João Filipe Tavares Natividade

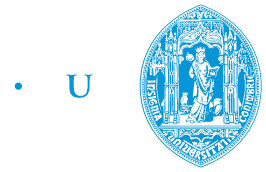
Vegetation Classification monitored with Multispectral Aerial Images

Thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in Electrical and Computer Engineering

February, 2017

· U C ·

Universidade de Coimbra

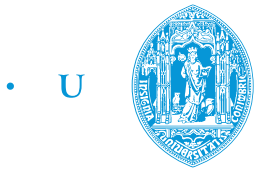


FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Vegetation Classification monitored with Multispectral Aerial Images

João Filipe Tavares Natividade

Coimbra, February 2017



Vegetation Classification monitored with Multispectral Aerial Images

Supervisor:

Prof. Dr. Lino José Forte Marques

Co-Supervisor:

Dr. José Augusto Soares Prado

Jury:

Prof. Dr. Jorge Manuel Moreira de Campos Pereira Batista

Prof. Dr. Paulo Jorge Carvalho Menezes

Prof. Dr. Lino José Forte Marques

Dissertation submitted in partial fulfillment for the degree of Master of Science in Electrical and Computer Engineering.

Coimbra, February 2017

Acknowledgements

Firstly, I would like to express gratitude to my advisers Professor Lino Marques and Dr. José Prado for their essential guidance, experience and encouragement. This dissertation would not be possible without their advice and dedication.

Certainly, without the affection given by my parents and brothers (my two older brothers are my daily source of inspiration), I surely wouldn't be able to complete my Master's degree. I would also like to express my thanks to my girlfriend, Andreia Cardoso, for all her patience, understanding and support and, lastly, to all my friends.

I must thank Institute of Systems and Robotics for providing me the best conditions and for all the investment that was made with the purchase of the Drone and all the underlying material. For each one who belongs to the Embedded Systems Laboratory I would also like to express my gratitude, with particular emphasis to my friend and colleague Eduardo Domingues.

Finally, it was a pleasure to have studied at the University of Coimbra for the past 5 and a half years.

Resumo

Na Agricultura de Precisão (AP), a detecção e classificação de vegetação em culturas herbáceas (*e.g.*, vinhas e árvores) é um passo crucial para a definição de objectivos consequentes, tais como a utilização de diferentes fertilizantes ou distintos níveis de hidratação. Um Veículo Aéreo Não Tripulado (VANT) foi montado e testado com o objetivo de desenvolver uma ferramenta de classificação (câmara "multiespectral") para ser usada em diferentes culturas. O sistema compreende uma plataforma aérea *quad-rotor*, capaz de voar até alturas de 100 m acima do nível do solo. O presente trabalho desenvolve diferentes abordagens de segmentação que resultam na identificação da Região de Interesse (RDI) e, posteriormente, na sua classificação de acordo com a previsão fornecida por dois classificadores: Máquinas de Vectores de Suporte (MVS) e Árvores de Decisão (AD). As imagens são capturadas por um sistema composto por uma câmera (*NoIR* Raspberry Pi (RPi) - com um conjunto de cinco filtros ópticos acoplados à sua lente) conectada a um RPi, colocado no VANT para detecção de vegetação. As imagens foram adquiridas através de voos sobre os seguintes campos agrícolas: três vinhas com diferentes variedades de casta e florestas de cinco espécies de árvores (eucaliptos, pinheiros, oliveiras, laranjeiras e magnólias). Os testes mostram o desempenho dos dois classificadores, de acordo com a RDI identificada previamente por um algoritmo baseado em *thresholding* através de valores fornecidos pelo índice de vegetação NDVI. Os dois classificadores recebem dados de entrada fornecidos pelo cálculo de oito índices de vegetação. O algoritmo gerou valores de exactidão de 72% e 73% (para o Sistema de Reconhecimento de Padrões (SRP) associado ao conjuntos de dados #1 e #2, respectivamente, relacionados com as MVS) e 74% e 79% (para o SRP de acordo com os conjuntos de dados #1 e #2, respectivamente, relacionadas com os AD). O algoritmo é totalmente automatizado e a classificação é fornecida a partir do RPi para uma base de controlo, em tempo real, através duma conexão por Wi-Fi.

Palavras-Chave: Agricultura de Precisão, Veículo Aéreo Não Tripulado, Índice de Vegetação, Raspberry Pi, Máquinas de Suporte de Vectores, Árvores de Decisão.

Abstract

In Precision Agriculture (PA), detecting and classifying the vegetation in herbaceous crops (*e.g.*, vineyards and trees) is a crucial step prior to address further objectives, such as specifying either different fertilizers or distinguished hydration levels. An Unmanned Aerial Vehicle (UAV) was assembled and tested with the aim of developing a classification tool ("multispectral" camera) for both types of vineyard and tree species. The system comprises a quad-rotor aerial platform capable of flying up to heights of 100 m above the ground level. The present research work develops different segmentation approaches which result in the identification of the Region of Interest (ROI) and afterwards their classification according to the prediction provided by two different machine learning classifiers: Support Vector Machines (SVM)s and Decision Trees (DT)s. Along with the general description of the procedure, remotely-sensed images captured with a sensor (a *NoIR* Raspberry Pi (RPi) camera with a five optical filters wheel attached) connected to an RPi and mounted on the UAV, were applied for vegetation detection. Images were acquired while hovering above both fields of three vineyard species (with different grape varieties) and forests of five tree species (eucalyptus, pine trees, olive trees, orange trees and magnolias). The tests show the performance of both classifiers, according to the ROI identified by a thresholding algorithm based on Normalized Difference Vegetation Index (NDVI) measurements. The two classifiers receive input data provided by the computation of eight Vegetation Indices (VI)s. The algorithm has led to accuracy values of 72% and 73% (for the Pattern Recognition System (PRS) according to datasets #1 and #2, respectively, related to the SVMs) and 74% and 79% (for the PRS according to datasets #1 and #2, respectively, related to the DTs). The entire algorithm is totally automated and the classification output is provided from the RPi to a ground station in real-time, by a Wi-Fi socket connection.

Keywords: Precision Agriculture, Unmanned Aerial Vehicle, Vegetation Index, Raspberry Pi, Machine Learning, Support Vector Machines, Decision Trees.

"Do not weep; do not wax indignant. Understand."

— Baruch Spinoza

Contents

List of Acronyms

1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
API	Application Programming Interface
B	Blue
CCD	Charge-Coupled Device
CI	Confidence Interval
COC	Circle Of Confusion
CV	Cross Validation
DOF	Depth Of Field
DOFo	Depth Of Focus
DHWT	Discrete Haar Wavelet Transform
DT	Decision Tree
DWT	Discrete Wavelet Transform
ESC	Electronic Speed Controller
ExG	Excess Green
FPR	False Positive Rate
G	Green

GNDVI	Green Normalized Difference Vegetation Index
GRVI	Green-Red Vegetation Index
GVI	Green Vegetation Index
HSV	Hue Saturation and Value
HW	Hardware
IP	Internet Protocol
LAI	Leaf Area Index
MCA	Multi Camera Array
MRSA	Multi Resolution Segmentation Algorithm
NDVI	Normalized Difference Vegetation Index
NIR	Near-Infrared
OOI	Object Of Interest
OpenCV	Open Source Computer Vision Library
PA	Precision Agriculture
PC	Personal Computer
PRS	Pattern Recognition System
PV	Precision Viticulture
PWM	Pulse-Width Modulation
QE	Quantum Efficiency
R	Red
RC	Radio Control
RBF	Radial Basis Function
RGB	Red Green and Blue
RNDVI	Red Normalized Difference Vegetation Index

ROC	Receiver Operating Characteristic
ROI	Region Of Interest
RPi	Raspberry Pi
SAR	Synthetic Aperture Radar
SAVI	Soil-Adjusted Vegetation Index
SW	Software
SR	Simple Ratio
SSH	Secure Shell
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TPR	True Positive Rate
WLAN	Wireless Local Area Network
UAV	Unmanned Aerial Vehicle
VI	Vegetation Index
V	Visible

List of Figures

List of Tables

1 Introduction

Nowadays, most Precision Agriculture (PA)[1] research is oriented towards the implementation of new sensors and instruments, able to detect crop patterns in real or *quasi* real-time. Discrimination among crop species can be extremely important when one wants to differentiate crop treatments, such as the use of fertilizers, herbicides or even the way crops should be hydrated. Monitoring crops with an Unmanned Aerial Vehicle (UAV) can strongly help farmers to improve their ability to manage vegetation treatments, once the identification of different features can be computed on a daily basis and with results provided in real-time. As a first step, the UAV should understand which is the species that it is looking at. The main goal of this work is focused on this subject, mainly on the identification of trees and vineyards of different species through the use of aerial images and the development of a Pattern Recognition System (PRS) with these data. From the moment vegetation is classified among different species, the final output can be used not only for territorial mapping, but also to compute different Vegetation Index (VI)s which can then be used to compare crop vigour between vegetation that belongs to the same species.

Color spectrum is one of the fundamental properties that define all matter. It is the selective absorption of some electromagnetic wavelengths and the reflection of others that give all matter its characteristic color. Monitoring these visible and invisible wavelengths provides insights into materials composition. An area where these principles are becoming an increasingly important application is in the remote sensing of the Visible (V) and Near-Infrared (NIR) radiation reflected by cultivated crops, forests and other eco-systems.

Since different species reflect distinct radiation on the V and NIR wavelengths of the electromagnetic spectrum, it was developed a low-cost "product", composed by an Raspberry Pi (RPi) NIR camera connected to an RPi and attached to the UAV. A Wi-Fi connection is established between the RPi (attached to the UAV) and the ground station. Images are captured on the UAV and then sent to the ground station, where a PRS is executed and

[1] PA is an approach to farm management that uses information technology to ensure that the crops receive exactly what they need for optimum health and productivity [1].

the final output is generated after a short time span (approximately 1 second). In contrast to most related academic works (see chapter 2), the final output is not a classification mapping for the entire sample. Instead, for each sample, the Region Of Interest (ROI) is identified (defined by the crop to be assessed) and features (VIs) computed. Finally, these features act as input data for the classification task, being then post-processed (the final crop classification is output with an associated probability).

The process of pattern recognition is fully-automated and controlled by the ground station which defines the exact moment samples should be captured and afterwards processed. The user can choose between PRS execution during flight or after UAV's landing. Before classification comes into play, the PRS takes care of the identification of crops (set of pixels which neither belong to the bare-soil nor to non-organic materials). It was assumed that a particular sample (set of {V, NIR} or {Blue (B), Green (G), Red (R), NIR} images) entirely belongs to the same crop species. Therefore, the UAV flew over several farmlands of a single crop species (vineyards and forests of either eucalyptus, pine trees, orange trees, olive trees or magnolias).

Throughout this thesis, few academic works which have used UAVs, and/or Support Vector Machine (SVM)s and Decision Tree (DT)s, for the detection and classification of vegetation were mentioned. Therefore, the description of the used Hardware (HW) was highlighted, as well as a theoretical background which includes a brief explanation of the deterministic and probabilistic machine learning models that were chosen for the classification task. Subsequently, the PRS was clarified and its steps (sensing, segmentation, feature extraction, classification and post-processing) were identified. Finally, experimental results for different segmentation techniques, classification accuracy/precision and the best decision for the choice of classifiers parameters (based on a set of different metrics) were shown.

As a new contribution to crop monitoring, a particular segmentation algorithm for low Depth Of Field (DOF) aerial images was tested offline. Furthermore, it was tested a Wi-Fi configuration between the RPi located in a UAV and a ground station fixed on the ground, which allows real-time classification for the crop vegetation to be sampled, based on two classifiers.

During the course of this work it was exciting to operate with a UAV for the capture/test of data, as well as to study the overall theory that underlies mathematical models for each machine learning technique. Moreover, studying the mathematical core for a PRS using Matlab and making the best of its source code when changing to a programming language (*i.e.*, C++), which allows the computation of real-time results, is always an interesting task.

2 Related Work

This thesis can be extended to several topics, spinning off discussions on a widely set of *cutting edge* works. Therefore, a few academic works were identified, which are associated to the use of UAVs to detect/classify vegetation crops as well as to the usage of either SVMs and/or DTs for the aforementioned subject.

2.1 Unmanned Aerial Vehicles on the Detection and Classification of Vegetation

The use of UAVs for PA was previously tested among distinguished crop classifications (*e.g.*, [2–14]).

D. Turner *et al.* developed an UAV capable of collecting hyper resolution visible, multispectral and thermal imagery for application in Precision Viticulture (PV) [2], sustaining that mapping with UAVs has the potential to provide imagery at an unprecedented spatial and temporal resolution. A thermal infrared camera is used to map soil moisture enabling the assessment of irrigation efficiency, and a six-band multispectral camera enables the calculation of VIs (*i.e.*, Normalized Difference Vegetation Index (NDVI), Simple Ratio (SR) and Leaf Area Index (LAI)) that relate to vineyard vigour and health. The aim is to increase the spatial resolution of data available from "conventional platforms" (such as satellites and manned aircraft) from 20-50 cm/pixel to 1 cm/pixel, due to UAVs capability to fly much lower and hence collect imagery at a higher resolution. This way, the sensory rig can be used to differentiate vigour and health of vineyards.

A. I. de Castro *et al.* proposed an object-based approach for crop row characterization from a list of color-infrared images taken with a UAV for weed management [3], identifies and classifies (plant characteristics were measured through the dimension and spectral properties - NDVI values - of the plant) the crop rows within a maize-crop-field. The ultimate objective is to distinguish small weed seedlings at early stages for in-season site-specific herbicide

treatment, by the execution of a PRS which is computed offline. The developed algorithm has achieved satisfactory results for the rows' detection and count, when compared to on-ground measures of weed emergence.

High resolution images were taken with two "onboard" cameras (Red Green and Blue (RGB) and NIR) [14] providing unsupervised classifications which were performed in order to test the algorithm's ability to distinguish between different bushes and trees species. It was proved that it is possible, with "low cost"[1] instruments, to obtain a classification of different crop vegetation species that can help in identifying specific variety.

Edoardo Fiorillo *et al.* assembled and tested a flexible UAV for PA with the goal of site-specific vineyard management [5]. The system acquired 63 multispectral images during 10 minutes of flight, which were then analyzed and classified vigour maps were produced based on NDVI.

Michaela De Giglio *et al.* analyzed cultivations of vineyards and tomatoes with a multispectral camera (mounted on a UAV) by the creation of "triband orthoimages"[2] of the surveyed sites [7]. Therefore, these data allowed the establishment of different VIs (*e.g.*, NDVI, Soil-Adjusted Vegetation Index (SAVI) and Green Normalized Difference Vegetation Index (GNDVI)) to examine the vegetation vigour for each crop.

The water status variability of a commercial vineyard was assessed by thermal and multispectral imagery using a UAV [10]. Moreover, it was achieved a relationship with thermal imagery and water status parameters that was combined with the computation of different VIs (*e.g.*, NDVI) for mapping the spatial variability of water status within the vineyard.

The idea of detecting single trees was improved for an application on palm plantation [12], by the creation of photogrammetric "point clouds".[3] These points were classified based on the geometric characteristics of the classes (*i.e.*, palm, other vegetation and ground), leading to results which can be seen with high capabilities for operation use, due to their accuracy on the entire study area (which comprised densely scattered growing palms, as well as abundant undergrowth and trees).

G. Bareth *et al.* deepened the idea of building a "mini" UAV (with a total weight lower than 5 Kg - such as the one used in the present thesis) in such a way that the UAV can be suited by a "mini" multispectral camera [13]. The main goal consists in the evaluation

[1] Still higher costs when compared to the work presented in this thesis.

[2] An *orthoimage* is an aerial photograph, geometrically corrected, such that the scale is uniform. The photo has the same lack of distortion as a map [15].

[3] A *Point Cloud* is a set of datapoints in some coordinate system. For the case, points are defined in a Three Dimensional (3D) coordinate system, with x, y and z coordinates [16].

Table 2.1: Different technologies used to capture data.

Reference	Sensor payload
[2]	Tetracam MCA, Canon 550D and FLIR[5]
[3]	Tetracam MCA
[14]	Pentax Optio A40 and Sigma DP1 (with a Foveon X3 sensor)
[5]	Tetracam ADC Lite
[7]	Tetracam ADC Micro
[10]	Tetracam MCA
[12]	Panasonic Lumix G3
[13]	Tetracam mini-MCA

of a sugar beet field experiment examining pathogens and drought stress. The classification method was based on the computation of the NDVI and the assessment of histograms related to the difference among temperatures of distinguished sugar beet vigour.

Each one of the aforementioned studies has the main goal of classifying crops. Despite classification results are used for different purposes, studies demonstrate that it is possible to make such classification from UAV images.

In conclusion, most of the studies exposed in table 2.1 requires the use of a "professional" multispectral camera (*e.g.*, Tetracam). Furthermore, even when multispectral data aren't needed (*e.g.*, [12]) or just two electromagnetic bands (*e.g.*, V and NIR) are required (*e.g.*, [14]), the overall cost remains much higher when compared to the one defined during this thesis.

2.2 Support Vector Machines and Decision Trees for Crop Classification

A particular PA application motivated by the need to estimate crop yield during the growing session "classifies"[6] the structure of grapevines into three "classes" (*i.e.*, leaves, branches and fruit) [17], identified from the color and shape of 3D point clouds. Furthermore, the fruit class is then distinguished between grapes which are prior to ripening and grapes during ripening, by the prediction provided by a SVM.

Arguing that forecasting the grape yield of vineyards is of critical importance to the wine industry, the work [18] estimates which pixels do belong to grapes, by the assessment of NDVI and the correlation between its values and crop yield. The classification stage resorts to the use of either RGB thresholding, color histogram or an SVM.

[5] See `http://www.flir.com/aboutFLIR/` and `http://www.tetracam.com/`.

[6] Classification which results from the segmentation process.

The use of SVMs was tested for crop classification using hyperspectral images [19] and different conclusions were drawn: 1. SVMs yield better performances than neural networks; 2. training neural models is unfeasible when working with high dimensional input spaces.

A framework was developed based on SVMs for crop classification using features provided by a Synthetic Aperture Radar (SAR) [20]. A set of kernel functions (which includes linear, polynomial and Radial Basis Function (RBF))[7] was employed and compared for mapping the input space into a higher dimension space, shown that RBF kernels increases the overall accuracy in 3% in comparison to the use of linear kernel, and up to 1% in comparison to a 3^{rd} degree polynomial kernel function.

The monitoring of sugarcane crops from "spectroradiometer" imagery [21] as well as the assess of different meteorological variables on the propagation of soybean *asian rust* disease [22], were classified resorting to DTs during the prediction stage.

A DT was used to identify crop types in an agricultural area, from multi-temporal images [23]. The separability of sugar beet, tomato, pea, pepper, and rice classes was significantly improved with the use of additional bands (provided by a hyperspectral approach) by the computation of different VIs (*e.g.* NDVI).

Indeed, all the studies mentioned above are just examples of different approaches using either SVMs or DTs during the classification task. As one can see from section 4.5, in this thesis the final output (prediction) is defined taking into account the classification provided by the two machine learning classifiers.

2.3 Multispectral Filter Wheel Approach

The combination of RGB and multispectral imagery for discrimination of *Cabernet Sauvignon* grapevine elements [24] was done by the construction of a *custom-made sensory rig that integrates a Charge-Coupled Device (CCD)*[8] *camera and a servo-controlled filter-wheel* for the acquisition of images during the experimental stage. It proposes a sequential masking algorithm based on the K-means [26], for the classification of image pixels into five clusters: leaves, stems, branches, fruit and background.

The idea behind data acquisition (namely for the multispectral assembly) in the present thesis was achieved by a similar approach to the one mentioned in [24].

[7] Set of kernels also tested during the present thesis.

[8] A *CCD* is a semiconductor sensor to capture images, formed by an integrated circuit containing an array of capacitors. Each capacitor can transfer energy to another neighbor capacitor [25].

