International Journal of Geo-Information

MDPI

Technical Note

Development of a QGIS Plugin to Obtain Parameters and Elements of Plantation Trees and Vineyards with Aerial Photographs

Lia Duarte [1,2,*], Pedro Silva [1] and Ana Cláudia Teodoro [1,2]

[1] Department of Geosciences, Environment and Land Planning, Faculty of Sciences, University of Porto, Porto 4169-007, Portugal; up201007485@fc.up.pt (P.S.); amteodor@fc.up.pt (A.C.T.)
[2] Earth Sciences Institute (ICT), Faculty of Sciences, University of Porto, Porto 4169-007, Portugal
* Correspondence: liaduarte@fc.up.pt; Tel.: +351-220-402-477

Received: 4 January 2018; Accepted: 12 March 2018; Published: 14 March 2018

Abstract: Unmanned Aerial Vehicle (UAV) imagery allows for a new way of obtaining geographic information. In this work, a Geographical Information System (GIS) open source application was developed in QGIS software that estimates several parameters and metrics on tree crown through image analysis techniques (image segmentation and image classification) and fractal analysis. The metrics that have been estimated were: area, perimeter, number of trees, distance between trees, and a missing tree check. This methodology was tested on three different plantations: olive, eucalyptus, and vineyard. The application developed is free, open source and takes advantage of QGIS integration with external software. Several tools available from Orfeo Toolbox and Geographic Resources Analysis Support System (GRASS) GIS were employed to generate a classified raster image which allows calculating the metrics referred before. The application was developed in the Python 2.7 language. Also, some functions, modules, and classes from the QGIS Application Programming Interface (API) and PyQt4 API were used. This new plugin is a valuable tool, which allowed for automatizing several parameters and metrics on tree crown using GIS analysis tools, while considering data acquired by UAV.

Keywords: unmanned aerial vehicle; multispectral imagery; NDVI imagery; QGIS; python; plugin; imagery segmentation; imagery classification; fractal analysis; plantations; orthophotos

1. Introduction

Remote sensing techniques provide useful information for agricultural management enhancing the reliable automation of mapping procedures [1]. In recent years, Unmanned Aerial Vehicle (UAV) imagery has allowed a new way of getting geographic information from remote sensing data [2]. UAV technology provides several advantages compared to a manned aircraft such as: (i) significantly lower qualification requirements for UAV pilots; (ii) allows for mapping small areas, resulting in very high spatial resolution (VHR) images, which are ideal for detecting small objects; (iii) low cost acquisition; and, (iv) it can couple different sensors [2]. UAVs are increasingly used in different contexts and in several research areas, such in cartography, environmental studies, cultural heritage, civil engineering, forestry, and agriculture/precision agriculture [3–8]. Therefore, when considering UAV imagery, it is possible to acquire information more efficiently and inexpensively [9]. UAV imagery have been used to estimate metrics from trees in several crops or farms, which can be of interest of farmers in order to obtain a better planning of the irrigation process, helping in the forecast of the production date and climate change effects as a consequence of deforestation and degradation of forests, which leads to global warming, precision forestry, forest management planning, and biomass estimation [2,10–13]. Most of the work conducted with UAV data has been related with forest areas [11,12,14–19]. However, in the field of agriculture crops, it is relevant to study the geometric characterization of trees and

plantations. This is a complex task, being particularly important to understand the tree growth and productivity [1,9,13,20–22]. In the field of precision agriculture, several image processing methods for automatic detection and delineation of individual trees were already implemented [1]. The information regarding tree crown areas of a specific crop is the key in the study of ecological value and ratio of the occupied area by that crop [9,15,23]. Traditionally, tree counting has been performed manually based on field information and statistical data, being an expensive process, time consuming, and a tedious task subject to several errors [13,14,22]. Satellite, manned aircraft, or UAV imagery can be used to obtain these data. Therefore, the implementation of automatic processes is crucial to extract information and provide forest or crops inventories from remote sensing data [2,13,22].

Several studies have been developed with different techniques to estimate the tree crown delineation, tree counting, and tree characterization from remote sensing imagery. Deng et al. [23] proposed an artificial intelligence algorithm based on seeded based region growth method to extract tree crown from Quickbird satellite images. In this work, after the segmentation process, a condition regarding the Normalized Difference Vegetation Index (NDVI > 0.38) was considered. Santoro et al. [1] proposed an algorithm based on an asymmetric weighted kernel optimized for tree counting using GeoEye-1 satellite imagery. Park et al. [10] developed a tree isolation algorithm based on aerial imagery for precise tree detection and crown delineation. Through UAV imagery acquired from a Hexacoptor, Bazi et al. [2] extracted a set of candidate key points using Scale Invariant Feature Transform (SIFT) analyzed with the Extreme Learning Machine (ELM), which is a kernel-based classification method. Processing operations using mathematical morphology were used to distinguish palm trees from other type of vegetation. Kang et al. [9] proposed an identification method for tree crown areas from imagery captured by UAV. The methodology was based on mathematical morphological filtering, unsupervised segmentation based on J-value SEGmentation (JSEG), local spatial statistics and Iterative Self-Organizing Data Analysis Technique Algorithm (ISODATA). Panagiotidis et al. [24] used the Structure from motion (Sfm) approach to generate a point cloud that was used to calculate a Digital Surface Model (DSM) and an orthomosaic. This information was used as input data for tree crown delineation using local maxima filtering (for tree heights) and Inverse Watershed Segmentation (IWS) for tree crown detection. Thiel and Schmullius [17] also used Sfm for generating remote sensing products from UAV imagery and LASTools software package to generate the point cloud. The tree detection was based on pit-free Canopy Height Models. Hassaan et al. [11] created a script in Matlab to remove barrel distortions from UAV images, k-means clustering in the segmentation stage, and texture analysis. Finally, to count the trees a circle fitting approximation technique was used. Nevalainen et al. [12] investigated the performance of UAV and hyperspectral imagery in individual tree detection and tree species classification in forests computing the spectral features using Matlab. Chianucci et al. [25] used Red-Green-Blue (RGB) UAV images to segment tree canopies (tree/non tree mask) based on visible-vegetation indices and colour space conversion and conclude that the true colour UAV images can be effectively used to obtain meaningful estimations of forest crown attributes.

