in real time based on the drone velocity, supposing its trajectory is aligned with the row, as explained in (5.01). When the requested flow rate overcomes the upper or lower limit that the pump can manage or to have an optimal droplet size, the UAV velocity must be changed. If, at velocity $v_{UAV}{}^{*}$ the requested flow rate FR_{req} is greater than the maximum (1,13 L/min) of ΔFR , than:

$$FR_{req} = FR_{max} + \Delta FR = \frac{PPP^{plant}}{0{,}015} v_{UAV}{}^{*} \tag{5.03}$$

To lower FR_{req} than ${v_{UAV}}^*$ must be reduced with a gain $K = \frac{FR_{max}}{FR_{max}+\Delta FR}$ to reestablish a request flow rate within its limits. If the requested flow rate is under FR_{min} the reasoning is complementary:

$$\begin{cases} K = \frac{FR_{max}}{FR_{max}+\Delta FR}, & \Delta FR = FR_{req} - FR_{max} \quad if\ FR_{req} > FR_{max} \\ K = \frac{FR_{min}}{FR_{min}+\Delta FR}, & \Delta FR = FR_{req} - FR_{min} \quad if\ FR_{req} < FR_{min} \end{cases} \quad (5.04)$$

This spraying mode has 3 main limits if applied in precision spraying:

- If the requested volume is too great $K \rightarrow 0$, and it means the UAV should remain stationary to accomplish the delivering. It basically collapses to a Point-to-Point strategy.
- When the proximity check is too strict (i.e the threshold radius is too small) the continuous spray risks to spray less volume than needed at the end of the mission. For this reason, unless the prescription map is particularly heterogeneous, is better to relax or deactivate the proximity check, but this could lead to larger amounts of weak PPP. Adjunctively a safety factor can be insert in the effective PPP sprayed volume, leaving enabled the proximity check, but increasing the requested volume.
- When the velocity must be changed to keep the pump operative range the control response must be fast enough to ensure that velocity before exiting the target plant area. Since the steady state velocity of the UAV is within 2-3 m/s and the characteristic length (aligned with the row) of the grapevine is 90 cm the UAV spent about 300 milliseconds over each plant, so the control must respond in a fraction of this time.
- As will be seen in the next sections the wind estimator performs better while the UAV is stationary, making the continuous spraying mode less reliable for path correction purposes.

Although those criticalities, the continuous spraying mode is the first choice when the request an intensive spraying session of all the plants in field and the velocity of the mission worth more than precision.

5.1.2 Point-To-Point spray

Differently by the continuous spraying, the point-to-point (PTP) is studied *to ensure* that each plant receive its own product demand volume, as far as the delivered volume estimation mechanism is reliable. To do this the PTP strategy consists of sequentially visiting each to-be-treated plant, overflight them, correct its position trusting on estimated wind and the consequent footprint center coordinates, and spraying when the footprint center is within the radius imposed by the proximity check. This mechanism permits a greater control of the per plant sprayed product than in continuous case. Not considering the error introduced by the flowmeter and eventual leakages, at the end of the mission the total sprayed volume will be exactly equal to the total requested volume, differently by the continuous spraying mode. This implies higher Delivered/Requested (equal in this case to Delivered/Sprayed) ratios for PTP than the continuous spraying mode, although the more is reduced the proximity threshold radius the more time is required to complete the mission. PTP is more suitable for chirurgical spraying purposes, while can become too much slow for substitute the continuous spraying when there is not an evident heterogeneous PPP demand among the plants, and most of the plants in the field must be treated. Is important to notice that, both for the continuous and PTP cases, the Requested volume does referment simply to the volume prescribed to be delivered but is not necessarily the same of the volume that the plant requests. Indeed, whatever is the way in which the quantity of product needed is computed, the analysis of the next session will provide the Delivered/Requested ratios in different cases to compensate them with, for example, a correspondent over estimation of the Volume demand in the prescription map computing in the PTP case, or in an under estimation of

the drone velocity or over estimation of the PPP-plant volume in the continuous spray mode. When a waypoints sequence is ready to be used, the UAV changes its trajectory in real time, stationing on the target plant until its mission is computed. PtP introduces a time delay in the offline computed path who is essentially a time-out of the path tracking active until the desired PPP volume is sprayed. As explained, the prescription map can provide a *2xN* matrix *PMxy* containing the *(x,y)* coordinates of the *N* waypoints sequence and a *N*-dimensional vector *PMp* containing the prescribed volume for each waypoint. Given a constant inter-waypoints velocity v (set to 3 m/s in all the PtP simulations), when the GPS recognizes the drone approaches a waypoint *f* within a certain tolerance distance, the system follows the following steps:

- The drone is stopped in hovering over the waypoint.

$$(x,y)_{ref,k} = (x,y)_{ref,k-1} \quad (5.05)$$

- Estimates the wind and correct the position to allow the footprint center coincides with the target plant barycenter.
- If the footprint center is within the proximity radius threshold the flow rate goes to its nominal value (1,13 L/min) otherwise the pump shuts off. Simultaneously the flowmeter signal starts to be integrated only when the pump is active (to avoid eventual bias integration during the quiescent phases) obtaining the estimated volume delivered to plant *f*.
- When the delivered volume reaches the requested volume *PMp(f)* of the target the counter *f* is increased (*f=f+1*) and each time step Ts the reference is computed as:

$$(x,y)_{ref,k} = (x,y)_{ref,k-1} + v \cdot [\cos(\theta_{k-1}), \sin(\theta_{k-1})] \quad (5.06)$$

where θ_k is the orientation angle in the inertial plane *(x,y)* of the line connecting $(x,y)_{ref,k-1}$ and PMx_f:

$$\theta_k = atan2\left(PMxy(2,f) - y_{ref,k-1}, PMxy(1,f) - x_{ref,k-1}\right) \quad (5.07)$$

Once the drone position (x,y) passes in the neighborhood of *PMxy(:,f)* the path is kept constantly to the current value.

$$(x,y)_{ref,k} = (x,y)_{ref,k-1} \quad (5.08)$$

And the routine is repeated.

- When the UAV reaches for the second time the starting point the simulation is automatically stopped.

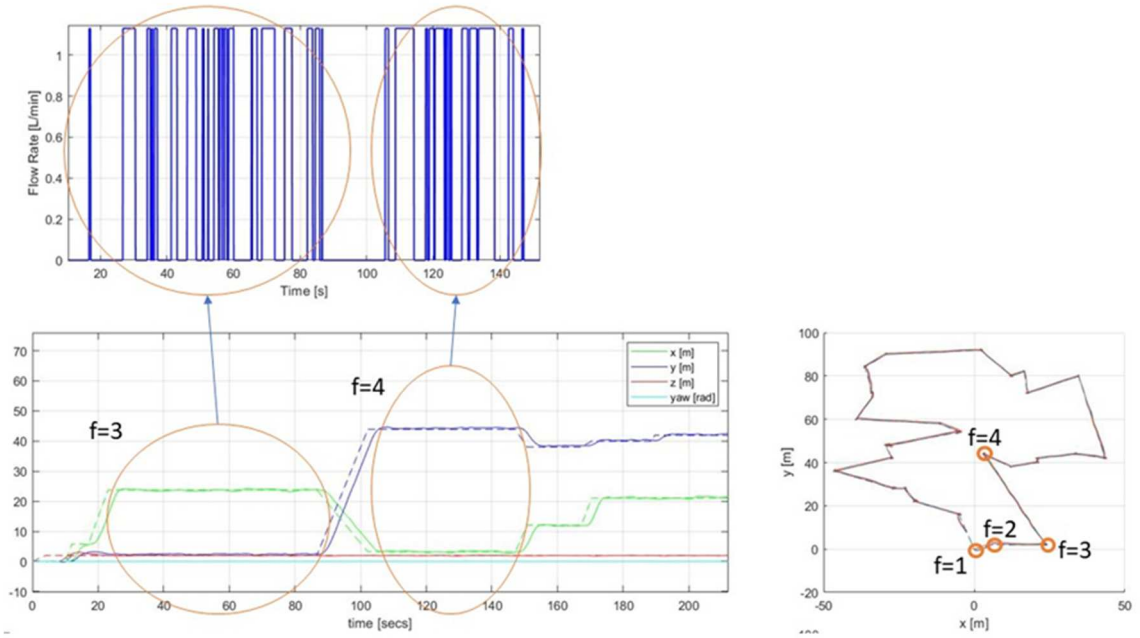


Figure 98 PtP strategy, state tracking

In figure 98 is presented a typical too sensible proximity check situation in which for each waypoint the drone overflight the plant in presence of 4 m/s wind mean and rapidly On/Off the pump at a nominal flow rate of 1,13 L/min. This happens because in this case the proximity radius was set to 22,5 centimeters, and the footprint center is continuously deviated by the variable wind intensity and by the UAV control intrinsic errors.

5.2 Offline path planning

In a first stage, small UAVs should diagnosticate diseases capturing images of all the plants, extract images features and use them to classify the pathological status of the plants. Fusing those data with a 3D map of the field, the system should be able to produce a prescription map whom elements correspond to each plant position in the axes of referment and the relative product demand of the plant in that position. From the prescription map a disordered set of waypoints is extracted under the form of a *2xN* matrix in which each column contains the *(x,y) N* coordinates of waypoints with respect to the inertial reference frame. Adjunctively a 1xN vector associates to each waypoint a prescribed PPP volume demand, e.g., extracted by the pathological status of each target plant by some continuous relations, or simply set to a constant value for all the positives targets. In 4.2 prescription maps was obtained posing non-zero a certain number of randomly picked elements of a matrix with dimension $n_{rows} \times n_{plants}$. The number of rows and plants is strictly linked to width and length of the operating area considered, since in chapter 4 has been fixed geometrical properties of plants, plant dimension (90x90 cm) and inter-row distance (2 meters). The 2D prescription map obtained is often normalized to have a maximum volume demand less than the tank capacity of 10 Liters. At the end of the chapter, where a complete simulation of all the mission stages will be presented, the prescription map will be derived from the use of dataset and CNN algorithms seen in chapter 2.

5.2.1 Modified Traveling Salesman Problem

Once the prescription map is submitted, the subsequent task is to find the visiting sequence of waypoints which minimize the charge consumption of the quadrotor. Concerning about the distance minimization only, the problem is the well-known Travelling Salesman Problem (TSP). The historical origin of the problem is unclear, but was formulated as:

"Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" (Wikipedia)

The problem belongs to the class of Non-deterministic Polynomial-time Hardness problems (NP-Hardness) and does not exist an exact analytical solution. If the number of "cities" is low enough an exact brute force algorithm can be applied, however the number of permutations is $(n-1)!/2$, where *n* is the number of cities. This means that if the number of ill plants is 100/hectare a brute force approach would requires $46663 \cdot 10^{151}$ permutations per hectare to find the global minima. In recent years have been developed many heuristics algorithms to tackle the problem. The most present in literature are genetic algorithms (GA), ant colony optimization (ACO), artificial neural networks (ANN), learning-based (LB) methods, particle swarm optimization (PSO), fuzzy logic (FL) and many others, as suggested in the survey [36]. The most used is the genetic algorithm, that shows high performance, in terms of computation time needed to find a local minimum of the problem, so for a higher presence of references and studies this has been our choice. The fundamental idea of the genetic algorithm is to generate an initial *generation* of *individuals* (in our case an initial bunch of waypoints sequence vectors). Then the *individuals* who best fit a certain cost function *survive* and parts of their *genes* (in our case sequence vectors elements) are combined and exchanged or permuted forming a new set of individuals who belong to the subsequent *generation.* The gene combinations follow different probabilistic logics that varies in function of the complexity of the algorithm. This work is based on the algorithm developed by Maxim Vedenyov in 2022 (Travelling salesman problem with Genetic algorithm; https://www.mathworks.com/matlabcentral/fileexchange/31818-travelling-salesman-problem-with-genetic-algorithm, MATLAB Central File Exchange. Retrieved January 26, 2022.). The algorithm presents three kinds of recombination, each one associated with its probability of occurrence per generation:

- Probability p_m of mutation of exchange 2 random cities of the same path.
- Probability $p_{m,2}$ of mutation of exchange 2 random pieces of the same path.
- Probability $p_{m,f}$ of mutation of flip random pieces of the same path.

We have used the value suggested by the author, namely $p_m = 0{,}01;\ p_{m,2} = 0{,}02;\ p_{m,f} = 0{,}08$. A first test of the algorithm was done comparing the optimal distances found by the GA with the actual global minimum searched with a brute force. In the comparison were inserted 10 sequences with a number of waypoints spacing between 4 and 12. For the first sequence the brute force needed 3 permutations to find the optimal, while for the last one 19 958 400. In all the cases the GA was able to find the optimal solution within few seconds. The number of generations was 3000 and the population size was of 30 individuals. Waypoints were generated randomly in a squared area of 30x30 m^2.

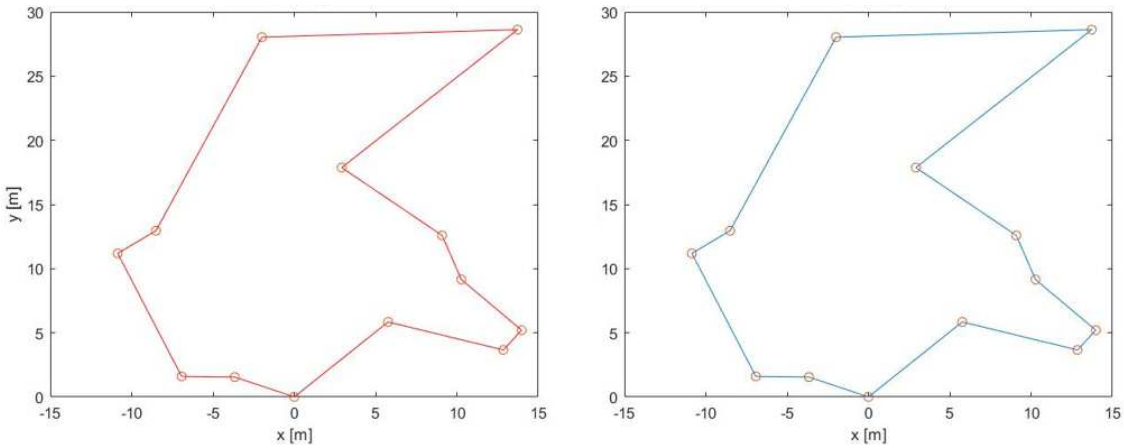


Figure 99 12 Waypoints, optimal solution found by GA (Left) and the Brute force method (Right)

Table 18 BF/GA comparison

	4	5	6	7	8	9	10	11	12 waypoints
GA solution[m]	63,08	43,47	77.75	95,9	75,58	91,91	98,129	91,551	102,64
BF solution[m]	63,08	43,47	77.75	95,9	75,58	91,91	98,129	91,551	102,64

To test the algorithm on higher number of waypoints the GA solutions were compared with a Nearest Neighborhood (NN) algorithm and with a modified version of the same which considers the vineyard geometry (modified NN). The NN found the optimal path choosing as next waypoint the next among all the others, without repetitions. NN is a simple algorithm widely used to solve rapidly the TSP in literature for its simplicity, speed, and reasonable results. The modified NN algorithm, as shown in the flow chart below, considers the discrete nature of the space in which waypoints can appear. Indeed, waypoints can be present only each row (each 2 meters in y direction) and each plant are distant by 90 cm. Before to pass to next row the algorithm chooses the nearest waypoint present in the row. Once the last waypoint in the last row is visited the algorithm chooses as successor the nearest tailender waypoint present in the nearest row. Both NN and modified NN chooses as last waypoint the starting point.

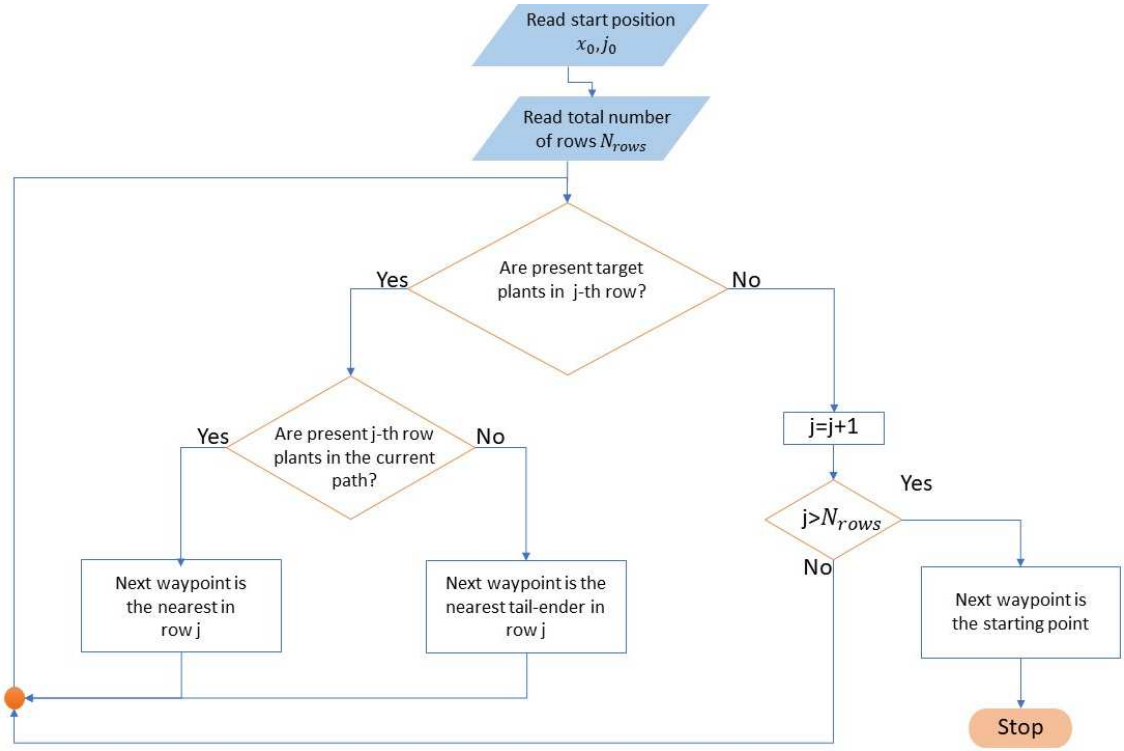


Figure 100 Modified Nearest Neighborhood algorithm for vineyard geometry

Is worth noticing the modified NN is substantially an optimized version of the path planning used in the continuous spray mode, where each row is completely visited. This is the distance that the UAV should cover if travels following the entire rows lengths and then come back to the starting point, covering all the plants. This distance, in 30x30 m^2 area is constantly equal to:

$$d_{ref,\ S_S=30\ m} = S_S \cdot \frac{S_S}{L_{inter-row}} + S_S + \frac{S_S}{2} + \frac{\sqrt{5}}{2} \cdot S_S = 523{,}4348\ m \qquad (5.09)$$

where has been considered the sum of the length of all the rows containable in a S_S square side ($S_S \cdot \frac{S_S}{L_{inter-row}}$), the summation of all the inter-row turnings (S_S), the half row that has to be travel

at the beginning, since the origin is define at the 15-th meter of the bottom first row, ($\frac{S_S}{2}$) and the minimum comeback length that the drone has to travel once overflight the last top row ($\frac{\sqrt{5}}{2} \cdot S_S$). In figure 101 can be seen that when the number of waypoints approaches the total number of plants in the area it starts to coincide with the referment path planning studied in the previous chapters.

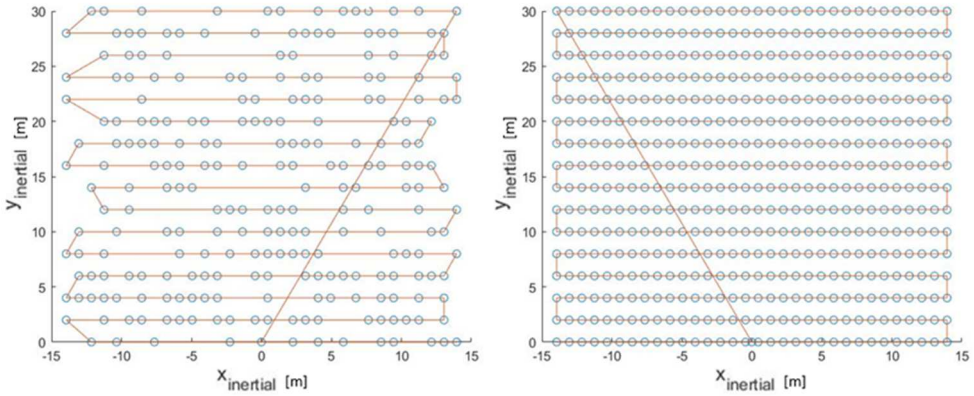


Figure 101 modified NN solution with 240 waypoints (Left) and 513 (100% filled map) (Right)

A 30x30 m^2 map was generated and 20 different waypoints sequences with a progressively increasing of their size were obtained. For each sequence each algorithm provided the sequence of the minimum distance path and their computation time. The GA was running with population and generation size proportional with the number of waypoints to leave the algorithm to adapt combinations and iterations with the increasingly complexity of the problem.

$$population\ size = 2N$$

$$generations = 800N$$

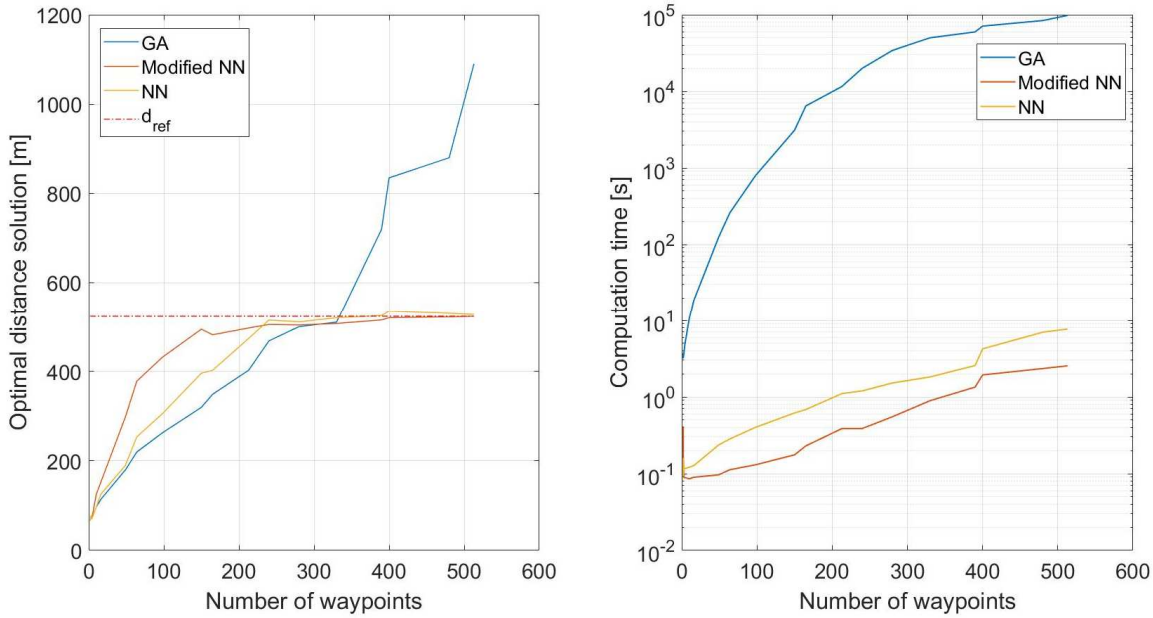


Figure 102 Genetic Algorithm, NN and modified NN best distances (Left) and computational time (Right) in function of the number of waypoints to be connected (the total number of plants in the 30x30 m^2 area is 513)

As can be noticed in figure 102, GA provides the minimal distance among the alternatives under 300 waypoints, then continuous to increasing someway linearly with the number of waypoints. Under 250 waypoints the conventional NN is able to find the second minimum distance over

which is overcame by the modified NN. As expected, the modified NN converges to the referment distance as the waypoints increase. In the table below are collected the rankings of the three algorithms, in referment with the solution found against the optimal, in different intervals of map filling density.

Table 19 GA, NN and modified NN ranking in function of map density, where $d = \frac{N}{513}100$

	$\boldsymbol{d < 45\%}$	$\boldsymbol{45\% \leq d < 50\%}$	$\boldsymbol{50\% < d \leq 68\%}$	$\boldsymbol{d > 68\ \%}$
GA	1st	1st	2nd	3rd
NN	2nd	3rd	3rd	2nd
Modified NN	3rd	2nd	1st	1st

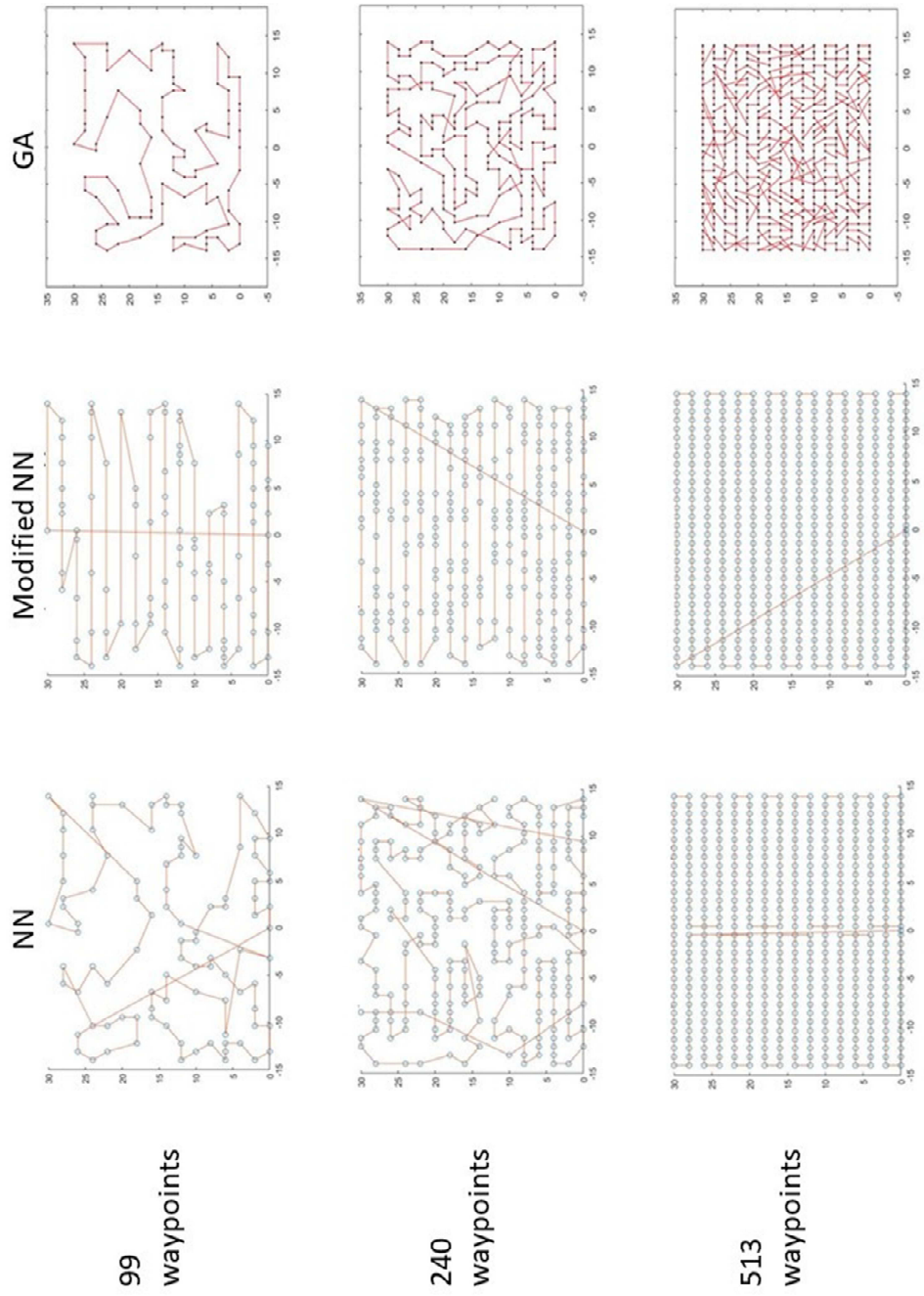


Figure 103 NN, modified NN and GA optimal sequences for different number of waypoints

It is possible to see that the GA solutions become rougher as the waypoints increase, suggesting that the number of generations and/or population size should increase more than linearly with the number of waypoints to reach similar optimality for increasing sizes of the problem. However, the computational time needed to solve for example the 240 waypoints TSP was about 4 hours, reaching up to 10 hours for the fully filled map against the almost constant few seconds needed by NN and modified NN (time was measured with the function tic-toc in MATLAB on Acer Nitro AN5N5 laptop). From those reasonings emerges that, when the prescription map starts to become too dense, the time needed to get a travel with distance less than d_{ref} could overcomes the reasonable amount of time that could passes between the missions, and the better choice could be to use the straight-forward path planning of referment, namely the modified NN. In this thesis is

supposed that the UAV covers an area of 900 m^2 in a single mission, where about the 10-30% of the plants can be diseased (i.e. about 50-150 plants) and several hours (or days) can passes between two missions. In this way, the ability of GA to find the best solution among the three algorithms is preferred to the extremely faster solutions provided by the NNs. However, a hybrid solution could be thought. For example, the map density check could be also applied at a lower scale, i.e., to face too dense sub-areas of the prescription map their barycenter can be treated as single waypoint by the genetic algorithm to find the global optimal sequence, then be interpolated with a continuous line that follows the rows between the first and the last point of the cluster. Since this problem implies many other questions, such as the use of continuous or discrete movement and spraying and how to define a mechanism to automatically distinguish between clusters and insulated plants, this topic has been left for future works and developments. By the way a possible scheme of the algorithm is proposed in figure 104.