3 Materials and Methods

3.1 Hardware Description

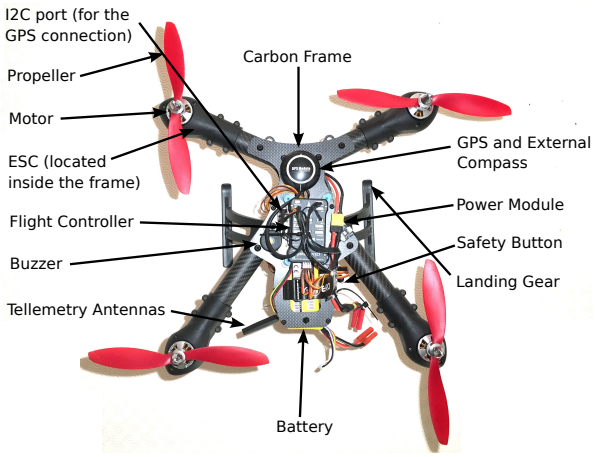
The UAV model (Sky Hero[1] Little Spyder SK00-104-RTF) is manually controlled by a Radio Control (RC) (FrSky Taranis X9D Plus 16CH RC Transmitter). The flight controller (3DR Pixhawk, aided by a 3DR GPS module) has got an external compass, and uses a barometer which measures air pressure as the primary means for determining altitude. Locating the throttle stick inside the "mid-throttle deadzone" (40% $\sim$ 60% of throttle), the quad-rotor maintains the altitude. Otherwise, outside this zone, the quad-rotor descends at 2.5 m/s (when the stick is completely down) or climbs at 2.5 m/s when it is located at the very top. These values are obtained by the tunning of the following set of variables, changed with the use of Mission Planner[2] Application Programming Interface (API): `Altitude Hold P` (converts the difference between the desired altitude and the actual altitude to a desired climb or descent rate), `Throttle Rate` (converts the desired climb or descent rate into a desired acceleration up or down) and `Throttle Accel PID` gain (converts the acceleration into a motor output).

Quad-rotor's firmware was downloaded from Mission Planner API. Furthermore, both compasses (internal compass and the one provided by the 3DR GPS module) as well as the RC were calibrated with Mission Planner API, by the use of "Mandatory Hardware Wizard Setup" option, provided by this API. The RC has assumed a set of 6 different channels: 1. Channel 1 (Roll); 2. Channel 2 (Pitch); 3. Channel 3 (Throttle); 4. Channel 4 (Yaw); 5. Channels 5 and 6 are both called "flight modes". The set of Electronic Speed Controller (ESC)s (T-Motor Air series 40 A, designed for operation on 2-6s Li-Po batteries) was calibrated by a sequence of steps as follows: 1. Radio control is turned on; 2. Throttle is set to maximum; 3. Li-Po battery is connected to power module (here, a cyclical pattern of red, blue and yellow LEDs should appear on the flight controller); 4. With the transmitter

[1] See `http://www.sky-hero.com/en/`.

[2] Software available at: `http://ardupilot.org/planner/index.html`

(a) Quad-rotor fitted with the "multispectral" camera.



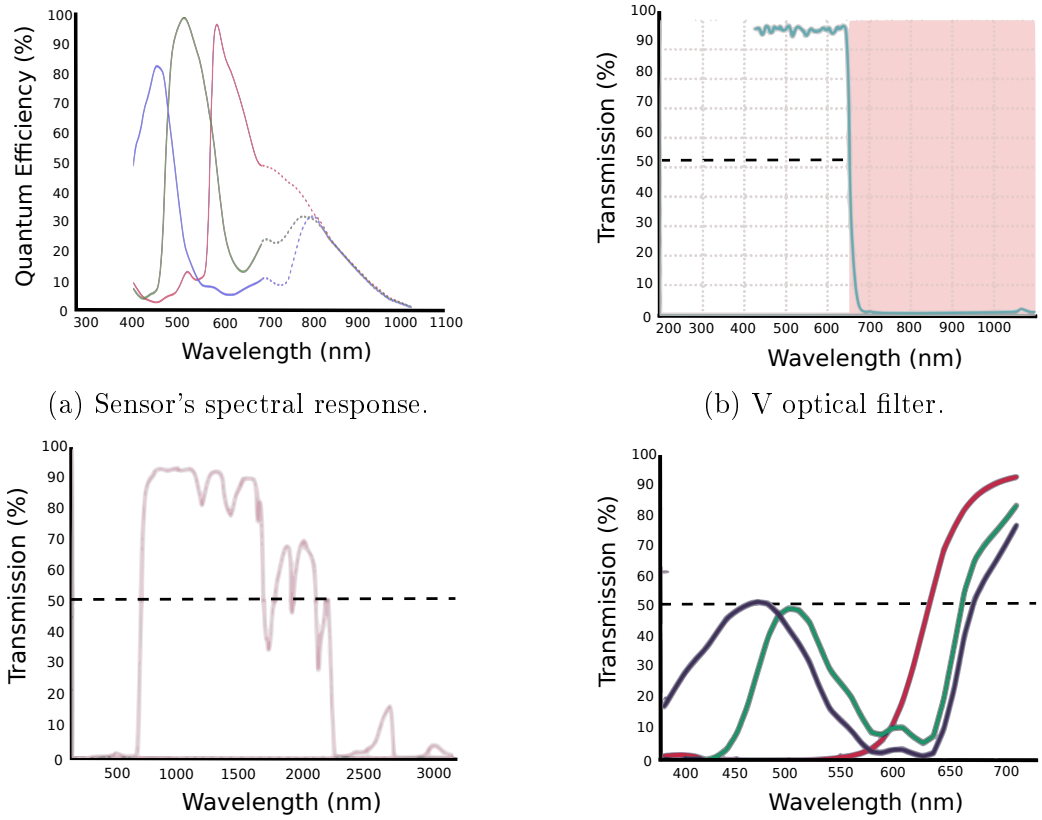
(b) Quad-rotor's main components.

Figure 3.1: Illustration of the used UAV.

throttle stick still high, the battery should be disconnected and afterwards reconnected; 5. The safety button (which is connected to the flight controller) should be pressed; 6. The flight controller should emit a musical tone composed by a regular number of beeps indicating battery's cell count (*i.e.*, 4 beeps) and then an additional 2 beeps signal to indicate that the maximum throttle has been captured; 7. Transmitter's throttle stick should be pulled down; 8. The flight controller should then emit a long tone indicating that the minimum throttle has been captured; 9. Finally, the battery should be disconnected to exit ESC-calibration mode.

The quad-rotor (see figures 3.1 and 3.4a) is equipped with 4 motors (Sky Hero 2806, 950 kV) and it is power supplied by a 4S (four cells) (3700 mAh, 14.8 V, Zippy Compact 25C Series) Li-Po battery. Batteries are charged with a synchronous balance charger/discharger (iCharger 406 DUO), at a rate of 5C ($5 \times 3700 = 14.8$ A), and the iCharger is supplied by a switched-mode power supply (1 output, RSP-3000-24, Mean Well) which allows output current up to 125 A. Depending on the farmland to be classified (vineyards or crop forests), the UAV has flown up to an height of either 3 or 20 meters, respectively, above the ground level. The UAV's takeoff occurred in *Stabilize* flight mode and so it remains until the moment when the target altitude is reached. Here, the flight mode is changed to *Altitude Hold* with the specifications mentioned above. Each flight, for data collection or even when tests were done, occurred over time spans which didn't exceed periods of 5 minutes (for security reasons the aim was to operate the UAV to a minimum of 60% of the batteries' power capacity).

The multispectral design (see figure 3.3 for the final assembled structure) was based on the concept of capturing different radiation (differentiated wavelengths) with the use of



(a) Sensor's spectral response.

(b) V optical filter.

(c) NIR optical filter.

(d) RGB optical filters.

Figure 3.2: Technical information (spectral responses) for different optical filters and camera's sensor.

distinct optical filters laid in front of the lens of an 8-MegaPixel camera board (Raspberry Pi NoIR Camera Module v2 - *Raspicam*). The camera has got a SONY IMX219 sensor and its spectral response can be seen[3] in figure 3.2a in terms of its Quantum Efficiency (QE).[4] The Raspicam was calibrated with a Matlab application (Single Camera Callibration App) in order to compute camera's intrinsic parameters ($f_x = 2768.2$ pixels, $f_y = 2765.7$ pixels). Consequently, since the size s of each pixel is 1.12 $\mu m \times 1.12$ μm ($s_x = s_y$), the focal length,[5] f, is (approximately) equal to 3.1 mm. Samples were captured by the use of either the combination of RGB with NIR or V with NIR optical filters (for the last option, RGB channels were split by the camera itself allowing the capture of data as exemplified in figures 3.4b and 3.4c). The selection of these filters (see figure 3.2 for spectral responses) is based on the fact that all photosynthetic plants, including grapevines and trees, are characterized by

[3] From figure 3.2a, one can see an estimation for the sensor's spectral response on the NIR band (dashed line) suggested by [27].

[4] QE is the number of signal electrons created per incident photon (QE>100% when more than 1 electron is created per incident photon [28]).

[5] The focal length (f) is defined by: $\begin{cases} f_x = f \times s_x \\ f_y = f \times s_y \end{cases}$.

Figure 3.3: Multispectral configuration (view from different sides).

a low reflectance in red wavelengths ([570, 700] nm [29]) because chlorophyll absorbs much of the incident energy for the photosynthesis. On the other hand, in the NIR wavelengths ([700, 950] nm [29]) *photosynthesising* plants reflect large proportions of the incident sunlight [30]. The remaining optical filters (*i.e.*, green and blue) were used to allow the computation of different features (*i.e.*, VIs - see section 4.4).

The RPi is power supplied by a 2S (two cells - 850 mAh, 7.4 V Turnigy nano-tech) Li-Po battery, where the voltage is adjusted by a module regulator (DC-DC LM2596 step down adjustable converter). The Wireless Local Area Network (WLAN) is established by the use of both an access point (Tube 2H, Alfa Network Inc., 150 Mbps, 2.4 GHz) and a USB Wi-Fi (150 Mbps) adapter connected to the RPi. Finally, a servo (TowerPro SG90 Mini Gear Micro Servo 9g) allows the filters' wheel to rotate, according to Pulse-Width Modulation (PWM) signals sent from the RPi.

Concerning the algorithms that were developed during this thesis, the RPi (1.2 GHz quad-core ARMv8 CPU) hasn't got enough processing capacity to provide real or *quasi* real-time results. Consequently, the main goal of using the RPi is only based on the way data are collected and afterwards sent to the ground station (Personal Computer (PC) - ASUS K450J, quad-core 2.4 GHz CPU) to be processed. However, an alternative to the use of the RPi and its camera was considered. Here, the Odroid XU-4 (2 GHz Cortex-A7 octa-core CPU) was tested with a Kurokesu (USB camera C1) attached. Nevertheless, due to the size of the Kurokesu camera (and subsequently the increase of the filters' size), this option was excluded from practice as well as the idea of processing data out of the ground station.

(a) UAV during flight. (b) V image. (c) NIR image.

Figure 3.4: Set of images related to UAV's flight.

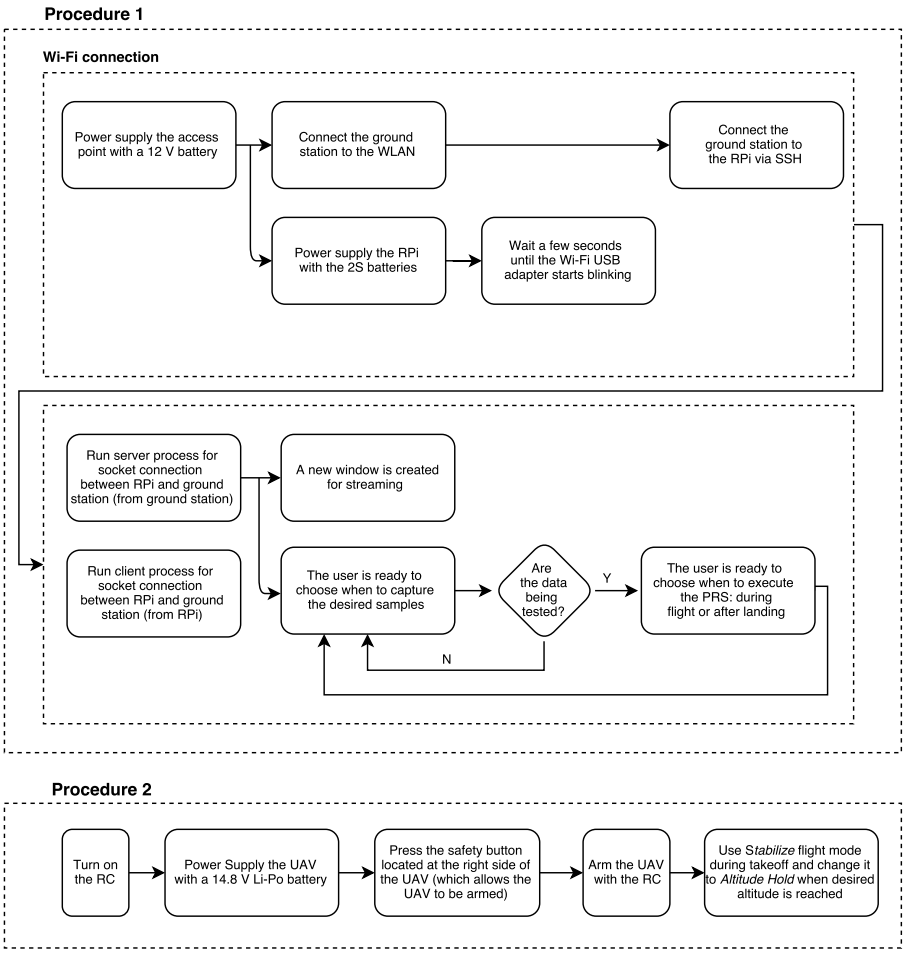


Figure 3.5: Sequence of steps required before flight.

Before any flight session, the UAV, the RPi, and the ground station must be subjected to different configurations suggested by a set of 2 procedures which are illustrated on diagram mentioned above (see figure 3.5).

3.2 Theoretical Background for Classification

The main difficulty of the classification process results from the variability of the feature values for objects in the same category, as well as the similar aspects among feature values for

objects in different categories. This work attempts to classify different crop species, proving that features mentioned in 4.4 provide reliable measurements to the problem mentioned above.

In the broadest sense, any method that incorporates information from training samples in the design of a classifier requires learning. These pattern classification algorithms were taught by a supervised [31] learning model, providing a category label for each pattern in a training set. Thus, a large set of images for different classes (*e.g.*, samples for several species) is used to compute several VIs which are then used as the input for the classification algorithm during the training stage. Considering X as the set of input variables (*e.g.*, VIs) for the entire training dataset and Y as the set of output variables (*e.g.*, crop's species), the output quality of a supervised model is based on the learning of the mapping function, f, from the input to the output ($Y = f(X)$). Learning stops when the algorithm achieves an acceptable level of performance (as mentioned in section 3.2.3). On the other hand, unsupervised learning models only have input data, X, and no corresponding output variables are generated. The main objective is to model the underlying structure or distribution in the data in order to learn more about it without being previously taught. Unsupervised learning can be further grouped into clustering (where one wants to discover the inherent groupings in the data - such as the definition of Object Of Interest (OOI) or non OOI by K-means algorithm, mentioned during segmentation technique 4.3.2) and association problems (where one wants to discover rules that describe large portions of data).

It is also important to clarify that the algorithm is not in continuous learning: after the training stage, a pattern classification method provides the final output, identifying, for instance, the specific type of crop (testing stage) and not providing its result as a feedback for future classification.

3.2.1 Support Vector Machines

SVMs [31–33] (see table 3.1) rely on pre-processing the data to represent patterns in a high dimension. They are based on the Structural Risk Minimization [33] principle from computational learning theory. The idea of structural risk minimization is to find a hypothesis, h, for which one can guarantee the lowest true error. The true error of h is the probability that h will make an error in an unseen and randomly selected test example. Considering H as the hypothesis space of which h belongs, SVMs find the hypothesis h which minimizes the bound on the true error. One remarkable property of SVMs is that their ability to learn

can be independent of the dimensionality of the feature space (F). SVMs measure the complexity of hypothesis based on the margin with which they separate the data. Indeed, SVMs do not rely on measuring the complexity of hypothesis based on the number of features. Therefore, one can generalize even in the presence of many features. In fact, classification performance was shaped according to the increase/decrease of several features (final results, shown in chapter 5, are classified according to the set of features mentioned in 4.4). Graphically (see figure 3.6a and example 1), one can see the concept of SVMs in the separation of data into classes according to the plane which maximizes the margin between the classes' closest points (hyperplane). The margin is optimized to reduce the number of weights that are nonzero to just a few ones (called Support Vectors)[6] that correspond to the important features that matter in deciding the separating hyperplane.

Each instance in the training set contains one "target value" (*i.e.*, the class labels) and several "attributes" (*i.e.*, the features). As discussed, SVM's goal is to produce a model (based on the training data) which predicts the target values of the test data considering only the test data attributes. In other words, a training set is composed by l pairs of *(feature, label)*, $(x_1, y_1), ..., (x_l, y_l)$, where $x_i \in R^n, i = \{1, ..., l\}$ ($n \equiv$ # of features $\equiv$ # of VIs $= 8$, see section 5.4) and $y \in \{1, ..., k\}$ ($k = \{5, 3\}$, depending on the dataset used for the PRS - see section 5.4) is the class of x_i. The resulting output consists in a set of weights (w_i), one for each feature, whose linear combination predicts the value of y. The hyperplane $H0$ (*e.g.*, *line* for the Two Dimensional (2D) case and *plane* for a feature space with dimension greater than 2) and its margin (distance between $H1$ and $H2$) can be defined[7] as suggested in equations (3.1), (3.2) and (3.3) (considering that $F \in R^2$). Furthermore, knowing that the distance (d) between H0 and H1 is represented as mentioned in equation (3.4), one should minimize $||w||$ to maximize the margin (2× d) between classes.

$$H1 : \mathbf{x} \cdot \mathbf{w} + b = 1 \quad (3.1) \qquad H0 : \mathbf{x} \cdot \mathbf{w} + b = 0 \quad (3.2) \qquad H2 : \mathbf{x} \cdot \mathbf{w} + b = -1 \quad (3.3)$$

$$d = \frac{|\mathbf{x} \cdot \mathbf{w} + b|}{||\mathbf{w}||} = \frac{1}{||\mathbf{w}||} \tag{3.4}$$

Moreover, assuming that there are no data points between H0 and H1, the set of two

[6] These weights are called Support Vectors because they "support" the separating hyperplane.

[7] $w \cdot x \equiv w^T x \equiv < w, x > :=$ Inner product between vectors of weights and input features, respectively.

inequalities mentioned above can be simplified as follows:

$$\left.\begin{array}{ll} \mathbf{x_i} \cdot \mathbf{w} + b \geq 1, & \text{if } y_i{=}1 \\ \mathbf{x_i} \cdot \mathbf{w} + b \leq -1, & \text{if } y_i{=}\text{-}1 \end{array}\right\} \equiv y_i(\mathbf{x_i} \cdot \mathbf{w}) \geq 1$$

Table 3.1: Important terminology related to SVMs.

Terminology	**Description**
Hyperplane	Margin between classes' closest points.
Support Vectors	Points lying on the boundaries.
Type of SVM	Considering only multi-class classification, SVMs can be either of type *C-SVM* or *nu-SVM*. For the later type, a *nu* parameter defines an upper bound on the fraction of margin errors and a lower bound of the fraction of support vectors relative to the number of training examples. Considering $nu = x\%$, one guarantees that $x\%$ of the training examples are at most misclassified and $x\%$ of the training examples are at least support vectors.
Nonlinearity	When one cannot find a linear separator, data points are projected into a higher-dimensional space where the data points effectively become linearly separable (this projection is realised via *kernel* techniques).
Kernel	It is a similarity function, which enables SVMs to operate in a high-dimensional implicit feature space, without computing the coordinates of the data in that space but rather by simply computing the inner products between the images of all pairs of data in the feature space. For the case, one can choose between *linear*, *polynomial* or *RBF* kernels (see table 3.2 and figure 3.6b).
Overlapped Classes	Data points on the "wrong" side of the discriminant margin are weighted down to reduce their influence (*soft margin*).
Multi-class Classification	Basically, SVMs can only solve binary classification problems. To allow for multi-class classification, `libsvm` (SVM's library chosen to work with during this thesis) uses the *one-against-one* technique by fitting all binary subclassifiers and finding the correct class according to a voting strategy.
One-against-one	Assuming that k is the number of classes, there are $Kc = \frac{k \times (k-1)}{2}$ different ways of combining binary classifications. This technique creates Kc classifiers where each one is trained on data from two classes. If[8] $sign(f(x))$ says x is in the i^{th} class, then the vote for the i^{th} class is added by one (otherwise, the vote for the j^{th} class is increased by one).