Object-Based Image Analysis (OBIA) has proven to be a valuable approach to image segmentation and classification when considering the form, color, and textures of the pixels, minimizing the boundary problem [22,26–29]. OBIA groups the local homogeneous pixels, i.e., it allows for mapping the VHR imagery into meaningful objects [27].

In the referred studies proprietary software were considered. However, there are several open source software available such as Spring, Orfeo ToolBox (OTB)/Monteverdi, GRASS and QGIS which are complemented with processing image algorithms [28]. The open source Geographical Information System (GIS) software, such as QGIS, presented some advantage since it integrates algorithms from OTB and GRASS through *Processing Toolbox* framework [30–33]. The UAV imagery can be considered as an input in GIS applications as it provides the tools that are required to correctly manipulate, analyze, and incorporate geographic information. In fact, the open source concept implies the creation/development of new applications. For instance, Grippa et al. [34] presented a semi-automated processing chain, developed in Python language, for object-based land-cover and land-use classification

using GRASS GIS and R open source software. Also, Duarte et al. [35] developed a GIS open source application for image processing automatizing the photogrammetric procedures, such as computing tie points between pairs of images, computing relative orientations based on calibration models, sparsing three-dimensional (3D) models, generating orthomosaics, point cloud, DSM, and shaded reliefs, through the integration of MicMac open source software in QGIS.

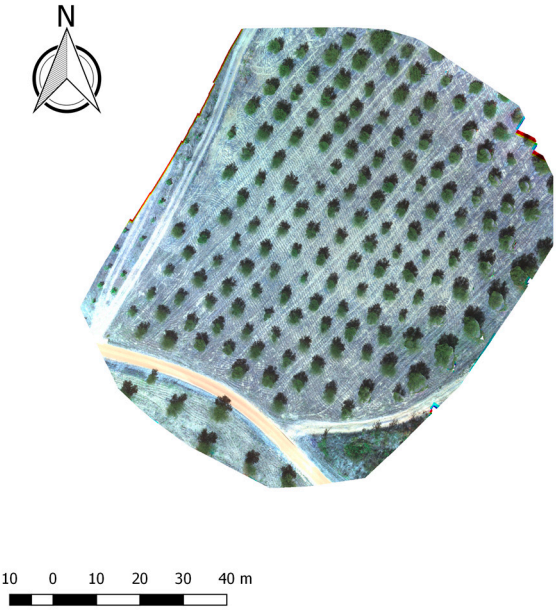
The objective of this work was the creation of a GIS open source application/plugin—Tree Crown—that estimates several parameters and metrics on tree crown through image analysis techniques (image segmentation and classification) and fractal analysis.

The innovation of this work is highlighted by: (i) the literature analysis—there is not a GIS application available that provides specifically the metrics regarding tree crown; (ii) the developing and implementation of the methodology in a new plugin, so the users that are not familiar with the algorithms presents in QGIS software can, easily, estimate the parameters with a simple tool; (iii) the GIS application was developed in Python language, which is a free programming language, so that it allows for the modification or adaptation by any user according to specific rules; (iv) we respect the freedoms of open source software using QGIS software which provide the possibility of incorporating algorithms and functions from others software; and, (v) the fractal analysis was only feasible with an automatic implementation.

2. Study case and Data Acquisition

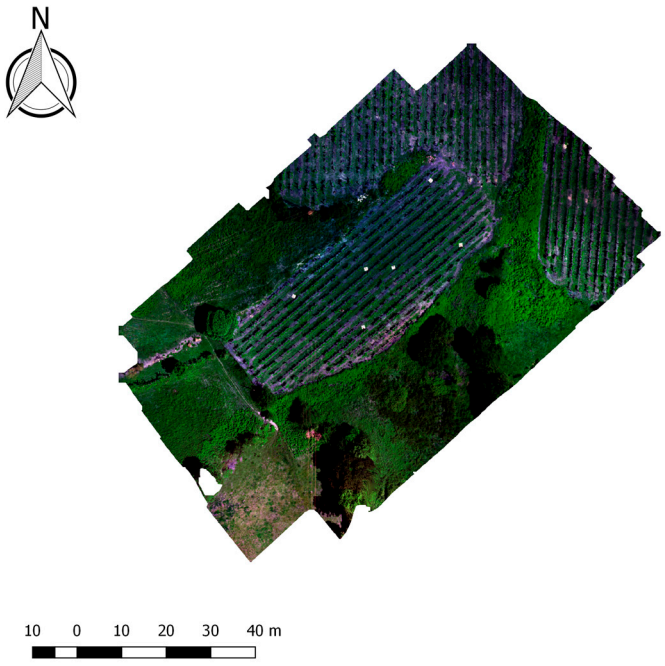
2.1. Study Case

Three study cases were considered: an orthomosaic from an olive crop acquired with an UAV imagery with 2.75 ha (Mirandela, Bragança, Portugal in August 2016, 41°37′18.895″ N; 7°10′20.182″ W), another from a eucalypt crop with 0.77 ha (Chamusca, Santarém, Portugal in December 2016, 39°20′39.686″ N; 8°23′28.989″ W) and the last one considered was from a vineyard crop with 1.57 ha (Corvo, Fafe, Portugal in May 2017, 41°31′03.124″ N; 8°13′00.997″ W). In Figure 1 are presented the three orthomosaics that are considered in this work.

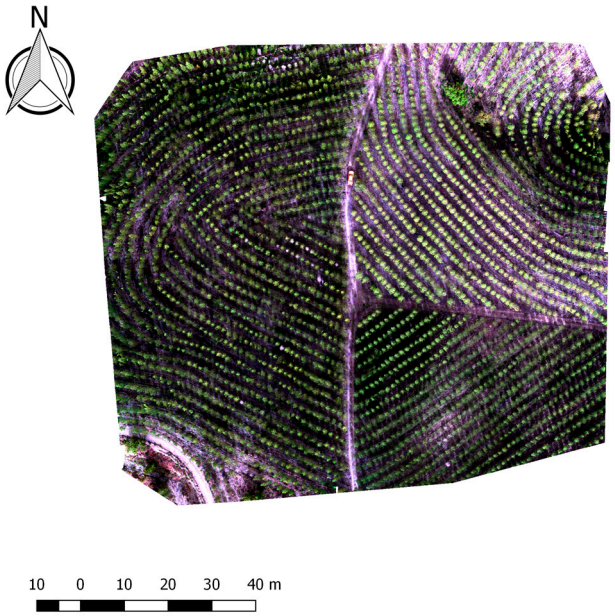


(**a**)

Figure 1. *Cont.*



(**b**)



(**c**)

Figure 1. Study cases: (**a**) olive crop; (**b**) eucalyptus crop; and, (**c**) vineyard crop.