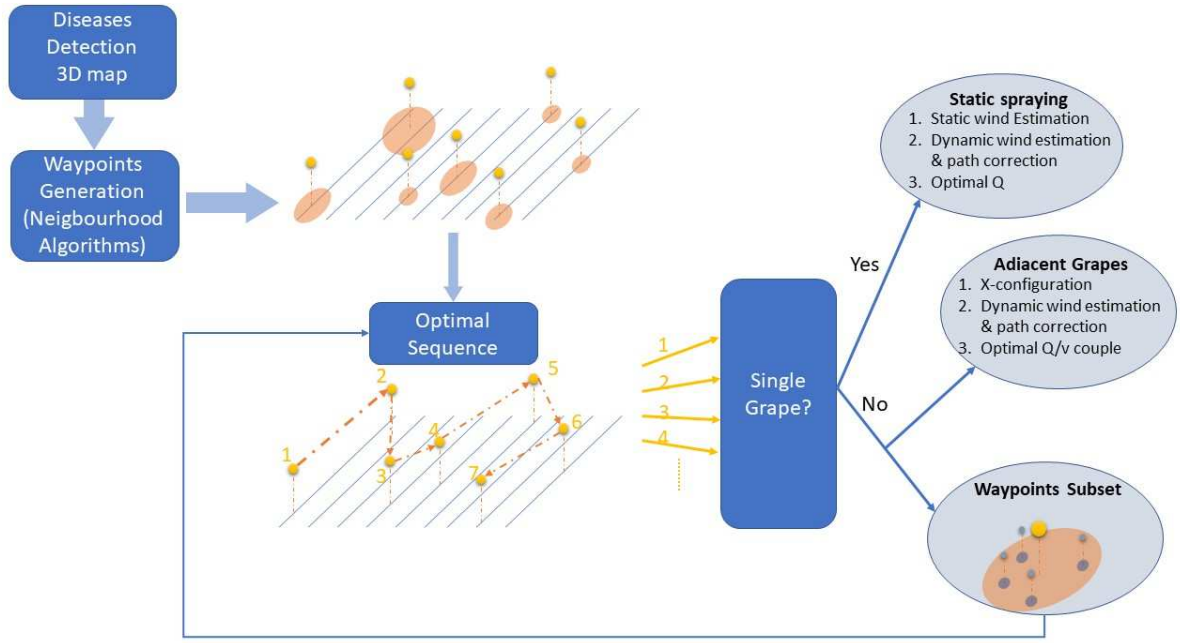


Figure 104 General purpose path planning optimization algorithm block scheme

Although these criticalities, can be seen that for a number of way points less than 50% of the total number of plants in 30x30 m^2 squared area, GA offers the opportunity to reduce the referment travel distance running for maximum few minutes, showing its suitability for low-medium density ill plants spraying purposes. This is the analyzed case. So, considering the simpler problem of reaching a certain number of insulated target plants, the Traveling Salesman Problem cost function, namely the travel distance, has been modified to further increase the UAV autonomy, keeping the same number of plants sprayed. Minimizing only the distance does not consider the emptying history of the PPP tank. During its travel the UAV loses weight and pushes its flight time horizon forward, so the earlier it reduces the payload the more is the gain in autonomy. In other word must be minimized the time integral of the yet on-board PPP volume, or, since in this case study the UAV is programmed to have always the same inter-waypoints velocity, the space integral of the same quantity. Supposing the UAV being in hover the thrust must be proportional to the total mass. Since the mechanical power is $P = K \cdot \omega^3 = C\omega = \frac{K_T}{K_F} \cdot Thrust \cdot \omega$, then results the power to be proportional to ${V_{onBoard,k}}^{1,5}$. Given N number of waypoints, counting the starting point two times as first and last waypoint, $V_{onBoard,k}$ the PPP volume, in Liters, still present in the drone tank between the waypoint k and k-1, and d_k the distance between the waypoints k and k-1, the cost function is:

$$J = \sum_{k=2}^{N} {V_{onBoard,k}}^{1,5} \cdot d_k \qquad (5.10)$$

Theoretically, minimizing this cost function means to prefer spraying first the subset of highest demanding plants that simultaneously require to cover the lowest distance to be reached. This means that minimizing only $J_{payload}$ can also lead to the minimization of the total travel distance. Anyway, simulations show that passing only $J_{payload}$ as a cost function to the GA the optimal travel distance found is typically higher than that one found passing as cost only the total distance. The third option is the definition of a coupled cost function weighting the two terms:

$$J_{payload} = w_1 \cdot distance + w_2 \cdot J \qquad (5.11)$$

Wight values was tested fixing $w_2 = 1$ and leaving $w_1$varying between 0 and 10. Has been noticed that for $w_1 \gg 1$ and $w_1 \ll 1$ the GA behaved too similarly the *only distance* approach or the *only weight time integral,* respectively. Therefore, in the *coupled* cost function the two weight was posed equal to 1, values at which the algorithm have shown a significant difference in the solution search with respect the two extreme cases. To demonstrate the performance of the algorithm of Maxim Vedenyov with the modified cost, two scenario was analyzed, using the Simulink model discussed in previous sections provided with all the designed features implemented, in absence of wind or other disturbances. In both scenarios there were 56 cities, 300 individuals per generations and 30000 generations. The simulation stops when the starting point is reached for the second time. The first scenario is the optimization of waypoints sequence in presence of a strongly heterogenous PPP volume demand. In this case few waypoints (that in general could indicate the barycenter of plant clusters) need most of the total volume. As can be seen in the three figures below (figure 105,106,107), the traditional *only distance* solver is able to find the shortest path, but not considering the payload history consumes the most. The *only weight time integral case saves* 1040 mAh but follows the longer path. Surprisingly the best is the *coupled* solution with the best J value, the 2nd shorter path (about 110 meters longer than the optimal) and the lower charge consumption, namely 15836 mAh, against 16022 mAh of the *only weight time integral* and 17062 mAh of the *only distance.*

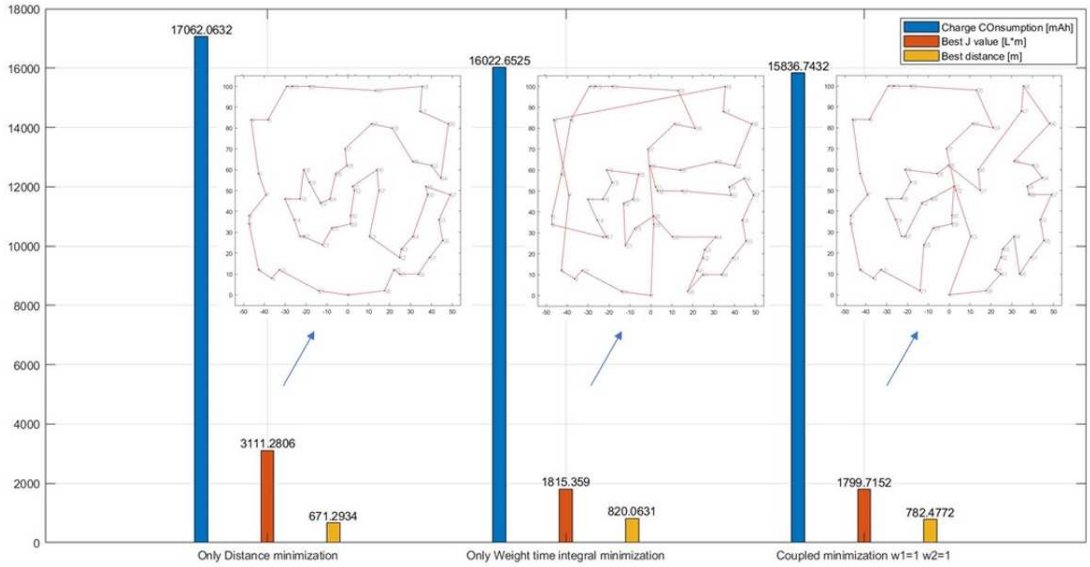


Figure 105 Charge consumption (Motors+Pump), optimal distance and optimal J value for different weights of the cost function, Heterogeneous volume demand

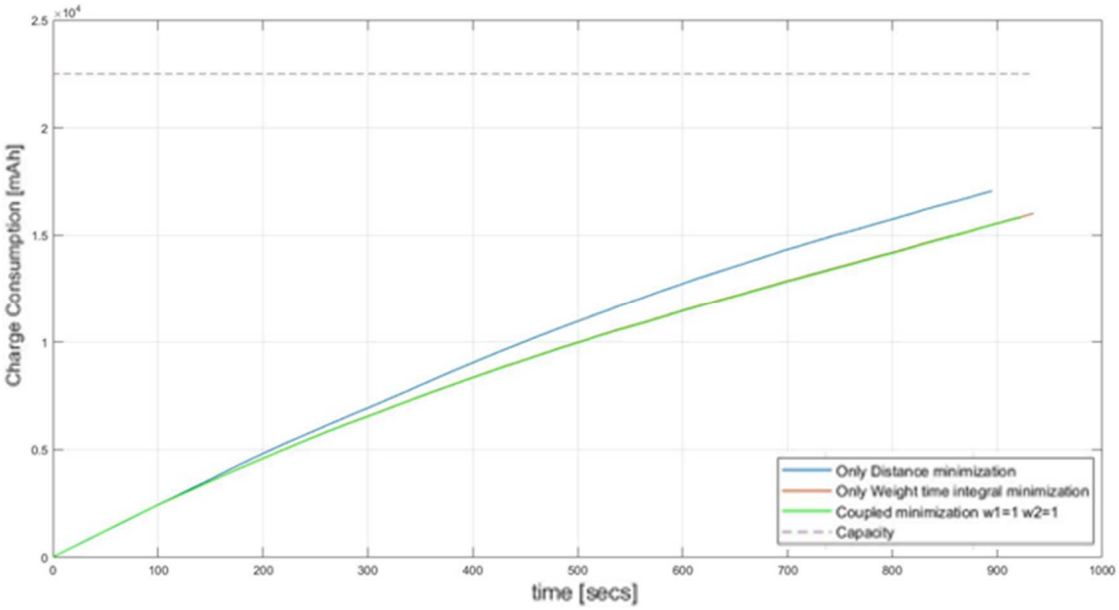


Figure 106 Charge consumption histories for the different cost function results, Heterogeneous volume demand

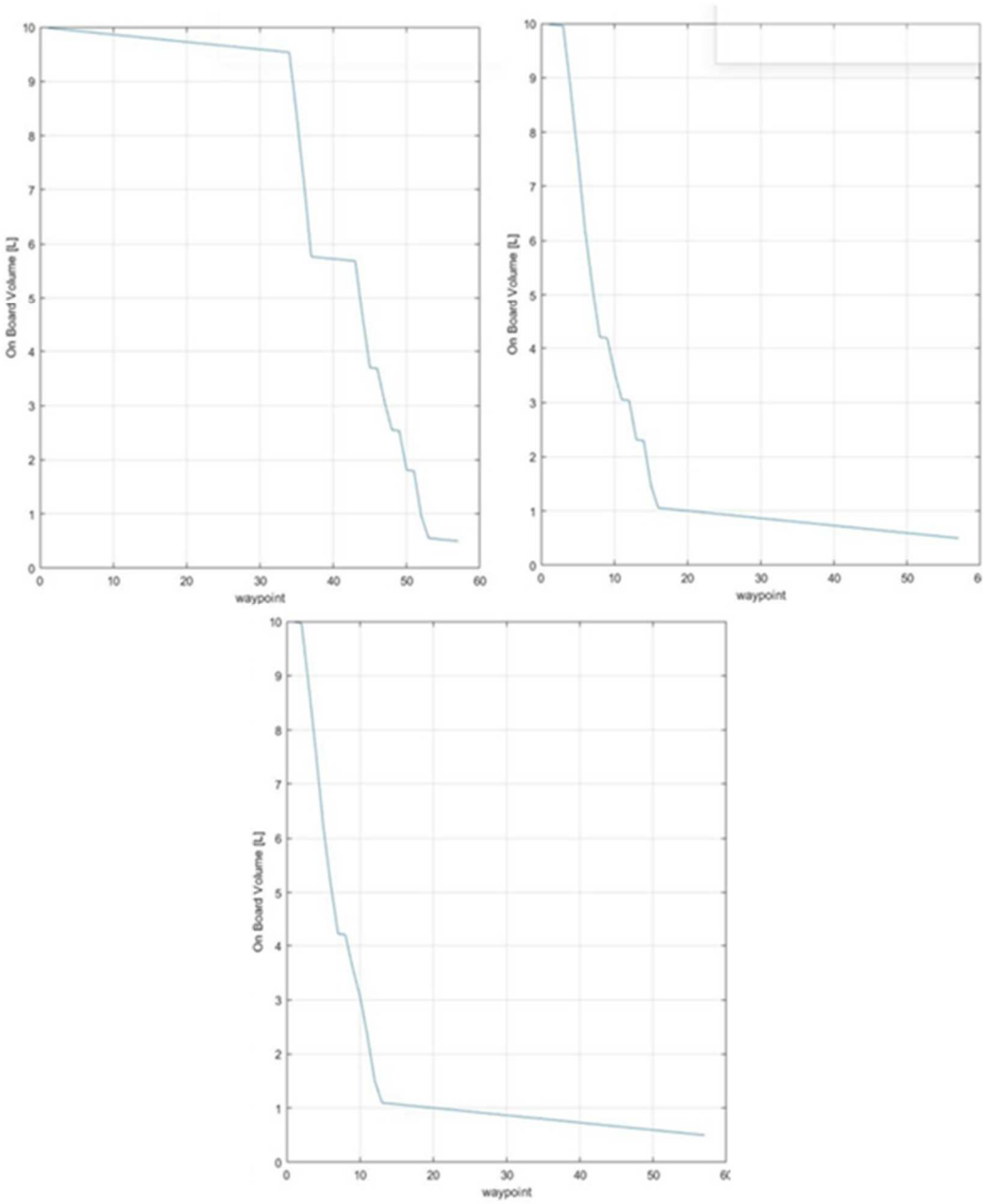


Figure 107 On board volume history, Only distance (Top-left), Coupled (top-right), Only weight (bottom), Heterogenous volume demand, 56 cities

The 2nd scenario consisted in a uniform distribution of PPP demand across waypoints. In this case the sequence of waypoint almost does not influence the J value anymore. Seeing the three graphs below, can be noticed that the worst option is the *only weight time integral case,* with 17157 mAh consumption, followed by the *coupled case* with 16274 mAh and the *only distance case,* with 16206 mAh. As in the previous case the *coupled* const function reveals able to reach shorter path and lower J values than the *only weight*. Since, in general, the *coupled* cost function seems to be able to find a short path and, simultaneously, shows great robustness against heterogeneous volume demands there are two options:

- Use always the *coupled* cost function; this could lead to often have the minimization of the autonomy unless the PPP demand is strongly uniform. In the last case, although for a small gap, the *only distance*, i.e., the traditional Salesman Problem sub-solution should be the best choice.

- Analyze, before the application of the solver, the PPP demand distribution and set the cost function weights following some relation found between them and the statistichal modes of the distribution (e.g. a monotonically increasing relation between the PPP distribution variance and the weight w_2).

Since the second option is laboriousness, and the gain in autonomy seemed to be not so large, we've opted to always use the *coupled* cost function.

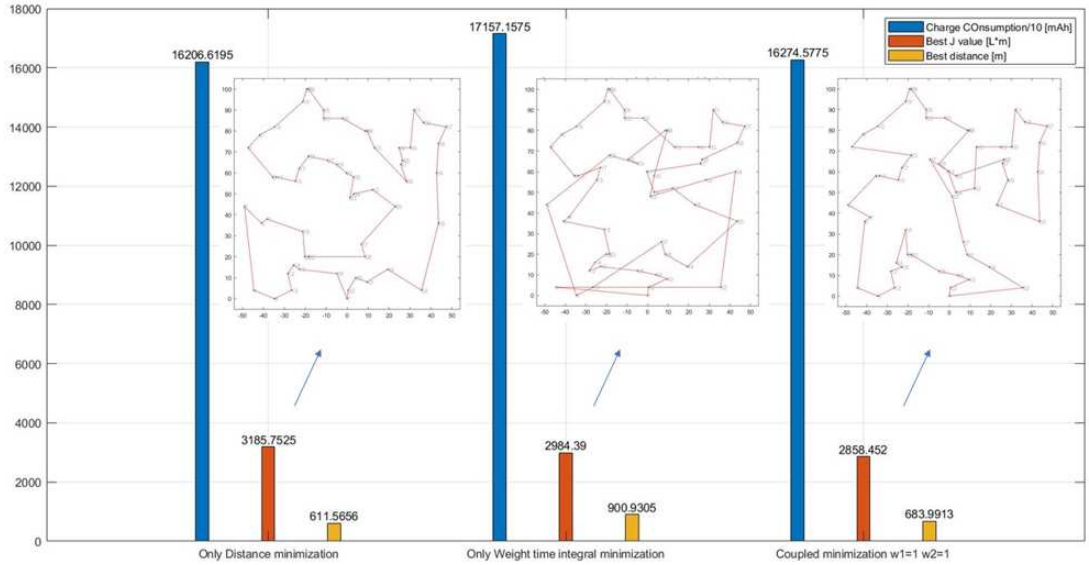


Figure 108 Charge consumption (Motors+Pump), optimal distance and optimal J value for different weights of the cost function, Uniform volume demand, 56 cities

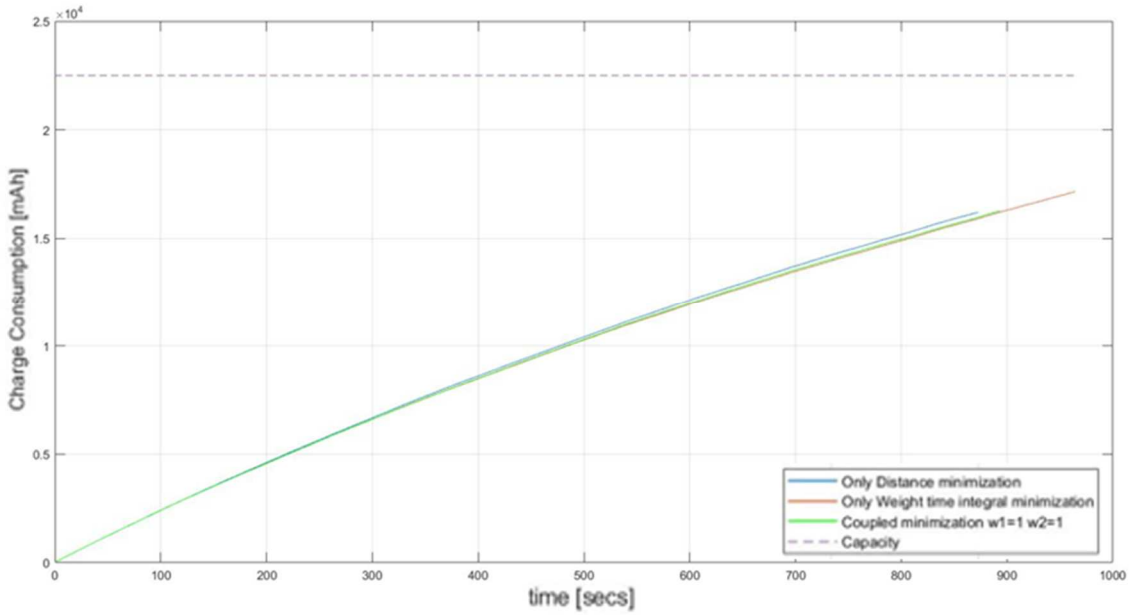


Figure 109 Charge consumption histories for the different cost function results, Uniform volume demand

The TSP studied here was developed in 2D, but the 3D case is simply obtainable with small modifications. As discussed in [6] the path finding task cannot be focused solely on the optimal waypoints sequence. In the working space there are fixed obstacles and moving objects that could intercept the straight lines connecting two subsequent waypoints. The A* algorithm and the Theta algorithm are two ways perform an offline/online path finding act to adjust the original path planning to find the minimum distance to connect two waypoints when the Line of Sight (LoS) is compromised. The offline path finding should be based on previous 3D mapping of the working

area, while an online path finding can be actuated through on-board sensors that perform a LoS checking in real time, such as Lidars and ultrasound proximity sensors. Adjunctively a Capacitated Vehicle Routing Problem (CVRP) solver should be used upward the TSP in order to consider the limited tank capacity of the UAV and thereby choose to visit only the waypoints who ensure the not exceeding of the tank volume, but those themes were not in the scope of this thesis and was left for feature works.

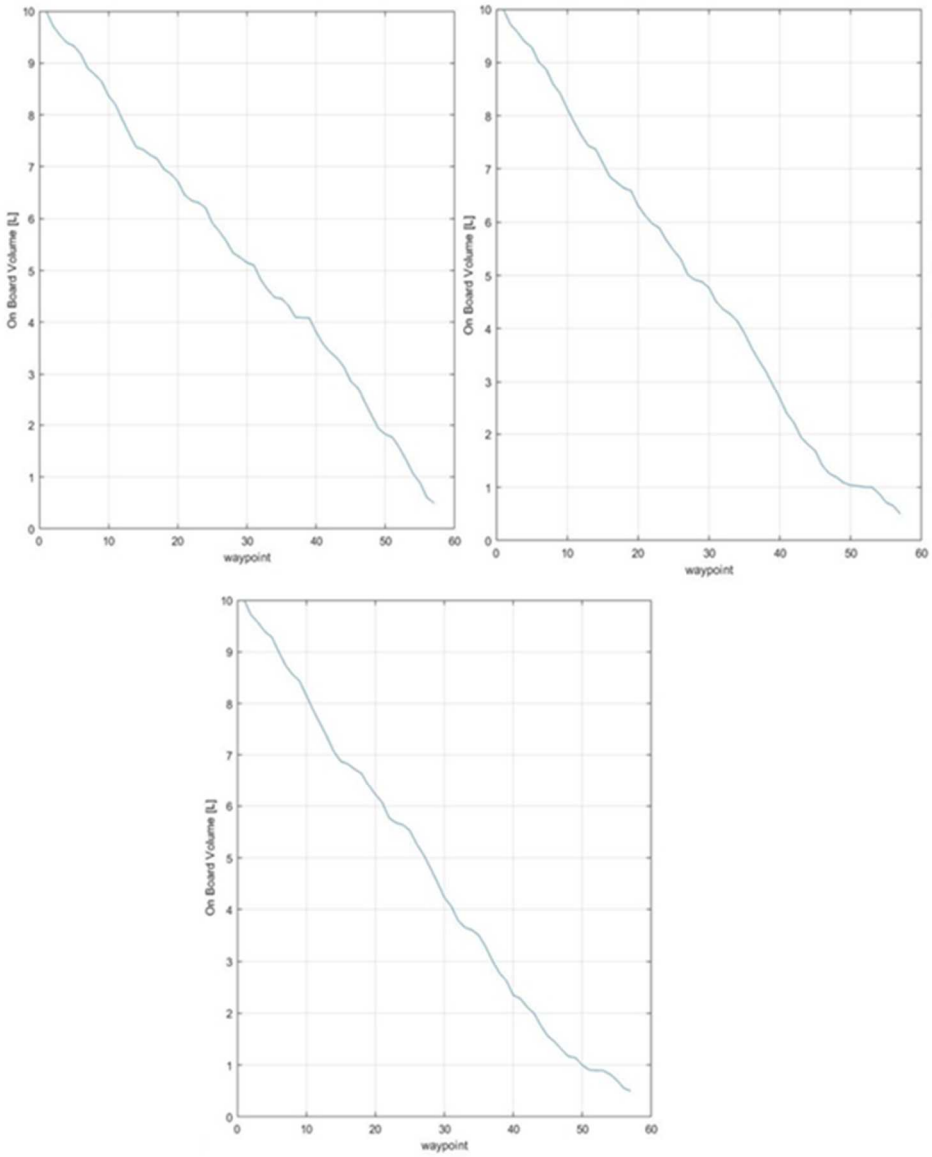


Figure 110 On board volume history, Only distance (Top-left), Coupled (top-right), Only weight (bottom), Uniform volume demand, 56 cities

5.3 Online adaptive path planning in wind environment

5.3.1 Footprint center coordinates estimation

Given a path, the UAS must be able to recognize the footprint center position to accomplish better the spray mission. A real time correction of the offline computed path planning can be used to increases the PPP effectively released on top the plants (so the *D/S* ratio) and knowing the relative position of flux and plants barycenter makes possible to shut off the pump when the footprint is out-of-target following a certain threshold distance. In this section will be analyzed the "proximity check" and the "path correction" mechanism and their influence on the spray system performance.