This way, one should solve a quadratic[9] minimization problem (which can be solved by

[8] $sign(x) = \begin{cases} 1 & \text{if x}{>}0 \\ 0 & \text{if x}{=}0 \\ -1 & \text{if x}{<}0 \end{cases}$

[9] Because it is quadratic, the surface is a paraboloid (avoid local *minima*).

the Lagrangian multiplier method), as follows:

$$\begin{aligned} \min \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to} \quad & [y_i(\mathbf{x_i} \cdot \mathbf{w}) - b] - 1 = 0. \end{aligned}$$

The idea is to find the intersection of two functions, f and g, at a tangent point (which is a minimum or a maximum for f, subjected to the constraint g). Hence, one should look for the solution of the following problem (L is a Lagrangian function which includes a new variable α):

$$\begin{cases} L(x,\alpha) = & f(x) - \alpha g(x) \\ \nabla(x,\alpha) = 0 \end{cases} \equiv \text{ (in general) } L(x,\alpha) = f(x) + \sum_i \alpha_i g_i(x),$$

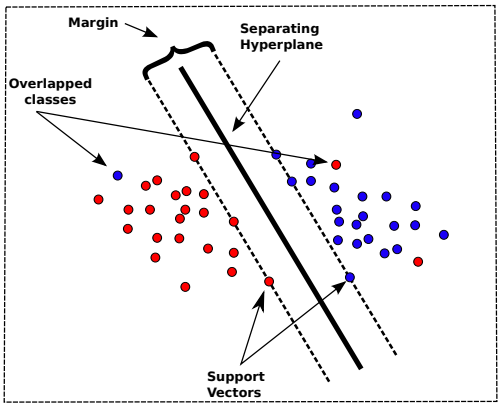
Considering l training points, the **primal** form of the optimization problem can be seen as:

$$\min \quad L = \frac{1}{2}||w||^2 - \sum_{i=1}^{l} \alpha_i y_i (x_i \cdot w + b) + \sum_{i=1}^{l} \alpha_i.$$

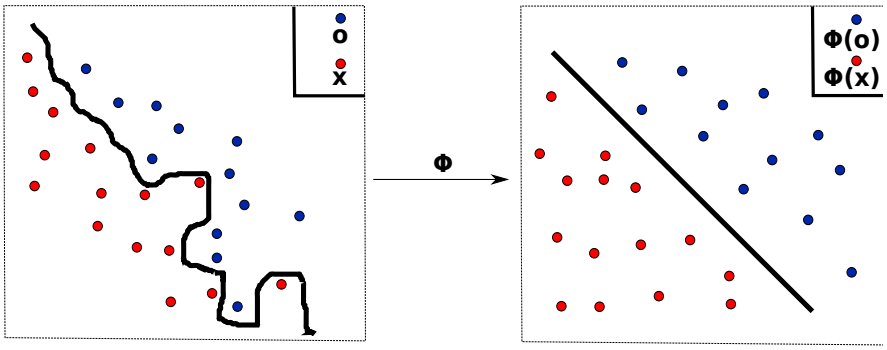
$$\frac{\partial L_p}{\partial w} = w - \sum_{i=1}^{l} \alpha_i y_i x_i = 0 \equiv w = \sum_{i=1}^{l} \alpha_i y_i x_i \qquad (3.5) \qquad \frac{\partial L_p}{\partial b} = w - \sum_{i=1}^{l} \alpha_i y_i = 0 \qquad (3.6)$$

The Lagrangian method partially derivatives (see equations (3.5) and (3.6)) the primal problem with respect to w and b. Accordingly, instead of minimizing over w and b (subject to constraints involving α), one should maximize over α (subject to the relations obtained for w and b). This way, dependencies of w and b are no longer required, and the problem can now be solved by computing the inner products between x_i and x_j (features for the i^{th} and j^{th} classes). The resultant **dual** optimization problem is represented as follows:

$$\begin{aligned} \max \quad & L(\alpha_i) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \\ \text{subject to} \quad & \sum_{i=1}^{l} \alpha_i y_i \alpha_i \geq 0, \forall i. \end{aligned}$$



(a) Illustration scenario.



(b) The "Kernel trick".

Figure 3.6: SVMs trivial scenario.

Finally, knowing α_i, one can find the weights (w) for the maximal margin hyperplane (*training* stage) as suggested in equation (3.5). The prediction of the class label (y) for a given unknown data point u is computed by the assessment of $sign(f(x))$ (see description for *one-against-one* in table 3.1), with $f(x)$ defined according to equation (3.7).

$$f(x) = w \cdot u + b = (\sum_{i=1}^{l} \alpha_i y_i x_i \cdot u) + b \tag{3.7}$$

However, when patterns are not linearly separable, data are mapped into a higher dimensional feature space, through the use of Kernel functions, defined by: $K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$ (see table 3.1 for the *Kernel* definition).

Table 3.2: Different kernels used on the PRS, for the two datasets, as mentioned during section 5.4.

Type of Kernel	**Formula**[10]
Linear	$K(x_i, x_j) = x_i^T x_j$
Polynomial	$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$
RBF	$K(x_i, x_j) = e^{-\gamma \|\|x_i - x_j\|\|^2}, \gamma > 0$

[10] γ, r and d are kernel parameters (hyperparameters).

Example 1. Imagining that in fact a pair of features from the set provided in 4.4 (*e.g.*, NDVI and GNDVI - $x_i \in R^2$) is a powerful tool when distinguishing (for instance) Eucalyptus (●) from Pine trees (●). One should also take into account that the feature space is only composed by these data. Thus, a trivial case for such classification can be seen as the one suggested in figure 3.6.

3.2.2 Decision Trees

DTs [31,34,35] (see table 3.3) are a non-parametric[11] supervised learning method. They have their basis on classifying a pattern through a sequence of questions in which the next question asked depends on the answer to the current question. This approach is particularly useful for non-metric data, because all of the questions can be asked in a "yes/no", "true/false" or "value (property) ∈ set of values" style that does not require any notion of metric. Such a sequence of questions is displayed in a DT, where the root (first) node is displayed at the top (for the DT presented in figure 3.7, the root node is displayed at the left just for the sake of text formating), connected by successive links to other nodes that are similarly connected until reaching the leaf nodes (which have no further links).

Considering once more a training set composed by l pairs of *(feature, label)* ($x_i \in R^n, i = \{1, ..., l\}$ and $y \in \{1, ..., k\}$ - see section 3.2.1), a decision tree recursively partitions the dataset such that the samples with the same labels are grouped together. Let the data at node m be represented by Q. For each split candidate, $\theta = (j, t_m)$, consisting of a feature j and a threshold t_m (limit LIM, exemplified in figure 3.7), the data are partitioned[12] into $Q_{left}(\theta)$ and $Q_{right}(\theta)$ subsets:

$$\begin{aligned} Q_{left}(\theta) &= (x, y) \mid (x_j <= t_m), \\ Q_{right}(\theta) &= Q \setminus Q_{left}(\theta). \end{aligned}$$

The algorithm finds the "best" split of data by the construction of binary trees using the θ that yield the largest information gain at each node (*i.e.*, which generate the most homogeneous

[11] Non-parametric (or distribution-free) refers to data which do not assume a specific distribution. Parametric data are defined by a normal distribution [36].

[12] $A \setminus B$ consists of all elements of A which are not elements of B (generic example).

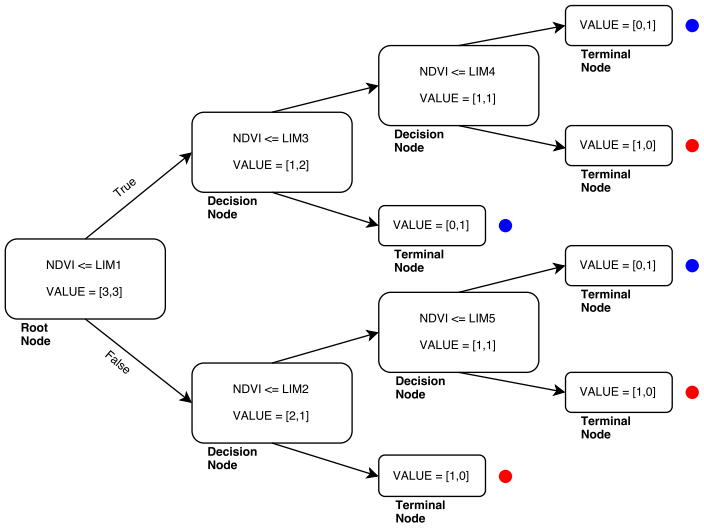


Figure 3.7: DTs trivial scenario.

branches). The *impurity* G at m is computed[13] using an impurity function H:

$$G(Q,\theta) = \frac{n_{left}}{N_m} H(Q_{left}(\theta)) + \frac{n_{right}}{N_m} H(Q_{right}(\theta)),$$

and the parameters (j^*, t_m^*) are then selected, according to the ones that minimize the impurity:

$$\theta^* = \text{argmin}_\theta G(Q,\theta).$$

Until the maximum allowable depth is reached (or N_m reaches a state where its value is lower than the minimum allowable samples at a leaf node) $Q_{left}(\theta^*)$ and $Q_{right}(\theta^*)$ are recursively computed. Taking on values $\{0, ..., k-1\}$ for the different classes at node m and representing a region R_m with N_m observations, let

$$p_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

be the proportion of class k in node m. Finally, the two *criterion* chosen for measuring the impurity were the *gini* and *entropy* defined, respectively, as follows:

$$H(X_m) = \sum_k p_{mk}(1 - p_{mk}),$$

$$H(X_m) = -\sum_k p_{mk} log(p_{mk}).$$

Taking into account a similar scenario (for this case, $x_i \in R^1$) to the one illustrated on

[13] n_{left}, n_{right} and N_m are, respectively, the number of nodes in the left and right subsets, and the sum of the number of left and right nodes, according to the data at a particular node m.

Table 3.3: Important terminology related to DTs.

Terminology	Description
Root Node	It represents the entire sample.
Splitting	Process of dividing a node into 2 or more sub-nodes.
Decision Node	Sub-node that is split into further sub-nodes.
Terminal Node	Sub-node that is not split.
Impurity	It is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly classified according to the distribution of labels in the subset.
Criterion	Defines how will attributes be selected for splitting.
Maximum Depth	Parameter used to restrict the size of the DT (to prevent over-fitting).
Maximum Features	Maximum # of features to consider when looking for the best split.
Minimum Samples to Split or Minimum Samples per Leaf	Minimum # of samples required to split a decision node.

example 1, a trivial case for DTs can be as similar as the one suggested in figure 3.7. Here, a set of 3 samples was considered for both Eucalyptus (●) and Pine trees (●).

3.2.3 Finding the Best Classifier

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake. A model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data (this is called *overfitting* [35]). To avoid it, it is common practice when performing a supervised machine learning experiment, to hold out part of the available dataset as a test set.

Different estimators (such as SVMs and DTs) require an evaluation of several settings (so-called *hyperparameters* [35]). For instance, C setting should be evaluated manually for SVMs. Thus, there is still a risk of overfitting on the test, because the parameters can be tweaked until the estimator performs optimally. To solve this problem, another part of the dataset can be held out as a so-called *validation set*: training proceeds on the training set, after which evaluation is done on a validation set and then, when the experiment seems to be successful, final evaluation can be done on the test set. At this point, another problem arises. For datasets which are not composed by a huge number of samples (for the case, just a few hundred samples were used) and by partitioning the available data into 3 sets, one drastically reduces the number of samples which can be used for learning the model, and the

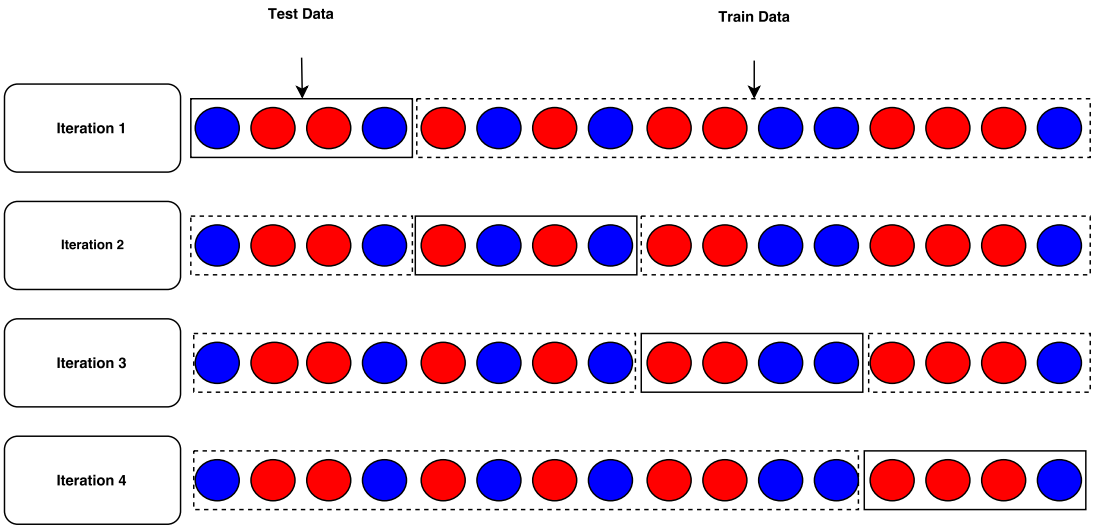


Figure 3.8: CV trivial scenario.[14]

results might depend on a particular random choice for the pair *(train, validation)* sets. A solution to avoid this problem is a procedure called Cross Validation (CV) (see figure 3.8). A test set should still be held out for final evaluation, but the validation set is no longer needed. For the *k-fold* CV approach (which was taken into account in this thesis), the training set is split into k smaller sets. For each of the k *folds*, the following procedure is ensued:

1. A model is trained using $k-1$ of the folds as training data;
2. The resulting model is validated on the remaining fold - *i.e.*, it is used as a test set to compute model's accuracy.

In conclusion, the performance measure reported by *k-fold* cross-validation is then the average of the values computed in the loop. Despite being computationally expensive (when compared to the method where data are split into 3 sets), this procedure does not waste too much data (which is a major priority in this work, due to the low number of available samples). It is also important to report that CV was just used to better understand which is the model that better suits the data provided for training stage. After achieving the "best" hyperparameters for the model, test data come then into play. Both SVM and DT libraries allow the inclusion of CV during training stage.

SVMs CV took into account the following specifications:

- *Type*: C-SVM (**0**) or nu-SVM (**1**);
- *Kernel Type*: Linear, (**0**) Polynomial (**1**) or RBF (**2**);

[14]Trivial example where $k = 4$ and just 2 classes considered.

- *Degree* (**D**, for polynomial kernel);
- **Γ** (for polynomial or RBF kernels);
- *Cost* (**C** for C-SVM type) and **nu** (for nu-SVM type).

The following specifications were taken into account for DTs CV:

- *Criterion*: gini (**0**) or entropy (**1**);
- *Maximum Depth*: **5**, **7**, **9**, **11** or *None* (**N**);
- *Minimum Samples to Split*: **1**, **2**, **3**, **4** or **5**;
- *Minimum Samples per Leaf*: **1** or **2**.

The PRS was evaluated according to the following approaches:

- Scoring parameter (model-evaluation tool using CV - assessing of *accuracy* for CV):
  - The k results from the folds are averaged to produce a single estimation for *accuracy*.
- Metric functions (set of functions assessing prediction errors: *precision*, *recall*, *accuracy* and Receiver Operating Characteristic (ROC) curve - relation between True Positive Rate (TPR) and False Positive Rate (FPR)):

$$\begin{aligned} TPR = Recall &= \frac{Tp}{Tp + Fn}, \\ FPR &= \frac{Fp}{Fp + Tn}, \\ Precision &= \frac{Tp}{Tp + Fp}, \\ Accuracy &= \frac{Tp + Tn}{Tp + Tn + Fp + Fn}. \end{aligned}$$

  These functions use the following set of statistical measurements for the computation of metrics:

  - True Positive (Tp): # of samples which belong to a class (*e.g.*, *class 1*) and are classified as *class 1*;
  - True Negative (Tn): # of samples which do not belong to *class 1* and are classified as "not" class 1 ($\overline{class\ 1}$);

- False Positive (Fp): # of samples which belong to $\overline{class\ 1}$ and are classified as *class 1*;
- False Negative (Fn): # of samples which belong to *class 1* and are classified as $\overline{class\ 1}$.

In conclusion, the final "optimized" models for the PRS were achieved according to the following enumerated items (for item 1. it was assigned the highest priority when choosing classifier's configuration, whereas item 3. has the lowest priority):

1. Choice of the minimum value for the Euclidean distance between the correspondent point on the ROC curve and the *sweet spot* ($\text{TPR} = 1, \text{FPR} = 0$);
2. Choice of the configuration which results on the best CV accuracy;
3. Choice of the configuration which results on the best precision.

Finally, the accuracy was used to compute the probability associated to the output's classification (see section 4.5) during test predictions.

4 Pattern Recognition System

This chapter describes how pattern classification algorithms were developed, some mathematical background (namely for segmentation techniques and VIs computation) and how the system processes the entire information (from *sensing* to *post-processing*). Deep clarifications according to each element mentioned in figure 4.1 are provided. It is also important to clarify that pattern classification differs from image processing. In image processing, the input and output are both images and it often includes transformations which preserve all the original information. On the other hand, feature extraction (such as the work done in this thesis) loses information and preserves everything relevant to the task at hand.

4.1 Introduction

In describing a crop classification system, distinctions were made among pre-processing (identification of the ROI), feature extraction, classification and post-processing (assessing of the final class according to the output provided by the two classifiers). Figure 4.1 shows a slightly more elaborate diagram of the components considered in a PRS. In fact, to design such a system, some problems can emerge. To understand the problem of designing this system, a short introduction to each component is considered.

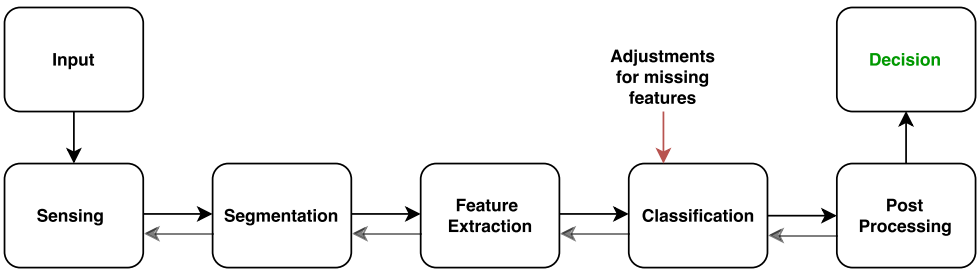


Figure 4.1: The PRS.

When *sensing*, the input of this PRS is a camera. Here, the difficulty of the problem depends on the camera characteristics, such as its resolution and the accuracy required by its sensor to detect electromagnetic radiation within the required wavelength span. Since the

idea was to develop a low-cost mechanism, it was very hard to find cameras on the market which can be mounted into a multispectral mechanism.

Once images are taken, patterns need to be segmented. A way to distinguish vegetation leaves from bare soil, non-organic materials, or even leaves which are from other plants, can be extremely hard to do if the radiation captured is similar (radiation which belongs to identical wavelength bandwidths) among different objects. Segmentation is one of the deepest problems in pattern recognition and was the hardest point to cope with when elaborating this work.

The concept boundary between *feature extraction* and *classification* is somewhat arbitrary: an ideal feature extractor would yield a representation that makes the job of the classifier trivial; conversely, an omnipotent classifier would not need the help of a sophisticated feature extractor. The distinction is forced upon us for practical, rather than theoretical reasons [31]. The main idea of feature extraction is to characterize an object in such a way that its measurements can be used to distinguish it from non similar objects. For instance, different VIs were used during this work, which relate the different radiation captured on the same image (sample). Once the ROI is well segmented, each VI is computed. As a result, their mean values are used as input data for the classification task.

The *post-processor* uses the output of both classifiers to assess which is the best decision (*i.e.*, which is the classification that suits the higher probability). If all classifiers agree with a particular pattern, there is no difficulty. Otherwise, how could this combination of classifiers achieve the best decision (see section 4.5)?

4.2 Sensing and Data Collection

Data collection can surprisingly account for the large amount of the cost of a PRS development. It may be possible to perform a preliminary feasibility PRS with a small set of typical examples, but much more data are usually needed to ensure a good performance in the field. In fact, a small dataset was used in the beginning, so that one can be able to test the entire algorithm. Afterwards, more data were needed to achieve higher accuracies when classification occurred, since the algorithm training set has increased.

Throughout this work, two different ways of capturing data were considered (*i.e.*, *Data Collection #1* and *Data Collection #2*). Since the main goal consists in equipping a UAV, the size of the "multispectral mechanism" became a crucial variable. On account of that and despite being later described, Data Collection #2 was only used during an embryonic phase

of the project. For both data collections procedures, a ground station (in the case, a PC) establishes a Wi-Fi connection to either 1 or 2 RPi.

4.2.1 Data Collection #1

The purpose of this data collection (see figure 4.2) is to use the least HW and at the same time make it possible to collect the desired bands of the electromagnetic spectrum. This is a non snapshot technique (a couple of seconds is needed) based on filter shifting, providing a total of N images per sample ($N = \{2, 5\}$ depending on the segmentation algorithm used). Computations based on data provided by either 5 optical filters (B, G, R, NIR and V) or 2 optical filters (V and NIR) were tested (namely during pre processing).