2.2. Data Acquisition

The image corresponding to each study case was obtained with a rotary-wing UAV (3DR Solo) with a *MicaSense RedEdge* camera which is composed by five bands: Red, Green, Blue, Red edge, and Near-Infrared (NIR). The coordinate reference system adopted was Universal Transverse Mercator (UTM) Zone 29 North—World Geodetic System 1984 (WGS84; EPSG:32629) for olive crop image (with a spatial resolution of 2.8 cm), and European Terrestrial Reference System 1989 (ETRS89) Portugal Transverse Mercator 2006 (PTTM06; EPSG:3763) for eucalypt (with a spatial resolution of 3.6 cm) and vineyard images (with a spatial resolution of 5.2 cm). It was performed one single cloud-free flight to

each study case. The flight height to olive crop was 50 m, for eucalypt was 120 m and for vineyard was 80 m. After, for each image, the NDVI was computed using Red and NIR bands [36].

3. Methodology

3.1. GIS Software, Python Libraries and Tree Crown Application

The application was created and implemented in QGIS software (version 2.18 *Las Palmas*), an open source software, which consider the Stallman four freedoms, namely the possibility to study and modify the code, to distribute the program with different modified versions [32,33]. QGIS allows for the development of new plugins in Python 2.7 programming language [37] and provide online support through forums, tutorials, and documentation [33]. It allows for vector and raster files manipulation, visualization, analysis, and acquisition. To create a new plugin, different libraries and/or Application Programming Interface (API) are commonly used, such as Geographic Resources Analysis Support System (GRASS), QGIS API, and PyQt4 API [30,38,39]. Also, from QGIS 2.0, a *Processing Toolbox*, a framework containing several algorithms of standalone GIS software, such as GRASS 7.2.2 and OTB 5.0.0, was integrated [31]. The OTB software is an open source project for remote sensing and it is composed by several algorithms, which allows for processing high resolution, multispectral, and RADAR images, and includes several image processing procedures, such as ortho-rectification, pan sharpening, classification algorithms, image manipulation, and segmentation [31].

This plugin was concretized as a button in QGIS main menu. The plugin graphic interface was developed through *Qt Designer* package where it is possible to create and personalize the widgets such as combo boxes, push buttons, labels, among others [40].

The main graphic interface is a window composed by four fields, two inputs and two outputs (Figure 2). This fields were defined as push buttons and line edits. The functions related to each push button are, respectively, *inputfile*, *inputfile2*, *output*, and *output2*. The main procedure was implemented in the function *run*. In this function the image details, such as extension, number of bands and multispectral bands, were accessed using functions from *QgsRasterLayer()* class [38]. In order to access to *Processing Toolbox* algorithms, the *Processing* module was imported and the function *runAlgorithm()* was used. The experimental procedure performed allowed to select the most accurate methodology to be implemented in the plugin. The methodology implementation is described in the Section 3.2.1, Section 3.2.2, and Section 3.2.3.

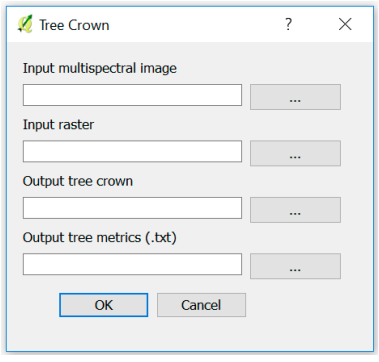


Figure 2. Plugin graphic interface.

3.2. Procedures Tested

To define the methodology to be implemented in the plugin, different tools were tested and compared. In the end, the more accurate methodology was chosen. Based on the literature consulted and in the tools existing in QGIS, the OBIA classification approach was chosen. The Sections 3.2.1 and 3.2.2 describe the tools that are available in QGIS to perform OBIA segmentation and classification.

3.2.1. OBIA Segmentation

For the segmentation process, two algorithms from GRASS and OTB, respectively, *i.segment* and *Watershed segmentation* were tested. The first approach performs the segmentation based in the region growing algorithm and the second uses the watershed algorithm [41]. The *i.segment* sequentially examines all of the segments in the raster file and calculates a similarity measure between the current segment and each of its neighbors, according to a given distance [30]. This algorithm was computed considering two parameters: the *thresholding*, which represents the similarity between 0 and 1; and, the object minimum size (*minsize*), which corresponds to the minimum size value that each object can have. In *thresholding*, a higher value aggregates objects with similar properties. Through a neighbor relation, the pixels that presented similar spectral and spatial characteristics to the seed points were aggregated, dividing the image into objects.

When considering the region growing method, different values of *thresholding* and *minsize* were tested in order to optimize the procedure. The *thresholding* was tested with values ranging from 0.05 to 0.95 (unitless), in which a *loop* cycle was created, and where the first image created (with 0.05 of *thresholding*) contained the seed points and the following images were obtained recursively from the first one; was considered an increment in *thresholding* value of 0.05 [41]. Also, the *minsize* value was randomly tested when considering the values 10, 100, and 200. In this segmentation process, the tool accepts the multispectral image with separated bands and with non-separated bands as inputs. The other parameters were defined by default.

The *Watershed segmentation* algorithm from OTB was tested with the default values that were provided by QGIS [33].

In Section 4.1.1, the results that were obtained with the two algorithms tested will be presented. The more robust algorithm implemented was the *i.segment*. In the Tree Crown application, to split the multispectral image by bands, the *split image*, algorithm from OTB, was employed. The first seed image was processed with 0.05 of *thresholding*. Then, a *for* loop was created to use *i.segment* algorithm with an increment of 0.05 and the *thresholding* values varied between 0.1 and 0.65. The final image was obtained considered a *thresholding* of 0.65.