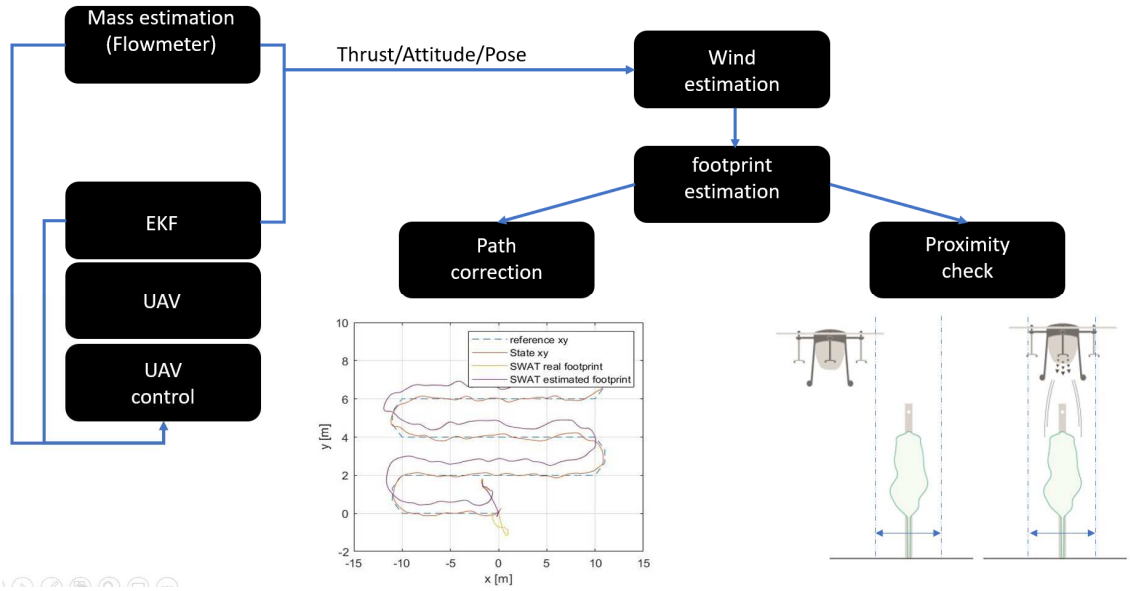


Figure 111 Online adaptive path planning, blocks scheme

For what concerns the path correction on the base of the wind speed we will assume that *the spraying is optimal only if the center of the elliptic section, obtained intersecting the ground plane with the 3D profile of the flux, coincides with the plant position.* The last sentence can be considered a good first approximation, although more specific requirements for the plant-flux interaction exist. The curve used is a quadratic relation between the altitude z of the nozzle and the deflection of the swath axis at the ground footprint, where the last is the Euclidian distance between the center of the spray cone and the vertical axis passing from the nozzle and orthogonal to the ground plane; the curve is obtained as average between inner and outer lateral profile curves discussed in chapter 4. This curve is defined in the same plane of the relative air-UAV velocity vector $\overrightarrow{v_r}$, and the r axis is directed in the same versus as $\overrightarrow{v_r}$.

$$r(W_{rotor}, \|\overrightarrow{v_r}\|) = \boldsymbol{a}(W_{rotor}, \|\overrightarrow{v_r}\|) \cdot z^2 + \boldsymbol{b}(W_{rotor}, \|\overrightarrow{v_r}\|) \cdot z \qquad (5.12)$$

5.3.2 Footprint-plant center proximity check

Once is available the coordinates of the to-be-treated plants, the UAS can easily compute the distance between the estimated footprint center and the target barycenter, as far as EKF positioning and wind/footprint estimation mechanism are reliable. Whatever is the modality in which the UAS approaches the plants and the spray strategy used, this is what happens in real time:

- Computing the minimum Euclidian distance between the footprint center and the target plants positions.
- If the distance is under a certain threshold the flow rate is the one computed, otherwise is zero.

The proximity check can be set in terms of threshold radius. The more the radius is short the more the mechanism is specific, namely unlocks the pump only if the UAS is almost over the target. To evaluate the proximity check in continuous spray was prepared a simulation in which the drone follows 5 20-meters-length rows starting from the half of the first and stopping at half of the last. The pump worked at the constant flow rate of 0,98 L/min. The UAS moved, over the rows, at the constant velocity of 1 m/s at 2 meters altitude. The UAS-crown distance was 50 cm (see 4.1). The path in which the drone overflight the active surface was 80 meters length, therefore the UAS overflight the grapevines in 80 seconds and the maximum sprayable volume was 1,32 L, that was set equal to the Requested volume. In one case the threshold was set to one half of the plant side,

namely 0,45 cm, and all the plants are seen as target plants with a needing of 0,015 L (1,32L/88 plants). The result of these reasonings is in figure 112:

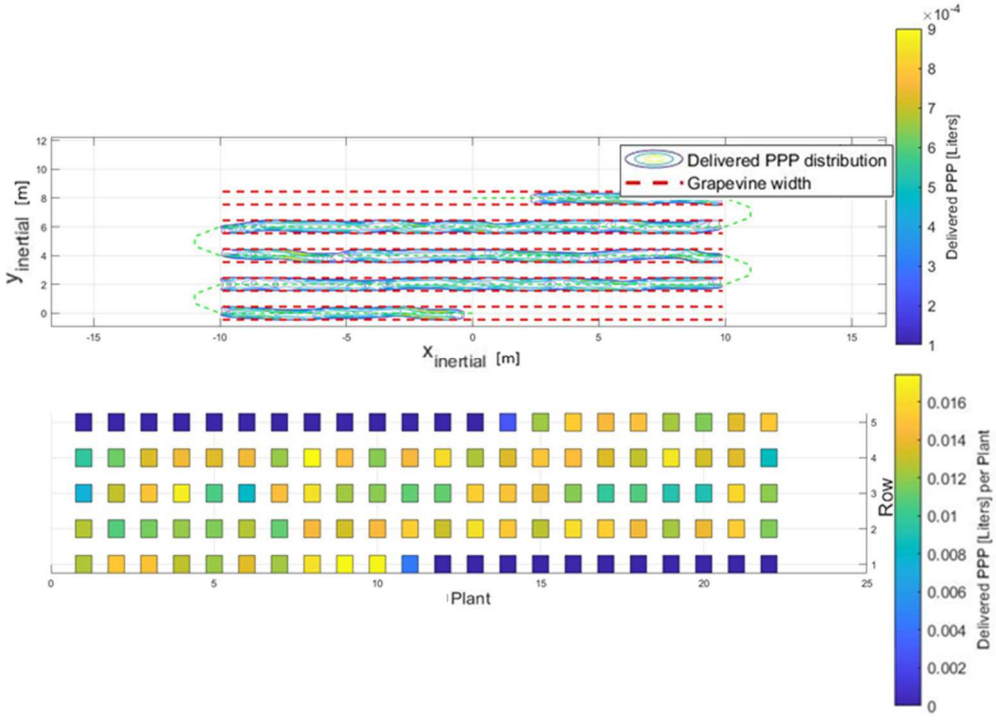


Figure 112 From top to bottom: Delivered PPP distribution in the plane, Delivered PPP per plant, first simulation with Threshold= 45 cm

In the bottom graph plants are enumerated by 1 to 22 for each row. Can be see, as expected a medium release of 0,015 Liters per plant. In this case the total released volume is of 1,21 L and the delivered volume of 1,149 L.

Table 20 Spray quality indexes with proximity threshold at 45 cm, Cross-configuration

Threshold	**Requested Volume**	**Sprayed Volume**	**Delivered Volume**	**Delivered/Requested Volume**	**Delivered/Sprayed Volume**
45 cm	1,32 L	1,21 L	1,149 L	1,149/1,32=*0,87*	1,148/1,21=*0,95*

In a second simulation the threshold was set to 0,9 m:

Table 21 Spray quality indexes with proximity threshold at 90 cm, Cross-configuration

Threshold	**Requested Volume**	**Sprayed Volume**	**Delivered Volume**	**Delivered/Requested Volume**	**Delivered/Sprayed Volume**
90 cm	1,32 L	1,31 L	1,19 L	1,19/1,32=*0,91*	1,19/1,31=*0,91*

It can be deduced that if the characteristic length threshold is reduced the spraying process increases its spray efficiency, so the Delivered/Sprayed ratio, while becomes energetically less efficient, i.e., less liters are released for unit time and so for charge consumption unit, and consequently decreases the Delivered/Requested ratio. As can be seen, increasing the threshold also leads to a more uniformity in the per plant delivered volume. In this case the process is fast and highly efficient because of no wind is introduced.

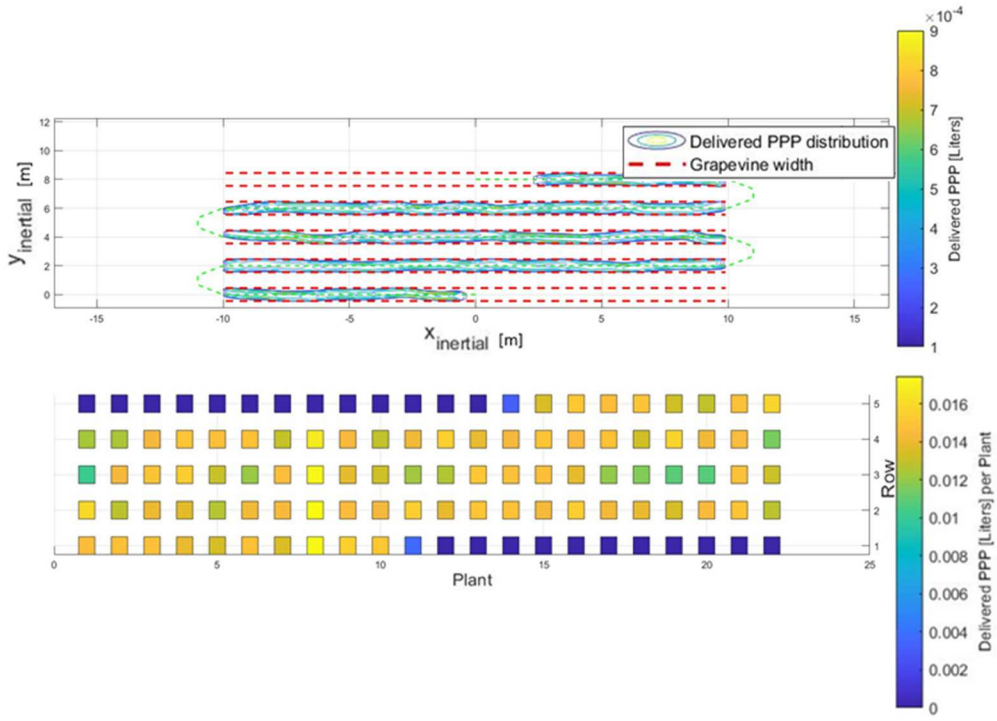


Figure 113 From top to bottom: Delivered PPP distribution in the plane, Delivered PPP per plant, first simulation with Threshold=90 cm

The comparison, in same environmental condition, was done between low and high threshold in Point-To-Point strategy applied to a random path, with the same total requested volume as before (1,32 L) uniformly divided by 22 targets. What can be seen is that the sprayed volume always equals the requested because of the closed loop control of the sprayed PPP amount. What changes with threshold is the time spent to accomplish the mission, indeed lower thresholds constrict the UAS to be in hover on the target for more time than with higher thresholds.

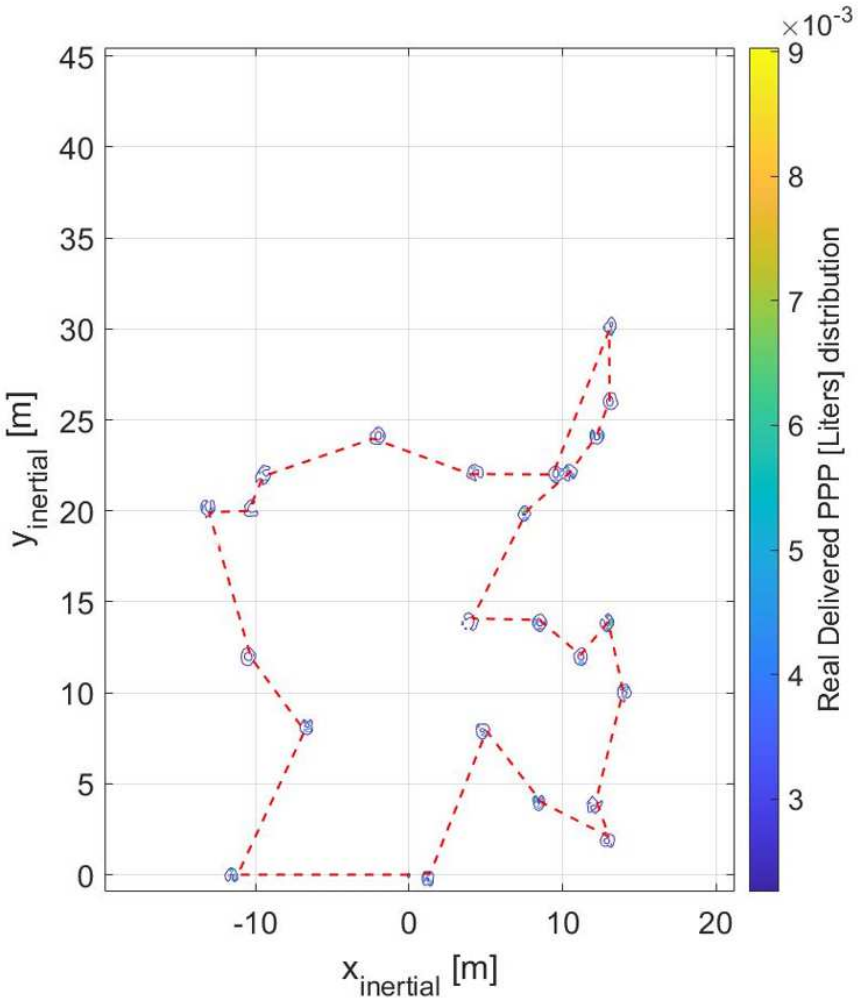


Figure 114 Delivered PPP liters, threshold=45 cm

With a too sensible proximity check for each waypoint the drone overflight the plant and rapidly startups/shutdowns the pump at a nominal flow rate of 1,13 L/min. This happens because in this case the proximity radius was set to 45 centimeters, and the footprint center is continuously deviated by the variable wind intensity and by the UAV control intrinsic errors. With this threshold the Delivered/Sprayed ratio reaches a mean of 0,75, as can be seen in figure 115. In a second simulation the threshold radius, in the same conditions of the first simulation, was doubled. The relaxation of the threshold leads to a Delivered/Requested ratio lower than the 12 % of the low threshold case but reduces also the time needed to complete the mission.

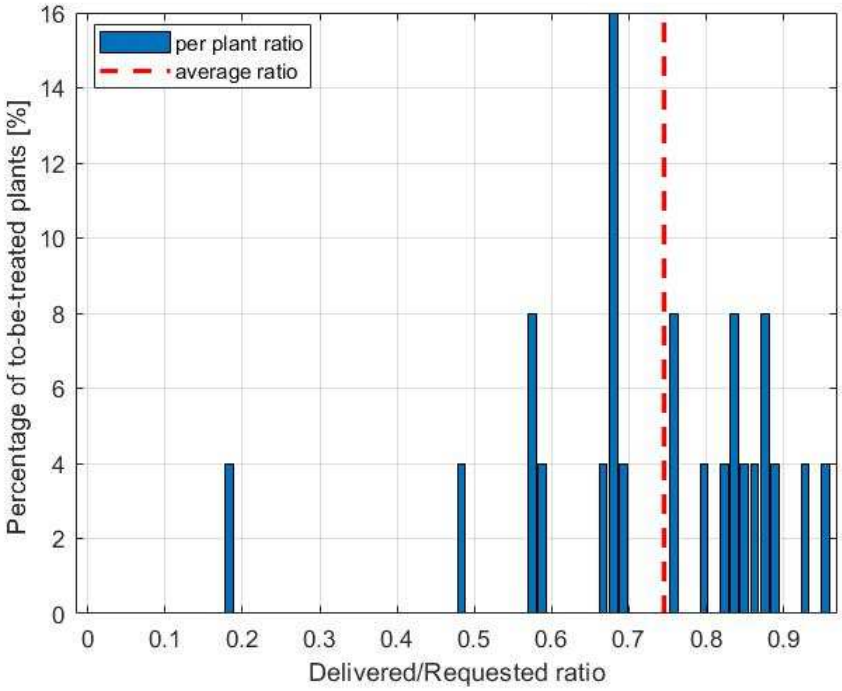


Figure 115 Per plant Delivered/Requested ratios, Proximity radius threshold=45 cm

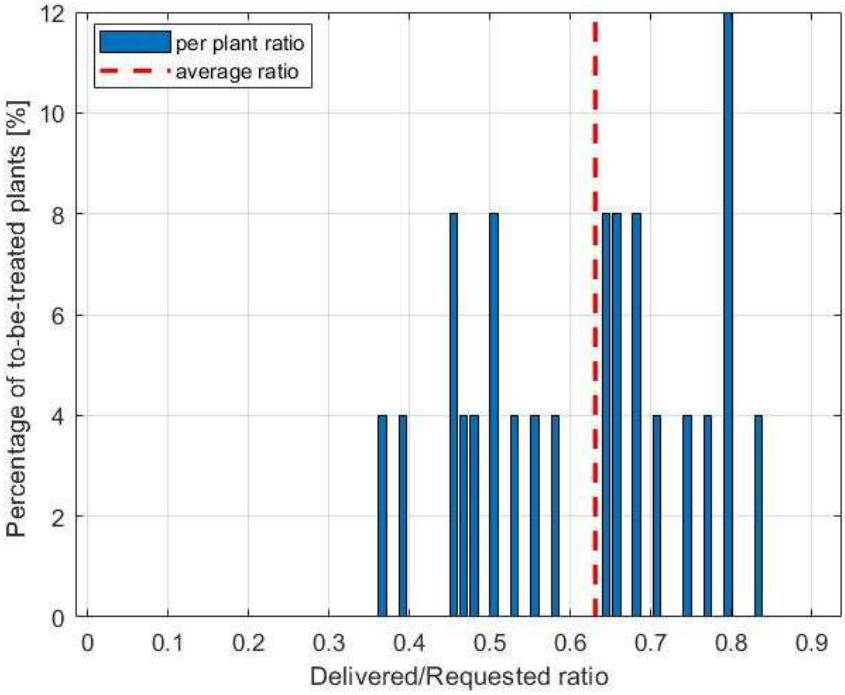


Figure 116 Per plant Delivered/Requested ratios, Proximity radius threshold=90 cm

Table 22 Spray quality indexes with proximity threshold at 45 cm, PtP

Threshold	Requested Volume	Sprayed Volume	Delivered Volume	D/R	D/S	time

45 cm	1,32 L	1,32 L	0,99 L	0,75	0,75	210 s

In a second simulation the threshold was set to 0,9 m:

Table 23 Spray quality indexes with proximity threshold at 90 cm, PtP

Threshold	Requested Volume	Sprayed Volume	Delivered Volume	D/R	D/S	time
90 cm	1,32 L	1,32 L	0,83 L	0,63	0,63	194 s

Decreasing too much the proximity threshold could be not suitable for the continuous spraying mode, because plants could receive less volume then needed, because if the footprint exits the active zone for a time longer than that needed to overflight a plant then the plant will remain unsprayed. A way to circumvent the problem is to command a higher PPP demand than the really requested or to slow the drone more than theoretically needed. However, if the prescription map is uniform and the aim is to spray all the plants in an intensive manner the proximity check should be shut off, since in those cases is more important to ensure a large-scale PPP coverage and less important the single plant spray precision. Another idea could be to return on the unsprayed plant controlling the delivered volume per plant, but this implies to station over the crop and substantially becomes a PtP strategy. Both in continuous and PtP strategies the flow rate command could become frenetic, especially if the proximity check is low. If the switch-off and the successive startup of the pump is commanded to happen in less than 1 second the response decreases its capacity to track the reference signal, producing artifacts as in figure 117:

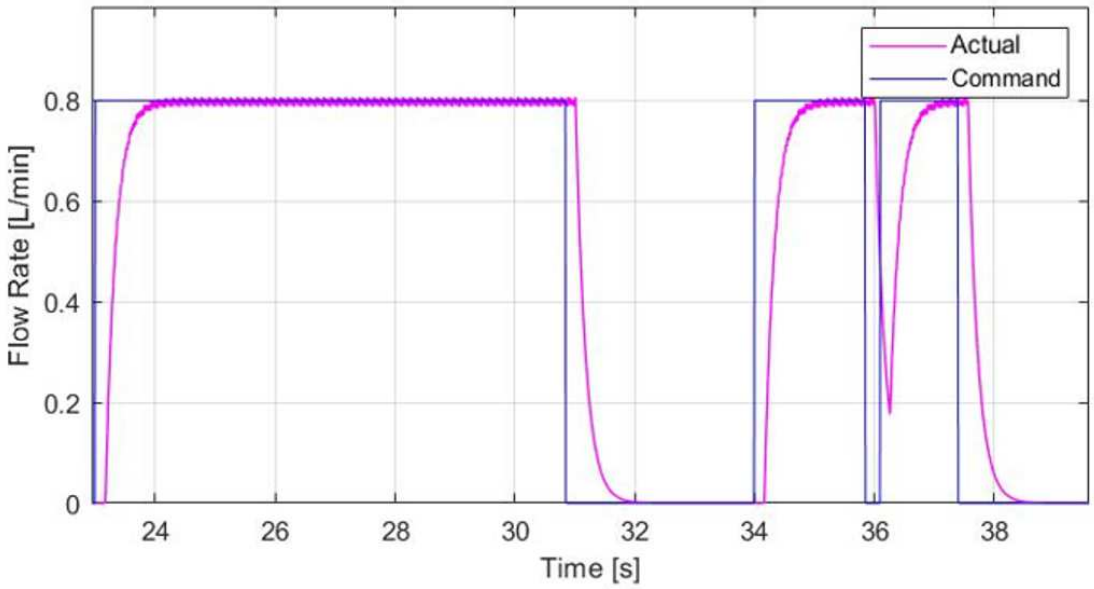


Figure 117 PtP strategy, too fast flow rate command variation

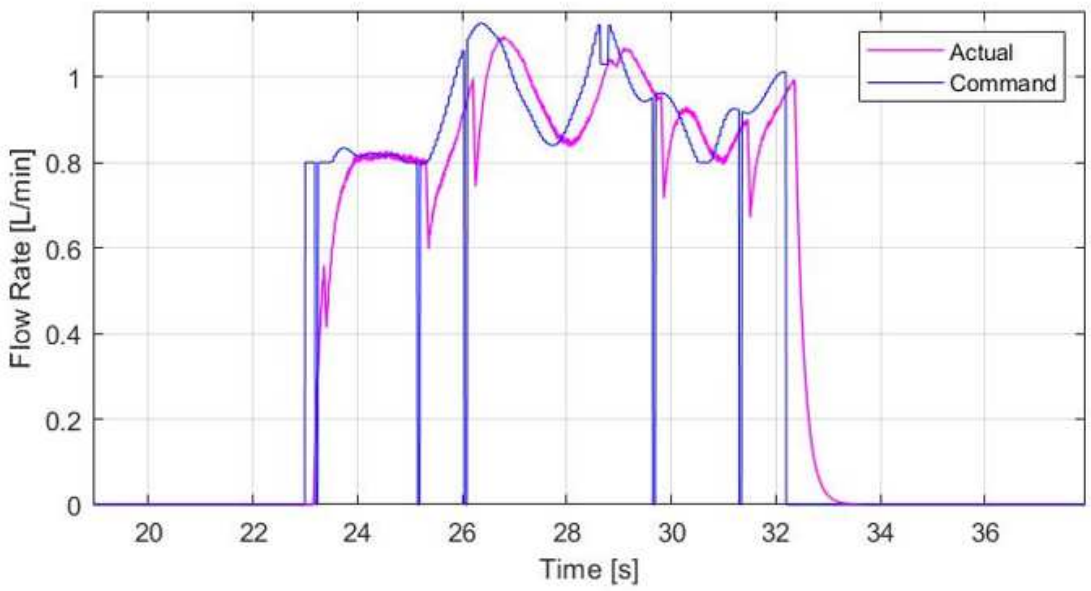


Figure 118 Continuous strategy, too fast flow rate command variation

5.3.3 Path planning correction and wind estimation

5.3.3.1 Swath deflection compensator

The air-UAV relative velocity in the inertial frame is the vectorial sum of the wind inertial vector and the opposite of the UAV velocity inertial vector:

$$\overrightarrow{v_r} = -\overrightarrow{v_{UAV}} + \overrightarrow{v_{wind}} \quad (5.13)$$

The following assumptions are done, some of those come from chapter 4:

1. Inertial properties of flux droplets are neglected.
2. The vertical component of the wind vector is considered null.
3. The nozzle is considered positioned in the barycenter of the UAV.
4. The hollow-cone nozzle has a perfectly axial-symmetric flux in stationary condition with no wind.
5. The effect of the air velocity on the frontal section flux axis (axis on the plane orthogonal to the air-UAV relative velocity) is negligible.
6. The upward nozzle pressure does not modify the deflection/altitude (*r/z*) relationship
7. The wind acts as disturbance for the spray but no drag is exerted to the drone (drag force will be introduced in the next simulations)

The first assumptions lead to consider the drift instantaneous and neglect any adjunctive deflections due to the acceleration of the UAV, since the flux, given an air-UAV relative velocity, solidly follows the drone. Furthermore, the same assumption, allows to neglect the contribution of the drone velocity for computing the path correction factors in the algorithm since the flux deformation due to the static relationship with drone velocity $\overrightarrow{v_{UAV}}$ is always pointing in the same direction of motion, but shifted of a certain quantity and this means that the flux always will reach the waypoints reached by the drone, but with a delay. For this reason, the algorithm will only focus on the deflection caused by the wind, that is a good approximation for small UAV accelerations. The second assumption literally reduces the problem to a 2D problem. This assumption simplifies the computations and can be consider reliable because of the small effect that vertical component of the relative velocity can have on the spray and the, typically, low vertical wind magnitude at low altitudes. The third assumption is a bit weak because is quite distant by the conditions for which the experimental curve was tested, since there are important relations between the spray profile and the turbulence introduced by the rotating propellers and

in the wind tunnel nozzle there were many nozzles, and all were mounted near the respective rotor. However, this assumption relaxes the computation leaving independent the flux deflection by the yaw movement of the UAV and leaves to further studies the analysis of the nozzle positioning effects on the path correction mechanism. The fourth and fifth assumptions ensures that the deflection effect of the air relative velocity is independent by the yaw orientation of the nozzle itself. The deflection r is positive in the direction and versus of $\overrightarrow{v_r}$ in inertial frame, so the x and y inertial components of the deflections are:

$$\begin{cases} r_{x\,In} = r \cdot \cos(\gamma) \\ r_{y\,In} = r \cdot sin(\gamma) \end{cases} \quad (5.14)$$

where γ is the the yaw angle in inertial frame of $\overrightarrow{v_r}$. Then for the consideration done until now:

$$\begin{cases} \overrightarrow{v_r} \cong (v_{wind,x}, v_{wind,y}, 0) \\ \gamma = atan2(v_{wind,y}, v_{wind,x}) \end{cases} \quad (5.15)$$

Once $r_{x\,In}$ and $r_{y\,In}$ are computed, in real time, the path planning reference signal can be updated and corrected with those quantities:

$$\begin{cases} {x^*}_{ref} = x_{ref} - r_{x\,In} \\ {y^*}_{ref} = y_{ref} - r_{y\,In} \end{cases} \quad (5.16)$$