The ground station establishes a connection to 1 RPi via Secure Shell (SSH), over a WLAN provided by a portable modem. The RPi is previously configured for booting with a specific Internet Protocol (IP) address (192.168.4.194). A client-server interprocess communication is used between both the RPi and the PC, respectively. The client (RPi) establishes a Transmission Control Protocol (TCP) socket connection to the server (PC), sending an image stream from the RPi to the PC (socket A). Streaming is done according to a resolution reducing factor of $\frac{1}{16}$ (decreasing the amount of data to be transmitted which allows real-time image transmission). At the same time and from another socket, a message is continuously sent towards the ground station, saying whether the streaming is in high or low resolution (socket B). This way, the ground station can choose in real-time between "streaming" (low resolution) or capturing desired samples (high resolution). Every time the ground station decides to capture a sample, the filter which is attached in front of the camera's lens changes its position, allowing the capture of the same sample in different bands. This control is based on PWM signals, sent by the RPi to a servo. For each sample, it is considered that all images are overlapped with each other (remark DC1).

4.2.2 Data Collection #2

Data collection #2 was tested and aims to guarantee that images for the same sample are matched correctly. Despite its implementation on the UAV was not considered when data were collected, its description is mentioned since one can see it as an alternative for Data Collection #1.

This data collection is reached by 2 RPi (*e.g.*, RA and RB) and a ground station, all connected to the same WLAN and both RPi give access to the ground station via SSH. One

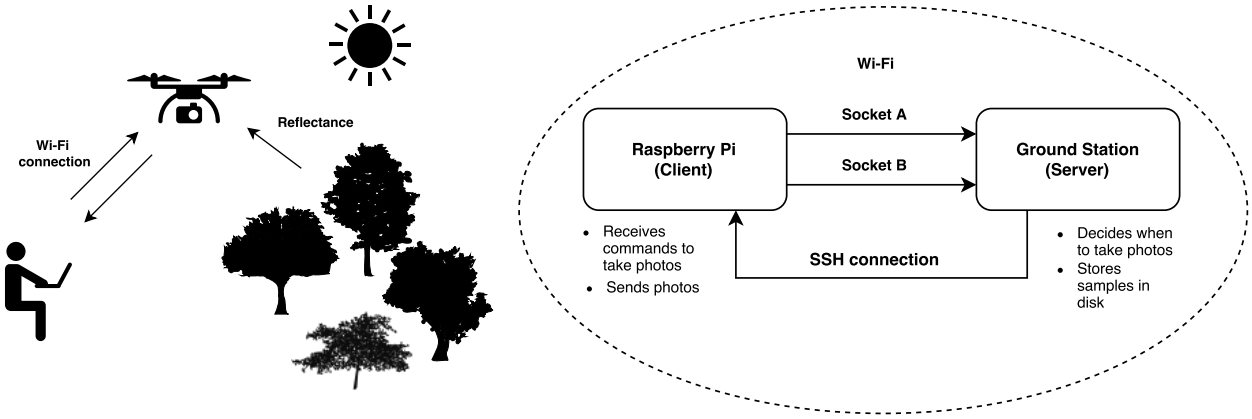


Figure 4.2: Data collection #1.

should also consider that a different camera is connected to each RPi (stereo configuration). RA has a "V" camera (which has got a NIR block filter between its sensor and lens) connected, allowing RGB spectral data. On the other hand, RB has a "NIR" camera (which hasn't a NIR block filter) connected, with a short-pass filter (blocks V and higher frequency radiation) attached, allowing the capture of NIR data.

RA works with the same logic of RPi from data collection #1, working with both sockets A and B. RB just streams data (high resolution NIR image) to the ground station (socket C), when receiving that order. Both messages (orders to take photos) from ground station to RA and RB are sent at the same moment.

Stereo calibration was made using Matlab (Stereo Calibration App). Once Data Collection #2 requires more HW and knowing that the final disparity between points on the left and right cameras may lead to errors (which might not be smaller when compared to remark DC1), Data Collection #2 was excluded from practice.

4.3 Segmentation

In this work, three segmentation processes were considered, which are either based on color or frequency data, provided by each set composed by 2 or 5 images taken in each sample.

1. *Segmentation based on NDVI thresholding* → Its goal is to reduce the number of thresholds (as also the number of used optical filters) and consequently to remove the empirical weight from the algorithm. Furthermore, since samples were taken during flight, using just two optical filters (*i.e.*, V and NIR) increases the probability of generating overlapped images (less "disparity") for the different bands, when compared to the use

of five optical filters.

2. *Segmentation based on low DOF V image* → This algorithm is seen as an alternative, since it allows the extraction of the ROI based only on the image provided by the V filter. In spite of being computationally heavier when compared to the other two techniques, it can be very convenient when low DOF samples are provided.

3. *Segmentation based on RGB and Hue Saturation and Value (HSV) thresholding* → It was developed in order to use data (captured within specific bandwidths) provided by the entire set of optical filters. However, it has the drawback of being composed by a huge set of empirically defined thresholds.

4.3.1 Segmentation based on Normalized Difference Vegetation Index Thresholding

This segmentation (see figure 4.3) is a multispectral remote sensing data technique, which finds the NDVI (described in section 4.4) for each sample. In contrast to 4.3.3, the algorithm has recourse to 2 different optical filters (V and NIR). It turns out that NDVI can distinguish vegetation from either bare-soil or non-organic materials with considerable accuracy. Accordingly, this technique can be extremely accurate for farmland which hasn't vegetation beyond the one that belongs to the ROI.

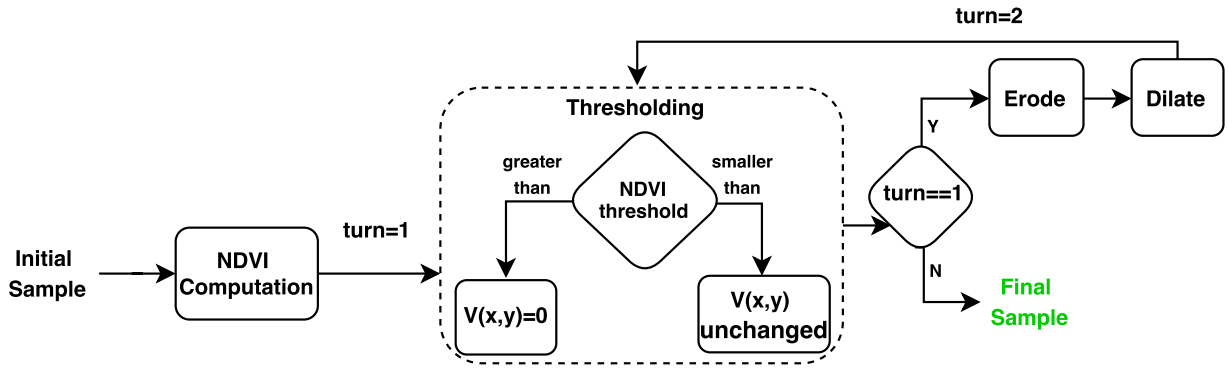


Figure 4.3: Block diagram for the segmentation based on NDVI thresholding.

After the thresholding process, images are eroded in a first step, causing the removal of small blobs which were not correctly segmented and subsequently dilated. Considering particular sizes for erosion and dilation elements, each pixel, $dst(x, y)$, of the destination image (either eroded or dilated), can thus be seen in relation to the source image, src, as suggested in equations (4.1) and (4.2), respectively (where x' and y' are the width and height, defined by the element [37], [38]).

Finally, images are again thresholded, according to the same values previously defined.

$$\min_{(x',y')\text{: element}(x',y')\neq 0} src(x+x', y+y') \qquad (4.1) \qquad \max_{(x',y')\text{: element}(x',y')\neq 0} src(x+x', y+y') \qquad (4.2)$$

4.3.2 Segmentation based on Low Depth of Field Visible Image

When moving the sensor in a digital camera in relation to its lens, the plane in object that is sharply focused moves as well. From this perspective, the sensor can be moved by a specific range, until the feature goes out of focus (*i.e.*, when the width of the blurred image of the feature has become larger on the sensor than the maximum allowed Circle Of Confusion (COC) - an arbitrary value, often equalized to the width of a pixel [39]). Therefore, when striking the sensor, objects which come to a focus too far in front of or behind the COC (C1) will spread out to a size larger than the circle. Examining the lines that connect the edges of the lens aperture to each side of the COC (see figure 4.4), one can see that an object gets out of focus, when positioned outside D1. The width of D1 (distance along the optical axis) is called Depth Of Focus (DOFo). The position in the object space that corresponds to the COC in the image space (C2) can be computed by Gauss' ray construction [40], based on a mathematical treatment of refraction at the interface between two different materials, provided by Snell's law (where A and B correspond to the farthest and nearest "in focus" points, respectively) [41]. Repeating on the object space an analogous procedure to the one made on the image space of the lens, another COC is achieved (C2) and D2 is created. Scene features inside D2 will focus to positions inside D1, leading to a blur which will not be greater than one COC (*i.e.*, features will appear "in focus" on the sensor). The width of the object space D2 is called DOF [39]. Considering the lens aperture the variable N, the DOF (width of D2, Z) can be approximately computed as equation (4.3) suggests [39]:

$$Z \approx \frac{2NCU^2}{f^2} \qquad (4.3)$$

With low DOF images, only the OOI is in sharp focus, whereas background and/or foreground are "out of focus" (typically blurred). This technique is used to create a sense of depth in two dimensional photographs. In fact, sharply focused objects have more details within the object than the ones which are not "in focus", leading to higher values of wavelet coefficients in the high frequency bands of the transform [42]. The high frequency energy is then measured by the standard deviation and variance of wavelet coefficients for those bands. Discrete Wavelet Transform (DWT) represents an image as a sum of wavelet coefficients,

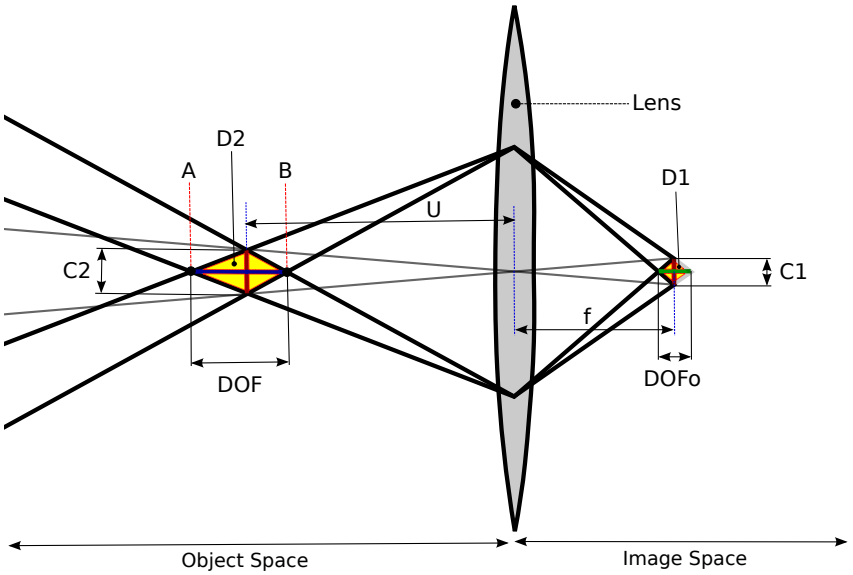


Figure 4.4: DOF illustration.

known as *wavelets*, with different location and scale. It represents the data into a set of high pass and low pass coefficients [43]. For the case of 2D DWT, the input data are passed through a set of both low pass and high pass filters in two directions (rows and columns). The outputs are then "downsampled" by 2 in each direction. To have a better understand-

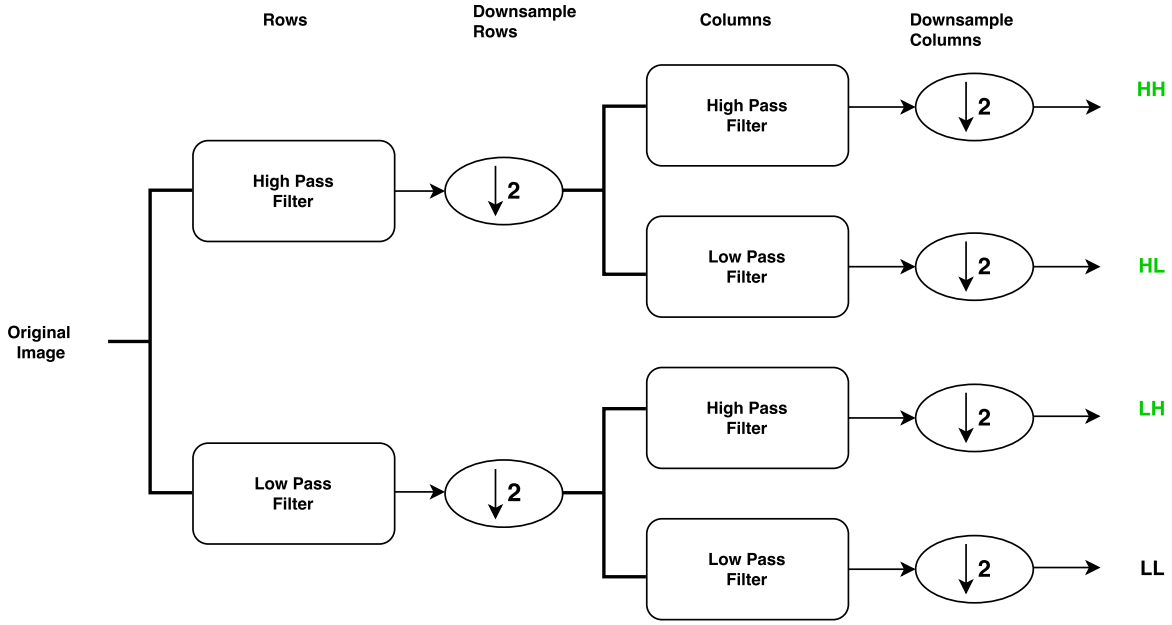


Figure 4.5: Block diagram of 2D DWT.

ing of the basis behind Discrete Haar Wavelet Transform (DHWT), one can consider the reasoning set below [44] and figure 4.5.

Defining $h = (h0, h1) = (\frac{1}{2}, \frac{1}{2})$ as a filter, one can see the convolution between two discrete signals, y and x, as $y[n] = h[n] * x[n] = \sum_{m=0}^{1} h[m]x[m-n] = \frac{1}{2}x[n] + \frac{1}{2}x[n-1]$. Making $z[n] = \frac{1}{2}x[n] - \frac{1}{2}x[n-1]$, the sequence $x[n]$ is defined as $x[n] = y[n] + z[n]$ and

$x[n-1] = y[n] - z[n]$. In fact, the system's matrix form is represented as (4.4) suggests (H and G are convolution matrices).

$$\begin{bmatrix} H \\ G \end{bmatrix} x = \begin{bmatrix} y \\ z \end{bmatrix}, \; H = \begin{bmatrix} \ddots & & & & & & \\ & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & \\ & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & \\ & & & & & & \ddots \end{bmatrix} \; G = \begin{bmatrix} \ddots & & & & & & \\ & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 & \\ & 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & \\ & & & & & & \ddots \end{bmatrix} \tag{4.4}$$

To be able to recover x, some redundancy can be omitted (some rows in H and G). Indeed, $y[n] + z[n] = y[n+1] - z[n+1]$, leading to a downsample of the rows of H and G. Defining N as the size of 1D data x, one can define DHWT, W_N, as $W_N = [H \; G]^T \sqrt{2}$. Therefore, to recreate x, one just need to compute: $W_N^T \; v = [H^T \mid G^T] \; v$. For the case of 2D data and assuming X as an $N \times N$ matrix, with N even, the mathematical procedure is defined as shown in equation (4.5).

$$W_N \; X \; W_N^T = \begin{bmatrix} H \\ G \end{bmatrix} \; X \; \begin{bmatrix} H \\ G \end{bmatrix}^T = \left[\begin{array}{c|c} HXH^T & HXG^T \\ \hline GXH^T & GXG^T \end{array}\right] = \left[\begin{array}{c|c} \mathbf{LL} & \mathbf{LH} \\ \hline \mathbf{HL} & \mathbf{HH} \end{array}\right] \tag{4.5}$$

HXH^T averages along the columns of X and then along the rows of X, producing a blur, β, of X. HXG^T averages along the columns of X and then differences along the rows of HX, producing vertical differences between β and X. GXH^T differences along the columns of X and then averages along the rows of GX, producing horizontal differences between β and X. GXG^T differences along the columns of X and then differences along the rows of GX, producing diagonal differences between β and X.

The algorithm for the segmentation based on low DOF V images was developed with the idea of being part of a target recognition process, which has as input images with low DOF. Therefore, desired vegetation would be "in focus" whereas background and/or foreground would be "out of focus".

Throughout the dissertation, the idea of using a camera capturing low DOF images fell through. However, the algorithm was developed and its results are exemplified in the next chapter. This technique has its pseudo-code as an adapted version of [42]. The image is partitioned into blocks, which are "classified" as background or OOI, according to their average intensity (feature 1) and their standard deviation and variance of wavelet coefficients of high frequencies (features 2 and 3, respectively). The Multi Resolution Segmentation Algorithm (MRSA) comprises 2 sequential phases: $r = 0$ (Phase 1) and $r > 0$ (Phase 2)

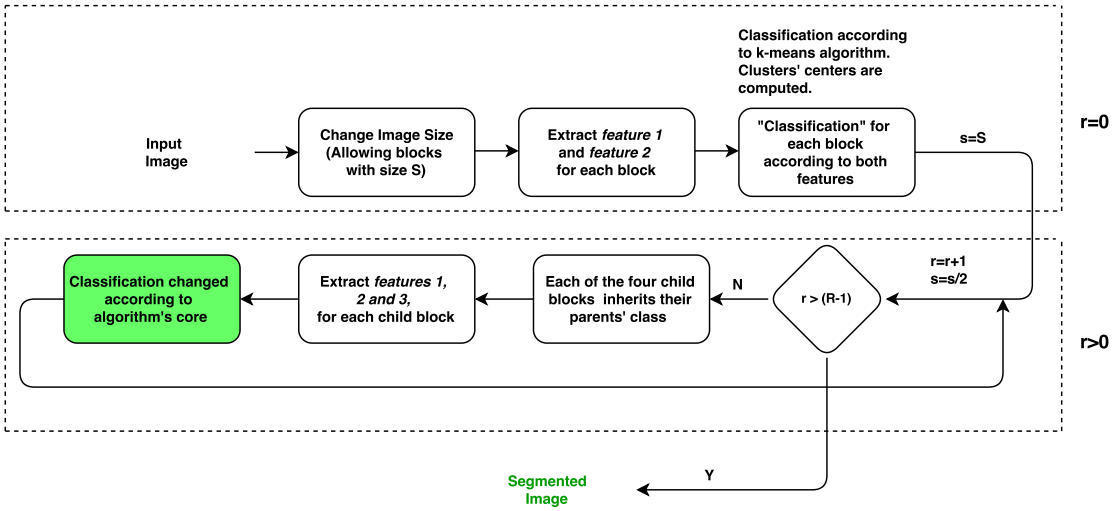


Figure 4.6: Block diagram for the segmentation based on low DOF V image.

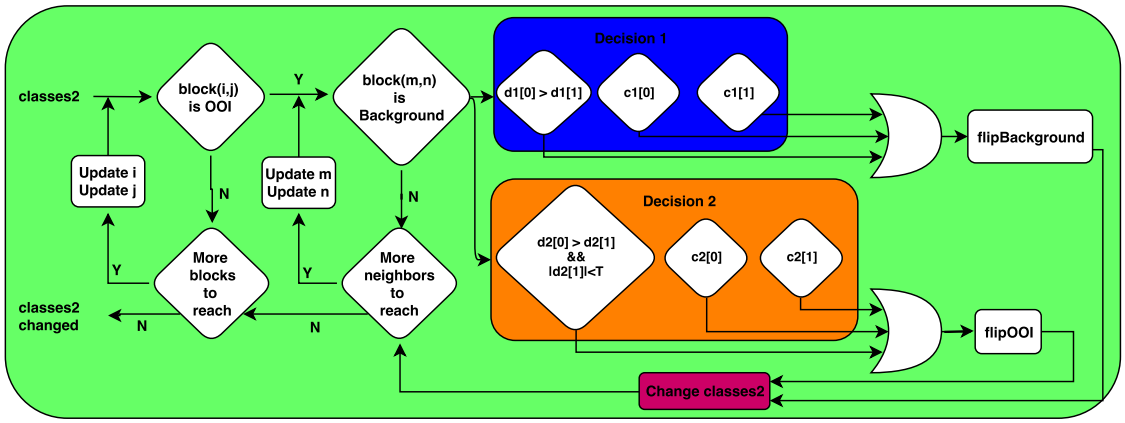


Figure 4.7: Block diagram for the core of the segmentation based on low DOF V image.