3.2.2. OBIA Classification

In image classification, two algorithms, from OTB, were tested: supervised classification using the *Image classification* algorithm, through the Support Vector Machine (SVM) classifier, and unsupervised classification through the *Unsupervised kmeans image classification* algorithm [42,43].

In the supervised classification, the mean and standard deviation values of each band were obtained from a XML file generated from *Compute images second order statistics* algorithm from OTB. The next step consisted in a shapefile creation with two training areas: the tree crown class (assigning as 1) and remaining objects (assigning as 0). The SVM method was used through *TrainImagesClassifier (svm)*, algorithm from OTB, to perform classifier training from pairs of images [30]. Finally, the classification was performed using the *Image Classification* algorithm with the multispectral image and the NDVI image as inputs, and the XML file containing the mean and standard deviation values, the segmented image generated before and the SVM resulting file [42]. The classification accuracy was obtained through *r.kappa* algorithm from GRASS GIS 7, which compute the error matrix of the classification result by crossing the classified map layer with respect to reference map layer (training areas). The error matrix and the kappa value were returned.

In the unsupervised classification (k-means algorithm), the clusters (center points of each class) are defined and the pixels are classified based on their distance to the cluster. When considering that the NIR or RED bands from low-cost UAV sensors are usually less radiometrically robust than RGB bands, the multispectral image, the NDVI image, and the RGB true colour combination were used as inputs into the classification process. The training areas were defined with the *Training set size* parameter, assuming the value of 100,000 pixels [44]. The validation mask corresponded to the segmented image and the other parameters were defined by default.

The classification algorithm chosen was the unsupervised classification algorithm implemented through the *Unsupervised kmeans image classification* from OTB. In this algorithm the input can be the RGB combination, the multispectral or the NDVI image, and the segmented image as validation mask. The training set size parameter was obtained when multiplying the total number of pixels by 0.00705 to obtain the value 100,000 and the remaining parameters were defined by default. Then, the maximum value of the resulted image was extracted and associated to the tree crown using functions from *QgsRasterLayer()*. To distinguish the tree crown from the other objects, the *Rastercalculator*, algorithm from GDAL, was used, assigning the value 1 to tree crown. The image obtained was converted to shapefile through *Polygonize* algorithm, from GDAL, which was interpreted using functions from *QgsVectorLayer()* class. To restrict the objects only to tree crown, were not considered: (i) objects with an area lower than 1 m^2 or higher than 50 m^2; (ii) elements with the class value different from 1; and, (iii) objects with the ratio between height and width of the polygon bounding box, which is considered as a regular polygon where the extension values (x minimum, x maximum, y minimum, and y maximum) correspond to the extension of an irregular polygon. In the case of tree crown, it is expected a quadrangular polygon, so this ratio should be closest to 1. However, in this plugin, a tolerance of 0.55 was considered. The ratio was obtained implementing a *for* loop to extract the area, height and width. In the end, the features, which satisfy at least one of the above criteria, were excluded from the shapefile.

3.2.3. Fractal Analysis

The fractal analysis was applied in order to obtain the tree crown parameters and metrics in shapefile format. To perform this, the resulting image from OBIA need to be converted to binary image, where 1 corresponded to the tree crown and 0 to the remaining objects. The resulted image was then converted to shapefile through *Polygonize*, algorithm from Geospatial Data Abstraction Library (GDAL) [45]. The area was estimated, and the tree crown objects were identified when considering an area between 1 m^2 and 50 m^2.

In the GIS application, the fractal analysis was generated and included in a text file composed by the following metrics: area, perimeter, centroid of each tree, total number of trees, total area of tree crown, and total number of missing trees. The information was saved in table format with the following headers: *Id*, *Area*, *Perimeter*, *x* (centroid) and *y* (centroid).

The code that was implemented used the *write()* function to insert each parameter. Through functions from *QgsVectorLayer()* class, the *Id* attribute was added to the shapefile attribute table. Two *for* loops were created to count the number of trees, sum the total area of tree crown, and to extract the trees metrics. To obtain the distance between the trees in each row, it was necessary to estimate the rhumb value. Because the trees positions are not regular, the rhumb between each tree was calculated using the azimuth function from *QgsPoint()* class. It was considered the positive rhumb. This value was calculated considered all of the centroids. From the list obtained, the mode value of the rhumb was extracted. Then, two *for* loops were defined to calculate the rhumb of one point related with the other points to find the distances in the existent rows, with a tolerance of 10. The minimum distance, which corresponds to the distance between two consecutive points in the same row, was extracted and listed. These values allowed for detecting the missing trees. For that, the mean distance was obtained and, if a distance value was higher than the mean value, with a specific tolerance of 1.5, a missing tree was detected. Finally, the total number of trees, the total area, the mean distance between trees and the number of missing trees was recorded in the text file.