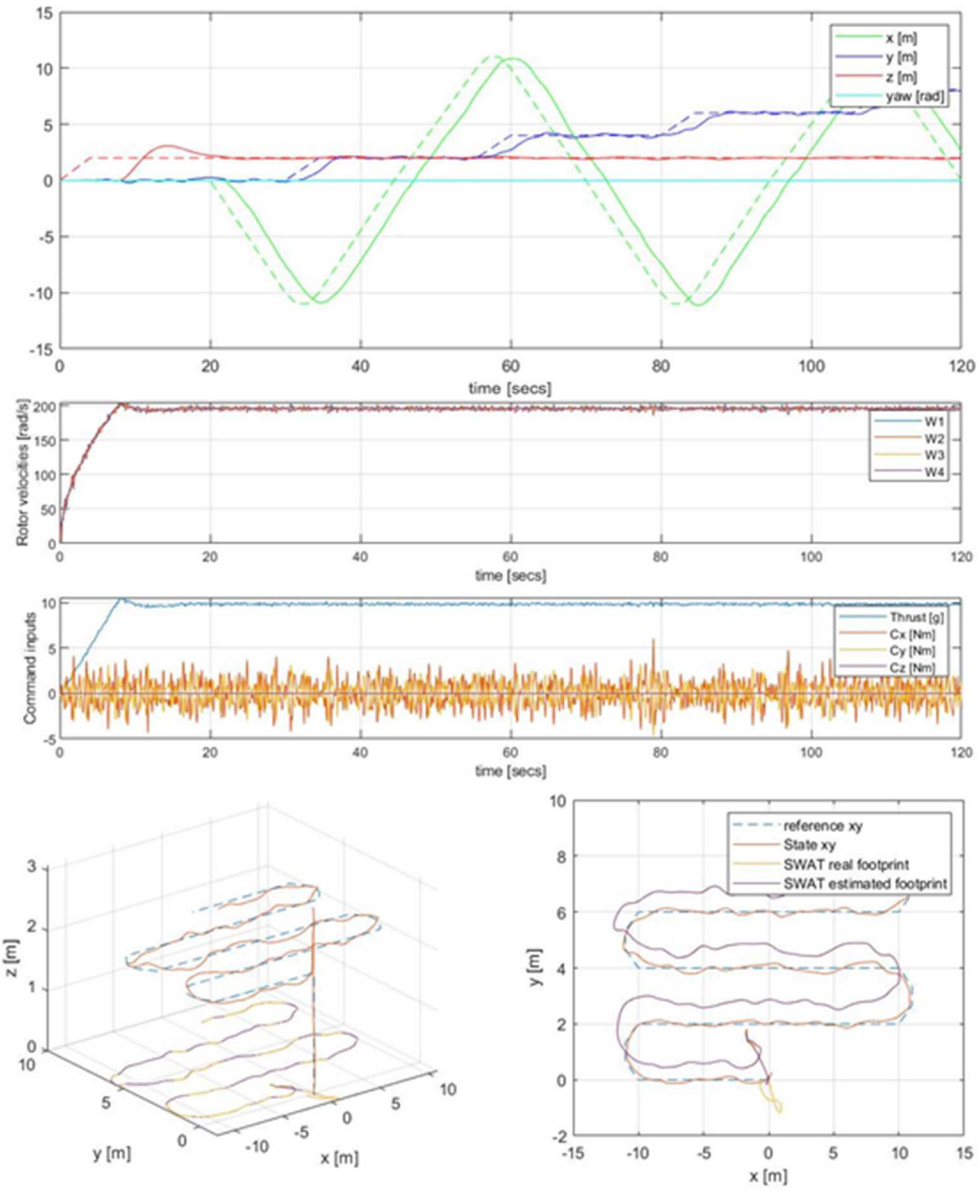


Figure 119 state tracking without path correction

These graphs present the comparison between the control without the real time path planning correction and with it when a wind directed at 45° in direction N-O (considering y aligned with north) with a constant magnitude of 8,48 m/s is applied:

$$\overrightarrow{v_{wind}} = (-6,6,0)\ m/s \qquad (5.17)$$

Is worth to notice that, while the algorithm supposes zero UAV velocity in computing the correction, in the graph the footprint centers orange line is the ground footprint deflection considering a closely complete relative velocity:

$$\overrightarrow{v_r} \cong \left(-\dot{x} + v_{wind,x}\ , -\dot{y} + \ v_{wind,y}\ , 0\right) \qquad (5.18)$$

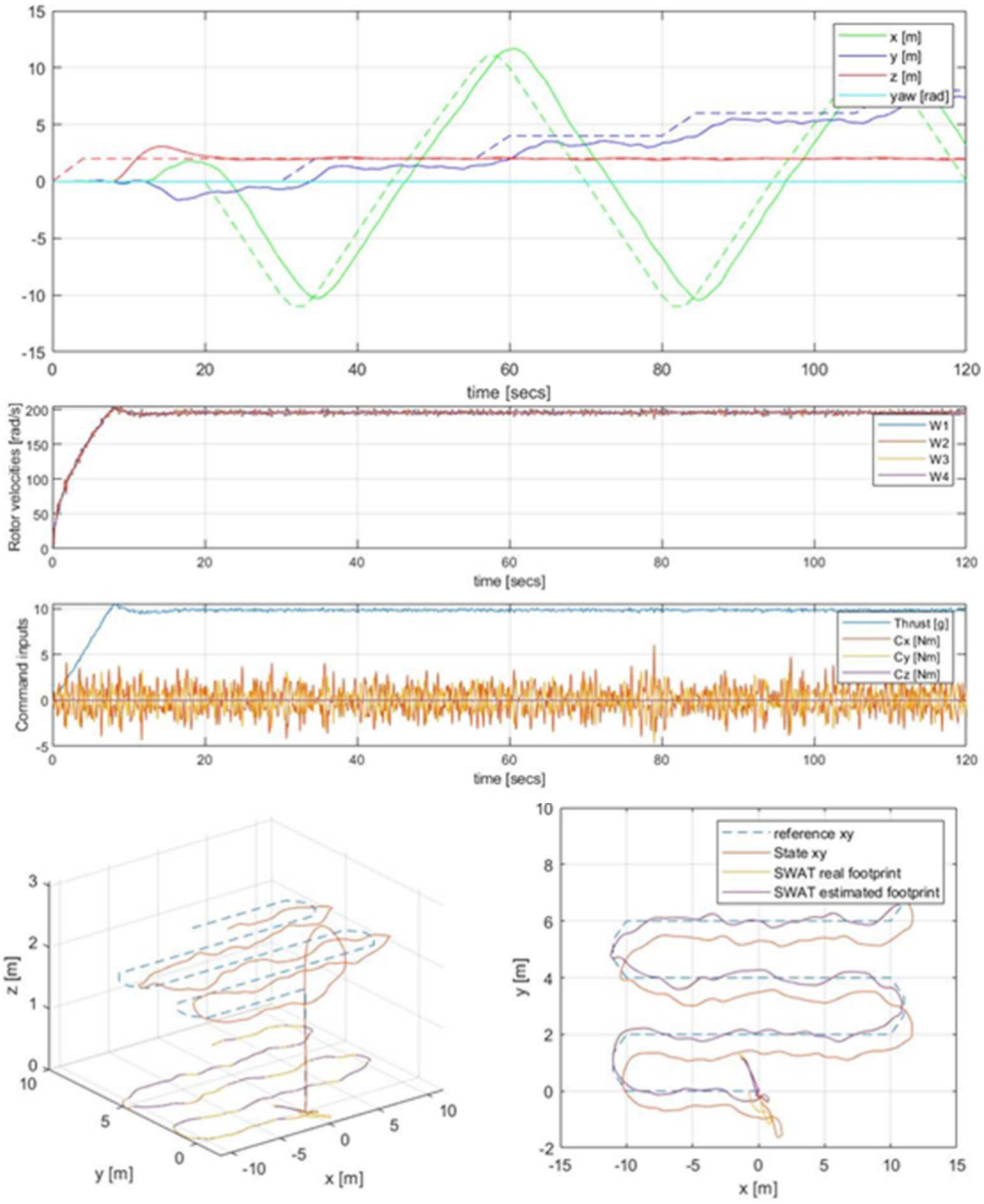


Figure 120 state tracking with path correction

5.3.3.2 Wind model and estimation

The proposed path correction algorithm is now used downstream a wind direction and magnitude estimation. In agricultural field wind measurement is typically executed by means of fixed meteorological stations equipped with anemometers. In slowly spatial wind variation, the station could be enough to provide an estimate of the local wind speed and direction in each point of the field. This is closely never the case if are present obstacles or wind spatial turbulences and the requested accuracy in wind estimation is within the m/s in magnitude and ten degrees in direction, as in our case study. Many efforts have been focused on localized wind estimation using UAV. Many of them consist in onboard sensor that directly measure air-speed, while indirect measurement strategies are present in literature, too [10].

1. Flow sensors

Typically, a flow sensor estimates air speed measuring the difference between internal static pressure and external dynamic pressure. The typical flow sensor used in large aerial vehicles and small fixed-wing UAV is the Pitot tube. The working principle of the Pitot tube is based on the pressure drop, due to air speed, between internal chamber and the external environment. The air

speed velocity is related to the pressure drop exploiting the Bernoulli equation. Pitot tube is considered light and cost-effective sensor, widely used in aerial vehicles. However, it measures pressure and air speed in the direction in which the tube is pointing. This means that a real time 3D air speed estimation would requires several sensors mounted, thereby increasing structure complexity and body weight. Furthermore, multirotor UAVs rotors blades generates constantly turbulences which could lead to a total misbehavior of the sensor unless it is mounted in specific optimal points which minimize the *rotor wash.* However, finding those points is a difficult task which requires CFD analysis and many experimentations to be effective.

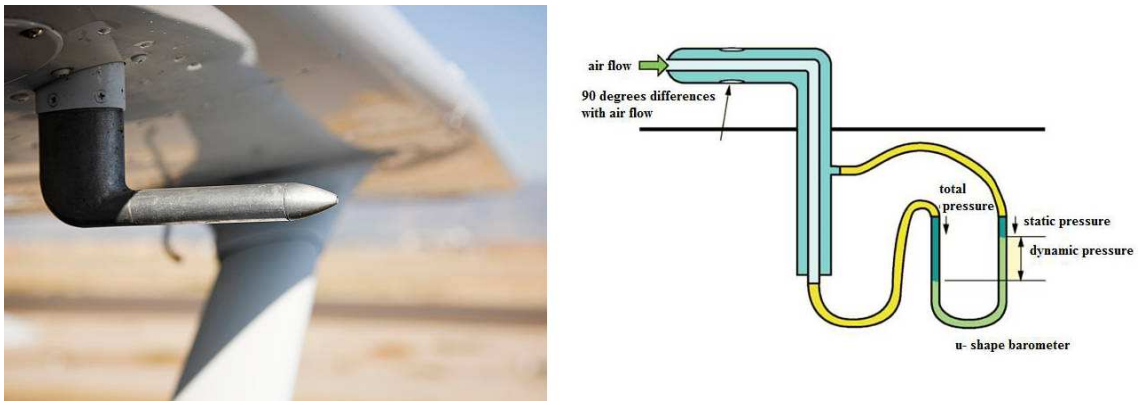


Figure 121 Pitot Tube, images from [37] and [38], modified

2. Ultrasonic anemometers

Two pairs of ultrasonic Transmitter-Receiver are mounted one facing the other. Is used a Time Of Flight approach consisting in the evaluation of the travel velocities of the wave produced by TX1 and sensed by RX2 and the second wave produced by TX2 and sensed by RX1. In static air condition the velocities are one equal the other. If the air speed is non-zero waves travelling in direction opposite to the air speed are slowed and the other are carried. Computing this difference, the air speed is estimated.

Figure 122 TOF based Ultrasonic Anemometer, image from [39](left) and [40](right) , modified

Another typology of ultrasonic anemometer is based on the acoustic resonance. They have a transmitter and two receivers mounted on the same plate. This plate is collocated on top a reflection plate at a known distance in order to create a cavity. The transmitter emits waves at the eigen frequency of the cavity, and an acoustic resonance is produced which increases the wave amplitude in time making it sensible enough when a good Signal to Noise ratio is reached. The phase shift between the emitted wave and the received one is proportional to the air velocity in the cavity and does not depend on airs temperature and pressure. The composition of two different sensors, allows to reconstruct the air speed vector components.

Figure 123 Acoustic resonance Ultrasonic Anemometer: DM50 FT7 series from FT Technology [41](Left), image taken from [42](Right), modified

As the flow sensors this anemometer are widely used as cost-effective solution for local wind estimation, but their size and structure increase the multirotor weight and complexity and suffers of rotor wash. Differently by flow sensors these sensors can measure magnitude and direction simultaneously. The main limit of the ultrasonic anemometers is their sensibility to rain drop and pressure/temperature changes because change the speed of sound. Some studies proposes to auto-calibrate sensors matching data with those one coming from a meteorological tower[10].

3. IMU and GPS based air speed estimation, Tilt angle approach

This kind of wind estimation is based only on data provided by the on-board electronics of the UAV. Trusting on the dynamical model of the quadrotor and on the control strategy a relation between the quadrotor state and the air speed can be deduced, from which wind magnitude and direction can be extracted. This method, compared to the other, has no need of external sensors. The on-board state estimation can provide enough information to estimate the wind speed. This method, as ultrasonic anemometers, can provides wind magnitude and direction simultaneously. However, 3D ultrasonic anemometers can measure the vertical component of the wind speed, while the IMU based tilt angle approach does not. The tilt angle approach can provide wind measurements only in the hovering condition; however, the problem could be circumvented through a correction phase based on the accelerometer measurements. Using no external sensors completely surpasses the accurate and laborious positioning phase needed by direct air flow sensors, while mass and structure complexity kept the same. As mentioned in [10] many studies have tried the experimentation of the Tilt angle approach, obtaining results and accuracies in the order of magnitude of our interest. For example [11] AirRobot AR100-B quadrotor in a study for gas source localization and mapping found an RMSE of 0.6ms^{-1} and 14,2° and [43] use a DJI 550 hexacopter with Pix-hawk 3DR in wind and meteorological study, obtaining a wind magnitude RMSE of 0.71ms^{-1} and 14,5° in direction. These studies validate the Tilt angle approach logging the quadrotor estimation while is in hovering near a meteorological tower provided by anemometer. The following wind estimation algorithm is based on the work in [10] at which a correction phase is added to improve the results, as suggested in[44]. Neglecting the vertical wind component and rotors mounting tolerances, the only two reasons for which the quadrotor is tilted are:

- Quadrotor acceleration in *x,y* plane.
- Air-UAV velocity induced drag force compensation.

The main problem, and so the main limitation of the following algorithm, is that while the quadrotor acceleration and the tilt angle are linked by a deterministic mechanical relationship, Air-UAV velocity drag force compensation produces a constant tilt angle at steady state, but its transient adjustment depends on angles control and positioning estimator responsiveness. However, the assumption of "instantaneous" drag force compensation produced enough good results to be accepted, although some regression techniques could be used to describe a posteriori the transient phase. The above limitation is evident when the quadrotor estimates the wind in motion operations, as will be shown later. The horizontal thrust component is the vectorial sum of the drag force compensation $\overline{F_D}$ and x-y acceleration $\overline{F_{acc}}$.

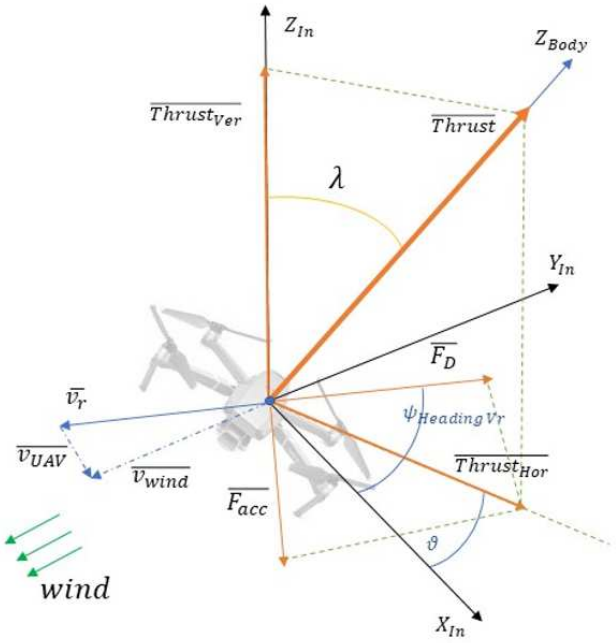


Figure 124 vectorial relationships of the wind estimation, image inspired and modified by [10]

The vertical thrust component magnitude, when the takeoff is done, is equal to:

$$\left|\overline{Thrust_{Ver}}\right| = m(g + \ddot{z_{In}}) \tag{5.19}$$

And the horizontal:

$$\left|\overline{Thrust_{Hor}}\right| = m(g + \ddot{z_{In}})\tan(\lambda) \tag{5.20}$$

The tilt angle λ is estimated inverting the relation of the scalar product between z_{In} and z_{body}:

$$z_{body}{}^{In} = R_{Ib}[0\ 0\ 1]^T = \begin{bmatrix} c(\phi)\,c(\gamma)\,c(\varphi) + s(\phi)\,s(\varphi) \\ c(\phi)\,s(\gamma)\,s(\varphi) - s(\phi)\,c(\varphi) \\ c(\phi)\,c(\gamma) \end{bmatrix} \tag{5.21}$$

$$z_{body}{}^{In^T} \cdot z_{In} = [0\ 0\ 1]^T R_{Ib}{}^T [0\ 0\ 1] = \cos(\phi)\cos(\gamma) = \cos(\lambda) \tag{5.22}$$

$$|\lambda| = |\text{acos}\,(\cos(\phi)\cos(\gamma))| \tag{5.23}$$

While the heading angle ϑ of $\overline{Thrust_{Hor}}$ is computed as the arctangent of the $z_{body}{}^{In}$ components on the inertial horizontal plane:

$$\vartheta = atan2\left(\frac{c(\phi)\,s(\gamma)\,s(\varphi) - s(\phi)\,c(\varphi)}{c(\phi)\,c(\gamma)\,c(\varphi) + s(\phi)\,s(\varphi)}\right) \tag{5.24}$$

At this point angle and magnitude of $\overline{Thrust_{Hor}}$ are known and the $\overline{F_D}$ vector can be derived from the vectorial relationship:

$$\overline{F_D} = -\overline{F_{acc}} + \overline{Thrust_{Hor}} \quad (5.25)$$

where $\overline{F_{acc}}$ can be directly measured rotating the accelerometer measurements in the inertial reference frame plane, multiplying resulted vector by the UAV mass. Results:

$$\overline{F_D} = \left[-m\ddot{x}_{In} + m(g + \ddot{z}_{In})\tan(\lambda)\cos(\vartheta) \;,\quad -m\ddot{y}_{In} + m(g + \ddot{z}_{In})\tan(\lambda)\sin(\vartheta) \;,\quad 0\right] \quad (5.26)$$

$$\psi_{Heading\,Vr} = atan2\left(\ddot{y}_{In} + m(g + \ddot{z}_{In})\tan(\lambda)\sin(\vartheta),\;\; -\ddot{x}_{In} + m(g + \ddot{z}_{In})\tan(\lambda)\cos(\vartheta)\right) \quad (5.27)$$

At the best the control system can do, the $\overline{F_D}$ force is equal and opposite to the drag force exerted by air velocity relative the UAV.

$$\left|\overline{F_D}\right| = \frac{1}{2}\rho \cdot C_D(\lambda) \cdot A_{proj}(\lambda) \cdot |\overline{v_r}|^2 \quad (5.28)$$

where:

$$\begin{cases} \rho = 1{,}225\ {}^{Kg}/_{m^3} & air\ density\ in\ standard\ condition \\ C_D(\lambda) & Air\ flow\ drag\ coefficient \\ A_{proj}(\lambda) & UAV\ exposed\ surface\ projection \\ \overline{v_r} & air - UAV\ relative\ velocity \end{cases} \quad (5.29)$$

So $\overline{v_r}$ is estimated in magnitude and angle:

$$\begin{cases} |\overline{v_r}| = \sqrt{\dfrac{2\left|\overline{F_D}\right|}{\rho \cdot C_D(\lambda) \cdot A_{proj}(\lambda)}} \\ \angle\overline{v_r} = \psi_{Heading\,Vr} + \pi \end{cases} \quad (5.30)$$

Knowing the horizontal UAV inertial velocity outputted by the Kalman filter the wind velocity vector is finally estimated:

$$\overrightarrow{v_{wind}} = \overrightarrow{v_{UAV}} + \overrightarrow{v_r} = \left[\sqrt{\frac{2|\overline{F_D}|}{\rho \cdot C_D(\lambda) \cdot A_{proj}(\lambda)}} \cdot \cos(\angle\overrightarrow{v_r}) + \dot{x}_{In},\quad \sqrt{\frac{2|\overline{F_D}|}{\rho \cdot C_D(\lambda) \cdot A_{proj}(\lambda)}} \cdot \sin(\angle\overrightarrow{v_r}) + \dot{y}_{In},\quad 0\right] \quad (5.31)$$

Knowing $\overline{F_D}$ can be computed the portion of the tilt angle engaged to contrast the air drag force, i.e. the angle between the projection of the ${z_{body}}^{In}$ axle on the vertical plane containing $\overline{F_D}$ and the z_{In} axis. It simply results in:

$$\lambda_{F_D} = atan2(|\overline{F_D}|, \left|\overline{Thrust_{Ver}}\right|) \quad (5.32)$$

As suggested in [10] to compute $C_D(\lambda)$ in a reliable way CFD analysis fused with wind tunnel tests are needed. The drag coefficient is derived measuring the drag force exerted by a controlled air speed for different tilt angles and different rotors velocities (to considering the effects of blades induced turbulence). To compute $A_{proj}(\lambda)$ a 3D design of the UAV body could be used to project the exposed area in the vertical plane toward which the quadrotor is facing. However, to give a first estimate of those two crucial parameters the simplified geometry of the UAV, supposed at the beginning of chapter 3, is used to model them. Modeling the quadrotor as a cylinder of radius $l/2$ and height h the tilt angle is used to orient the 3D shape in the space. For small 2D accelerations the corrected tilt angle λ_{F_D} can be confused with the tilt angle λ, so the tilting direction of the cylinder is the same direction of $\overline{F_D}$. The projected surface is limited between the maximum projected surface, namely the base circle surface of the cylinder, and the minimum given by the rectangular surface projected by the semi-lateral surface.

$$\begin{cases} A_{proj}(90°) = \pi\left(\dfrac{l}{2}\right)^2 = A_{proj,max} \\ A_{proj}(0°) = hl = A_{proj,min} \end{cases} \quad (5.33)$$

In between this interval the circle is projected as an ellipse with major semi-axis equal to l and minor semi-axis equal to $l \cdot \sin(\lambda)$, while the lateral semi-surface is approximated a rotated rectangle projection.

$$A_{proj}(\lambda) = hl\left|\cos\left(\lambda_{F_D}\right)\right| + \left|\sin\left(\lambda_{F_D}\right)\right|\pi\left(\frac{l}{2}\right)^2 \quad (5.34)$$

For the drag coefficient function was used the theoretical drag coefficient for Reynolds numbers in the range $(10^4 - 10^6)$ of three extreme cases:

- For $\lambda_{F_D} = 0°$ the lateral surface of a cylinder is exposed at the laminar flux, the $C_D(0°) = 1{,}17$.
- For $\lambda_{F_D} = 90°$ the flat base surface is exposed so $C_D(90°) = 1{,}17$.
- For $\lambda_{F_D} = 45°$ the prism assumes a configuration not presented in the table, so was assimilated to a cuboid tilted of 45°, with $C_D(45°) = 0{,}8$.

Heuristically:

$$C_D\left(\lambda_{F_D}\right) = 0{,}985 + 0{,}185\cos\left(4\,\lambda_{F_D}\right) \quad (5.35)$$

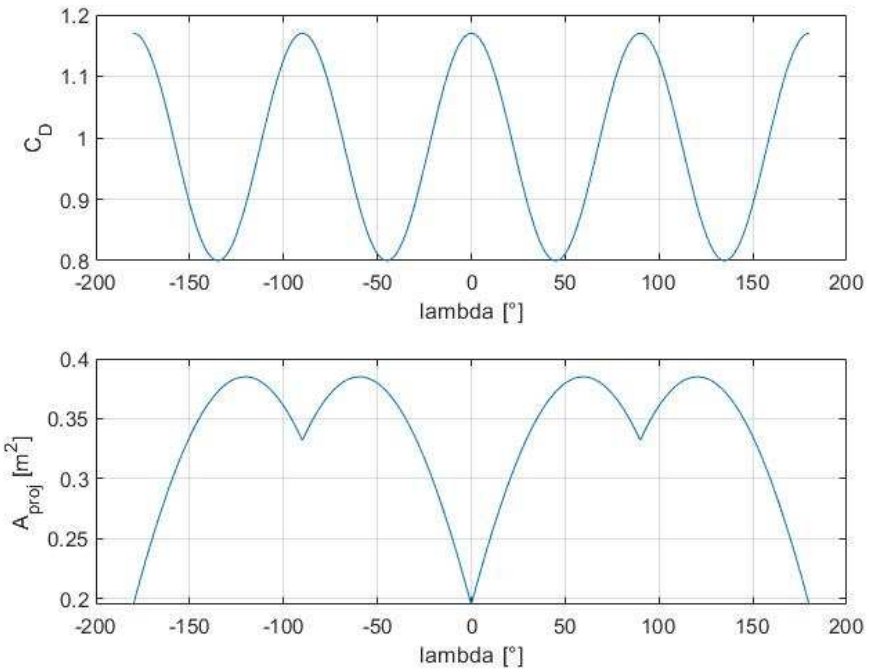


Figure 125 Drag coefficient (top) and projected area (bottom) varying the tilt angle

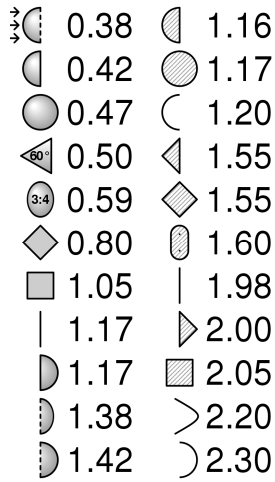


Figure 126 Table of drag coefficients of assorted prisms (right column) and rounded shapes (left column) at Reynolds numbers between 10000 and 1000000 with flow from the left, from Fluid Dynamic Drag by Sighard Hoerner [45], redrawn and reordered by CMG Lee.

To simulate the wind was firstly defined the actual drag force exerted on the quadrotor, coherently with the model used by the algorithm. The drag force vector is in general independent by the tilt direction θ (the direction toward the drone is facing to), but if its angle exactly equals $\theta + k\pi$ the drag coefficient and the projected area are the same computed before. That is because it sees the elliptical surface and the rectangle projected with the tilt angle and the same 3D geometry, so the same drag coefficient, as far the model is good. Leaving the $\theta + k\pi$ direction, the ellipse and the rectangle projected over the plane orthogonal to the air flux field (parallel to the ground) start to be compressed in their major axis direction and with an angle equal to the difference between the drag force vector angle and θ. In the same way increasing the angle the air flux continues to see a rhomboidal geometry with the frontal edge the inclines as more as the angle increases, until reaches a flat rectangle when the drag force is orthogonal to θ.

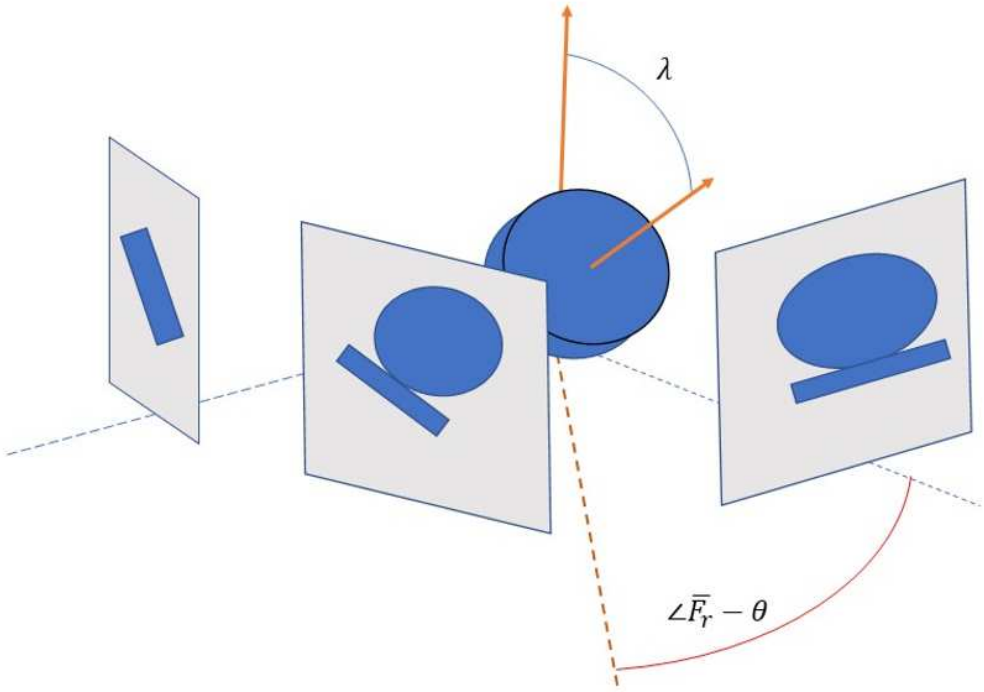


Figure 127 Schematic representation of projected area seen by wind coming from different directions while the tilt angle is non-null

Analytically, the drag force model written in this form:

$$\overline{F_D} = \overline{F_{wind}} - \overline{F_{vUAV}} \quad (5.36)$$

$$\left|\overline{F_D}\right| = \frac{1}{2}\rho \cdot |\overline{v_r}|^2 \cdot \left(C_D(\lambda) \cdot A_{proj}(\lambda) \cdot \cos(\angle\overline{v_r} - \theta)^2 + C_D(0) \cdot A_{proj}(0) \cdot \sin(\angle\overline{v_r} - \theta)^2\right) \quad (5.37)$$

$$\overline{F_D} = \left|\overline{F_D}\right| \cdot [\cos(\angle\overline{v_r}), \sin(\angle\overline{v_r}), 0] \quad (5.38)$$

Is not pretended to have a right model of the drag force, both because there is a lack of experimental data for infer C_D and A_{proj}, and, mainly, because it was used only to validate the wind estimation mechanism. Therefore, the priority was to make $\overline{F_D}$ values and its tilt angle dependencies coherent with those inserted in the wind estimation algorithm model. To compute $\overline{v_r} = \overline{v_{wind}} - \overline{v_{UAV}}$ a probabilistic model of $\overline{v_{wind}}$ in direction and magnitude is proposed. There are in general 2 possible approaches[46]:

- Probabilistic micro-approach: different regions of the UAV could be subjected to different wind direction and magnitude.
- Probabilistic macro-approach: the quadrotor is seen collapsed in its CoG, namely affected by the same wind in the whole structure.