(see figure 4.6). During Phase 1, the image is partitioned into blocks of size S, and their grayscale mean as well as their wavelet coefficients are computed. Therefore, each block is classified as being an OOI or background, through K-means algorithm [26,45], according to 6 "sub-features" (variance and standard deviation of LH, HL and HH). At the end of this stage, both clusters' centers for each "classification" are computed and afterwards used in Phase 2. Throughout Phase 2, objects are subdivided by a factor of 2, in each iteration. Typically, $S = 32$ for $r = 0$, leading to 4 iterations in Phase 2 ($r = 5$ is despised as at this iteration blocks shrink to the size of a pixel and so are likely to be smooth - variance of wavelet coefficients are no longer good features). Defining R as the number of iterations required to have objects with a size of a pixel, for each iteration $r \in [0, R-1]$, features 1, 2 and 3 are computed for every object. Depending on conditions specified on the algorithm's core, each object class is maintained or changed. Algorithm 1 specifies the pseudo-code of this technique. K-means algorithm used in 1 (Phase 1) splits the data from 6 different sets

(for all high frequencies - LH, HL and HH - standard deviation and variance are computed) into a group of 2 clusters (Background and OOI). Random initial centers are selected at a first attempt, for a total of 10 trials, or a different number of attempts until accuracy is achieved. This implementation of k-means algorithm is reached by repeatedly assigning points to the closest centroid nearby using Euclidean [46] distance from data points to a centroid. The algorithm's core is outlined on figure 4.7 and its conditions clarified on table 4.1.

Algorithm 1 MRSA for low DOF images.

▷ **r=0** (Phase 1)
procedure rEqualToZero(I) ▷ I←grayscale low DOF image
  $S \leftarrow 32$ ▷ Size of each block
  for $i \in 1, ..., \frac{m}{S}$ **do** ▷ m ←lines of I
    for $j \in 1, ..., \frac{n}{S}$ **do** ▷ $n \leftarrow$ columns of I
      [rOI**]**← **Rect[**$j * S, i * S, S, S$**]** ▷ Region of block (i,j)
      [$\mu(i,j)$**]**← **mean[**rOI**]** ▷ Feature 1
      [$wC(i,j)$**]**← **haarWavelet[**rOI**]** ▷ Wavelet coefficients
      [$\sigma(i,j)$**,**$\sigma^2(i,j)$**]**← **stdDevVar[**$wC(i,j)$**]** ▷ Features 2 and 3
    end for
  end for
  [classes,clustersC]← **kMeans[**σ**,**σ^2**]** ▷ Classes and clusters' centers
  return $classes, clustersC, \mu$
end procedure
▷ **r>0** (Phase 2)
procedure rGreaterThanZero($I, classes, clustersC, \mu$)
  for $r \in 1, ..., R-1$ **do**
    $S[r] \leftarrow \frac{S}{2^r}$
    for $i \in 1, ..., \frac{m}{S} \times 2^r$ **do**
      for $j \in 1, ..., \frac{n}{S} \times 2^r$ **do**
        [rOI**]**← **Rect[**$j * S[r], i * S[r], S[r], S[r]$**]**
        [$\mu 2(i,j)$**]**← **mean[**rOI**]**
        [$wC(i,j)$**]**← **haarWavelet[**rOI**]**
        [$\sigma(i,j)$**,**$\sigma^2(i,j)$**]**← **stdDevVar[**$wC(i,j)$**]**
      end for
    end for
    $classes2 \leftarrow classes$ [1]
    [$classes2$**]**← **core[**$classes2, \mu 2, \mu, \sigma^2, \sigma, clustersC, r, \frac{m}{S} \times 2^r, \frac{n}{S} \times 2^r$**]**
    $classes \leftarrow classes2$ ▷ Update classes
    $\mu \leftarrow \mu 2$ ▷ Update mean
  end for
end procedure
▷ **Core**
procedure core($classes2, \mu 2, \mu, \sigma^2, \sigma, clustersC, r, iMax, jMax$)

[1] Labels from *classes2* (parents) are matched to *classes* (children), by a scale factor (each parent generates 4 children).

```
    for i ∈ r, ..., iMax − r do
        for j ∈ r, ..., jMax − r do
            if classes2(i, j) == 1 then                          ▷ If block(i,j) is OOI
                neighbor = {(i − 1, j), (i + 1, j), (i, j − 1), (i, j + 1)}
                if classes2(neighbor) == 0 then      ▷ At least 1 neighbor is Background
                    for k ∈ 1, ..., 4 do                          ▷ Loop between all neighbors
                        flipBackground ← 0
                        flipOOI ← 0
                        [l_mn, c_mn] ← neighbor(k)
                                                                          ▷ Decision 1
                        d1[0] ← μ2(l_mn, c_mn) − μ(l_mn × 0.5, c_mn × 0.5)
                        d1[1] ← μ2(l_mn, c_mn) − μ(l_i × 0.5, c_i × 0.5)
                        c1[0] ← (σ²(i, j) − clustersC(1))² < (σ²(i, j) − clustersC(0))²
                        c1[1] ← (σ(i, j) − clustersC(1))² < (σ(i, j) − clustersC(0))²
                        if d1[0] > d1[1] || c1[0] || c1[1] then
                            flipBackground ← 1
                        end if
                                                                          ▷ Decision 2
                        d2[0] ← μ2(i, j) − μ(i × 0.5, j × 0.5)
                        d2[1] ← μ2(i, j) − μ(l_mn × 0.5, c_mn × 0.5)
                        c2[0] ← (σ²(i, j) − clustersC(1))² > (σ²(i, j) − clustersC(0))²
                        c2[1] ← (σ(i, j) − clustersC(1))² > (σ(i, j) − clustersC(0))²
                        if (d2[0] > d2[1] && |d2[1]| < T) || c2[0] || c2[1] then
                            flipOOI ← 1
                        end if
                                                                      ▷ Change classes2
                        if !flipBackground && flipOOI then
                            classes2(i, j) = 0
                        end if
                        if flipBackground && !flipOOI then
                            classes2(l_mn, c_mn) = 1
                        end if
                    end for
                end if
            end if
        end for
    end for
    return classes2
end procedure
```

Table 4.1: Conditions used to assess the change of class labels.

Condition	Description (when condition verified)
$d1[0] > d1[1]$	Block (m, n) has its μ closer to OOI than to Background
$c1[0]$	Block (i, j) has its σ^2 closer to OOI cluster than to Background cluster
$c1[1]$	Block (i, j) has its σ closer to OOI cluster than to Background cluster
$d2[0] > d2[1]$	Block (i, j) has its μ closer to Background than to OOI
$d2[1] <$ T	Bounding condition of changing block (i, j) to Background
$c2[0]$	Block (i, j) has its σ^2 closer to Background cluster than to OOI cluster
$c2[1]$	Block (i, j) has its σ closer to Background cluster than to OOI cluster

4.3.3 Segmentation based on Red Green Blue and Hue Saturation Value Thresholding

The main target of this algorithm consists in finding whether pixels belong to the ROI, according to the data provided by 5 different filters: V, NIR, B, G and R. To achieve this purpose, a set of thresholds was empirically defined, for some RGB and HSV data.

Table 4.2: Different thresholds used in the segmentation based on RGB and HSV thresholding.

Threshold ID	**Threshold name**	**Brief description**
1	thIR	Threshold for NIR grayscale data
2	thRedMin	Threshold for R RGB data (minimum value)
3	thRedMax	Threshold for R RGB data (maximum value)
4	thGreenMin	Threshold for G RGB data (minimum value)
5	thBlueMin	Threshold for B RGB data (minimum value)
6	thIRMin	Threshold for NIR RGB data (minimum value)
7	thVMax	Threshold for V RGB data (maximum value)
8	thVHueMax	Threshold for V HSV data (maximum value)

Thresholds mentioned in table 4.2 were obtained by an empirical process. For the first N images, each value (threshold) was optimized, leading to ROIs which covered most of the vegetation leaves. For example, for the assessment of each sample which corresponds to each vineyard, 3 sets were defined, according to thresholds related to vineyards A, B and C. For the same case, let X_{ijk} be the mean of values for threshold i, for $i = [1, 8]$, $j = [1, 3]$ and $k = N_j$ (number of samples used in vineyards A, B and C, knowing that $N = \sum_{j=1}^{3} N_j$). Considering that X_{ijk} follows a Gaussian distribution, each threshold can be defined as the mean (m) of measurements taken from the related vineyard, within a Confidence Interval (CI). A level of confidence of 0.95 for threshold i was considered, knowing that $X \sim N(m, \sigma)$, m, $\sigma \in \mathbb{R}^+$ and seeing Z (equation (4.6)) as the core variable (T is the Student's distribution and $\hat{S}$ the standard deviation - computed afterwards - in probability and statistics [47]):

$$Z = \frac{\overline{X_{ijk}} - m}{\frac{\hat{S}}{\sqrt{N_j}}} \sim T(N_j - 1) \tag{4.6}$$

Its block diagram is similar to 4.3, apart from the fact that threshold conditions are no longer defined by the NDVI (the computation of NDVI is also despised) but by a set of thresholds related to RGB and HSV data.

4.4 Feature Extraction

The capacity for distinguishing features is a critical step and depends on the characteristics of the problem domain. In selecting or designing features, one obviously would like to find features that are simple to extract, invariant to irrelevant transformations, insensitive to noise and useful for discriminating patterns in different categories.

For PA, multiple VIs are powerful tools to describe different features in herbaceous crops. Spectral VIs reduce the multiple-waveband data at each pixel to a single numerical value (index). Due to limitations on both the camera's sensor and optical filters' spectral response, computed features are composed by radiation between, approximately, 380 nm and 850 nm. NDVI (equation (4.7), [48], [49]), GNDVI (equation (4.8), [50]), Green-Red Vegetation Index (GRVI) (equation (4.9), [49]), SAVI[2] (equation (4.10), [49]), SR (equation (4.11), [50]), Green Vegetation Index (GVI) (equation (4.12), [49]),Red Normalized Difference Vegetation Index (RNDVI) (equation (4.13), [49]) and Excess Green (ExG)[3] (equation (4.14), [51]), were computed for each sample. Depending on the case, a sample is linked to a pair of V and NIR photos or a set composed by B, G, R, NIR and V photos, depending whether it was chosen segmentation 4.3.1/segmentation 4.3.3, or segmentation 4.3.2, respectively, to identify the ROI. Then, their mean values are stored as future inputs for the classification method.

Every VI mentioned above is related to the reflectance that comes from electromagnetic radiation with different frequencies. Specifications for different bands used during computations for each VI were mentioned in figure 3.2.

$$\mathrm{NDVI} = \frac{\mathrm{IR} - \mathrm{V}}{\mathrm{IR} + \mathrm{V}} \qquad (4.7)$$

$$\mathrm{GNDVI} = \frac{\mathrm{IR} - \mathrm{G}}{\mathrm{IR} + \mathrm{G}} \qquad (4.8)$$

$$\mathrm{GRVI} = \frac{\mathrm{IR}}{\mathrm{G}} \qquad (4.9)$$

$$\mathrm{SAVI} = \frac{\mathrm{IR} - \mathrm{R}}{\mathrm{IR} + \mathrm{R} + L} \times (1 + L) \qquad (4.10)$$

$$\mathrm{SR} = \frac{\mathrm{IR}}{\mathrm{R}} \qquad (4.11)$$

$$\mathrm{GVI} = \frac{\mathrm{G} - \mathrm{R}}{\mathrm{G} + \mathrm{R}} \qquad (4.12)$$

$$\mathrm{RNDVI} = \frac{\mathrm{IR} - \mathrm{R}}{\mathrm{IR} + \mathrm{R}} \qquad (4.13)$$

$$\mathrm{ExG} = 2 \times g - r - b \qquad (4.14)$$

This set of features is used not to distinguish healthy from ill crops, but to be a powerful tool when one wants to differentiate crop species. For this reason, it was not considered the relation between each VI and crop health condition.

[2] L = 0.5 for intermediate vegetation density.
[3] $g = \frac{G}{R+G+B}$, $r = \frac{R}{R+G+B}$, $b = \frac{B}{R+G+B}$

4.5 Post-Processing

When performing classification, the final decision is achieved by the correlation between prediction of both classifiers. Here, this decision (final prediction) is computed as an *accuracy* function related to the two classifiers:

- Accuracy → Each classifier works on a testing dataset, with a total of N (N differs between dataset #1 and dataset #2, as mentioned in section 5.3) different unseen and randomly selected samples. According to the achieved ratio between correct and incorrect decisions, accuracies for both classifiers are computed: $accSVM$ and $accDT$ (considering SVMs and DTs, respectively).

Therefore, the final prediction of the unseen test (probability, *prob*, for crop class) is computed as a probability measurement related to the accuracy of both classifiers (equation (4.15)):

$$prob = accSVM \times accDT + accSVM \times (1 - accDT) + accDT \times (1 - accSVM) \quad (4.15)$$

This procedure is only considered in case the two classifiers agree in the same class. Otherwise, the final result is disregarded and a new test should be done.

5 Experimental Results

5.1 Software Implementation

The pattern classification algorithm described on chapter 4 was implemented and tested in *C++* programming language. Furthermore, the core of the algorithm (different segmentation techniques as well as testing classification libraries) was previously studied using *Matlab* environment. Matlab was chosen due to its ease handling of data in the form of matrix as well as allowing *script* implementations oriented to numerical programming.

Therefore, to create a real or *quasi* real-time algorithm, different libraries were used in C++ to manipulate data. Open Source Computer Vision Library (OpenCV) is (essentially) a C++ (and C, Java or Python) API which provides several computer vision algorithms to manipulate data. In short, its *Mat* object was used, allowing the storage of data in One Dimensional (1D), 2D or 3D arrays, smoothing the transition from Matlab implementation into C++. Furthermore, OpenCV allows the display of data in image format for different precision (*e.g.*, 8, 32 or 64 bit). Every collected sample was provided according to particular configurations on the RPi camera. Such configurations (exposure, brightness, contrast, saturation and gain) were optimized and changed according to the API provided by [52]. As far as classifications are concerned, two distinct libraries were used: SVM's library [53] (`libsvm`) allows both *C* and *nu* SVM and supports multi-class classification (source code available for C++). DT's library [35] (`scikit-learn`) (*Machine Learning* in *Python*) has simple and efficient tools for data mining and data analysis and provides a particular class (`DecisionTreeClassifier`) which is capable of performing multi-class classification.

The pattern classification algorithm is run on the ground station. Its backbone is defined by an object named `myAlgorithm`, composed by the following *public methods*:

1. `void setVariables()` → Initializes a set of variables needed for segmentation algorithms;

2. `void readInfo(cv::Mat*,cv::Mat*,int,int,int)` → Reads input images for both

V and NIR bands;

3. `void segmenAndFExtraction(cv::Mat*,cv::Mat)` → Depending on the chosen segmentation algorithm, it identifies the ROI and subsequently computes (considering the ROI) the set of features mentioned in 4.4;

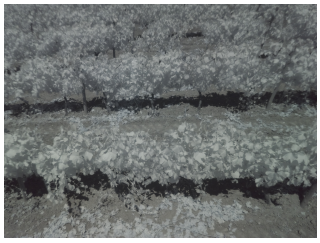
4. `createTrainDataSVMandDT(int)`/`createTestDataSVMandDT(int)` → Depending either training or testing is selected, data are put in such a way that both libraries ([53] and [35]) are capable of reading data.

After the execution of this set of methods, data are then processed according to the executables `svm-train` (when training SVMs) or `svm-predict` (when testing SVMs), and Python scripts `decisionTreesTrainning.py` (when training DTs) or `decisionTreesTesting.py` (when testing DTs). Finally, data are post-processed as mentioned in 4.5.

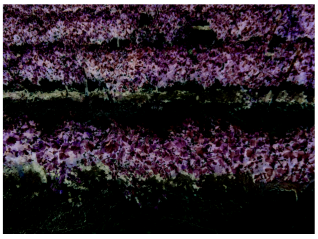
5.2 Segmentation

The goal of any segmentation process is to distinguish the ROI from the rest of the image. However, each segmentation method (4.3) presented in this thesis has different accuracies, depending on the conditions of the crop to be evaluated. During this chapter, classification results were obtained taking into account algorithms described during sections 4.3.1 and 4.3.2 for the segmentation process. Despite algorithm 4.3.3 was obtained considering a probabilistic analysis, its results led to machine learning overfitted results (therefore, data which are unknown for the algorithm - data that don't belong to the training dataset - were not classified with the desired accuracy). For this reason, segmentation 4.3.3 was discarded from implementation tests. Figure 5.1 shows, step by step, the result of algorithm 4.3.1. There, one can consider that the ROI was well defined. On the other hand, the samples that captured undesired vegetation might introduce significant distortions in what would be the intended ROI, since NDVI cannot make that distinction. How to capture the desired ROI when not wanted vegetation belongs to the camera's scene plane? Furthermore, one might want to assess a particular region in the image even if other regions belong to the same "type" of vegetation. At this point, algorithm 4.3.2 comes into play. In fact, by the manipulation of the DOF during UAV flights, one can obtain the ROI in sharp focus, whereas background and/or foreground are out of focus. This work did not use HW capable of changing in real-time the camera's focus. Nevertheless, in order to test the algorithm 4.3.2, data were

(a) Input V image.

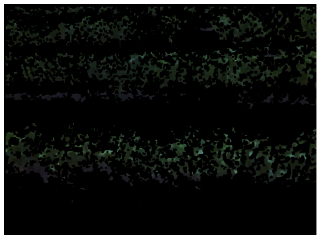


(b) Input NIR image.

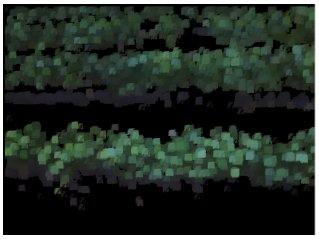


(c) NDVI computation.

(d) Thresholding with NDVI.



(e) Erosion.



(f) Dilation.

(g) Matching.

(h) Final result.

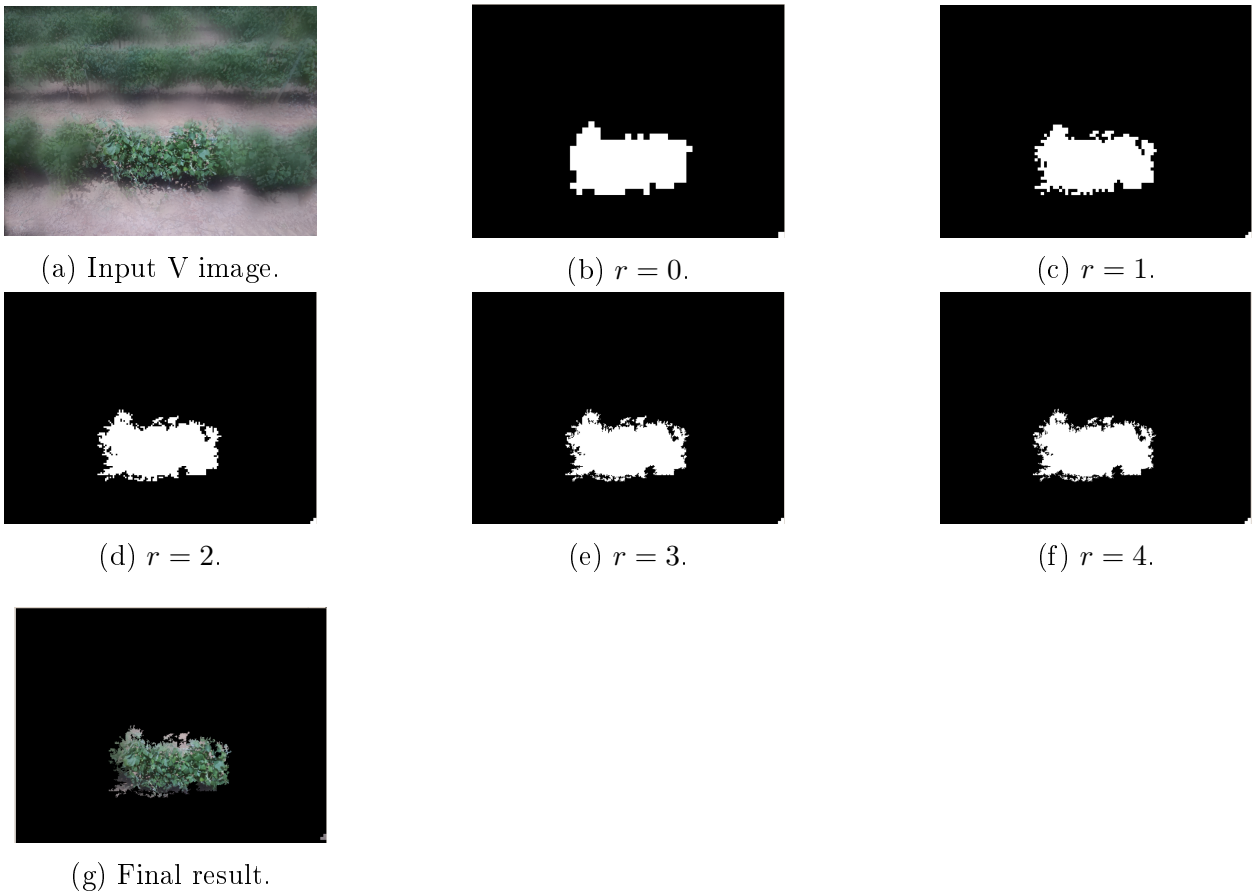
Figure 5.1: Example of segmentation based on NDVI thresholding for the "entire" vineyard.

replicated and then changed by introducing blur for every region which does not belong to the ROI. The results for this algorithm are shown, step by step, in figure 5.2.