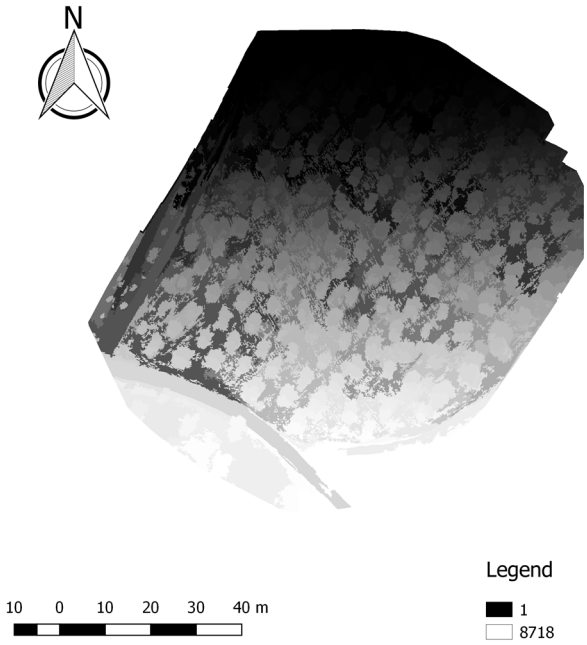
4. Results and Discussion

4.1. Experimental Procedure

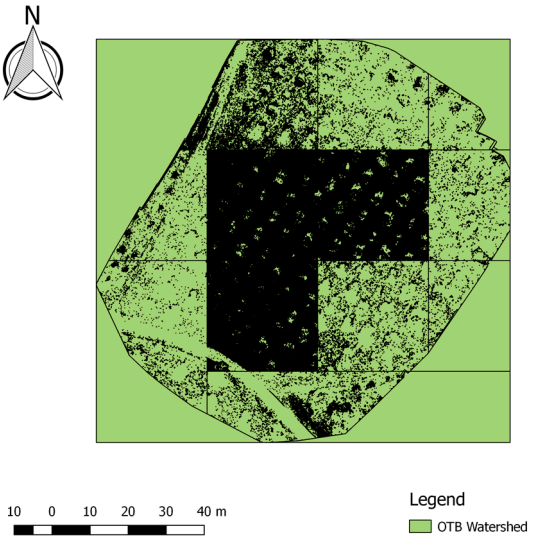
In order to define the methods to be implemented in the new plugin, some tests were performed in the segmentation and classification stages when considering the olive crop. This (Section 4) section presents the results obtained in these tests.

4.1.1. OBIA Segmentation

Figure 3 presents the segmentation result obtained with *i.segment* algorithm (Figure 3a) and considered the *Watershed algorithm* (Figure 3b), both performed when considering the default values.



(**a**)



(**b**)

Figure 3. (**a**) Segmentation with *i.segment* algorithm; (**b**) segmentation with *Watershed algorithm*.

Several tests were computed using different values of *thresholding* and *minsize*. Table 1 presents the results obtained regarding the tests that were performed using different values of *thresholding* (0.65 or 0.95) and *minsize* (10, 100, or 200) for segmentation process and different inputs (multispectral image, RGB image, or NDVI) for classification process with *i.segment* algorithm. Also, the algorithm can be applied with separated bands or joined bands. So, the tests presented considered these two cases.

Table 1 presents the results of several tests performed considered the multispectral image, NDVI image and RGB true colour image as inputs in unsupervised classification. For this study case, the results obtained considering the multispectral image and the RGB image presented worst accuracy when compared with NDVI image (as input). For this reason, the NDVI image was used as input.

The segmentation accuracy was evaluated based on "Goodness for fit" statistical model, as provided by *i.segment* algorithm, which allows for describing how well it fits a set of observations, summarizing the discrepancy between observed and expected values. In the GRASS library (*i.segment* algorithm), this parameter is based on the distance of the pixel to the object that it belongs to. A value of 1 means identical values, a perfect fit, and a value of 0 means maximum possible distance, worst possible fit. Given the lack of field data to validate the results obtained, this model was used. In order to estimate the percentage of pixels with specific values, *r.report* (algorithm from GRASS) was computed. The model was applied to the segmentation tests that were performed in Table 1. Based on the report that was returned it was concluded that: in case (a) all of the image pixels were above 0.96 and 47.33% of them were above 0.99; in case (b) all of the pixels were above 0.94 and 69.82% of the pixels were above 0.99; in the case (c) all the pixels were above 0.95 and 44.85% of them were above 0.99; in case (d) all the pixels were above 0.93 and 73.79% of them were above 0.99; in case (e) all the pixels were above 0.95 and 44.85% were; in case (f) all the pixels were above 0.97 and 64.94% of them were above 0.99. The values obtained proved that the segmentation using *i.segment* algorithm was satisfactory. Figure 4 presents an example of "Goodness for fit" map.

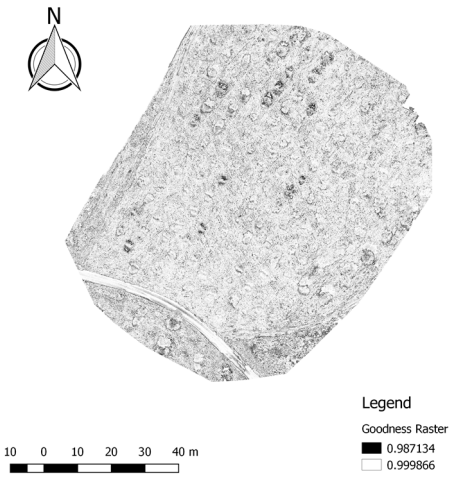


Figure 4. "Goodness for fit" map considering a *thresholding* of 0.65 and a *minsize* of 100, with separated bands.

The *i.segment* presented consistent results (Figure 3a), whereas the results that were obtained with the OTB algorithm were not satisfactory (Figure 3b). For this reason, the *i.segment* algorithm was chosen to be implemented in the plugin. The tests performed with different *thresholding* and *minsize* values allowed to define the more adequate methodology (Table 1). The *thresholding* parameter was defined as 0.65 as it presented a significant agglomeration of the pixels in the objects with smaller values. The *minsize* value was defined as 100 pixels as the results obtained were better with smaller values and very similar with higher values (Table 1).

Table 1. Tests performed for segmentation and classification processes considering different values of *thresholding* and *minsize* for *i.segment* algorithm, in the segmentation process and considering different inputs (multispectral image, Red-Green-Blue (RGB) image, or Normalized Difference Vegetation Index (NDVI)) in the classification process.