It will be used the micro-approach for simplicity and stability of the simulation environment. As discussed in [10] the wind simulation consists in a random vectors generation, representing the wind velocity at a specific point in the space, composed by a constant term and a variation term representing the gust. The gust can be modeled following the continuous gust model or a discrete gust model. The continuous gust model makes use of suitable shaping functions (Karman, etc..) that output the Power Spectral Density (PSD) of the wind velocity taking in input a white gaussian noise. The discrete gust approach, used in this thesis, models the gust as a step following certain time characteristics that gives randomness to the wind magnitude and direction. The proposed model of the wind will consist of a slowly variable systematic 2D vector summed with a fast variable gust vector:

$$V(t) = V_s(t) + V_g(t) \quad (5.39)$$

In the same way the direction $\beta(t)$ of the wind (the wind vector angle) in the x-y plane is modeled as a random vector in the range -180°, 180°. The three variables were simulated through three random number generators in Simulink, with the following characteristics:

$$\begin{cases} E\{V_s(t)\} = 4\frac{m}{s} \\ \sigma\{V_s(t)\} = 2\frac{m}{s} \\ E\{V_g(t)\} = 0\frac{m}{s} \\ \sigma\{V_g(t)\} = 0{,}33\frac{m}{s} \\ E\{\beta(t)\} = -\frac{\pi}{2} \\ \sigma\{\beta(t)\} = 6{,}5° \end{cases} \quad (5.40)$$

The random number generator of $V_s(t)$ updates with a sampling time of 10 seconds, that one of $V_g(t)$ with a sampling time of 0,1 seconds and a sampling each 6 seconds for $\beta(t)$. Downward these three random generators were inserted a low pass filter to smoothy the instantaneous changing in the values. The cutting frequency was of 0,1 rad/s for $V_s(t)$ and $\beta(t)$ and 1 rad/s for $V_g(t)$.

5.3.3.3 Path correction through wind estimation

Unifying the swath deflection compensator described in 5.3.3.1 and the wind estimator of 5.3.3.2, a set of simulations were prepared to evaluate the global performance of the mechanism in different conditions. The in-motion spray mode and the static spray mode were analyzed, to

highlight different responses that the algorithm could give in the two individuated spray strategies, continuous and point-to-point respectively. To show the stand-alone performance of the algorithm, the input Euler angles were taken from the model in the next simulations, then at the end of the demonstration the wind estimation was based on the Euler angles estimated by the EKF. A first simulation was performed with an ideal constant-magnitude/constant-direction wind of 4 m/s coming from north, so with a wind vector angle of $-\pi/2$, while a second simulation with a wind mean of 4 m/s, a wind standard deviation of 2 m/s, a gust with zero mean and 0,3 m/s standard deviation, a direction with mean 270° and a variance of 40°. In the simulations is implemented also the path correction that tries to keep the footprint position at the plant position (-0,45,0) in the plane. As done in the proximity check section, the continuous spray is tested considering 0,015 L per-plant demand, so a total of 1,32 L in the 88 plants of the 80 meters path, and for the PtP the requested volume was left coincided with the total sprayed, while the proximity check was disabled in both the strategies to highlight the autonomous capacity of the path correction algorithm to deliver liters on top of the plants. The wind estimation starts to be reliable after the take-off because the drag force estimation is based on the equilibrium thrust, reached in hovering, so also the footprint estimation starts after take-off. Is worth noticing a variance in the quadrotor and footprint position in S-W/N-E directions. It could be misleading; indeed, this oscillation is not due to the path correction algorithm but is the UAV control reaction to the small but constant EKF estimation error of pitch, roll and yaw. As can be seen in figure 128 the two errors are quite similar in magnitude, while the pitch is negative and roll is positive, causing an inclination toward S-W shifted clockwise by the yaw error.

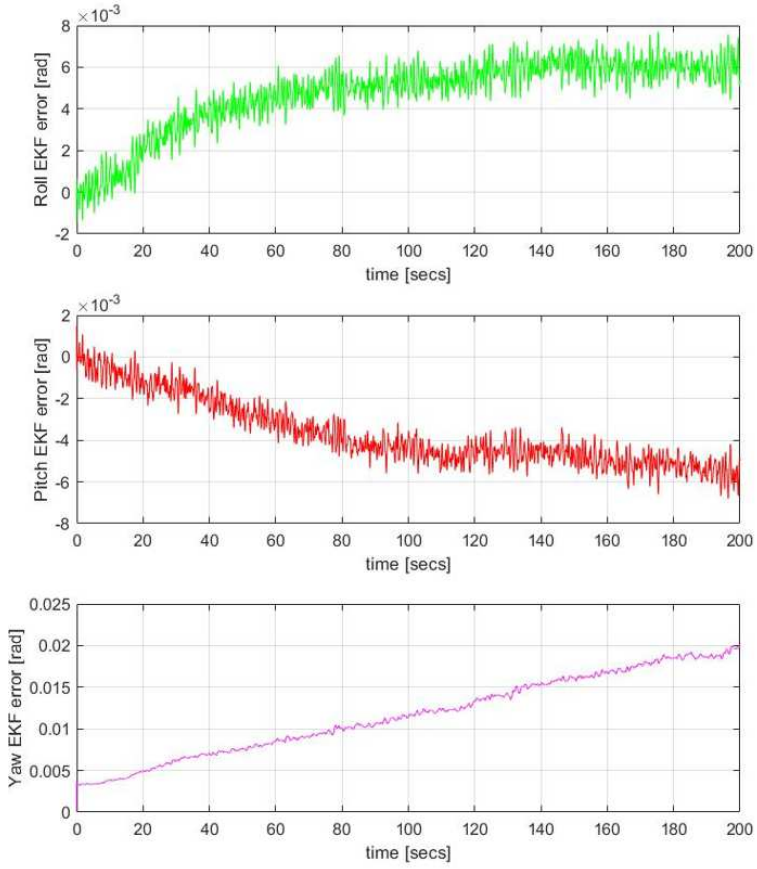


Figure 128 EKF Euler angles estimation and errors

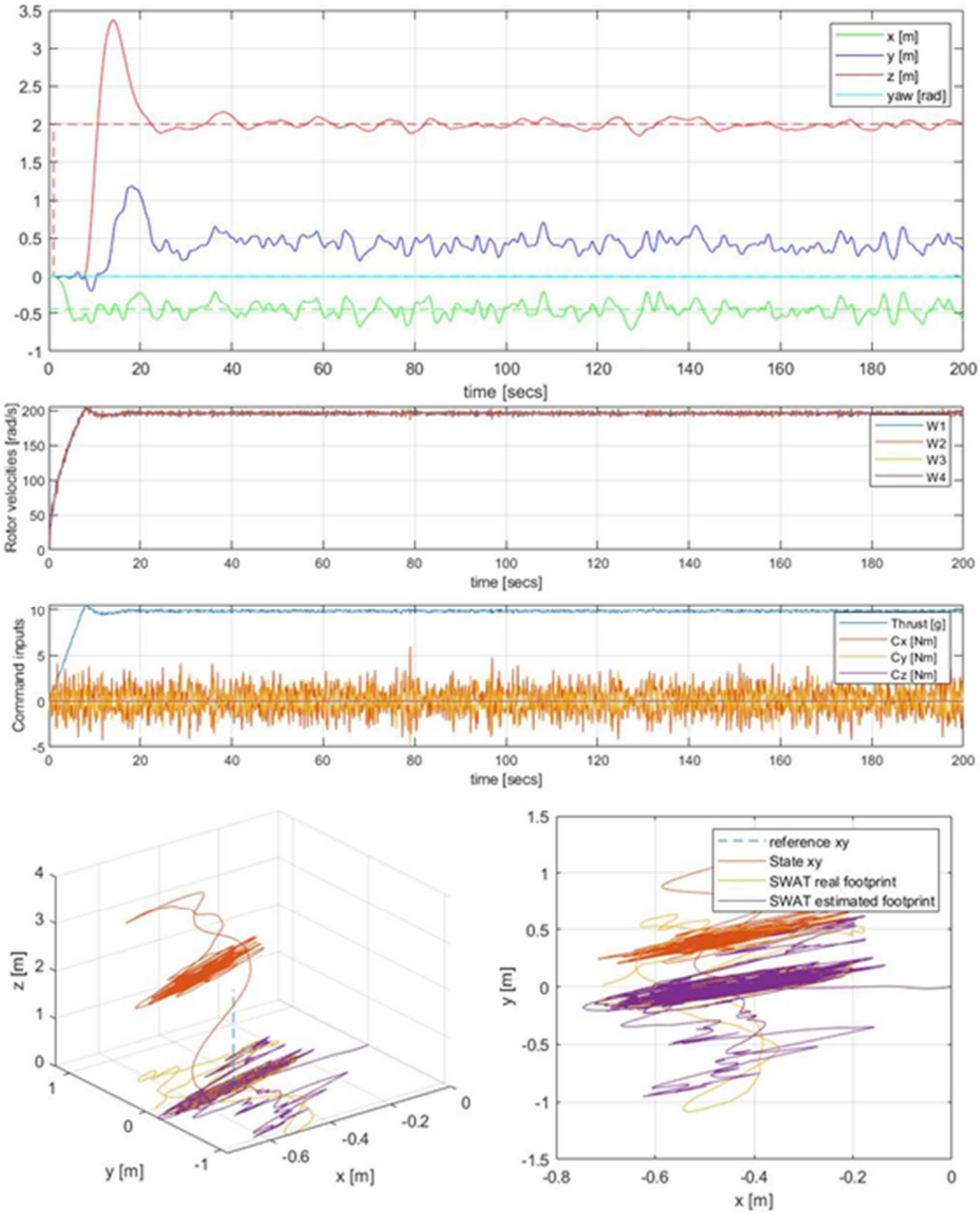


Figure 129 Static case, path correction based on the estimated ideal constant wind, directed from N to S

Coherently with the previous swath analyses the deviation is about of 40 cm; the drone adjusts its position to track the footprint at the plant position. In figure 130, is represented the wind estimation in magnitude and direction, and relative errors:

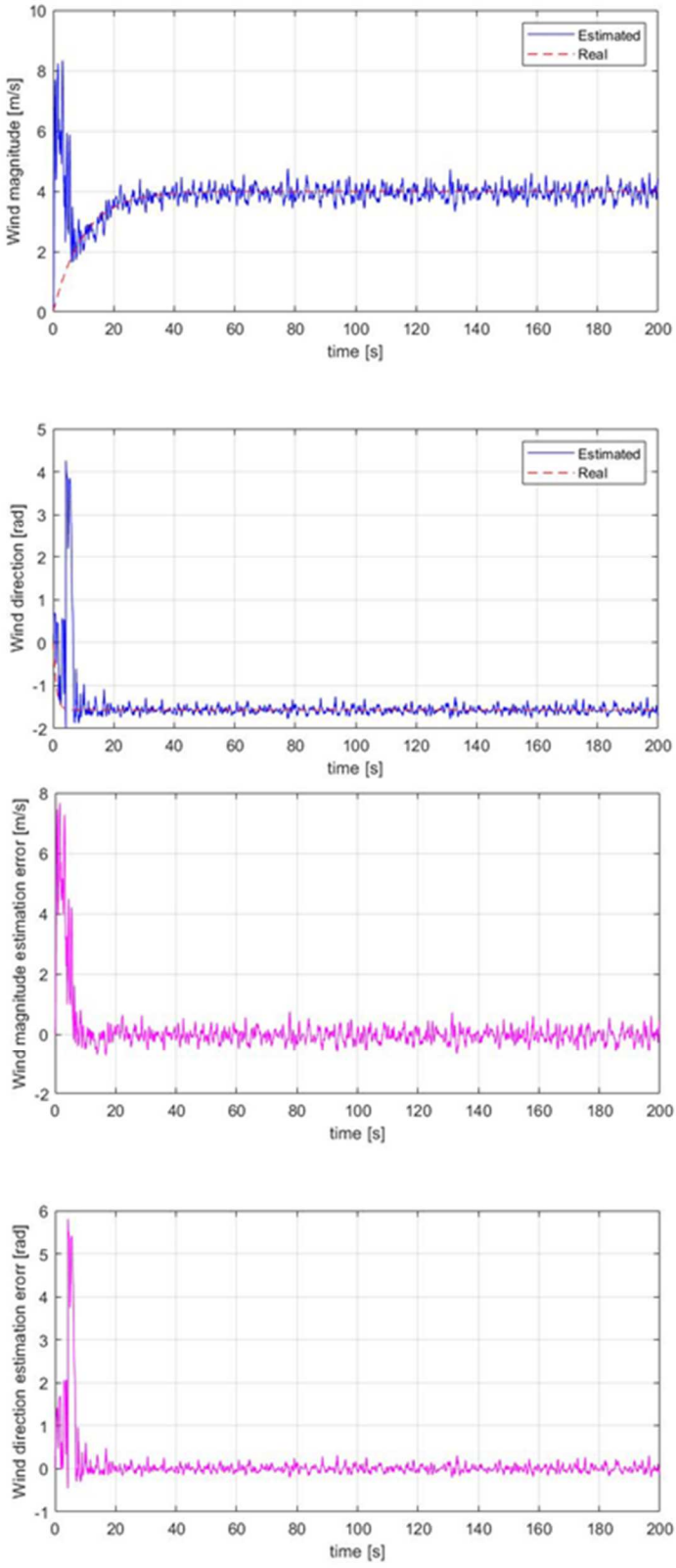


Figure 130 Wind estimation in magnitude and direction, static case, constant wind

Using, instead the wind model presented above the footprint distribution is quite more chaotic, but the path correction algorithm performed as well as in the constant wind case, correctly estimating the wind and adjusting consequently the drone position. The wind direction estimation has some spikes caused by the arctan domain comprised between 360° and -360°, so those spikes are not considered as errors, since they exactly equal 2π.

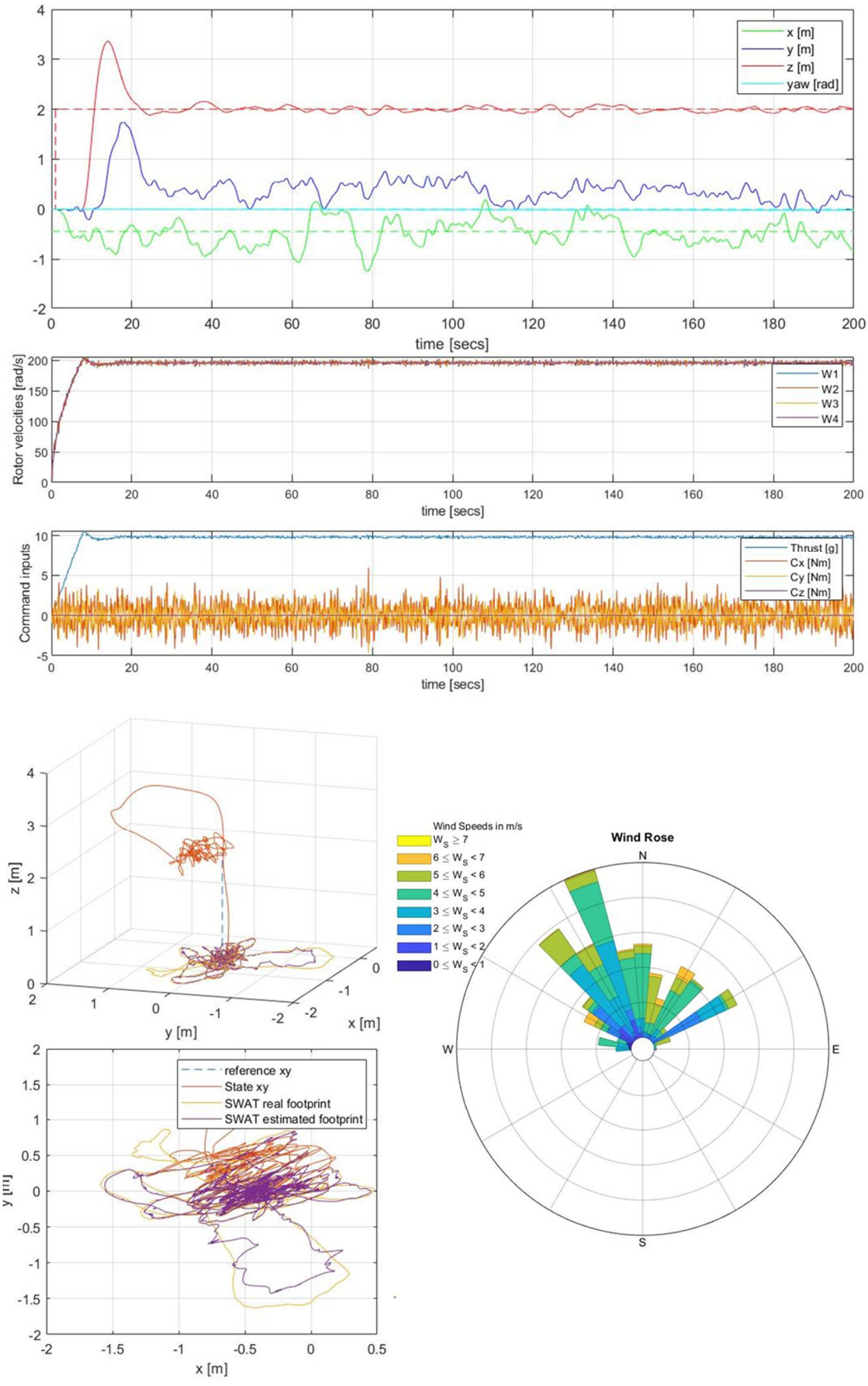


Figure 131 UAV State tracking, static case, variable wind

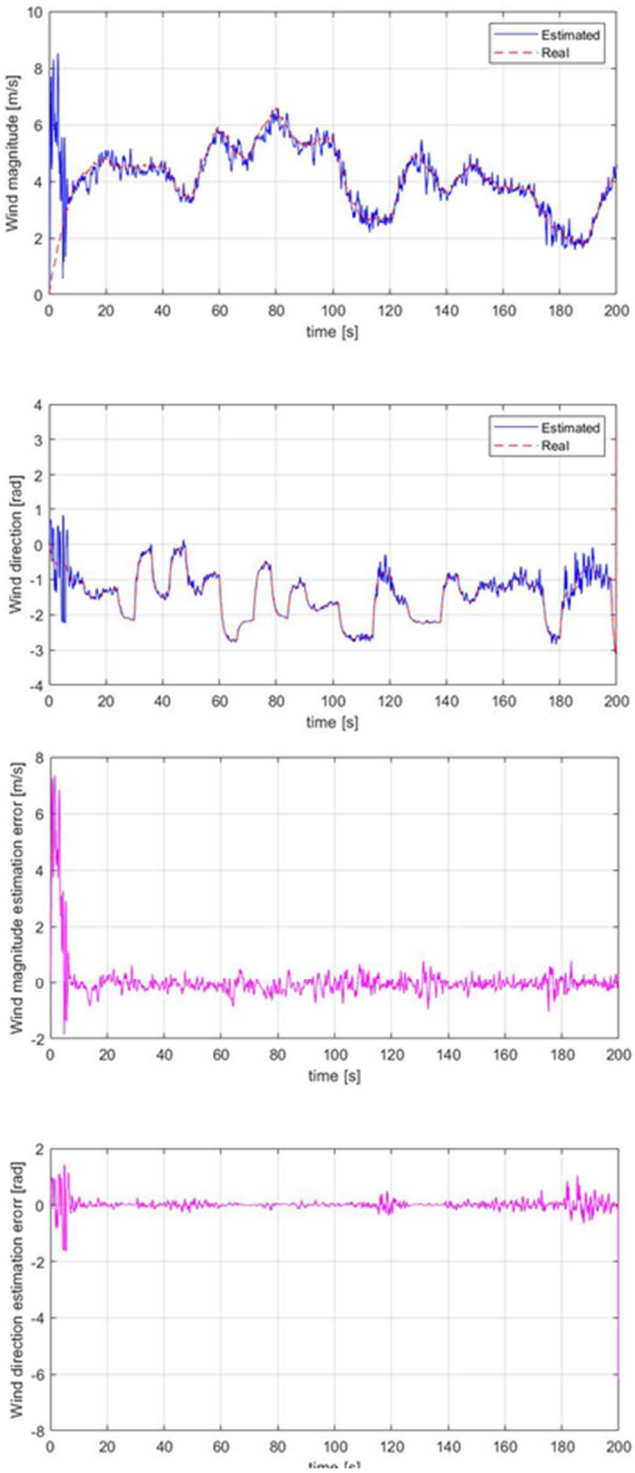


Figure 132 Wind estimation in magnitude and direction, static case, variable wind

In this case starts to emerge the main limit of the algorithm. The fast-varying wind leads the drone to accelerate in few seconds in different directions to keep the position. This situation causes the increasing in number of transients affecting the capacity of the attitude to reach a stationary condition, namely the principal requirements to obtain a precise estimation of the wind. In the following graphs are shown the footprint center position in inertial frame overlapped by the estimated footprint center position, and the footprint centers distribution in real and estimated

case. Can be seen that, despite the increased error in wind estimation, the footprint center estimation tracks the real in a reasonable way, with errors in the order of few centimeters:

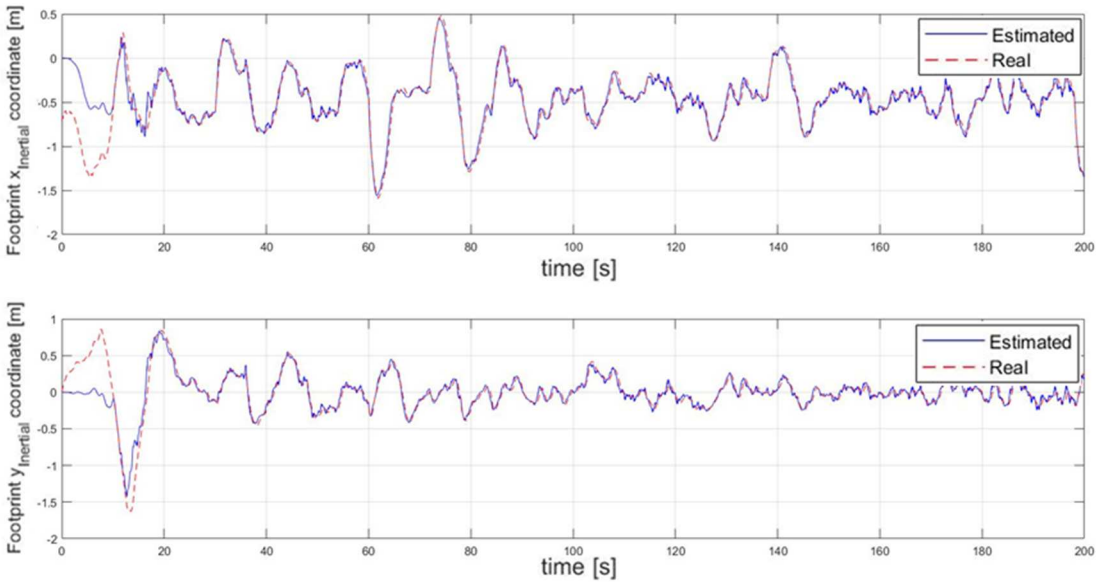


Figure 133 Footprint center coordinates, Estimated vs Real, static case, variable wind

Finally, with the same last wind statistics, the path correction algorithm was also tested when the quadrotor followed the Cross-configuration path planning:

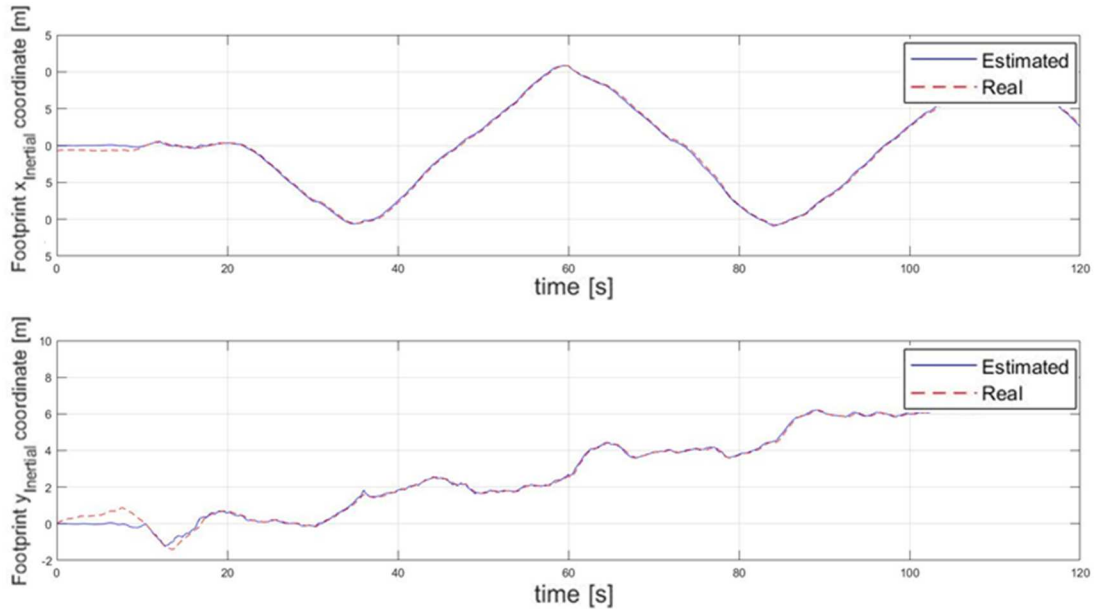


Figure 134 Footprint center coordinates, Estimated vs Real, cross-configuration case, variable wind