5.3 Feature Computation

Features were computed according to the identified ROI (5.2). Further classifications tests showed that variance for each VI was not a good indicator in distinguishing crop species. In fact, variances were significantly different within classes, which led to their exclusion for classification input data. Data were then used for the same PRS concerning 2 different datasets:

- Dataset for **Trees Classification**: distinguishing from *pine trees*, *orange trees*, *olive trees*, *eucalyptus* and *magnolias* → Dataset #1;
- Dataset for **Vineyards Classification**: believing that grapevines from different "qualities" generate different VIs, the PRS has the goal of distinguishing from 3 different vineyards → Dataset #2.



(a) Input V image. (b) $r = 0$. (c) $r = 1$.

(d) $r = 2$. (e) $r = 3$. (f) $r = 4$.

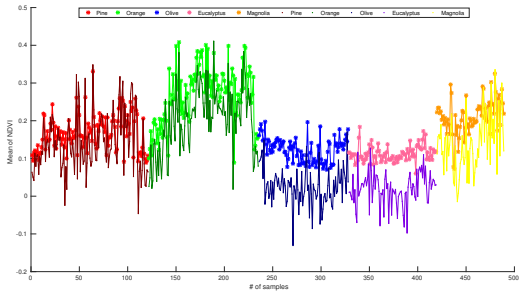
(g) Final result.

Figure 5.2: Example of segmentation based on low DOF V image for a "particular region" of the vineyard.

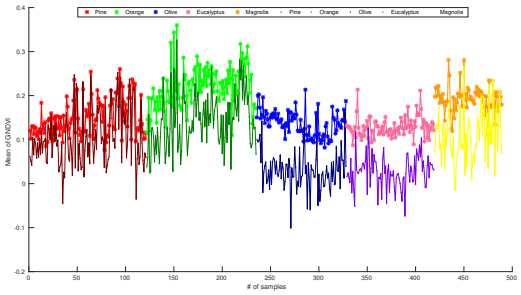
A priori, olive trees and eucalyptus don't have significant distinctions. Therefore, classification methods aim to build a recognition pattern for a feature space composed by this set of mean values (see figures 5.3[1] and 5.4[2]) for the VIs shown above, which differentiates even from classes that appear very similar at a first glance.

[1] 489 samples: 121 Pine Trees, 114 Orange Trees, 93 Olive Trees, 91 Eucalyptus and 70 Magnolias. Both segmentations (5.2 (•) and 5.1 (∗)) were considered. Different colors suggest distinguished tree species whereas different hues indicate distinct segmentations.

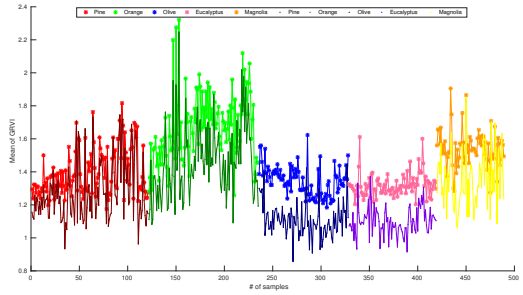
[2] 635 samples: 215 for Vineyard 1, 256 for Vineyard 2 and 164 for Vineyard 3. Just segmentation 5.1 was considered. Different colors suggest different vineyard "qualities".



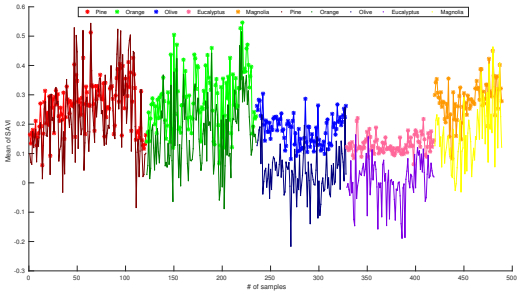
(a) NDVI.



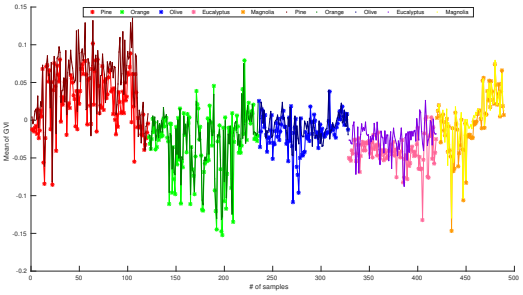
(b) GNDVI.



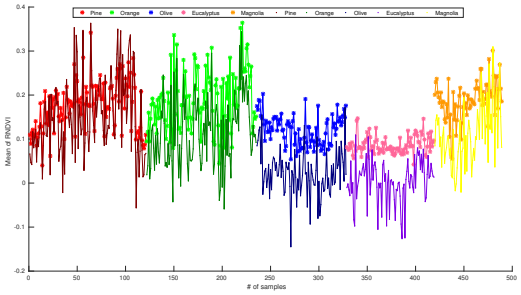
(c) GRVI.



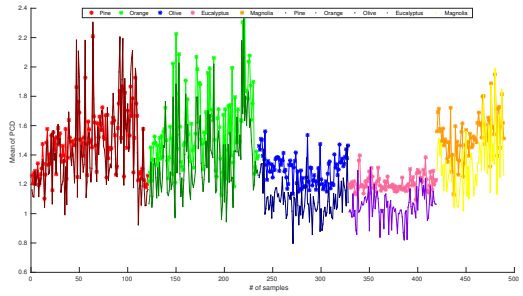
(d) SAVI.



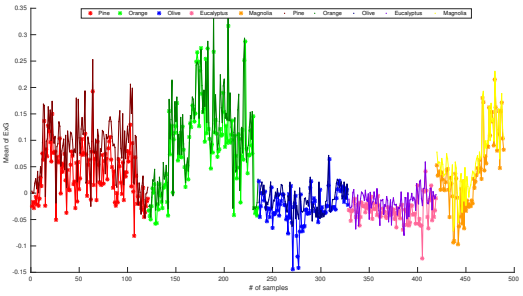
(e) GVI.



(f) RNDVI.

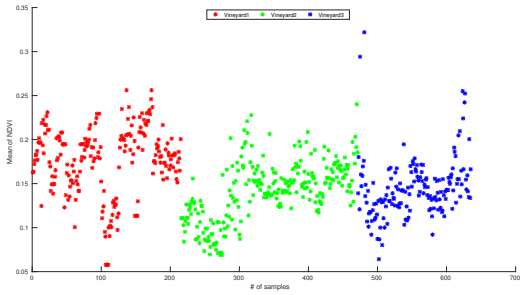


(g) SR.

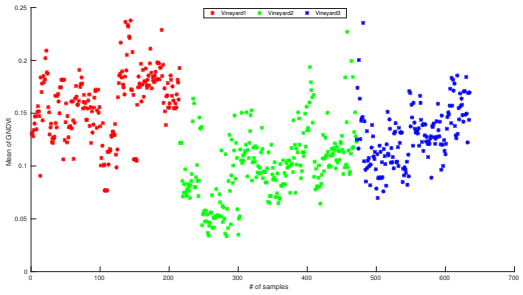


(h) Final result.

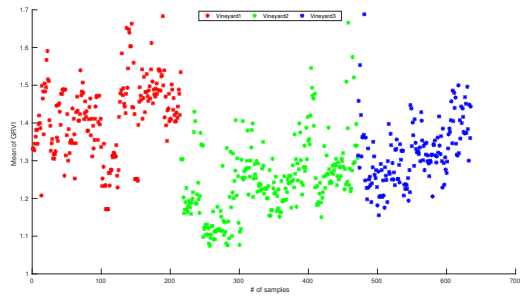
Figure 5.3: Mean values of each computed VI for trees classification.



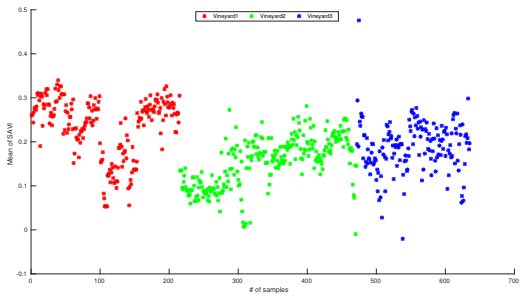
(a) NDVI.



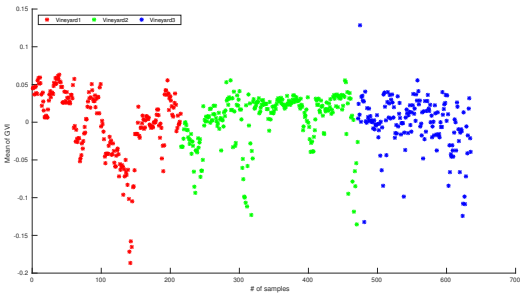
(b) GNDVI.



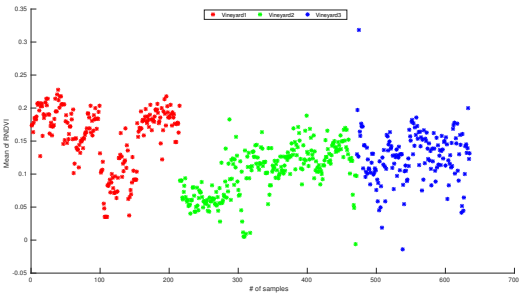
(c) GRVI.



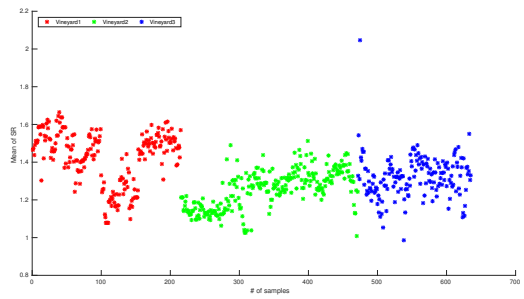
(d) SAVI.



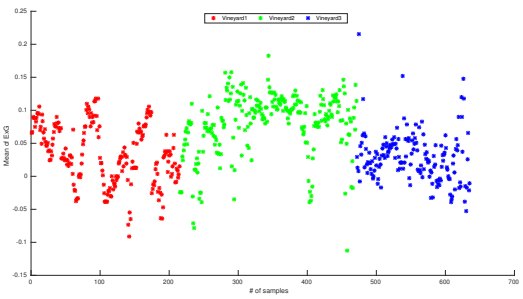
(e) GVI.



(f) RNDVI.



(g) SR.



(h) Final result.

Figure 5.4: Mean values of each computed VI for vineyard classification.

5.4 Classification

Test results were obtained considering segmentation 4.3.1: the main goal for this thesis consists on a real or *quasi*-real time classification (autonomous PRS) which can only be reached using this approach (knowing that DOF images cannot be taken with the available HW).

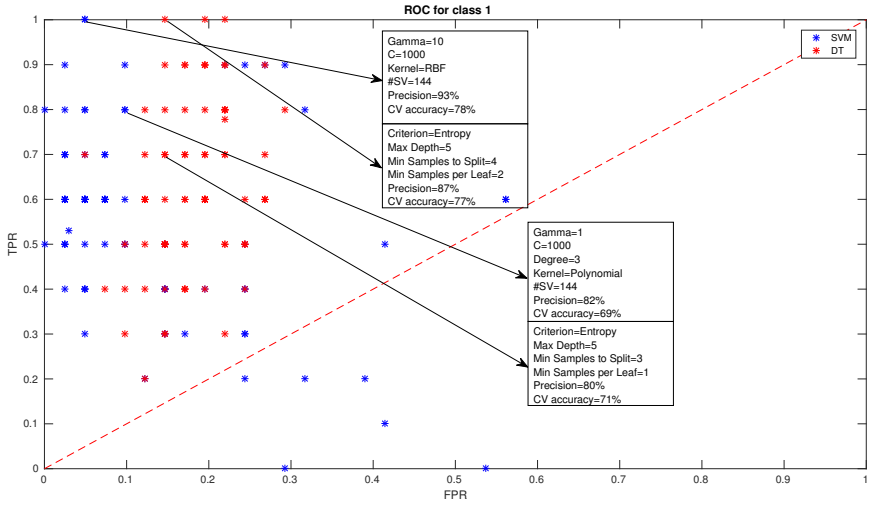
A set of 50 samples was used for the test of set #1, concerning the classification of 5 different trees (5 subsets of 10 images for each species). On the other hand, 175 samples were taken for the classification test when considering test set #2 (71 samples for Vineyard 1, 57 for Vineyard 2 and 47 for Vineyard 3).

For the considered metric functions for the assessment of the PRS, the entire data were split which led to N ($N = \{5, 3\}$, depending on the dataset) binary classifications (see example 2). For the two datasets, the PRS was tested according to 82 configurations for SVMs and 80 for DTs, by the change of specific hyperparameters.

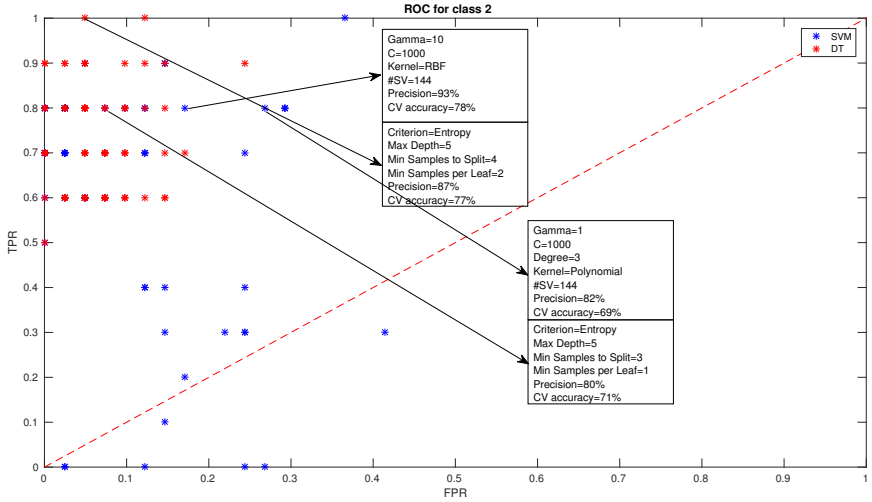
Example 2. For the PRS one should consider that a set of 50 samples (a_i, b_i, c_i, d_i and e_i, $i \in [1, 10]$) needs to be classified as being part of class 1, class 2, class 3, class 4 or class 5. The *ground truth*[3] for this set is defined as follows: $a_i \in$ class 1, $b_i \in$ class 2, $c_i \in$ class 3, $d_i \in$ class 4 and $e_i \in$ class 5. Therefore, 5 binary classifications are assessed ($a_i \in$ *class 1* $||$ $a_i \in \overline{class\ 1}$, (...), $e_i \in$ *class 5* $||$ $e_i \in \overline{class\ 5}$) and their ROC is represented. For each classifier configuration, its performance is computed and shown in the correspondent ROC, concerning thresholds of 50% (which means that the classifier defines the binary classification considering equal weights for both classes, during the decision moment). The final precision is computed as the mean of the 5 different precisions obtained for each binary classification.

For dataset #1 and considering both SVMs CV and DTs CV, data were cross validated with 10 folds. SVMs accuracy depends on the trade-off between a high-complexity model (which may overfit the data) and a large margin which will incorrectly classify some of the training data in the interest of better generalization. Therefore, the number of support vectors for the final SVM configuration was taken into account (the greater the number of support vectors, the greater the probability of overfitting). From table A.1 (see appendix) and figure 5.5, one can see that SVM's kernel can be defined by a RBF ($\Gamma = 10$) of type C-SVM ($C = 1000$), offering a precision of 83% and a CV accuracy of 78% (where 144 support vectors were identified) (option #1). Indeed, option #1 provides the best ROC point for

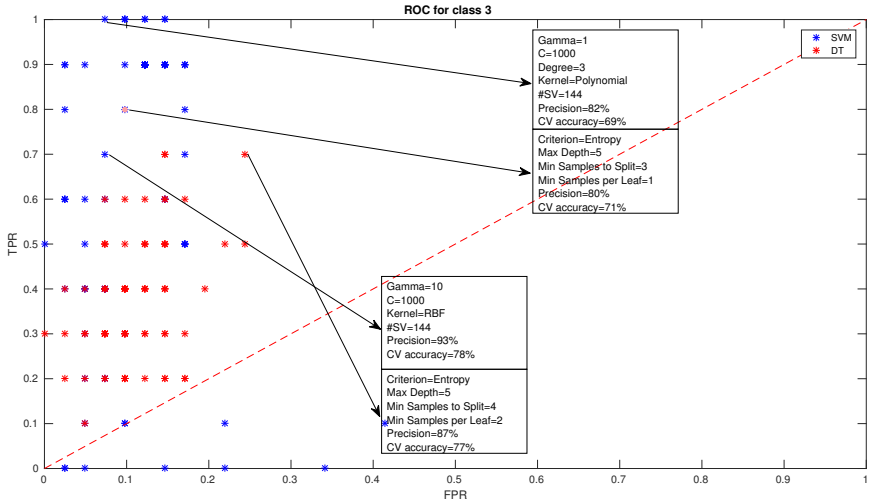
[3]It is the standard to which the learning algorithm needs to adapt (*i.e.*, it is a set of labels that "tells" the machine learning algorithm what to learn).



(a) Class 1.

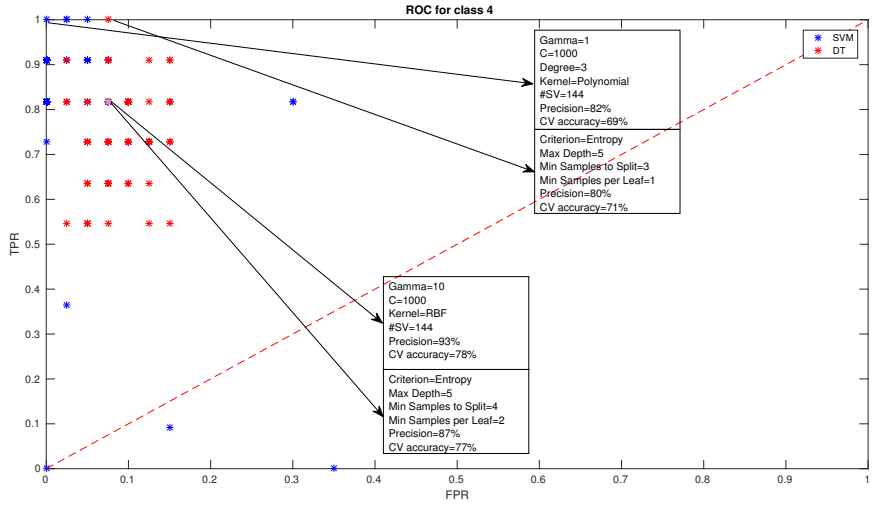


(b) Class 2.

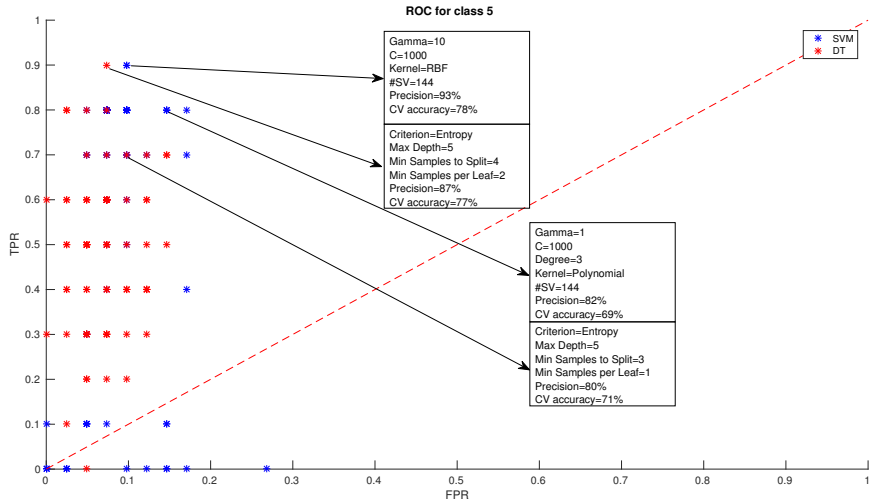


(c) Class 3.