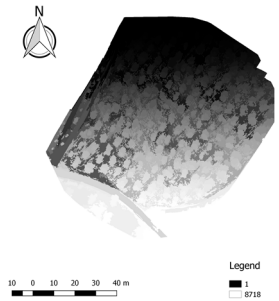
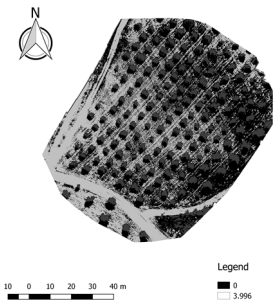
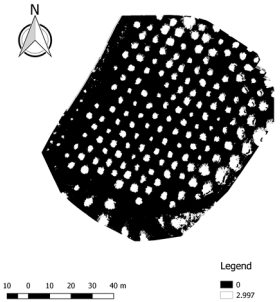
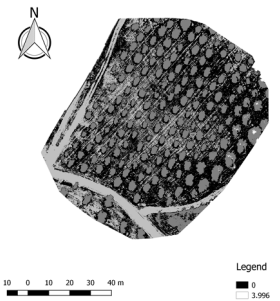
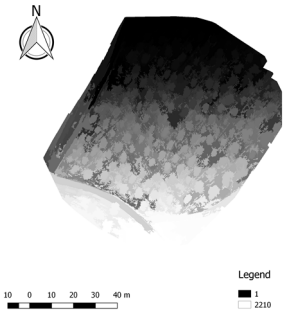
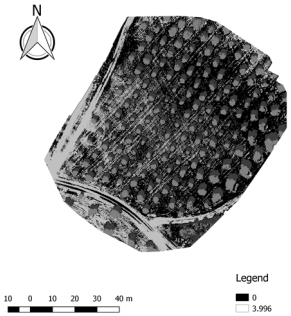
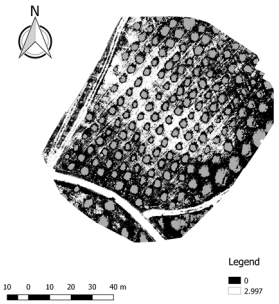
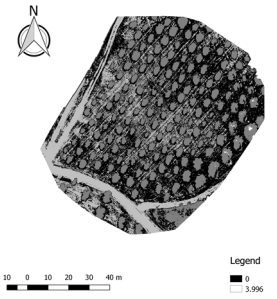
Segmentation	Unsupervised Classification with Segmented Image and Multispectral Image as Inputs	Unsupervised Classification with Segmented Image and NDVI Image as Inputs	Unsupervised Classification with Segmented Image and RGB Image as Inputs
			
(a) Segmentation: Thresholding = 0.65, minsize = 100, separated bands			
			
(b) Segmentation: Thresholding = 0.65, minsize = 100, joined bands			

Table 1. *Cont.*

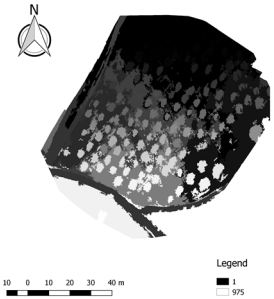
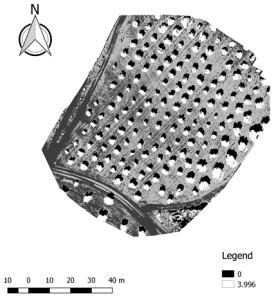
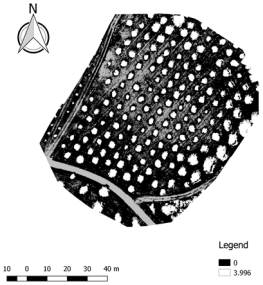
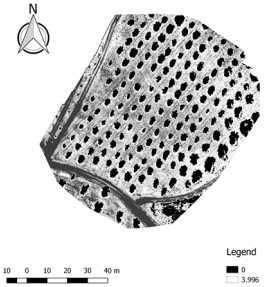
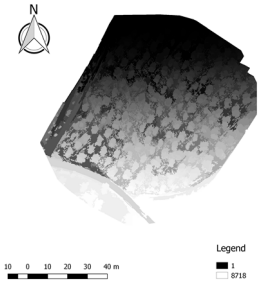
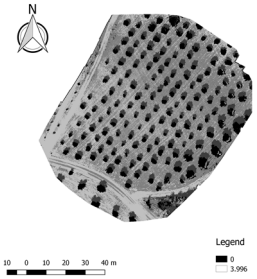
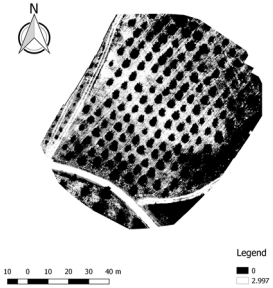
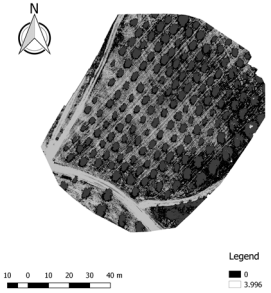
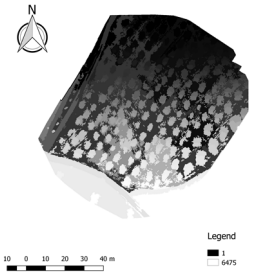
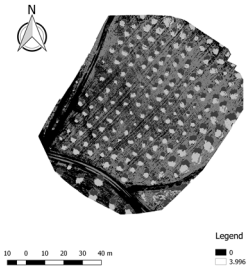
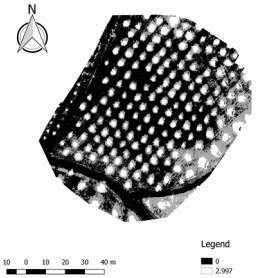
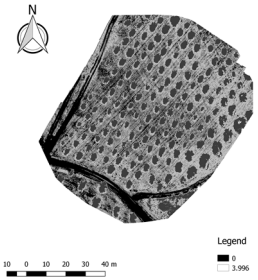
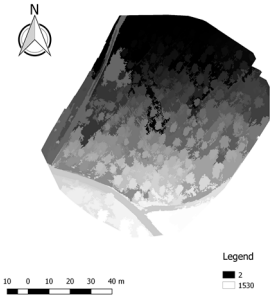
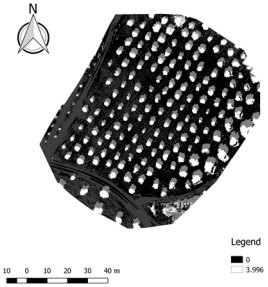
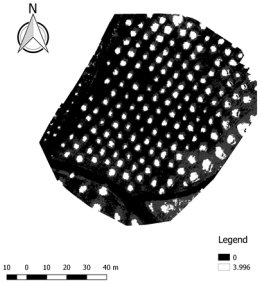
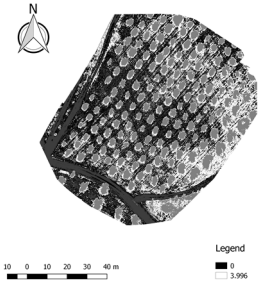
Segmentation	Unsupervised Classification with Segmented Image and Multispectral Image as Inputs	Unsupervised Classification with Segmented Image and NDVI Image as Inputs	Unsupervised Classification with Segmented Image and RGB Image as Inputs
			
(c) Segmentation: Thresholding = 0.95, minsize = 100, separated bands			
			
(d) Segmentation: Thresholding = 0.65, minsize = 10, joined bands			

Table 1. *Cont.*

Segmentation	Unsupervised Classification with Segmented Image and Multispectral Image as Inputs	Unsupervised Classification with Segmented Image and NDVI Image as Inputs	Unsupervised Classification with Segmented Image and RGB Image as Inputs
			
(e) Segmentation: Thresholding = 0.95, minsize = 10, separated bands			
			
(f) Segmentation: Thresholding = 0.65, minsize = 200, separated bands			

4.1.2. OBIA Classification

In this process, both classification, supervised and unsupervised, obtained similar results, as presented in Figure 5.