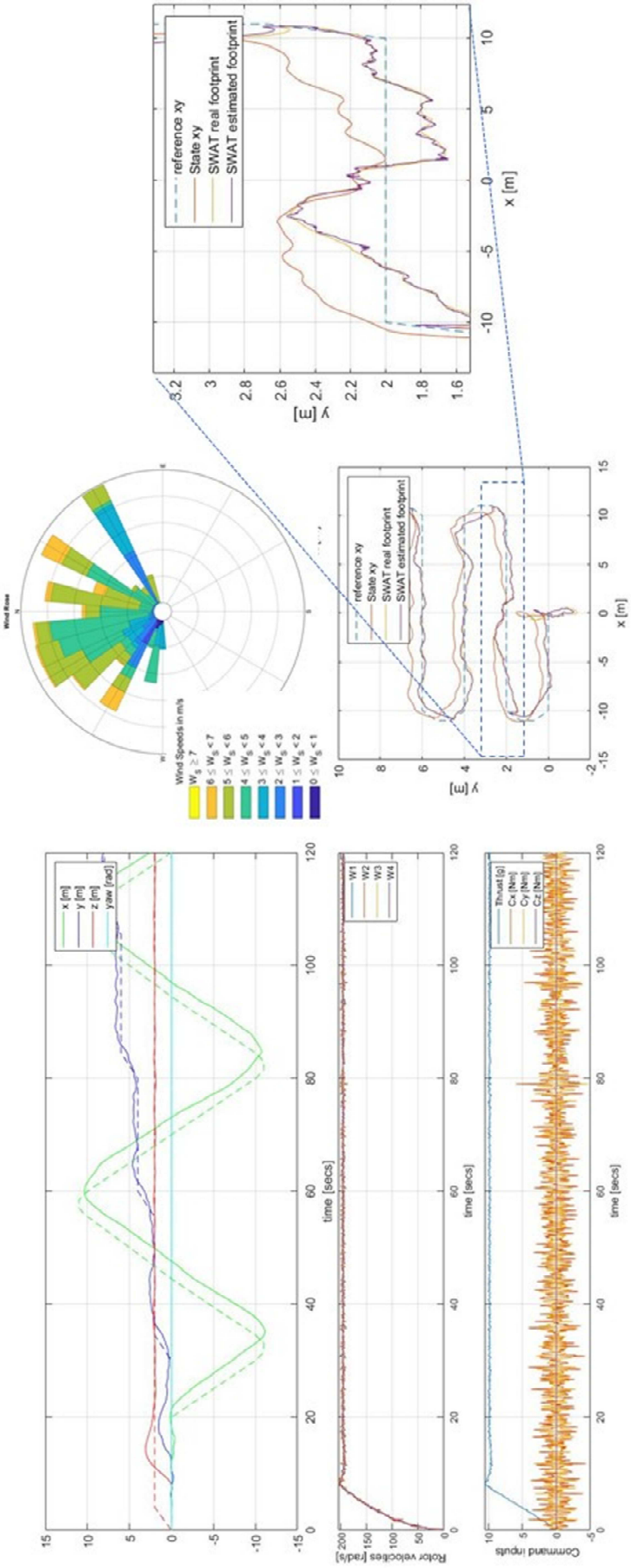


Figure 135 UAV State tracking, cross-configuration case, variable wind

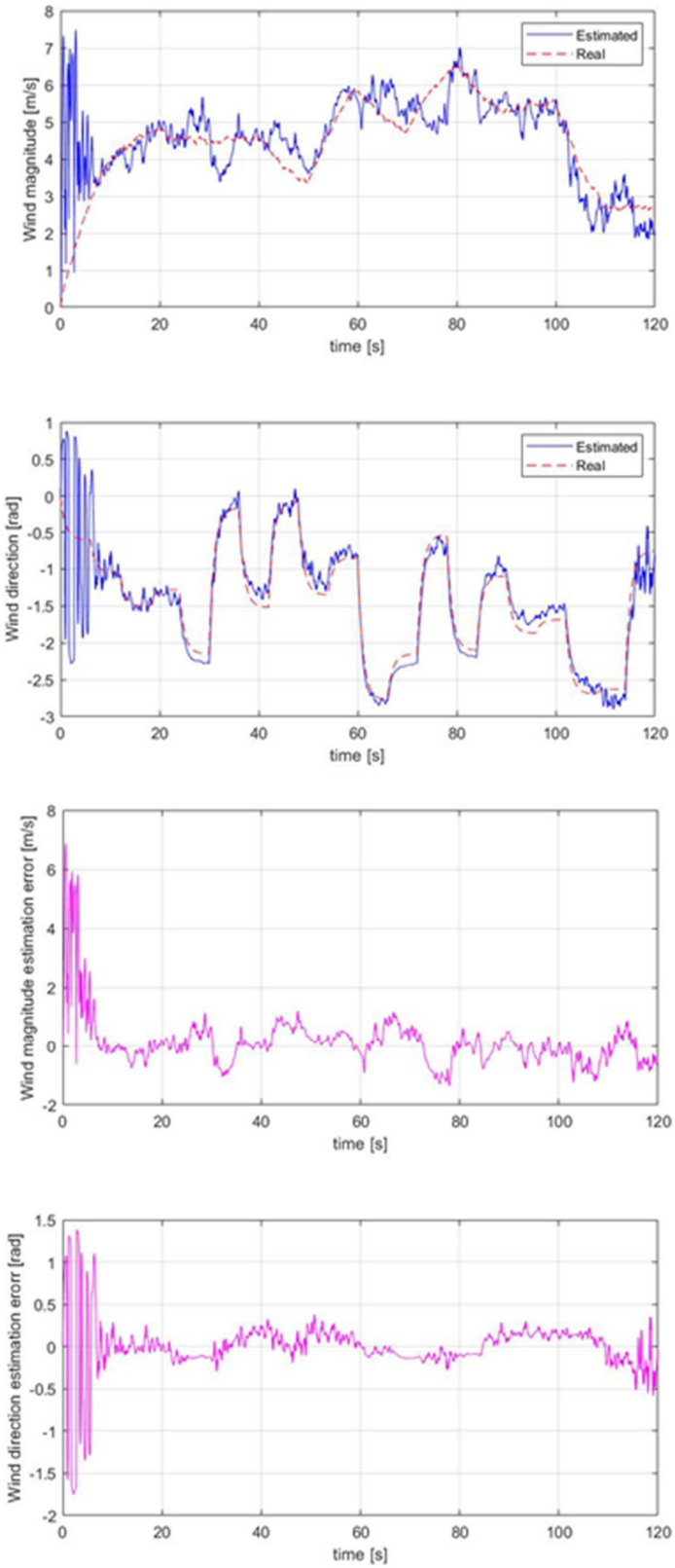


Figure 136 Wind estimation in magnitude and direction, cross-configuration case, variable wind

Table 24 Wind estimation algorithm performance in simulative environment, model aided

UAV model aided	Wind magnitude estimation RMSE [m/s]	Wind direction estimation RMSE [°]
Static/Constant wind	0,21	4,5
Static/Variable wind	0,23	12,7
Dynamic/Variable wind	0,51	8,61

In cross-configuration movements the wind estimation root mean squared error increases both in terms of magnitude, passing by 0,2 m/s of the static case to about 0.5 m/s, and in terms of direction, passing by 4° at about 9°. Main error's peaks are in correspondence of the driving inversion of the drone at 35,60, 85 and 110 seconds and in correspondence of the principal wind direction variations. Also, in this case the magnitude estimation error does not affect so much the footprint estimation because the input wind velocity is low relatively the deflection that can be produced, and, mainly, the UAV height is low (the deviations are computed at 50 cm from the UAV, as discussed in previous sections). However, the deflections are in the order of 60 centimeters and the compensation of this quantity does the difference if the target is the maximization of the delivered PPP liters. To validate the indispensability of the wind estimation and path correction algorithm another simulation of the cross-configuration was performed with the same wind condition and the same prescription map, always leaving the pump neighborhood control disactivated.

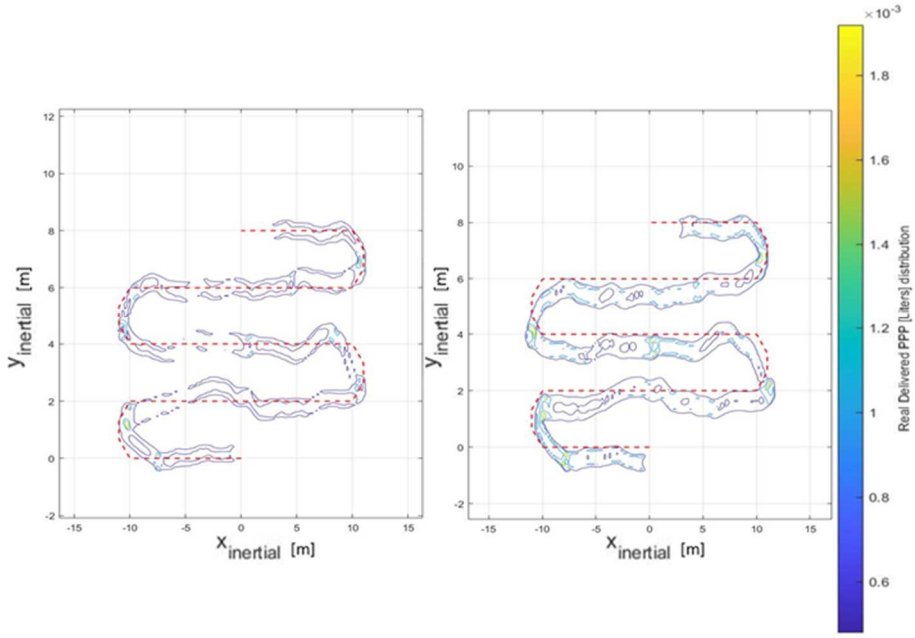


Figure 137 Actual PPP volume distribution with path correction (Top) and without (bottom)

As can be seen by figure 137, the absence of path correction in ordinary wind conditions (around 4 m/s) can strongly affect the delivering of the PPP product also flying as near as possible to the plants. The product released in active surface in the "blind" condition decreases of about the 40%. As seen before, in cross-configuration the sprayed volume is always greater than the requested if there is not a proximity check of the pump, because the needed flow rate is computed, for each plant, as the flow rate needed to reach the prescribed volume, at a given UAV velocity, when the

time over flighting the plant is the minimum possible, i.e., the time needed to cross the side of the square modeling the plant surface.

Table 25 Spray quality indexes with and without path correction, representative of Continuous (Dynamic) strategy

	Cross-configuration with path correction	**Cross-configuration without path correction**
PPP volume requested [L]	1,32	1,32
PPP volume sprayed [L]	1,5	1,5
Delivered/Sprayed ratio	0,73	0,46
Delivered/Requested ratio	0,83	0,52

Another simulation was done with the UAV spraying continuously while tries to keep the plant position (-0,45, 0) against the same wind, with and without wind estimation and path correction. As in the cross-configuration case the result is a strong optimization of the delivered volume in the target surface.

Table 26 Spray quality indexes with and without path correction, representative of PtP (Static) strategy

	Static configuration with path correction	**Static configuration without path correction**
Delivered/Sprayed ratio	0,87	0,44

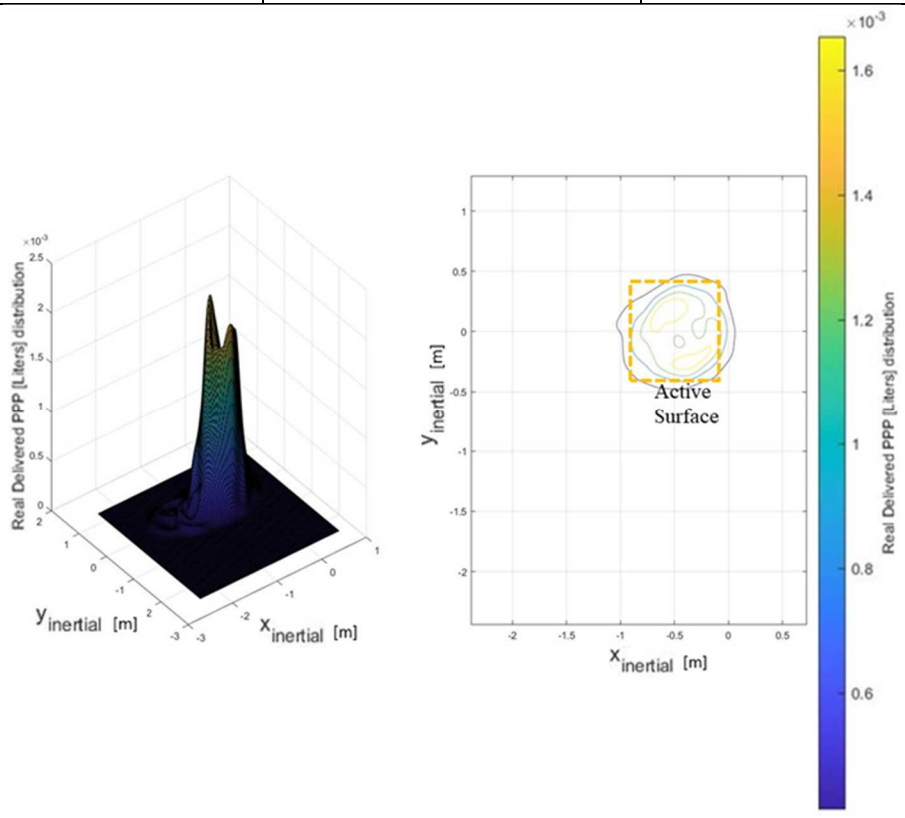


Figure 138 PPP Volume distribution and Active surface in static case, variable wind, With path correction

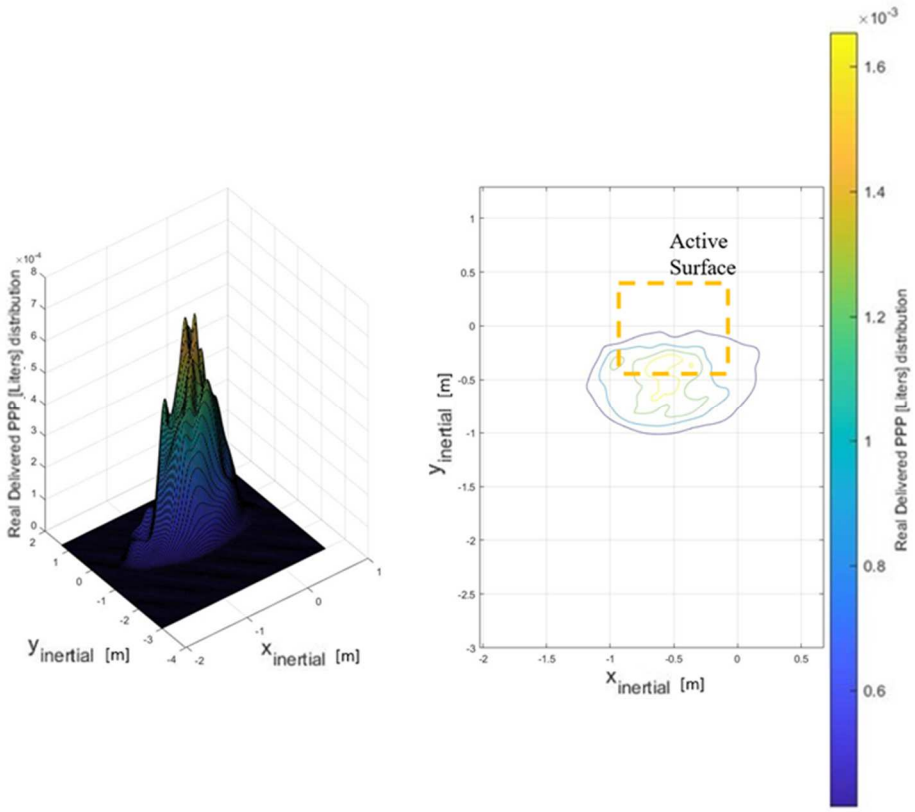


Figure 139 Volume distribution and Active surface in static case, variable wind, Without path correction

As anticipated, the wind estimation algorithm musts take in input the estimated Roll pitch and yaw in order to compute the tilt angle and the heading angles. In figure 140, is presented the wind estimation in static configuration with a variable wind where the wind estimation is based on the estimated EKF quantities. The error increases, both in magnitude and direction, but continues to be limited in a reasonable range. From now the algorithm will always uses the estimated quantities.

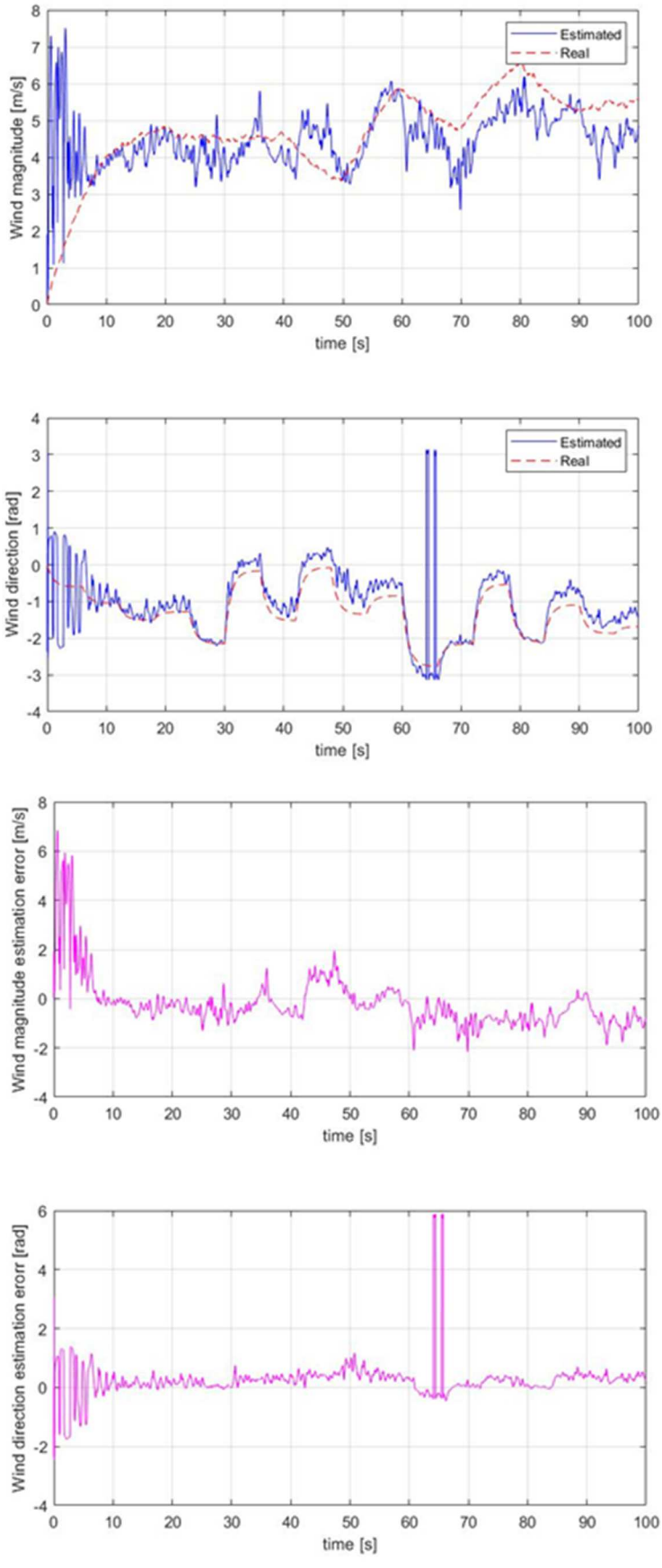


Figure 140 Wind estimation in hovering, EKF aided

Table 27 Wind estimation performance in simulative enviromenr, EKF aided

EKF aided	Wind magnitude estimation RMSE [m/s]	Wind direction estimation RMSE [°]
Static/Variable wind	0,75	22 °

5.3.4 Variable mass estimation

The mass estimation problem becomes crucial for 3 reasons:

- The update of the control gain matrix to its optimal value
- The correct execution of the wind estimation algorithm
- To keep track of the volume delivered to the plant.

As considered in the beginning of chapter 3, inertia matrix is considered independent by the mass variation and posed equal to the approximated cylindrical body inertia matrix computed when the tank is empty. Instead for what concerns the mass its variation estimation is of paramount importance because can decreases up to the 40% of its initial value. This decrement also causes the wind drag force to accelerate more the UAV, being the geometry invariant. The idea is to make a gain scheduling of the estimated new mass to update the control gain matrix solving a new LQR problem each second, so ten times the MCU sampling time, in order to decrease the latency due to the heavy computation of the LQR matrix. Another option is to compute the gain only when the UAV is in nearly stationary condition, simply by reading the accelerometer, because of the latency that the matrix computation could cause to the control system stability. For what concern the wind estimation algorithm the mass estimation is essential to get the thrust vector and rotate it obtaining the drag force.

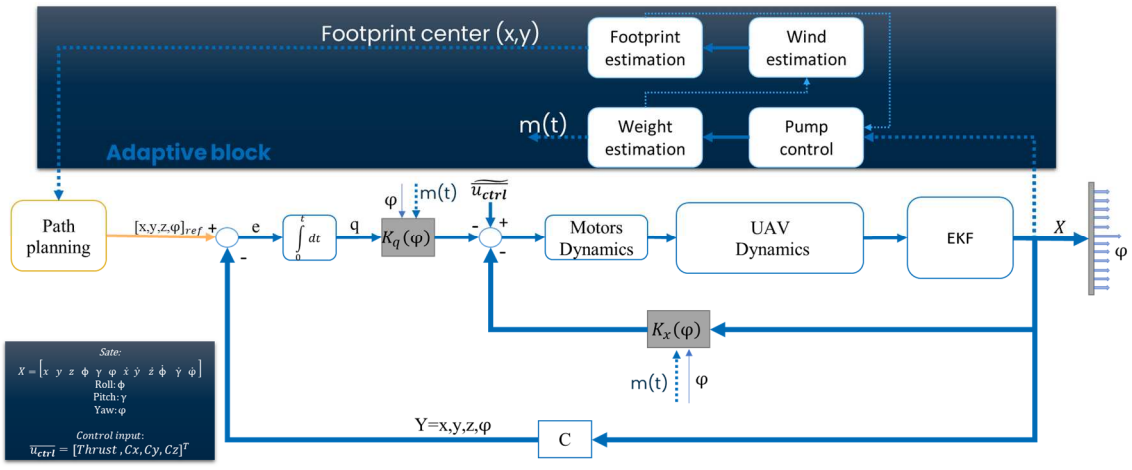


Figure 141 Block scheme of whole adaptive control

The mass variation was introduced in the simulation of the static-configuration seen until now. The results with and without mass estimation and mass gain scheduling were collected. The wind was of 5 m/s and directed toward N-O, the threshold distance 90 cm and the target plant in position (-0,45 0). Data were collected for 600 seconds in order to empty totally the tank. The PPP density was set to 1 Kg/L.

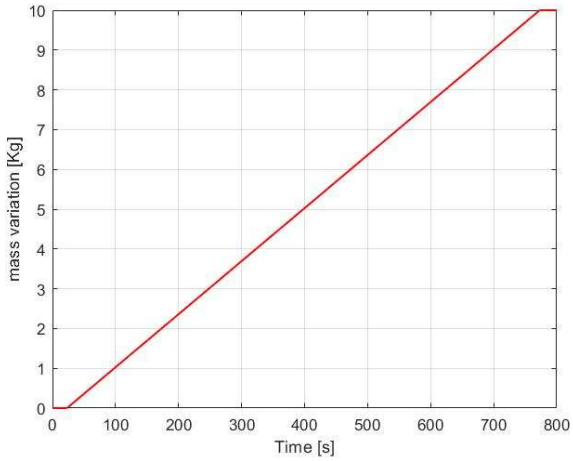


Figure 142 Sprayed mass with a constant flow rate of 0,8 L/min

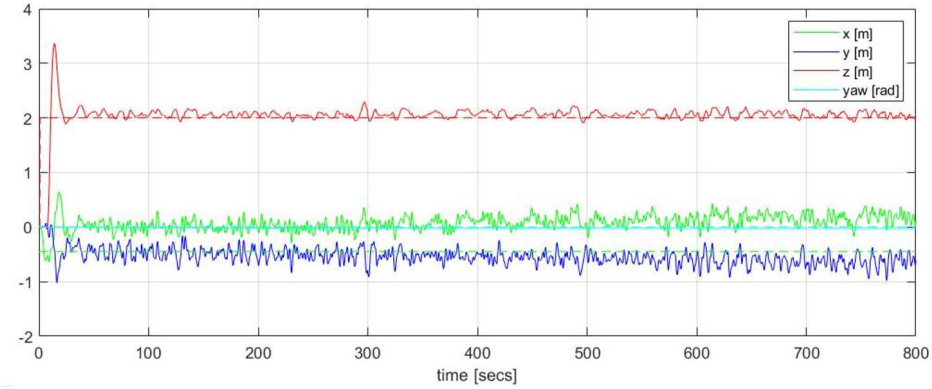


Figure 143 UAV state during the emptying, without mass estimation

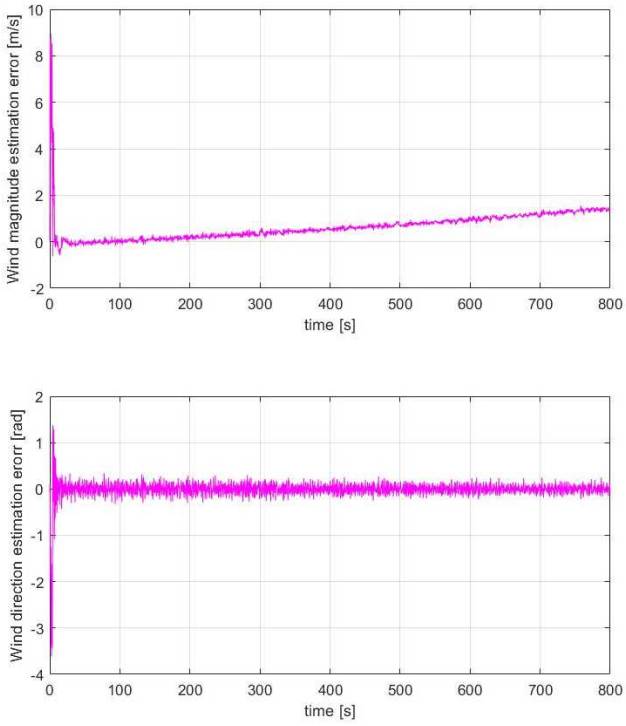


Figure 144 Wind Magnitude and angle error (Right) without mass estimation

In the second simulation all the parameters were left unchanged, while mass estimation and mass gain scheduling were activated.

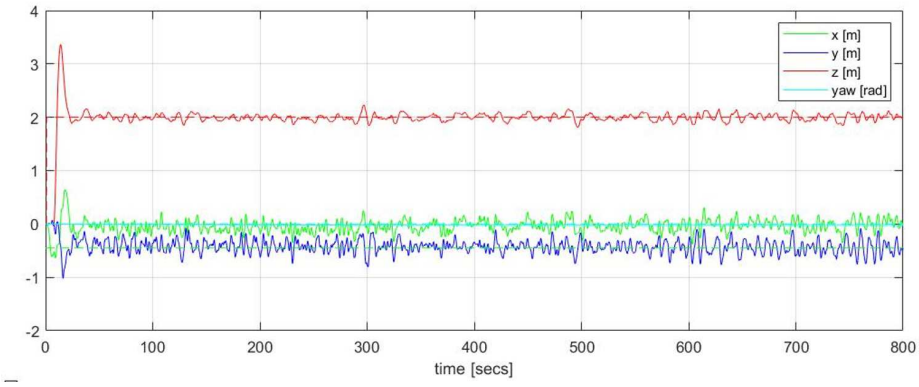


Figure 145 UAV state during the emptying, with mass estimation

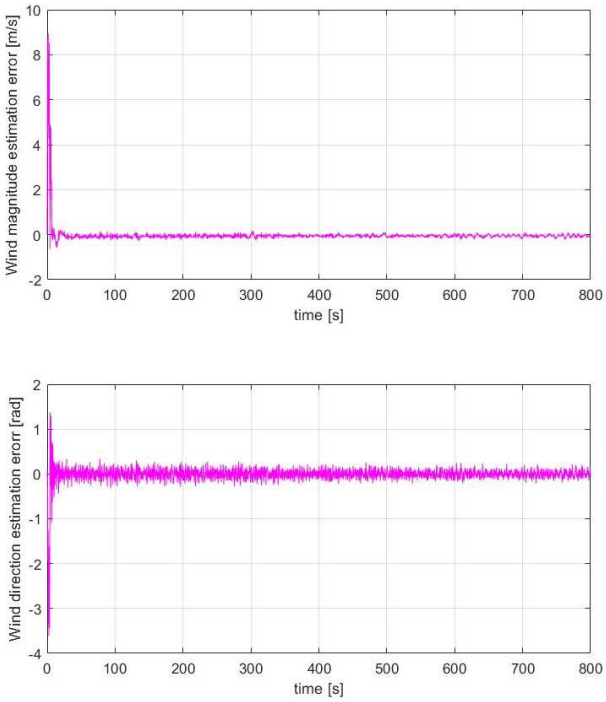


Figure 146 Wind Magnitude and angle error (Right), with mass estimation

The advantage introduced by the mass correction in wind estimation is clear, since without it the error continuously increases reaching 2 m/s and the same does the respective footprint center estimation. The advantage apported to the control system is less evident. Although the mass correction provides to the control system the optimal gain matrix, the control system clearly absorbs the mass variation reducing the thrust when the altitude starts to increase and augmenting it when the wind force causes greater accelerations in the plane caused by the mass decrement. However, this control processes, without mass gain scheduling, must wait the mass variation induced error to start. Indeed, seeing the z of the first simulation can be seen that it oscillates around a value greater than the reference value of 2 meters, while in the second simulation the z oscillates around 2 meters. The altitude error in the first case is about of 15 cm and is enough to has a lower Delivered/Actual ratio than in the second simulation. Is important to noticing that bigger flow rates could empathize this behavior, so the mass estimation is fundamental to have a good spray control system.