Figure 5.5: TPR FPR relationship for each class, according to dataset #1.



(d) Class 4.



(e) Class 5.

Figure 5.5: TPR FPR relationship for each class, according to dataset #1 (cont).

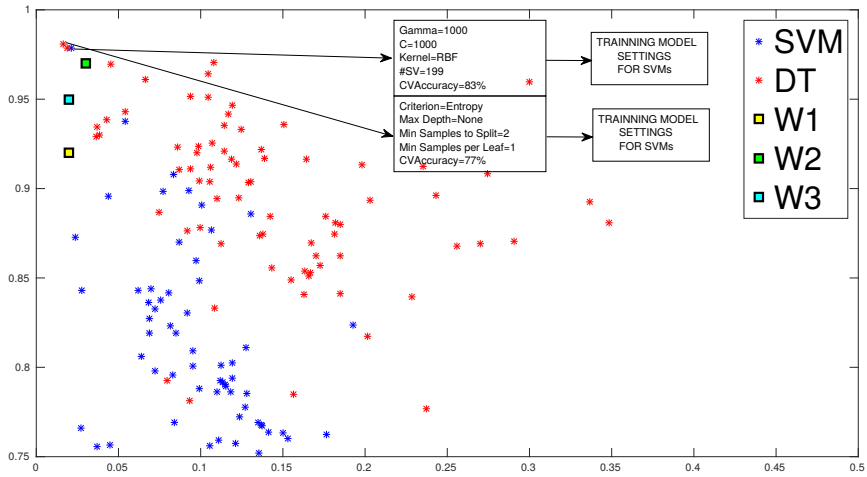
the binary classifications of classes 1, 2 and 5 (see figures 5.5a, 5.5b and 5.5e). Another possibility might be defining the kernel by a polynomial function of 3^{rd} degree ($\Gamma = 1$) of type C-SVM ($C = 1000$) (option #2, which coincidentally computes the same number of support vectors that option #1 does), since this configuration offers the best position on the ROC curve, for the binary classifications of classes 3 and 4 (5.5b and 5.5c). Options #1 and #2 led to testing accuracies of 72% (36/50) and 68% (34/50), respectively. With regard to DTs, the training model can be configured with two different sets of hyperparameters (see appendix, table A.3). In fact, as mentioned for SVMs during PRS for training set #1, a specific configuration for DTs better suits data for classes 1, 2 and 5 (option #1, see figures 5.5a, 5.5b and 5.5e), whereas another configuration achieves better performances for classes 3 and 4 (option #2, see figures 5.5c and 5.5d). These two options are defined,

respectively, as follows: *Criterion=entropy*, *Maximum Depth=5*, *Minimum Samples Split=4* and *Minimum Samples per Leaf=2* (option #1, also defined by a precision of 87% and CV accuracy of 77%) and *Criterion=entropy*, *Maximum Depth=5*, *Minimum Samples Split=3* and *Minimum Samples per Leaf=1* (option #2, also defined by a precision of 80% and CV accuracy of 71%). These models led to testing accuracies of 74% (37/50) and 70% (35/50), respectively.

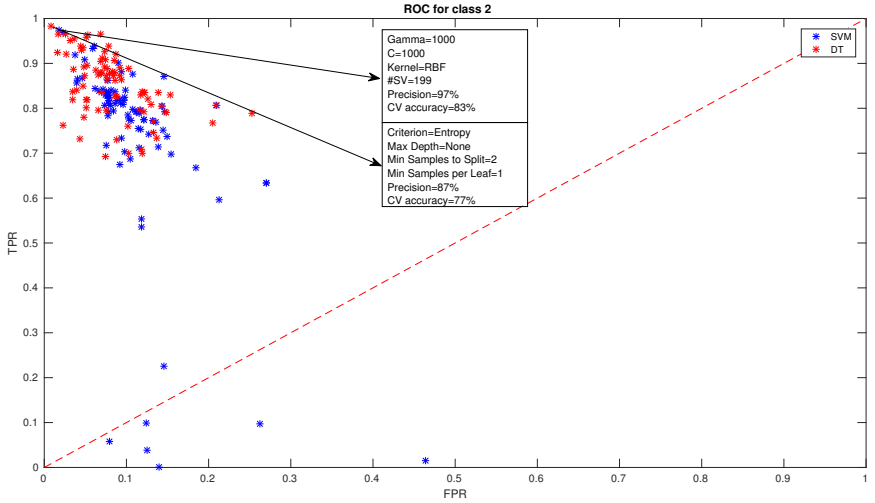
As it was considered for dataset #1, input data for the two classifiers (SVMs and DTs) CV were cross validated with 10 folds during the training stage of the PRS for dataset#2. From table A.2 (see appendix) and figure 5.6, one can see that SVM's kernel can be defined by a RBF ($\Gamma = 1000$) of type C-SVM ($C = 1000$), offering a precision of 97% and a CV accuracy of 83% (where 199 support vectors were identified). This model led to a testing accuracy of (approximately) 73% (127/175). With regard to DTs (see appendix, table A.4), the training model that better describes the dataset should be defined by the following set of hyperparameters: *Criterion=entropy*, *Maximum Depth=None*, *Minimum Samples Split=2* and *Minimum Samples per Leaf=1*, defined by a precision of 87% and CV accuracy of 77%. This model led to a testing accuracy of (approximately) 79% (138/175).

As figure 5.3 suggests, the feature space for the set of considered VIs is similar between Eucalyptus and Olive trees. The radiation reflected by these two tree species cannot be well differentiated for the achieved electromagnetic radiation. For this reason, either SVMs or DTs achieved better performances (for binary classifications) considering particular configurations for the two tree species mentioned above. Nevertheless, both configurations for each classifier led to satisfactory predictions for the entire set of data (see both CV and testing stage accuracy results for PRS related to dataset #1, as mentioned above), when multi-class is considered.

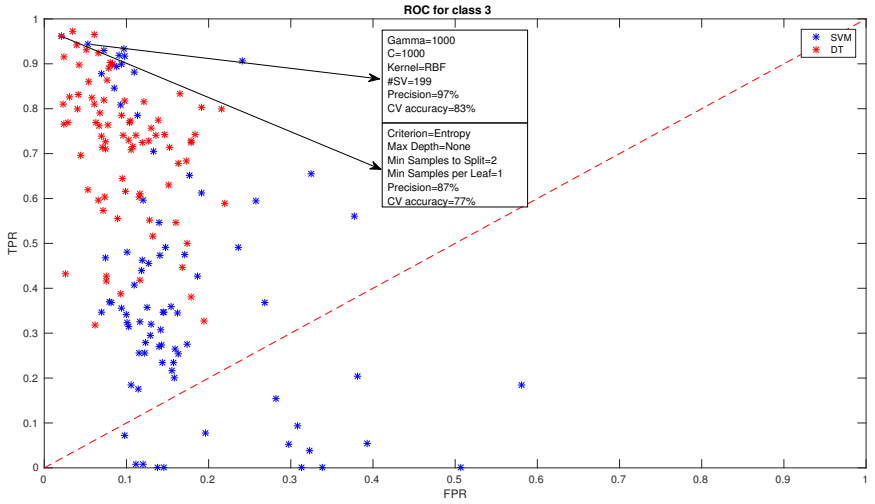
On the other hand, the PRS for dataset #2 was trained for one particular configuration for each classifier. The same set of hyperparameters was located near to the sweet spot for all the three binary classifications. For this reason, it was chosen a specific configuration for training each learner, knowing that the closer a point follows the left-hand border and then the top border of the ROC (sweet spot), the more accurate the test is. It is also important to clarify that each precision and recall results mentioned among tables A.1, A.2, A.3 and A.4 (see appendix), were computed as being an average among binary classes. This means that their values are greater than any other metric. Indeed, these metrics are increased by the correct distinction among samples from belonging or not to a specific class (binary classification), whereas CV accuracy's rate is only increased if the algorithm hits the correct



(a) Class 1.



(b) Class 2.



(c) Class 3.

Figure 5.6: TPR FPR relationship for each class, according to dataset #2.

class, from the entire set of possibilities (multi-class classification). Hence, precision was only considered as a measure to untie from different configurations, after concluding the assessment of both the ROC curves and CV accuracies for a particular configuration (see section 3.2.3).

Table 5.1: Final results obtained for the two PRSs according to SVMs.

SVMs configuration	**CV Accuracy**	**Precision**	**Recall**	**Accuracy**
Dataset #1 (option #1)	78%	93%	86%	72%
Dataset #1 (option #2)	69%	82%	78%	68%
Dataset #2	83%	97%	94%	73%

Table 5.2: Final results obtained for the two PRSs according to DTs.

DTs configuration	**CV Accuracy**	**Precision**	**Recall**	**Accuracy**
Dataset #1 (option #1)	77%	87%	88%	74%
Dataset #1 (option #2)	71%	80%	82%	70%
Dataset #2	77%	87%	90%	79%

The performance for the classification of the PRS according to dataset # 2 was compared[4] to the relationship between TPR and FPR of three different studies. One can see such binary classifications from figure 5.6:[5] 1. [18] classifies pixels which belong to grapes, according to different color spaces and bin sizes during the "segmentation" process (performance defined as W2 in figure 5.6); 2. [17] differentiates vines prior to ripening from grapevines during ripening (performance defined as W1 in figure 5.6); 3. [24] sustains that a classification of image pixels into five clusters (leaves, stems, branches, fruit and background) can be accurately measured, and achieves its best results for the identification of stems (performance defined as W3 in figure 5.6).

In conclusion, the correct choice of kernel parameters was crucial for obtaining good results, which practically means that an extensive search should be conducted on the parameter space before results can be trusted.

[4]Performances labeled as W1, W2 and W3 in figure 5.6 are related to approximated values presented in references [17], [18] and [24], respectively.

[5]Figure 5.6a was magnified for the purpose of comparing data points on the top left corner.

6 Conclusion and Future Work

Results for both datasets showed that one can classify farmland with a low-cost camera and a UAV. The algorithm has achieved better performances in the distinction between vineyards species (dataset #2). Indeed, learning which is the type of grape present in a vineyard could provide insights for a farmer in order to decide the best way to treat these crops. Moreover, during the harvest process farmers might desire to differentiate from grapevine varieties to increase vine's quality. Here, segmentation 4.3.1 can be extremely accurate. In fact, datasets for each vineyard were collected for farmlands composed by just one variety. Therefore, there's no need to identify the ROI with low DOF samples. The PRS for dataset #1 has shown serious difficulties when distinguishing between olive trees and eucalyptus. Notwithstanding, the overall classification task has resulted in satisfactory results. The UAV is now capable of flying over farmlands of five monocultures and predict what is the class of which it is part of (*see* tables 5.1 and 5.2 for classification accuracy according to the two classifiers).

The ability to discriminate crops is significantly affected by the imagery spectral (type of camera), spatial (flight altitude) and temporal (the date of the study) resolutions. As a future work and in order to increase the classification accuracy, data might be caught with a multispectral camera allowing the capture of radiation of greater wavelengths. The more widely a camera's sensor bandwidth is, the more potentially accurate a PRS for farmland would be, once the number of features increases as well. Another interesting idea would be the use of GPS signals to create a territorial analysis (map) according to the classification predictions provided by the PRSs. Finally, the Mission Planner API allows the creation of grid maps to define what should be the flight plan for the UAV. However, the user is subject to application limitations. Therefore, the manipulation of the GPS information gathered by the flight controller would enable the user to program flight plans totally automated.

This work was partially submitted as a paper to the IEEE International Conference on Autonomous Robot Systems and Competitions [54].

References

[1] Alex McBratney, Brett Whelan, Tihomir Ancev, and Johan Bouma. Future directions of precision agriculture. *Precision Agriculture*, 6(1):7–23, February 2005.

[2] D.Turner, A. Lucieer, and C.Watson. Development of an unmanned aerial vehicle (UAV) for hyper resolution vineyard mapping based on visible, multispectral, and thermal imagery. *Proceedings of* 34^{th} *International Symposium on remote sensing of environment*, 2011.

[3] A. I. de Castro, F. López-Granados, J. M. Peña-Barragán, and M. Kelly. Object-based approach for crop row characterization in UAV images for site-specific weed management. *Proceedings of the* 4^{th} *GEOBIA*, 2012.

[4] Livio Pintoa, Diana Pagliaria, Daniele Masseroni, Bianca Ortuani, Arianna Facchi, Giovanna Sona, and Daniele Passoni. UAV multispectral survey to map soil and crop for precision farming applications. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLI-B1, July 2016.

[5] Edoardo Fiorillo Lorenzo Genesio Emanuele Lugato Alessandro Matese Francesco Primo Vaccari Jacopo Primicerio, Salvatore Filippo Di Gennaro. A flexible unmanned aerial vehicle for precision agriculture. *Precision Agric*, 13(4):517, August 2012.

[6] J.A.J. Berni, P.J. Zarco-Tejada, L. Suárez, V. González-Dugo, and E. Fereres. Remote sensing of vegetation from UAV platforms using lightweight multispectral and thermal imaging sensors. *Int. Arch. Photogramm. Remote Sens. Spatial Inform. Sci*, 38(6), 2009.

[7] Michaela De Giglio, Marco Dubbini, Sebastian Candiago, Fabio Remondino, and Mario Gattelli. Evaluating multispectral images and vegetation indices for precision farming applications from UAV images. *Remote Sensing*, 7(4):4026–4047, April 2015.

[8] Seiich Nohara Korehisa Kaneko1. Review of effective vegetation mapping using the UAV (unmanned aerial vehicle) method. *Journal of Geographic Information System*, 6:733–742, December 2014.

[9] Adam J. Mathews and Jennifer L. R. Jensen. Visualizing and quantifying vineyard canopy lai using an unmanned aerial vehicle (UAV) collected high density structure from motion point cloud. *Remote Sensing*, 5(5):2164–2183, April 2013.

[10] Javier Baluja, Maria P. Diago, Pedro Balda, Roberto Zorer, Franco Meggio, Fermin Morales, and Javier Tardaguila. Assessment of vineyard water status variability by thermal and multispectral imagery using an unmanned aerial vehicle (UAV). *Irrigation Science*, 30:511–522, 2012.

[11] Alberto-Jesús Perea-Moreno, Maria-Jesús Aguilera-Ureña, José-Emilio Meroño-De Larriva, and Francisco Manzano-Agugliaro. Assessment of the potential of UAV video image analysis for planning irrigation needs of golf courses. *Water*, 8(12):1–19, December 2016.

[12] T. Kattenborn, M. Sperlich, K. Bataua, and B. Koch. Automatic single tree detection in plantations using UAV-based photogrammetric point clouds. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL(3):139–144, September 2014. Gottingen: Copernius GmbH.

[13] G. Bareth, J. Bendig, and A. Bolten. Introducing a low-cost mini-UAV for thermal and multispectral-imaging. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIX(B1):345–349, September 2012.

[14] L. Pinto, G. Sona, R. Gini, and D.Passoni. Aerial images from an UAV system: 3D modeling and tree species classification in a park area. *Remote Sensing and Spatial Information Sciences*, 47(B1):251–269, September 2012.

[15] Robert Gafinkel, Elena Fernandez, and Roman Arbiol. Mosaicking of aerial photographic maps via seams defined by bottleneck shortest paths. *Operational Research*, 46(3):293–304, June 1998.

[16] Lars Arge, Pankaj K, Agarwal, and Andrew Danner. From point cloud to GRID DEM: A scalable approach. *Proceedings of International Symposium on Spatial Data Handling*, 2006. Wien, Austria.

[17] Rahul Sukthankar, Debadeepta Dey, and Lily Mummert. Classification of plant structures from uncalibrated image sequences. *Applications of Computer Vision (WACV)*, 2012.

[18] Mark Whitty, Scarlett Liu, and Samuel Marden. Towards automated yield estimation in viticulture. In *Australasian Conference on Robotics and Automation*. University of New South Wales, Sydney, Australia, December 2013.

[19] G. Camps-Valls, L. Gómez-Chova, J. Calpe-Maravilla, E. Soria-Olivas, J. D. Martín-Guerrero, and J. Moreno. Support vector machines for crop classification using hyperspectral data. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial intelligence and Lecture Notes in Bioinformatics)*, 2652:134–141, 2003.

[20] S. Homayouni, M. Hasanlou, B. Yekkehkhany, and A. Safari. A comparison study of different kernel functions for svm-based classification of multi-temporal polarimetry sar data. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL(2/W3):281–285, November 2014.

[21] J.F.G. Antunes, L.H.A. Rodrigues, and S.R. de M. Oliveira. Data mining for sugarcane crop classification using nodis data. *European Federation for Information Technology in Agriculture, Food; The Environment World Congress on Computers in Agriculture*, 8, 2011.

[22] Guilherme A.S. Megeto, Stanley R. de M. Oliveira, Emerson M. del Ponte, and Carlos A.A. Meira. Decision tree for classification of soybean rust occurrence in commercial crops bases on weather variables. *Engenharia Agrícola, Journal of the Brazillian Association of Agricultural Engineering*, 34(3):590–599, May 2014.

[23] Seçil Sencan. Decision tree classification of multi-temporal images for field-based cropmapping. Master's thesis, The Graduate School of Natural and Applied Sciences of Middle East Technical University, August 2004.

[24] Carlota Salinas, Javier Sarria, Roemi Fernández, Héctor Montes, and Manuel Armada. Combination of rgb and multispectral imagery for discrimination of cabernet sauvignon grapevine elements. *Sensors*, 13(6):7838–7859, June 2013.

[25] Roberto Cid Fernandes Jr., Antônio Kanaan, and Jean Michel S. de M. Gomes. As ferramentas do astrônomo: O que medimos, como medimos e o que aprendemos. Technical report, Observatórios Virtuais, March 2002.

[26] David Arthur and Sergi Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[27] Koen Hufkens. Raspberry pi camera spectral response curves. Software available at: `https://github.com/khufkens/pi-camera-response-curves#raspberry-pi-camera-spectral-response-curves`.

[28] Sean E. Shaheen, Christoph J. Brabec, N. Serdar Sariciftcia, Franz Padinger, Thomas Fromherz, and Jan C. Hummelen. 2.5% efficient organic plastic solar cells. *Applied Physics Letters*, 78(6):841–843, February 2001.

[29] Isabel Pôças, Arlete Rodrigues, Sara Gonçalvesa, Patrícia M. Costa, Igor Gonçalves, Luís S. Pereira, and Mário Cunha. Predicting grapevine water status based on hyperspectral reflectance vegetation indices. *Remote Sensing*, 7(12):16460–16479, December 2015.

[30] G. A. Blackburn. Spectral indices for estimating photosynthetic pigment concentrations: A test using senescent tree leaves. *International Journal of Remote Sensing*, 19(4):657–675, 1998. Published on-line November 2010.

[31] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2nd edition, 2001.

[32] Andrew Ng. Cs229 lecture notes support vector machines part v. See `http://cs229.stanford.edu/notes/cs229-notes3.pdf`.

[33] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, 2000.

[34] Lior Rokach and Oded Maimon. *Data Mining and Knowledge Discovery Handbook*. Number 2. Springer US, 2010.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. Software available at: `http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier`.

[36] Vishwa Nath Maurya, Diwinder Kaur Arora, and Er. Avadhesh Kumar Maurya. A survey report on non-parametric hypothesis testing including Kruskal-Wallis ANOVA and Kolmogorov-Smirnov goodness-fit-test. *International Journal of Information Technology & Operations Management*, 1(2):29–40, May 2013.

[37] G. Bradski. Image filtering (erode). *Dr. Dobb's Journal of Software Tools*, 2014. See `http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=erode`.

[38] G. Bradski. Image filtering (dilate). *Dr. Dobb's Journal of Software Tools*, 2014. See `http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=dilate#void`.

[39] Andrew Adams Marc Levoy and Nora Willett. Depth of field, March 2011. See `http://graphics.stanford.edu/courses/cs178-11/applets/dof.html`.

[40] Andrew Adams Marc Levoy and Nora Willett. Operation of a thin lens, March 2011. See `http://graphics.stanford.edu/courses/cs178-11/applets/thinlens.html`.