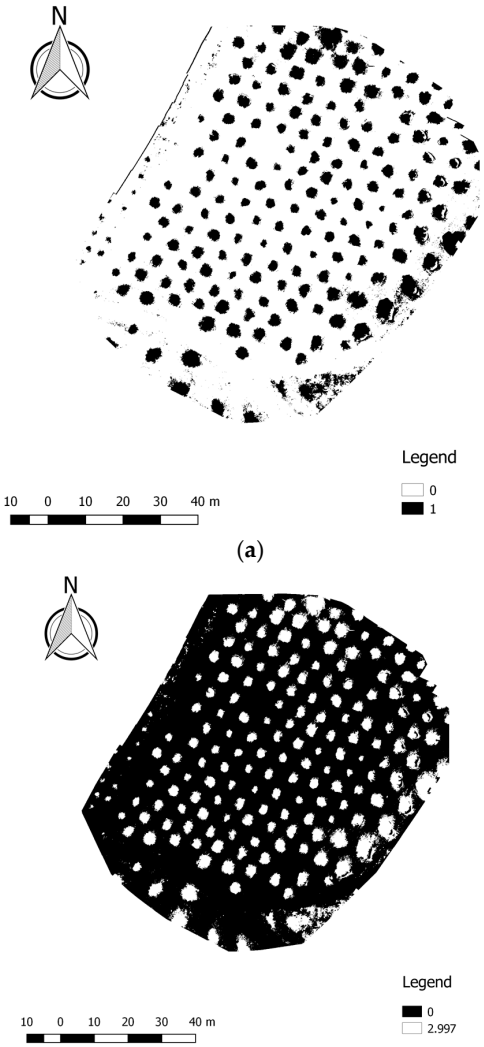


Figure 5. (**a**) Supervised classification result; and, (**b**) Unsupervised classification result, for olive crop.

The input defined in the supervised classification (Figure 5a) was the multispectral image, because the NDVI image did not presented valid results. Regarding the classification accuracy, in the supervised classification was obtained an overall accuracy of 92.84% and a kappa value of 0.71. While in the unsupervised classification the NDVI image presented results more robust (Figure 5b). Both classifications used the segmented image with separated bands as input. As both provided very similar results, the algorithm that was implemented in the plugin was the unsupervised classification because it is a procedure more objective when compared to the supervised classification, given the fact that the user doesn't need to define the training areas to classify.

The final result was obtained from a segmentation through *i.segment* algorithm using the multispectral image with separated bands, followed by an unsupervised classification through *Unsupervised kmeans image* classification when considering the NDVI image as input (Figure 5a).

4.1.3. Fractal Analysis

The classified image was then converted to binary image using *Raster Calculator*, resulting in an image with values 1 to tree crown and 0 to the remaining objects. After, this image was converted to shapefile (Figure 6).

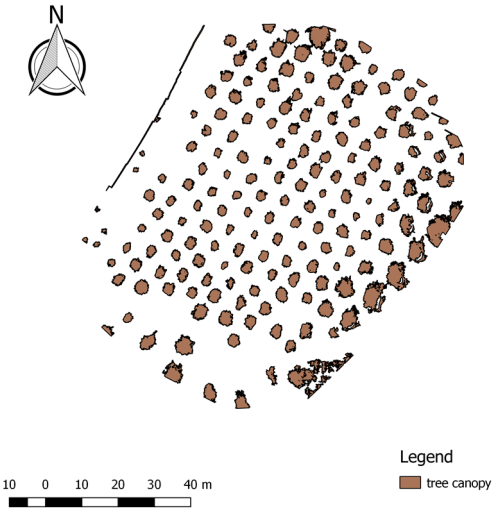


Figure 6. Shapefile with tree crown well identified.

In the shapefile showed in Figure 6, some objects were not correctly classified. Based on in situ knowledge, objects with an area different from (1–50) m^2 or objects with an atypical shape for the general tree crown should be removed. The remaining objects are considered as tree crown (Figure 5b).

4.2. GIS Open Source Application

To test the plugin created, the study case presented in the experimental procedure was considered. The result obtained was the shapefile with the tree crown and the text file with the metrics estimated. Figure 7a,b present the resulting shapefile from plugin and the shapefile obtained manually, respectively.

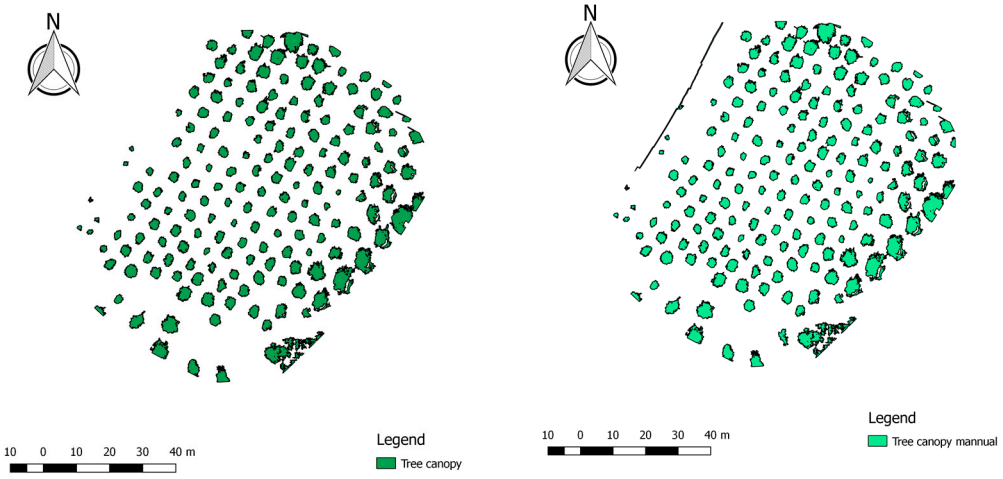


Figure 7. (**a**) Shapefile obtained with the plugin; and, (**b**) shapefile obtained with the manual process.