Table 28 D/R ratio with and without mass estimation

Delivered/Sprayed ratio without mass estimation	Delivered/Sprayed ratio with mass estimation
0,88	0,93

This is because the positive altitude tracking error, together with the increasing footprint center estimation error, causes to spray a larger area. The mass variation can be easily evaluated trusting on the on-board flow meter signal. Since the volume can be extracted by the flowmeter integrating its signal the drift problem can occur with dead reckoning. For these reasons should be adopted a flow meter with a bias under a certain limit that makes the drift error negligible with respect the total tank volume in the operation time wind (about 20 minutes). In addition, an extended Kalman filter can be used to estimate and correct the flow meter bias through a long-term stable signal able to correct the short-term reliability of the flowmeter, such as the thrust estimated knowing rotors velocities.

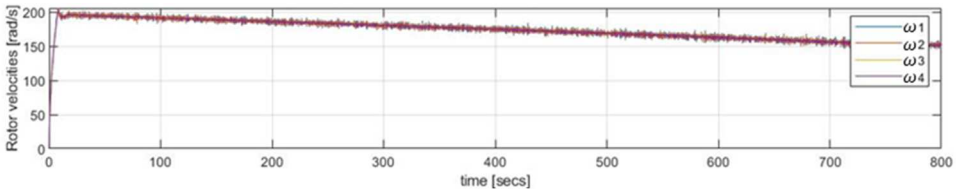


Figure 147 Rotor's velocities during the emptying

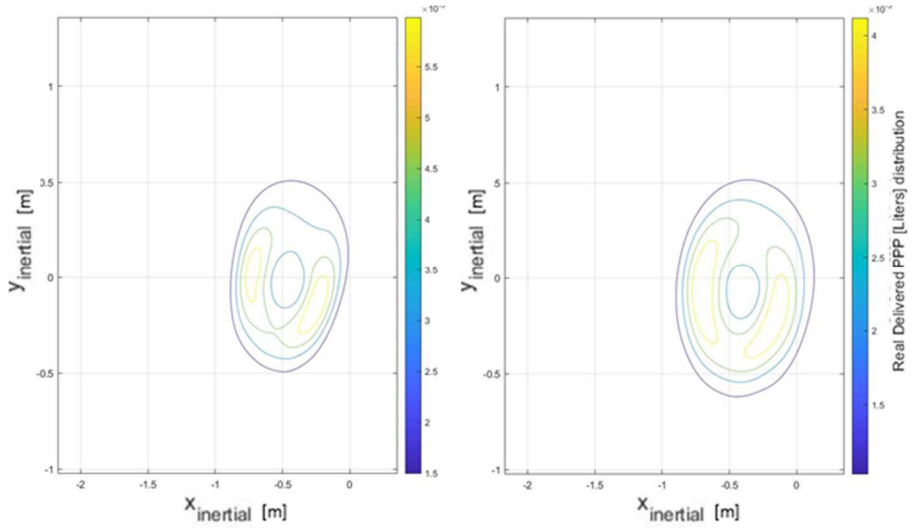


Figure 148 Footprint area with mass estimation (Left) and without (Right)

5.4 Main results

A final simulation was ran highlighting all the stages of the mission and evaluate how the whole mechanism is capable to ensure precision spray against wind, autonomy constraints, control and state estimation imprecisions. First, was generated a 2D squared map of 900 m^2 containing 513 plants (16 rows and 32 plants per row). Plants were represented by leaves picked up by the Plant Village dataset studied in chapter 2. The map was produced once for all, then all the subsequent simulations share the same leaves images in the same positions. For simplicity was considered only "Healthy" plants (94% of the total) and "Black Rot" deseeded plants (6%), therefore producing a low-density prescription map with 30 real targets, easily manageable by the GA, solving the modified Travelling Salesman Problem. To introduce randomness in the per plant volume demand, each plant PPP need was computed as its CNN posterior outcome subtracted by the chosen boundary threshold, then all the volumes of the prescription ma was normalized to

obtain a total volume request of 10 Liters, leaving their mutual ratios. To extract the waypoints from the images map was used the full trained MobileNetv2 without object recognition.

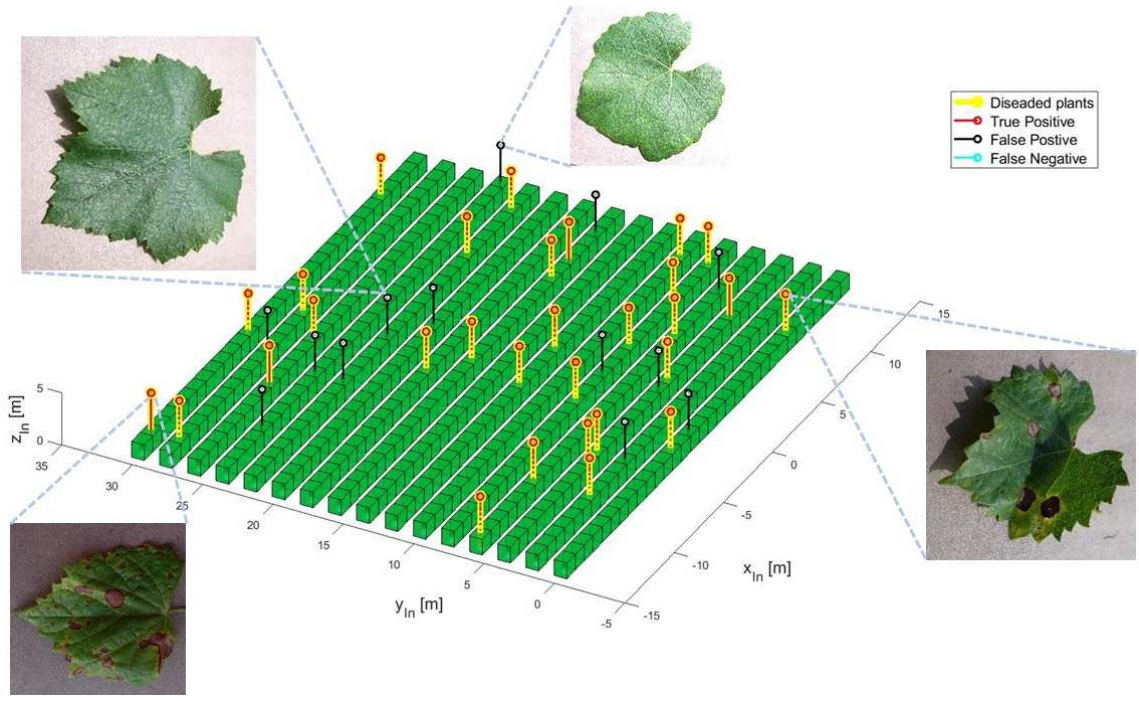
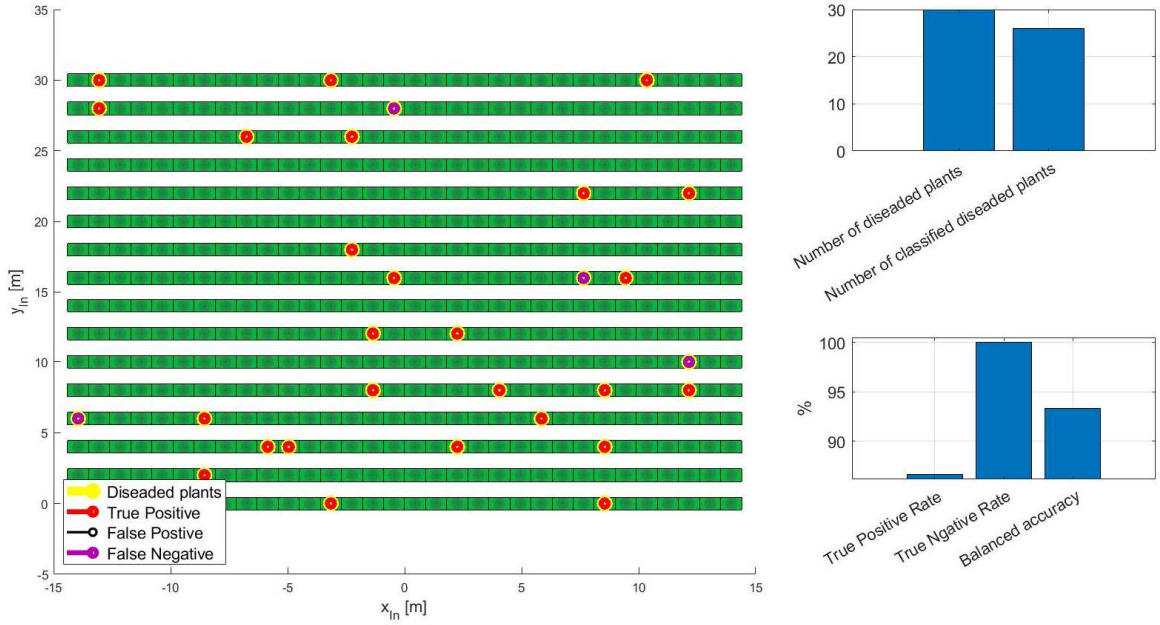


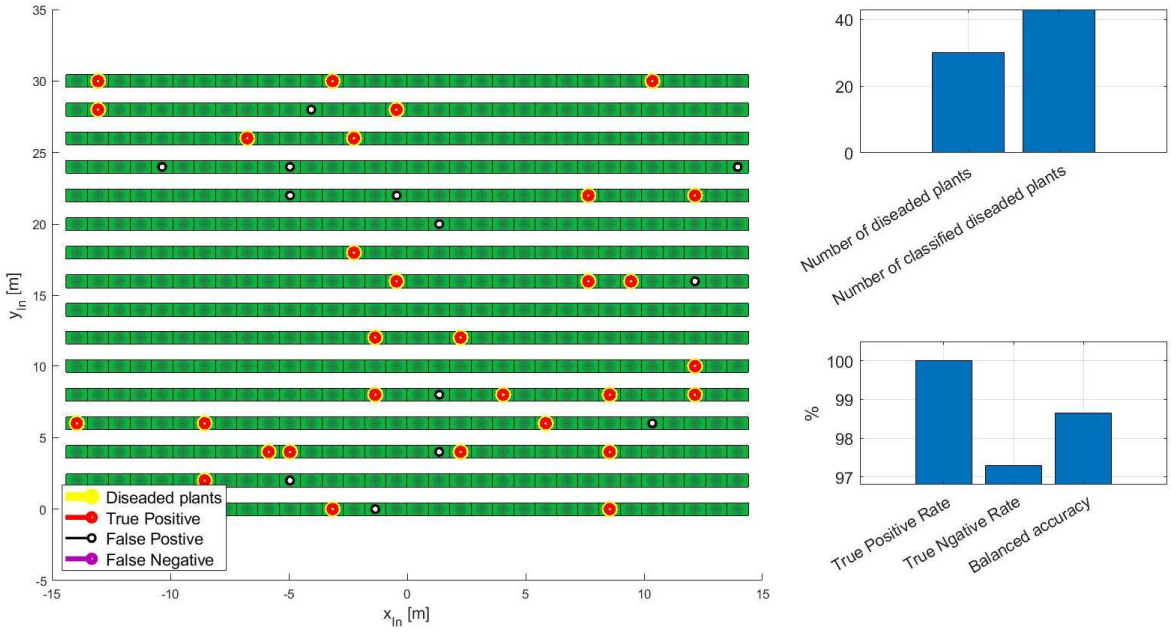
Figure 149 Generic prescription map waypoints with too sensible disease recognition

It is preferable to have a low FNR than a high FPR to conservatively spray the largest possible number of to be treated plants. For this reason, 3 thresholds have been tested. In fact, although in chapter 2 the optimal thresholding for mobnet2 was reported, that had been tested on a large database (about 4000 validation photos) and the photos that we will use here are the same but recombined (Flip, transpose, 180 ° rotation and lighting), a new rapid validity has been made with these 3 thresholds:

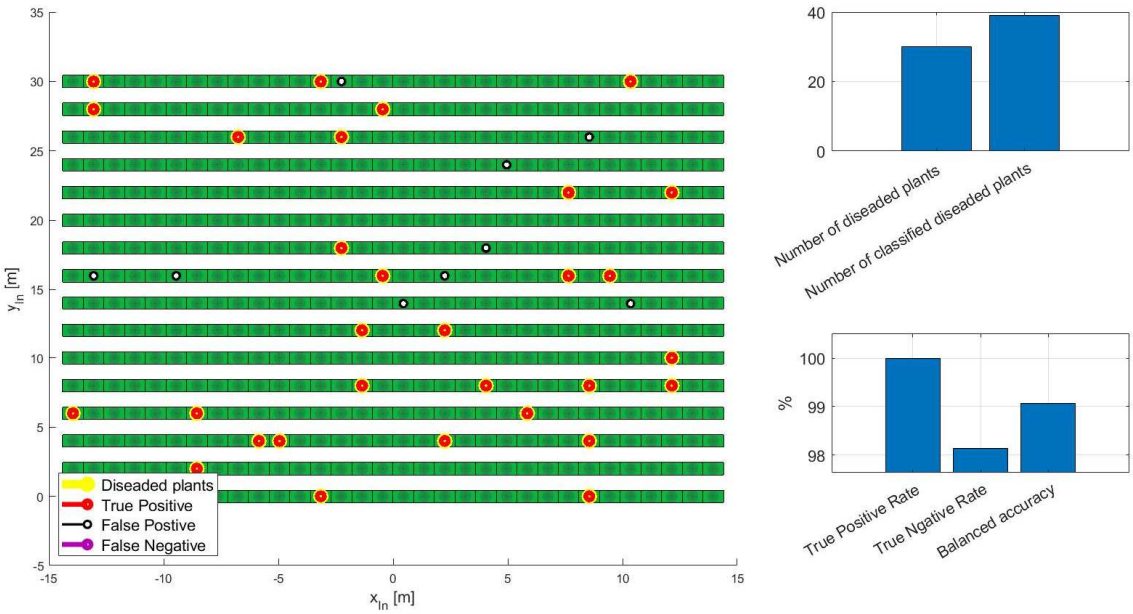
- Boundary threshold 0,93: the number of recognized plants is less than those actually sick, even if with the very high TNR all the recognized are actually sick (low waste of PPP, high risk of leaving plants without treatment)



- Boundary threshold 0.33: the number of recognized plants is greater than those sick, even if with the very high TPR all the diseased plants have actually been recognized (waste of PPP, low risk of leaving plants without treatment)



- Boundary threshold 0,43: is a good compromise. There are no false negatives and false positives are few. This threshold will be used in future simulations.



The last prescription map, sent to the GA with coupled cost function, provided a path planning on which 240 simulations were run. For each simulation was set a different wind statistic and a different proximity threshold. The wind sweep went from a systemic wind mean of 0 m/s to 6 m/s, from a wind systemic variance of 0 m/s^2 to 1,2 m/s^2, a fixed gust variance of 0,1 m/s^2 and north direction with 20° standard deviation. The proximity threshold values went from 23 cm to 270 cm (basically the same of a disabled proximity check) for a total of 7 values.

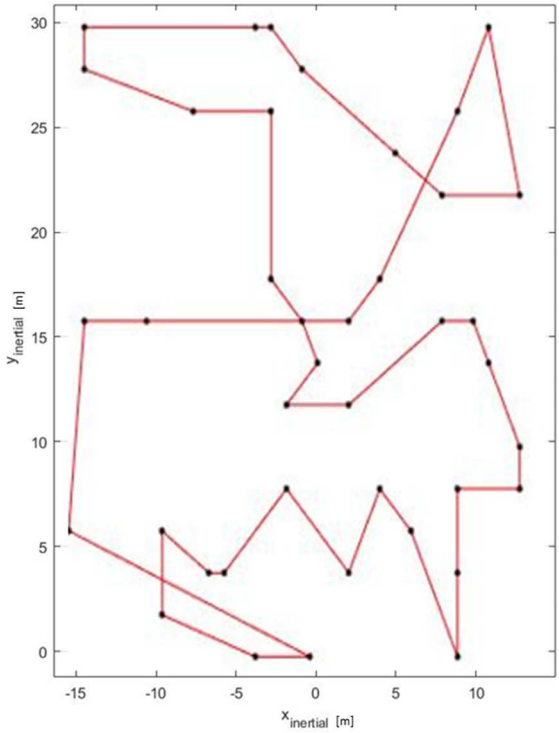


Figure 150 optimal distance=195.6 meters, J optimal=694,5 L1,5 m, found in 200 000 generations and 200 individuals

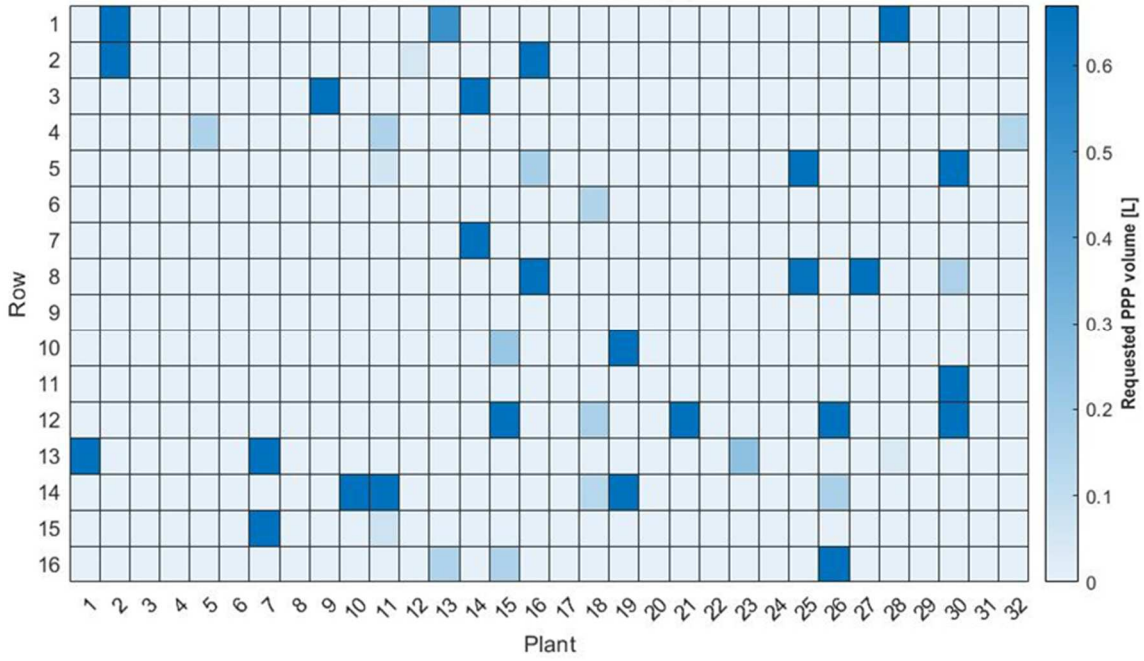


Figure 151 Definitive prescription map

As a result, interesting relationships between wind, proximity check sensibility, autonomy and spray performances were deduced:

- When the proximity threshold decreases, the average D/R, even in the absence of wind, rises but reaches a maximum limit, lower than 1, that depends on the combined quality of control and estimation of the wind. In addition, as the threshold decreases, the D/R begins to be less dependent on the wind speed remaining almost constant, while for larger thresholds the average starting D/R is monotonic decrescent with respect the wind speed. For a given wind velocity D/R ratio, in function of the proximity threshold, experiences a plateau over 1,5 meters, suggesting that threshold higher than that basically are equivalent to a disabled proximity check.
- At the same time to ensure such a high spray efficiency (high $D/R=D/S$) and robustness against the wind, the pump must turn off too frequently and the time of the PtP mission becomes so high as to exceed the maximum autonomy of the drone in the case 22.5 cm even in this mission low-density Prescription Map.
- In this case between 0.25 and 0.63 cm there is a proximity threshold that guarantees excellent wind resistance, a D/R between 0.7 and 0.8 and a consumption within the limits of 22500 mAh even up to 6 m/s (which can be considered already well beyond the maximum wind to take off).
- Path correction increases D/R by 15% (no wind) up to 150% in presence of windy environment.
- Proximity threshold further increase the D/R up to about 10 % in combination with path correction.

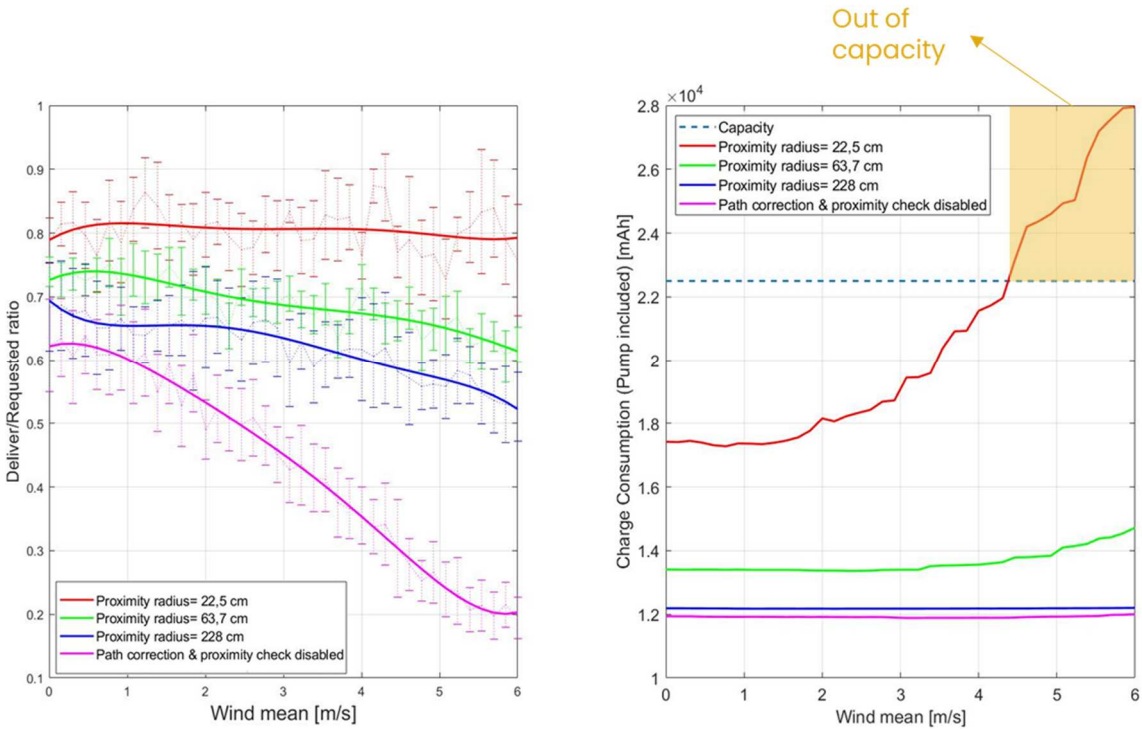


Figure 152 Delivered/Requested ratio and charge consumption in function of different wind magnitudes, for different proximity thresholds

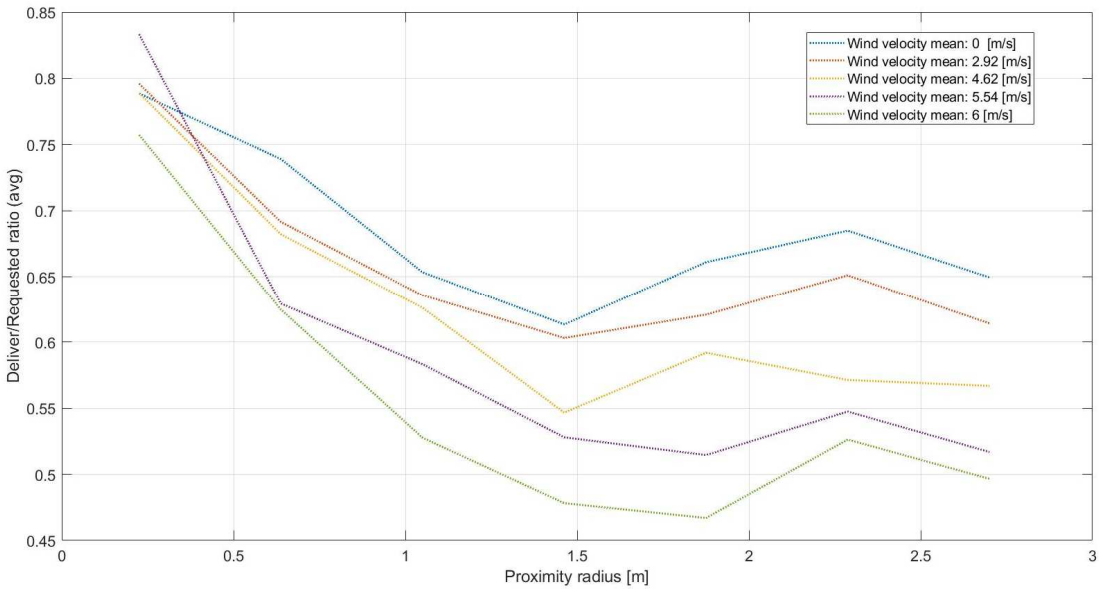


Figure 153 Delivered/Requested ratio in function of proximity threshold, at different systematic wind means

In the charts below is shown example mission with 90 cm proximity threshold, and 4 m/s systematic wind mean.