[41] David M. Pozar. *Microwave Engineering*. John Wiley & Sons, Inc., University of Massachusetts at Amherst, 4^{th} edition, 2012.

[42] Robert M. Gray, James Z.Wang, Jia Li, and Gio Wiederhold. Unsupervised multiresolution segmentation for images with low depth of field. *IEEE Transactions on pattern analysis and machine intelligence*, 23:85–90, January 2001.

[43] Monika Rathee and Alka Vij. Image compression using discrete haar wavelet transforms. *International Journal of Engineering and Innovative Technology (IJEIT)*, 3(12):47–51, June 2014.

[44] Patrick J. Van Fleet. *Discrete Wavelet Transformations: An Elementary Approach with Applications*. John Wiley & Sons, Inc., 2nd edition, February 2008.

[45] G. Bradski. K-means clustering in opencv. *Dr. Dobb's Journal of Software Tools*, 2014. See `http://docs.opencv.org/2.4/modules/core/doc/clustering.html`.

[46] J.C. Gower. Properties of euclidean and non-euclidean distance matrices. *Linear Algebra and its Applications*, 67(1):81–97, June 1985.

[47] Ana Cristina Rosa Esmeralda Gonçalves, Emília Nogueira. *Noções de Probabilidade e Estatística*. Universidade de Coimbra, February 2013.

[48] A.K. Bhandari, A. Kumar, and G.K. Singh. Feature extraction using normalized difference vegetation index (NDVI): a case study of Jabalpur city. *Procedia Technology*, 6(1):612–621, June 2012. 2^{nd} International Conference on Communication, Computing And Security.

[49] Takeshi Motohka, Kenlo Nishida Nasahara, Hiroyuki Oguma, and Satoshi Tsuchida. Applicability of green-red vegetation index for remote sensing of vegetation phenology. *Remote Sensing*, 2(1):2369–2387, October 2010.

[50] Angela Kross, Heather McNairn, David Lapen, Mark Sunohara, and Catherine Champagne. Assessment of RapidEye vegetation indices for estimation of leaf area index and biomass in corn and soybean crops. *International Journal of Applied Earth Observation and Geoinformation*, 34(1):235–248, February 2015.

[51] J. Torres-Sánchez, F. López-Granados, and J.M. Peña. An automatic object-based method for optimal thresholding in UAV images: Application for vegetation detection in herbaceous crops. *Computers and Electronics in Agriculture*, 114:43–52, June 2015.

[52] Rafael Muñoz Salinas. Raspicam: C++ api for using raspberry camera with/without opencv, 2014. Software available at: `http://www.uco.es/investiga/grupos/ava/node/40`.

[53] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, pages 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[54] João Natividade, José Prado, and Lino Marques. Low-cost multi-spectral vegetation classification using an unmanned aerial vehicle. January 2017. Submitted to IEEE International Conference on Autonomous Robot Systems and Competitions.

Appendices

A Classifiers Cross Validation

A.1 Support Vector Machines Configuration

Table A.1: Settings for the configuration of SVMs related to dataset #1.

Type	**Kernel**	**C**	**nu**	**D**	**Γ**	**CV Accuracy**	**# of Support Vectors**	**Precision**
1	2	—	0.05	—	1	0.44	144	0.45
1	2	—	0.1	—	1	0.62	183	0.47
1	2	—	0.15	—	1	0.68	195	0.71
1	2	—	0.2	—	1	0.72	208	0.84
1	2	—	0.05	—	10	0.66	189	0.77
1	2	—	0.1	—	10	0.73	215	0.80
1	2	—	0.15	—	10	0.77	221	0.87
1	2	—	0.2	—	10	0.78	240	0.87
1	2	—	0.05	—	100	0.72	265	0.68
1	2	—	0.1	—	100	0.73	287	0.60
1	2	—	0.15	—	100	0.74	307	0.75
1	2	—	0.2	—	100	0.76	315	0.75
1	2	—	0.05	—	1000	0.67	398	0.49
1	0	—	0.05	—	—	0.35	103	0.28
1	0	—	0.1	—	—	0.37	148	0.40
1	0	—	0.15	—	—	0.43	193	0.60
1	0	—	0.2	—	—	0.5	225	0.66
0	2	1	—	—	1000	0.66	408	0.58
0	1	1	—	1	1	0.57	387	0.33
0	1	1	—	1	10	0.68	338	0.68

Continued on next page

Table A.1 – continued from previous page

Type	Kernel	C	nu	D	Γ	CV Accuracy	# of Support Vectors	Precision
0	1	1	—	1	100	0.734	298	0.81
0	1	1	—	1	1000	0.75	255	0.80
0	1	1	—	2	1	0.67	338	0.64
0	1	1	—	2	10	0.75	261	0.79
0	1	1	—	2	100	0.77	204	0.81
0	1	1	—	2	1000	0.8	157	0.76
0	1	1	—	3	1	0.72	312	0.76
0	1	1	—	3	10	0.78	209	0.80
0	1	1	—	4	1	0.75	280	0.81
0	1	1	—	4	10	0.79	173	0.85
0	1	1	—	5	1	0.77	246	0.79
0	1	1	—	6	1	0.78	219	0.77
0	1	1	—	7	1	0.76	218	0.80
0	2	10	—	—	1	0.7	325	0.78
0	2	10	—	—	10	0.76	269	0.85
0	2	10	—	—	100	0.75	302	0.80
0	2	10	—	—	1000	0.66	399	0.49
0	1	10	—	1	1	0.68	338	0.68
0	1	10	—	1	10	0.69	325	0.81
0	1	10	—	1	100	0.73	299	0.80
0	1	10	—	1	1000	0.75	255	0.80
0	1	10	—	2	1	0.76	243	0.80
0	1	10	—	2	10	0.79	224	0.78
0	1	10	—	3	1	0.76	267	0.81
0	1	10	—	4	1	0.78	237	0.78
0	2	100	—	—	1	0.76	266	0.82
0	2	100	—	—	10	0.78	218	0.84
0	2	100	—	—	100	0.75	288	0.66
0	2	100	—	—	1000	0.66	398	0.53
0	1	100	—	1	1	0.73	298	0.82

Continued on next page

Table A.1 – continued from previous page

Type	Kernel	C	nu	D	Γ	CV Accuracy	# of Support Vectors	Precision
0	1	100	—	1	10	0.75	255	0.80
0	1	100	—	1	100	0.76	244	0.80
0	1	100	—	2	1	0.75	261	0.79
0	1	100	—	2	10	0.78	244	0.81
0	1	100	—	3	1	0.79	234	0.78
0	**2**	**1000**	—	—	**1**	**0.78**	**144**	**0.93**
0	2	1000	—	—	10	0.78	197	0.89
0	2	1000	—	—	100	0.73	274	0.71
0	2	1000	—	—	1000	0.66	398	0.53
0	1	1000	—	1	1	0.75	255	0.80
0	1	1000	—	1	10	0.76	245	0.80
0	1	1000	—	2	1	0.79	224	0.78
0	**1**	**1000**	—	**3**	**1**	**0.69**	**144**	**0.82**
0	0	1	—	—	—	0.57	387	0.33
0	0	10	—	—	—	0.68	338	0.68
0	0	100	—	—	—	0.73	298	0.81
0	0	1000	—	—	—	0.75	255	0.80
0	0	10000	—	—	—	0.76	245	0.80
1	1	—	0.05	1	1	0.62	398	0.28
1	1	—	0.05	1	10	0.64	200	0.38
1	1	—	0.05	1	100	0.61	148	0.20
1	1	—	0.05	1	1000	0.63	193	0
1	1	—	0.05	2	1	0.68	225	0.39
1	1	—	0.05	2	10	0.67	408	0.27
1	1	—	0.05	2	100	0.63	397	0.18
1	1	—	0.05	3	1	0.64	338	0.64
1	1	—	0.05	3	10	0.7	298	0.24
1	1	—	0.05	4	1	0.69	255	0.26
1	1	—	0.05	4	10	0.75	338	0.53
1	1	—	0.05	5	1	0.66	261	0.33

Continued on next page

Table A.1 – continued from previous page

Type	Kernel	C	nu	D	Γ	CV Accuracy	# of Support Vectors	Precision
1	1	—	0.05	6	1	0.65	204	0.39
1	1	—	0.05	7	1	0.69	197	0.45

Table A.2: Settings for the configuration of SVMs related to dataset #2.

Type	Kernel	C	nu	D	Γ	CV Accuracy	# of Support Vectors	Ratio
1	2	—	0.05	—	1	0.5	118	0.73
1	2	—	0.1	—	1	0.64	170	0.69
1	2	—	0.15	—	1	0.73	192	0.71
1	2	—	0.2	—	1	0.78	230	0.78
1	2	—	0.05	—	10	0.8	186	0.70
1	2	—	0.1	—	10	0.83	200	0.82
1	2	—	0.15	—	10	0.84	214	0.84
1	2	—	0.2	—	10	0.85	246	0.87
1	2	—	0.05	—	100	0.81	215	0.97
1	2	—	0.1	—	100	0.82	236	0.95
1	2	—	0.15	—	100	0.81	254	0.92
1	2	—	0.2	—	100	0.82	272	0.90
1	2	—	0.05	—	1000	0.75	450	0.97
1	0	—	0.05	—	—	0.51	82	0.56
1	0	—	0.1	—	—	0.48	138	0.50
1	0	—	0.15	—	—	0.62	174	0.49
1	0	—	0.2	—	—	0.7	204	0.62
0	2	1	—	—	1000	0.77	463	0.95
0	1	1	—	1	1	0.67	417	0.79
0	1	1	—	1	10	0.72	376	0.73
0	1	1	—	1	100	0.74	359	0.74
0	1	1	—	1	1000	0.76	330	0.74
0	1	1	—	2	1	0.72	374	0.74

Continued on next page

Table A.2 – continued from previous page

Type	Kernel	C	nu	D	Γ	CV Accuracy	# of Support Vectors	Precision
0	1	1	—	2	10	0.77	332	0.78
0	1	1	—	2	100	0.85	251	0.78
0	1	1	—	3	1	0.74	369	0.75
0	1	1	—	3	10	0.85	253	0.77
0	1	1	—	3	100	0.82	131	0.82
0	1	1	—	4	1	0.76	357	0.75
0	1	1	—	4	10	0.86	187	0.82
0	1	1	—	5	1	0.78	329	0.76
0	1	1	—	6	1	0.8	292	0.76
0	1	1	—	7	1	0.85	258	0.78
0	2	10	—	—	1	0.74	356	0.78
0	2	10	—	—	10	0.79	124	0.78
0	2	10	—	—	100	0.79	271	0.78
0	2	10	—	—	1000	0.75	454	0.90
0	1	10	—	1	1	0.72	376	0.96
0	1	10	—	1	10	0.74	359	0.73
0	1	10	—	1	100	0.76	330	0.74
0	1	10	—	1	1000	0.77	289	0.74
0	1	10	—	2	1	0.74	364	0.69
0	1	10	—	2	10	0.82	285	0.74
0	1	10	—	3	1	0.76	345	0.76
0	1	10	—	4	1	0.8	308	0.73
0	2	100	—	—	1	0.77	316	0.77
0	2	100	—	—	10	0.82	241	0.78
0	2	100	—	—	100	0.82	233	0.82
0	**2**	**100**	—	—	**1000**	**0.83**	**199**	**0.97**
0	1	100	—	1	1	0.74	359	0.98
0	1	100	—	1	10	0.76	330	0.74
0	1	100	—	1	100	0.77	290	0.74
0	1	100	—	2	1	0.77	332	0.69

Continued on next page

Table A.2 – continued from previous page

Type	Kernel	C	nu	D	Γ	CV Accuracy	# of Support Vectors	Precision
0	1	100	—	2	10	0.85	249	0.74
0	1	100	—	3	1	0.81	298	0.78
0	2	1000	—	—	1	0.82	273	0.76
0	2	1000	—	—	10	0.87	196	0.78
0	2	1000	—	—	100	0.82	207	0.87
0	2	1000	—	—	1000	0.75	449	0.97
0	1	1000	—	1	1	0.76	330	0.98
0	1	1000	—	1	10	0.77	290	0.74
0	1	1000	—	2	1	0.86	208	0.68
0	1	1000	—	3	1	0.84	251	0.77
0	0	1	—	—	—	0.67	417	0.78
0	0	10	—	—	—	0.72	376	0.80
0	0	100	—	—	—	0.74	359	0.73
0	0	1000	—	—	—	0.76	330	0.74
0	0	10000	—	—	—	0.78	291	0.74
1	1	—	0.05	1	1	0.73	256	0.56
1	1	—	0.05	1	10	0.72	245	0.53
1	1	—	0.05	1	100	0.73	278	0.29
1	1	—	0.05	1	1000	0.70	199	0.46
1	1	—	0.05	2	1	0.69	210	0.55
1	1	—	0.05	2	10	0.66	230	0.28
1	1	—	0.05	2	100	0.69	235	0.14
1	1	—	0.05	3	1	0.71	260	0.38
1	1	—	0.05	3	10	0.74	225	0.61
1	1	—	0.05	4	1	0.75	231	0.67
1	1	—	0.05	4	10	0.72	267	0.51
1	1	—	0.05	5	1	0.68	238	0.34
1	1	—	0.05	6	1	0.67	198	0.43
1	1	—	0.05	7	1	0.68	198	

A.2 Decision Trees Configuration

Table A.3: Settings for the configuration of DTs related to dataset #1.

Criterion	Maximum Depth	Minimum Samples to Split	Minimum Samples per Leaf	CV Accuracy	Precision
0	5	2	1	0.54	0.59
0	5	3	1	0.6	0.61
0	5	4	1	0.58	0.62
0	5	5	1	0.59	0.63
0	7	2	1	0.62	0.55
0	7	3	1	0.63	0.58
0	7	4	1	0.63	0.64
0	7	5	1	0.64	0.49
0	9	2	1	0.63	0.68
0	9	3	1	0.63	0.67
0	9	4	1	0.64	0.47
0	9	5	1	0.65	0.70
0	11	2	1	0.61	0.52
0	11	3	1	0.59	0.61
0	11	4	1	0.63	0.58
0	11	5	1	0.6	0.63
0	N	2	1	0.6	0.70
0	N	3	1	0.59	0.54
0	N	4	1	0.63	0.66
0	N	5	1	0.6	0.71
0	5	2	2	0.59	0.63
0	5	3	2	0.62	0.66
0	5	4	2	0.58	0.43
0	5	5	2	0.6	0.68
0	7	2	2	0.61	0.60
0	7	3	2	0.6	0.66
0	7	4	2	0.61	0.55

Continued on next page

Table A.3 – continued from previous page

Criterion	Maximum Depth	Minimum Samples to Split	Minimum Samples per Leaf	CV Accuracy	Precision
0	7	5	2	0.61	0.64
0	9	2	2	0.6	0.66
0	9	3	2	0.61	0.62
0	9	4	2	0.61	0.58
0	9	5	2	0.6	0.61
0	11	2	2	0.59	0.54
0	11	3	2	0.6	0.69
0	11	4	2	0.59	0.50
0	11	5	2	0.58	0.61
0	N	2	2	0.56	0.65
0	N	3	2	0.59	0.63
0	N	4	2	0.6	0.69
0	N	5	2	0.64	0.60
1	5	2	1	0.61	0.55
1	**5**	**3**	**1**	**0.71**	**0.58**
1	5	4	1	0.62	0.55
1	5	5	1	0.58	0.62
1	7	2	1	0.6	0.64
1	7	3	1	0.63	0.69
1	7	4	1	0.64	0.57
1	7	5	1	0.6	0.55
1	9	2	1	0.6	0.56
1	9	3	1	0.64	0.63
1	9	4	1	0.59	0.60
1	9	5	1	0.63	0.71
1	11	2	1	0.62	0.55
1	11	3	1	0.62	0.73
1	11	4	1	0.59	0.57
1	11	5	1	0.59	0.59
1	N	2	1	0.63	0.58

Continued on next page

Table A.3 – continued from previous page

Criterion	Maximum Depth	Minimum Samples to Split	Minimum Samples per Leaf	CV Accuracy	Precision
1	N	3	1	0.6	0.61
1	N	4	1	0.63	0.57
1	N	5	1	0.67	0.65
1	5	2	2	0.63	0.54
1	5	3	2	0.6	0.52
1	**5**	**4**	**2**	**0.77**	**0.87**
1	5	5	2	0.57	0.70
1	7	2	2	0.61	0.57
1	7	3	2	0.58	0.74
1	7	4	2	0.62	0.55
1	7	5	2	0.63	0.65
1	9	2	2	0.63	0.54
1	9	3	2	0.58	0.64
1	9	4	2	0.61	0.65
1	9	5	2	0.62	0.52
1	11	2	2	0.61	0.54
1	11	3	2	0.63	0.64
1	11	4	2	0.58	0.57
1	11	5	2	0.59	0.69
1	N	2	2	0.58	0.53
1	N	3	2	0.61	0.56
1	N	4	2	0.63	0.56
1	N	5	2	0.62	0.61

Table A.4: Settings for the configuration of DTs related to dataset #2.

Criterion	Maximum Depth	Minimum Samples to Split	Minimum Samples per Leaf	CV Accuracy	Precision
0	5	2	1	0.63	0.81
0	5	3	1	0.64	0.73

Continued on next page

Table A.4 – continued from previous page

Criterion	Maximum Depth	Minimum Samples to Split	Minimum Samples per Leaf	CV Accuracy	Precision
0	5	4	1	0.61	0.80
0	5	5	1	0.68	0.73
0	7	2	1	0.64	0.85
0	7	3	1	0.65	0.88
0	7	4	1	0.64	0.86
0	7	5	1	0.65	0.88
0	9	2	1	0.63	0.93
0	9	3	1	0.65	0.93
0	9	4	1	0.64	0.92
0	9	5	1	0.67	0.93
0	11	2	1	0.64	0.98
0	11	3	1	0.67	0.95
0	11	4	1	0.62	0.96
0	11	5	1	0.63	0.92
0	N	2	1	0.64	0.98
0	N	3	1	0.64	0.98
0	N	4	1	0.64	0.96
0	N	5	1	0.65	0.95
0	5	2	2	0.67	0.80
0	5	3	2	0.67	0.82
0	5	4	2	0.65	0.77
0	5	5	2	0.63	0.80
0	7	2	2	0.67	0.83
0	7	3	2	0.66	0.78
0	7	4	2	0.66	0.85
0	7	5	2	0.67	0.85
0	9	2	2	0.63	0.85
0	9	3	2	0.64	0.84
0	9	4	2	0.65	0.89
0	9	5	2	0.66	0.92

Continued on next page

Table A.4 – continued from previous page

Criterion	Maximum Depth	Minimum Samples to Split	Minimum Samples per Leaf	CV Accuracy	Precision
0	11	2	2	0.66	0.93
0	11	3	2	0.66	0.94
0	11	4	2	0.66	0.96
0	11	5	2	0.65	0.90
0	N	3	2	0.63	0.93
0	N	4	2	0.65	0.93
0	N	5	2	0.64	0.94
1	5	2	1	0.64	0.94
1	5	3	1	0.69	0.82
1	5	4	1	0.67	0.78
1	5	5	1	0.61	0.73
1	7	2	1	0.68	0.87
1	7	3	1	0.64	0.86
1	7	4	1	0.67	0.85
1	7	5	1	0.64	0.86
1	9	2	1	0.62	0.87
1	9	3	1	0.66	0.86
1	9	4	1	0.64	0.94
1	9	5	1	0.65	0.86
1	11	2	1	0.66	0.87
1	11	3	1	0.66	0.94
1	11	4	1	0.67	0.97
1	11	5	1	0.60	0.96
1	**N**	**2**	**1**	**0.77**	**0.87**
1	N	3	1	0.67	0.98
1	N	4	1	0.65	0.96
1	N	5	1	0.68	0.95
1	5	2	2	0.68	0.95
1	5	3	2	0.65	0.77
1	5	4	2	0.68	0.81

Continued on next page

Table A.4 – continued from previous page

Criterion	Maximum Depth	Minimum Samples to Split	Minimum Samples per Leaf	CV Accuracy	Precision
1	5	5	2	0.64	0.77
1	7	2	2	0.67	0.69
1	7	3	2	0.66	0.84
1	7	4	2	0.63	0.87
1	7	5	2	0.67	0.85
1	9	2	2	0.64	0.82
1	9	3	2	0.65	0.90
1	9	4	2	0.65	0.86
1	9	5	2	0.66	0.87
1	11	2	2	0.66	0.89
1	11	3	2	0.64	0.92
1	11	4	2	0.65	0.92
1	11	5	2	0.65	0.93
1	N	2	2	0.65	0.92
1	N	3	2	0.64	0.94
1	N	4	2	0.67	0.91
1	N	5	2	0.63	0.90