\

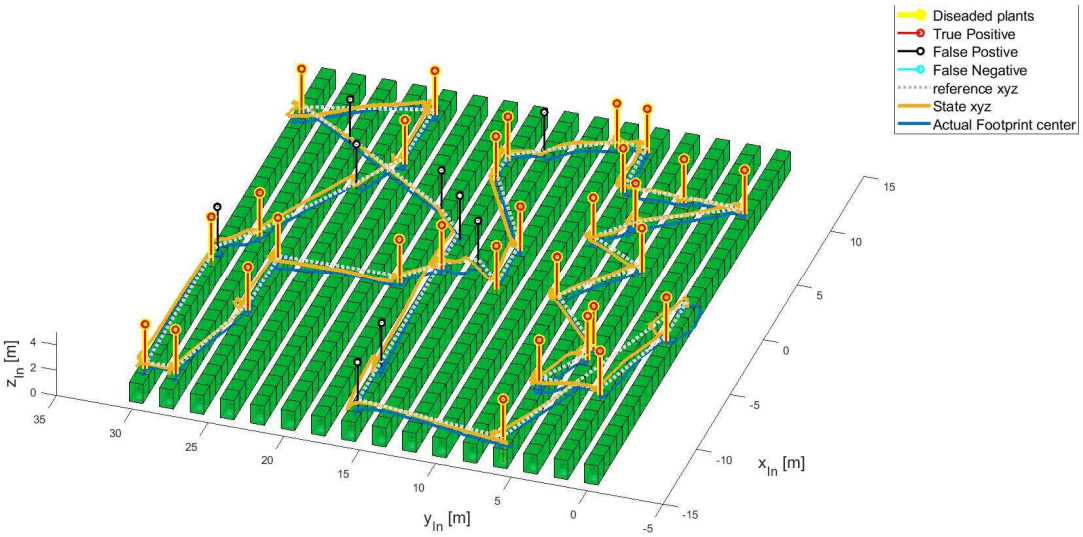


Figure 154 3D state tracking representation of the example mission

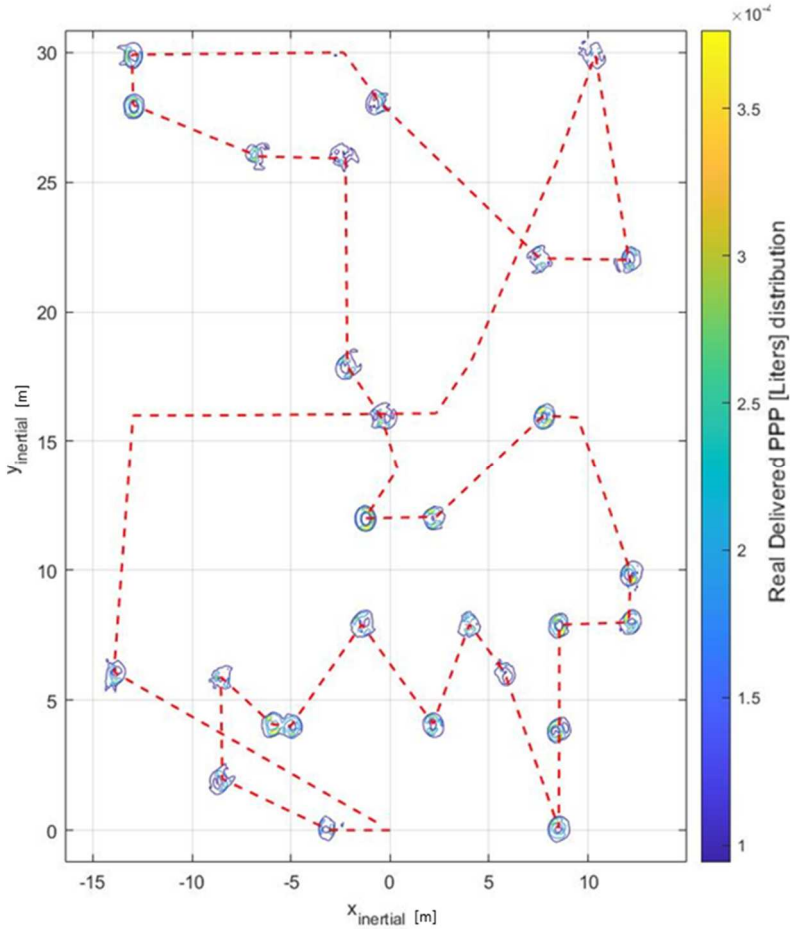


Figure 155 Delivered PPP distribution of the example mission

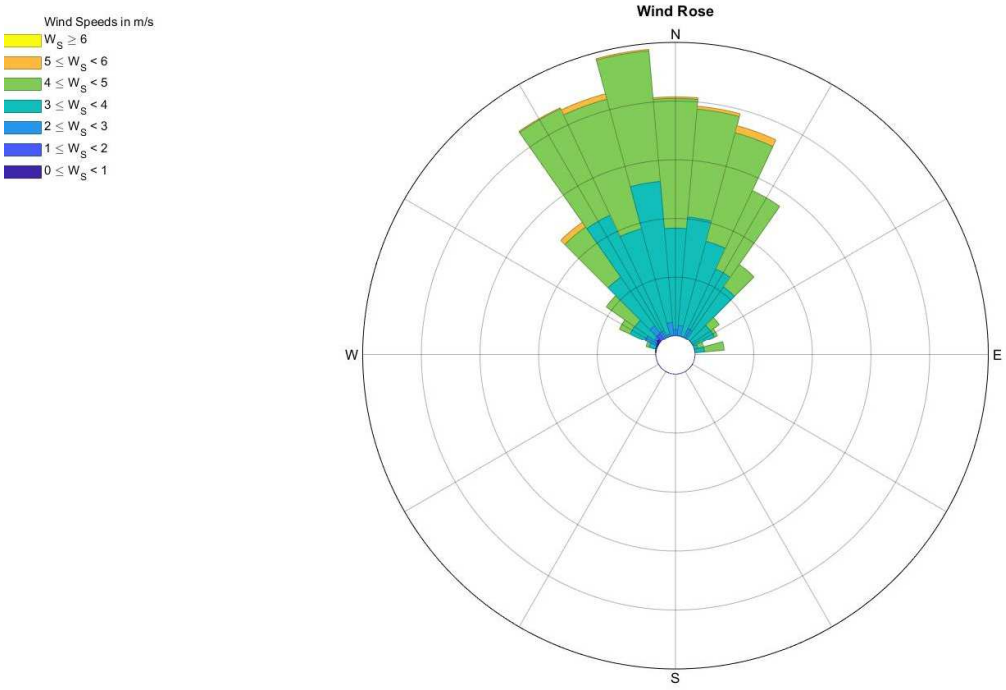


Figure 156 Wind rose of the example mission

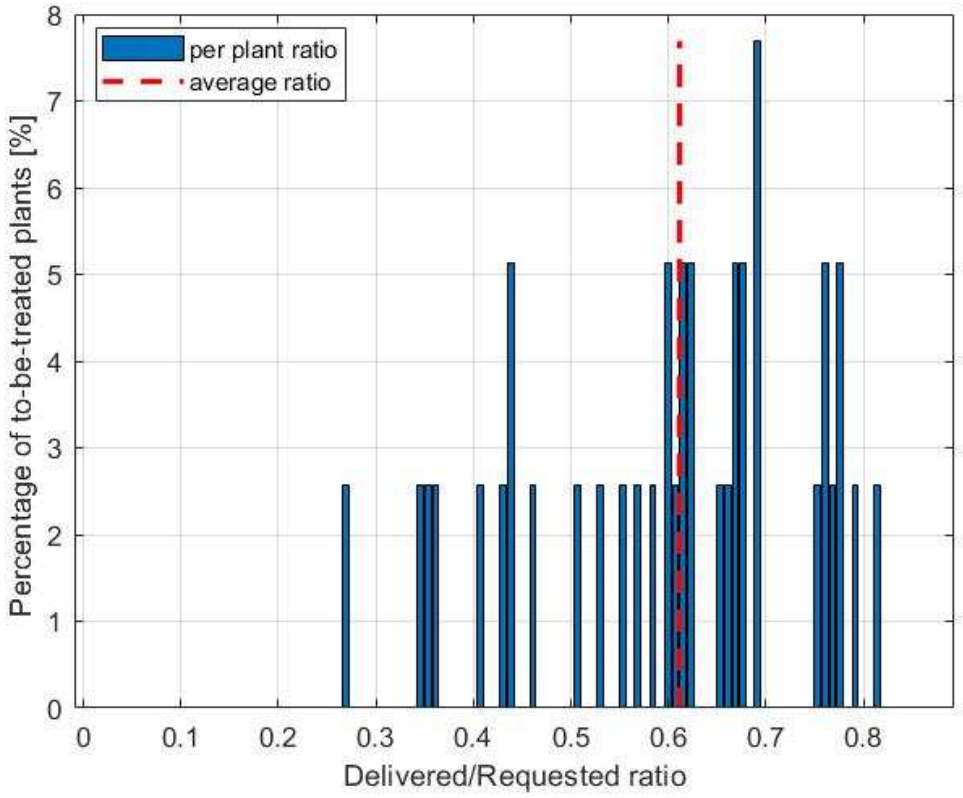


Figure 157 Per plant D/R distribution of the example mission

6 Conclusions

In any case, the mechanism of path correction and proximity control leads to a sharp increase in spray performance, compared to the total absence of the same, making this study useful in the actual implementation in-field of the precision spray via UAV. The ideal, in the application, would be to use these simulation curves, together with other experimental ones, to find the right trade-off between consumption and precision, knowing an estimate of the expected wind.
Beyond the case-specific considerations already discussed for each technique developed, can be said that the main factors that affect the spray precision and UAV autonomy has been:

- The accuracy of disease recognition, because of the presence of false positive increase the product waste, while false negatives leave diseased plants uncovered.
- The maximum distance of the proximity check above which the pump must be turned off to avoid waste. Over 1,5 meters of proximity threshold the delivered/requested ratio is not improved by the proximity check, while under about 25 centimeters the mission duration increased so much that overcomes the UAV autonomy, in presence of wind higher than about 4 m/s (in a map with 41 detected target plants, total PPP demand of 10 L and 196 meters path length).
- The quality of wind estimation in direction and amplitude, which in turn achieves maximum accuracy for low accelerations of the drone.
- The combination of control quality and state estimation, because of their influence to the drone ability to maintain the position over the target plant.
- The type of algorithm used for the optimization of path planning. The right solver leads to a minimization of the distance function in a reasonable amount of computational time.
- The parameters used in the path planning optimization algorithm. The preliminary examination of the prescription map demands distribution is essential to understand which cost function should be used.

Furthermore, collecting reasonings and results obtain until now emerges that the optimal spray strategy must be deduced across three dimensions:

- Per-plant precision: In applications in which is prominent the intensive spray of an entire field (e.g., seasonal pesticide treatments, preventive PPP spray etc..), is important to guarantee a specific volume released per hectare and to minimize the time spent to do so. Such intensive operations can be accomplished over flighting rows without stopping the drone (continuous spray) since the per-plant demand simply derives by the requested per-hectare volume and the number of plants contained in one hectare, but not so high emphasis is given to the individual plant pathological status. Although the rougher control of single targets PPP delivering, the path correction is always present because it only could improve the spray mission efficacy, while the proximity check can go in contrast with the priority of ensuring a certain amount of sprayed product (nevertheless if enabled it tends to reduce wastes). Instead, if the mission consists in precision spray, with a strictly correct PPP quantity, single plants, or pandemic clusters, then a closed loop control of the delivered volume is requested, and the priority becomes more the sureness of ill plants treatments than mission speed and area coverages. In the last case the combination of path correction and proximity check is fundamental.
- Prescription map density: In presence of a recognition of greatly sparse insurgence of ill plants in the area, the optimal obtained path often appears shattered and does not follow the field rows, since under a certain density the Modified NN does not offers the minimal distance solution. The lack of row-following path makes impossible to apply the cross-configuration. X-configuration (following the path lines and spray only while passing over the target) is however possible but, since this strategy does not implies drone

stopping if the estimated delivered volume is less than requested, does not ensure the precise PPP deliver and, furthermore, if the overflight velocity is high makes difficult to precisely track the optimal path at its elbows. As a result, is convenient and simpler to stop the UAV over the waypoint, spray the necessary, and restart toward the successive waypoint (PtP). When the prescription map starts to become so dense that the Modified NN solution is comparable or better than GA (found in reasonable computational time) continuous or PtP strategies can be both used based on precision levels specifications.

- Prescription map distribution: The more the volume demand is heterogeneously distributed among targets, the more the modified Salesman problem must be solved with the coupled cost function. The Genetic algorithm studied supports the coupled cost function, while NN and modified NN not. This implies that in sparse prescription maps the GA is enough to tackle both distance minimization and average mass transportation, while in presence of heterogeneous and dense prescription map the unique solution is to minimize the coupled cost function with GA and compare its expected charge consumption with the modified NN solution, although reasonable results of GA algorithm in this scenario have shown to appear after tens of hours computational time.

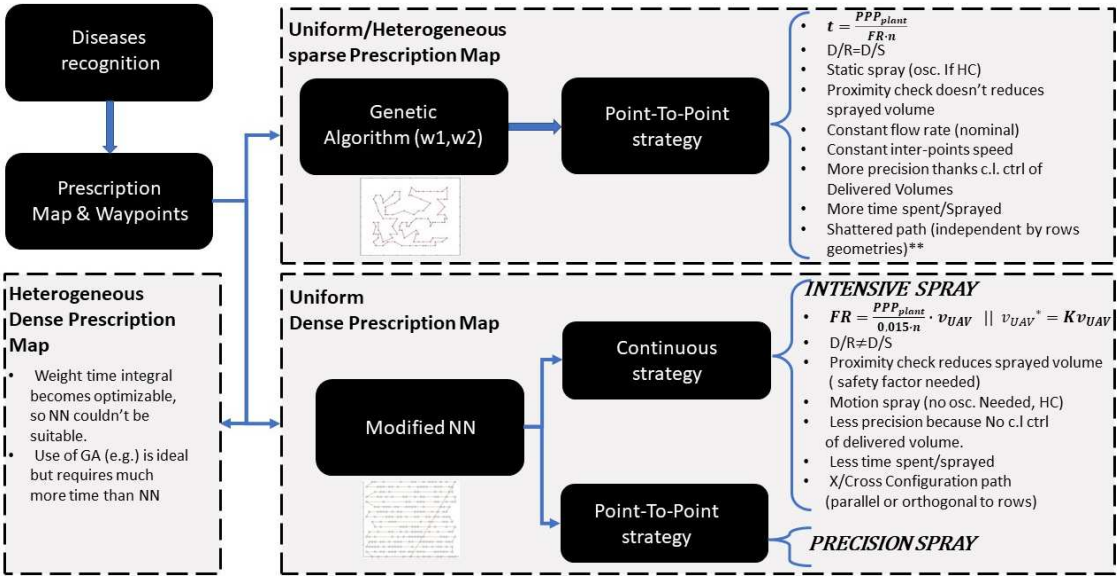


Figure 158 Flow chart of the spray strategy choice

Adjunctively, as seen in the path correction simulations, the wind estimation (and so the footprint center estimation) works better if the UAV is in hovering, leading to make furtherly precise the PtP strategy with respect the Continuous spray.

7 Future works

Future developments of the presented study will have to focus on the extension of disease recognition datasets to multi or hyperspectral images, so as to bring out non-extractable patterns in the visible spectrum. A more accurate and field imaging campaign should also be carried out, perhaps providing additional metadata to individual images, such as plant placement, weather, and lighting conditions. Once will be available a design and a prototype of the quadrotor, should be implemented the control algorithm in a micro-controller unit and calibrated the main parameters of the LQR. The state estimation should be extended putting in communication the IMU with an external positioning signal, such as the GPS. Wind tunnel tests are necessary to extract relations between the UAV tilt angle and the drag force, and the 3D CAD model of the drone should be used to define the projected area in function of the tilt angle. More accurate

models and validation could be carried out for the pump system, providing also a Proportional Integrative Derivative (PID) control to manage the pump delay and dynamics. An effective correlation between the output of the disease's classifiers and the real needs of the plant will have to be studied. Path planning should be enriched with real-time corrections related to the presence of obstacles not recorded in the one-time 3D mapping, using path finding algorithms. A Capacitated Vehicle Routing Problem (CVRP) could also be studied to select only the waypoints that guarantee not to exceed the capacity of the tank. Finally, the models and controls studied will have to be analyzed and validated experimentally.

Acknowledgements

My gratitude goes to my supervisor Professor Elisa Capello and my two co-supervisor Ing. Manuel Carreño Ruiz and Ing. Nicoletta Bloise, for constantly sustaining my ideas, for providing priceless advice during the preparation of the thesis and for offering valuable data and statistic which I used in my project. I am deeply grateful to the Polytechnic of Turin for the possibilities and the tools it has provided me for the thesis work and for the experience it has permitted me in these years. I would also like to thank my parents, the rest of my family and my girlfriend whom without this would have not been possible. Lastly, my heartful thank goes to all the friends for their human support during these months.

Bibliography

[1] K. Wiebe *et al.*, “Climate change impacts on agriculture in 2050 under a range of plausible socioeconomic and emissions scenarios,” *Environ. Res. Lett.*, vol. 10, no. 8, p. 085010, Aug. 2015, doi: 10.1088/1748-9326/10/8/085010.

[2] United Nations, Department of Economic and Social Affairs, Population Division, "World Population Prospects: The 2015 Revision, Key Findings and Advance Tables",2015, Working Paper No. ESA/P/WP.241.

[3] A. Tarabella, L. Trivelli, and A. Apicella, *Precision Agriculture. In: Food Products Evolution: Innovation Drivers and Market Trends. SpringerBriefs in Food, Health, and Nutrition.* Cham: Springer, 2019. Accessed: Mar. 24, 2022. [Online]. Available: https://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=5803026

[4] S. O. Araújo, R. S. Peres, J. Barata, F. Lidon, and J. C. Ramalho, “Characterising the Agriculture 4.0 Landscape—Emerging Trends, Challenges and Opportunities,” *Agronomy*, vol. 11, no. 4, p. 667, Apr. 2021, doi: 10.3390/agronomy11040667.

[5] M. Mammarella, L. Comba, A. Biglia, F. Dabbene, and P. Gay, “Cooperative Agricultural Operations of Aerial and Ground Unmanned Vehicles,” in *2020 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, Trento, Italy, Nov. 2020, pp. 224–229. doi: 10.1109/MetroAgriFor50201.2020.9277573.

[6] L. Becce, N. Bloise, and G. Guglieri, “Optimal Path Planning for Autonomous Spraying UAS framework in Precision Agriculture,” in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, Athens, Greece, Jun. 2021, pp. 698–707. doi: 10.1109/ICUAS51884.2021.9476690.

[7] Rançon et al., "Comparison of SIFT Encoded and Deep Learning Feature for the Classification and Detection of Esca Disease in Bordeaux Vineyards", *Remote Sens.* 2019, *11*, 1., doi: https://doi.org/10.3390/rs11010001

[8] C. L. Zitnick and P. Dollár, “Edge Boxes: Locating Object Proposals from Edges,” in *Computer Vision – ECCV 2014*, vol. 8693, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 391–405. doi: 10.1007/978-3-319-10602-1_26.

[9] N. Bloise, M. Carreno Ruiz, D. D’Ambrosio, and G. Guglieri, “Wind Tunnel Testing of Remotely Piloted Aircraft Systems for Precision Crop-Spraying Applications,” in *2021 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, Trento-Bolzano, Italy, Nov. 2021, pp. 378–383. doi: 10.1109/MetroAgriFor52389.2021.9628600.

[10] P. Abichandani, D. Lobo, G. Ford, D. Bucci, and M. Kam, “Wind Measurement and Simulation Techniques in Multi-Rotor Small Unmanned Aerial Vehicles,” *IEEE Access*, vol. 8, pp. 54910–54927, 2020, doi: 10.1109/ACCESS.2020.2977693.

[11] P. P. Neumann and M. Bartholmai, “Real-time wind estimation on a micro unmanned aerial vehicle using its inertial measurement unit,” *Sens. Actuators Phys.*, vol. 235, pp. 300–310, Nov. 2015, doi: 10.1016/j.sna.2015.09.036.

[12] J. Zhu, M. Cheng, Q. Wang, H. Yuan, and Z. Cai, “Grape Leaf Black Rot Detection Based on Super-Resolution Image Enhancement and Deep Learning,” *Front. Plant Sci.*, vol. 12, p. 695749, Jun. 2021, doi: 10.3389/fpls.2021.695749.

[13] W. V. Pereira, E. A. R. Czaja, L. G. R. Virmond, and L. L. May-De-Mio, “First Report of Leaf Spot Caused by *Pseudocercospora vitis* on *Bidens pilosa* in Brazil,” *Plant Dis.*, vol. 103, no. 4, pp. 772–772, Apr. 2019, doi: 10.1094/PDIS-05-18-0829-PDN.

[14] M. Alessandrini, R. Calero Fuentes Rivera, L. Falaschetti, D. Pau, V. Tomaselli, and C. Turchetti, “A grapevine leaves dataset for early detection and classification of esca disease in vineyards through machine learning,” *Data Brief*, vol. 35, p. 106809, Apr. 2021, doi: 10.1016/j.dib.2021.106809.

[15] L. Zheng, Y. Yang, and Q. Tian, “SIFT Meets CNN: A Decade Survey of Instance Retrieval,” *ArXiv160801807 Cs*, May 2017, Accessed: Feb. 17, 2022. [Online]. Available: http://arxiv.org/abs/1608.01807

[16] "Understanding Support Vector Machines: A Primer," *applied Machine Learning*, Accessed: Feb. 21, 2022. [Online]. Available: https://appliedmachinelearning.blog/2017/03/09/understanding-support-vector-machines-a-primer/
[17] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, Antalya, Aug. 2017, pp. 1–6. doi: 10.1109/ICEngTechnol.2017.8308186.
[18] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *ArXiv170404861 Cs*, Apr. 2017, Accessed: Feb. 17, 2022. [Online]. Available: http://arxiv.org/abs/1704.04861
[19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *ArXiv180104381 Cs*, Mar. 2019, Accessed: Feb. 17, 2022. [Online]. Available: http://arxiv.org/abs/1801.04381
[20] V. Aliyev, "A definitive explanation to the Hinge Loss for Support Vector Machines" *towardsdatascience*, Accessed: Feb. 21, 2022. [Online]. Available: https://towardsdatascience.com/a-definitive-explanation-to-hinge-loss-for-support-vector-machines-ab6d8d3178f1
[21] O. Gómez-Carmona, D. Casado-Mansilla, F. A. Kraemer, D. López-de-Ipiña, and J. García-Zubia, "Exploring the computational cost of machine learning at the edge for human-centric Internet of Things," *Future Gener. Comput. Syst.*, vol. 112, pp. 670–683, Nov. 2020, doi: 10.1016/j.future.2020.06.013.
[22] A. M. Harrington and C. Kroninger, "Characterization of Small DC Brushed and Brushless Motors:," Defense Technical Information Center, Fort Belvoir, VA, Mar. 2013. doi: 10.21236/ADA577582.
[23] Advanced Textbooks in Control and Signal Processing; Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo, " Robotics,Modelling, Planning and Control" Springer (2008)
[24] "DJI website, Matrix 600 exacopter." https://www.dji.com/it/matrice600
[25] S. Bouabdallah, A. Noth, and R. Siegwart, "PID vs LQ control techniques applied to an indoor micro quadrotor," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, Sendai, Japan, 2004, vol. 3, pp. 2451–2456. doi: 10.1109/IROS.2004.1389776.
[26] J.-M. Coron, "Controllability of nonlinear systems", *University Pierre et Marie Curie (Paris 6) BCAM OPTPDE summer school,* 2019, p. 119.
[27] A. Ataka *et al.*, "Controllability and observability analysis of the gain scheduling based linearization for UAV quadrotor," in *2013 International Conference on Robotics, Biomimetics, Intelligent Computational Systems*, Jogjakarta, Indonesia, Nov. 2013, pp. 212–218. doi: 10.1109/ROBIONETICS.2013.6743606.
[28] S. K. Phang, K. Li, B. M. Chen, and T. H. Lee, "Systematic Design Methodology and Construction of Micro Aerial Quadrotor Vehicles," in *Handbook of Unmanned Aerial Vehicles*, K. P. Valavanis and G. J. Vachtsevanos, Eds. Dordrecht: Springer Netherlands, 2015, pp. 181–206. doi: 10.1007/978-90-481-9707-1_116.
[29] M. Pettersson, "Extended Kalman Filter for Robust UAV Attitude Estimation," p. 105.
[30] X. Guo, N. Ansari, F. Hu, Y. Shao, N. R. Elikplim, and L. Li, "A Survey on Fusion-Based Indoor Positioning," *IEEE Commun. Surv. Tutor.*, vol. 22, no. 1, pp. 566–594, 2020, doi: 10.1109/COMST.2019.2951036.
[31] H. Xiang and L. Tian, "Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (UAV)," *Biosyst. Eng.*, vol. 108, no. 2, pp. 174–190, Feb. 2011, doi: 10.1016/j.biosystemseng.2010.11.010.
[32] Y. Feng and J. Wang, "GPS RTK Performance Characteristics and Analysis," *J. Glob. Position. Syst.*, vol. 7, no. 1, pp. 1–8, Jun. 2008, doi: 10.5081/jgps.7.1.1.
[33] P. Balsari, D. M. Tamagnone, and D. G. Oggero, "prove eseguite in conformità al capitolato enama n°46a presso il laboratorio crop protection technology del dipartimento di scienze agrarie, forestali e alimentari dell'università degli studi di torino" p. 6.

[34] A. de Castro, F. Jiménez-Brenes, J. Torres-Sánchez, J. Peña, I. Borra-Serrano, and F. López-Granados, "3-D Characterization of Vineyards Using a Novel UAV Imagery-Based OBIA Procedure for Precision Viticulture Applications," *Remote Sens.*, vol. 10, no. 4, p. 584, Apr. 2018, doi: 10.3390/rs10040584.
[35] Q. Lian *et al.*, "Design of precision variable-rate spray system for unmanned aerial vehicle using automatic control method," *Int. J. Agric. Biol. Eng.*, vol. 12, no. 2, pp. 29–35, 2019, doi: 10.25165/j.ijabe.20191202.4701.
[36] Y. Zhao, Z. Zheng, and Y. Liu, "Survey on computational-intelligence-based UAV path planning," *Knowl.-Based Syst.*, vol. 158, pp. 54–64, Oct. 2018, doi: 10.1016/j.knosys.2018.05.033.
[37] A. Goli, J. Khazaei, and S. Kouravand, "Studying and examining the acceleration and air flow pressure in wheat head stripper," *Int. Acad. J. Sci. Eng.*, vol. 06, no. 01, pp. 72–84, Jun. 2019, doi: 10.9756/IAJSE/V6I1/1910007.
[38] F. Ziegler, "Analyzing sensor system flaws in modern machines (series) The Air France AF 447 disaster," *Invensity*, Accessed: Feb. 21, 2022. [Online]. Available: https://www.invensity.com/2020/09/18/analyzing-sensor-system-flaws-in-modern-machines-1st-article-of-the-series/?lang=en
[39] W. Thielicke, W. Hübert, and U. Müller, "Towards accurate and practical drone-based wind measurements with an ultrasonic anemometer," Others (Wind, Precipitation, Temperature, etc.)/Remote Sensing/Instruments and Platforms, preprint, Sep. 2020. doi: 10.5194/amt-2020-258.
[40] "YOUNG Model 81000V Ultrasonic Anemometer," *YOUNG Usa*. https://www.youngusa.com/product/ultrasonic-anemometer-2/
[41] "Direct Mount, DM50," *FT Technologies*. https://fttechnologies.com/wind-sensors/ft7-series/direct-mount-dm50/
[42] M. Kvernvik, "Graphite and FT Technologies team up to develop 3D printed drone-specific wind sensor," *TCT magazine*, Accessed: Feb. 21, 2022. [Online]. Available: https://fttechnologies.com/wind-sensors/ft7-series/direct-mount-dm50/
[43] C. Brosy *et al.*, "Simultaneous multicopter-based air sampling and sensing of meteorological variables," *Atmospheric Meas. Tech.*, vol. 10, no. 8, pp. 2773–2784, Aug. 2017, doi: 10.5194/amt-10-2773-2017.
[44] Y. Song, Q.-H. Meng, B. Luo, M. Zeng, S.-G. Ma, and P.-F. Qi, "A wind estimation method for quadrotors using inertial measurement units," in *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Qingdao, China, Dec. 2016, pp. 426–431. doi: 10.1109/ROBIO.2016.7866359.
[45] S. F. Hoerner, "Fluid-Dynamic Drag," pp. 3–17.
[46] Solovyev Viktor V. et al., "Simulation of wind effect on a quadrotor flight", 2015