The code implemented allowed for removing the objects with a bounding box higher than the ratio defined. The metrics estimated presented a total count of 178 trees and a total area of 1401.3 m^2. The mean distance between the trees was 7.8 m and it were detected 23 missing trees.

In the plugin testing phase some limitations were founded. For instance, two trees were wrongly removed and some of them were not totally separated. The proximity between them was not detected by the algorithm (Figure 8).

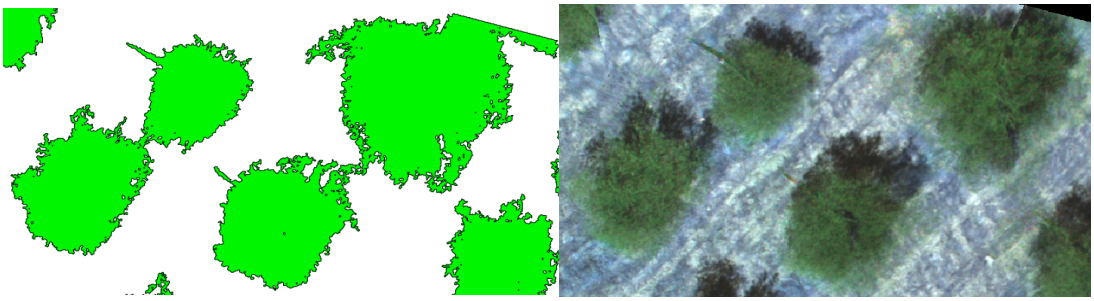


Figure 8. Incorrect objects identification (closer objects).

Also, some vegetation species with similar spectral characteristics with olives were wrongly identified as olive crop (Figure 9).

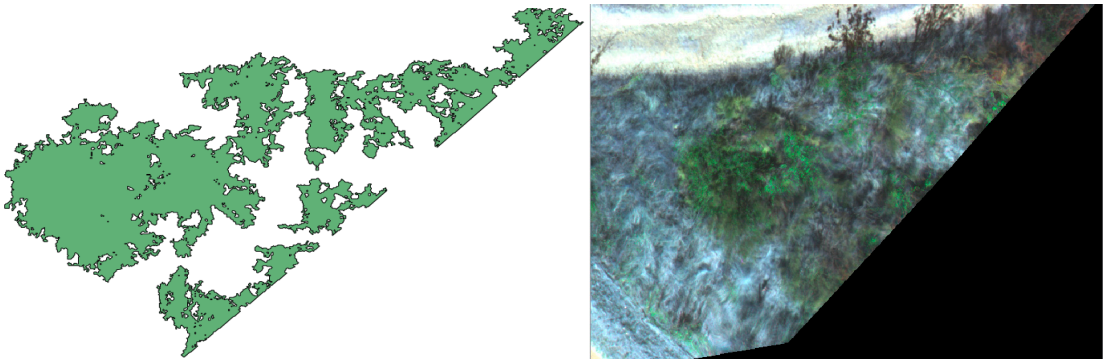


Figure 9. Incorrect objects identification (other type of vegetation).

The detected number of missing trees seems to be higher than the expected. This fact can be related with the roads that cross this area, so it was identified as a missing tree (Figure 10).

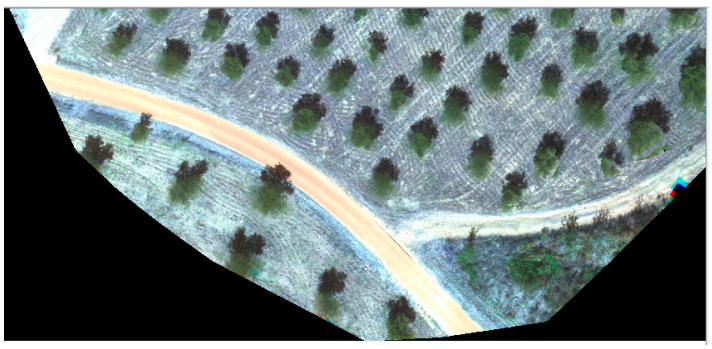


Figure 10. Representation of the road that cross the study area, which can influence the results.

Two different crops were also used to test the plugin, eucalyptus, and vineyards. In the case of eucalyptus, the resulted shapefile was not satisfactory, because most of the trees were not recognized and some shadows were identified as trees (Figure 11).

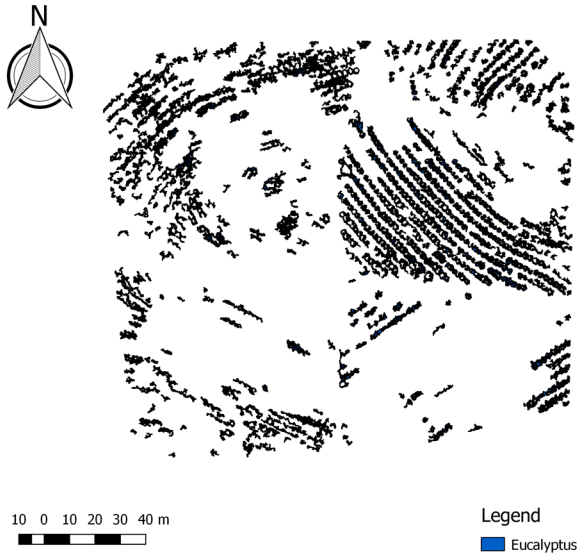


Figure 11. Eucalyptus crop plugin result.

In this study case, some polygons were not completely separated, because the eucalyptus trees are very close together. Also, the shade of the trees is different, so it was not possible to obtain a homogeneous colour maybe due to the light conditions. In this test, 558 eucalyptuses were detected and an area of 2632.1 m^2 was estimated. The mean distance obtained was 14.9 m and 101 eucalyptuses were not identified.

In vineyard crop study case, some species vegetation and other trees, different from vineyard, were identified. In this case, 254 trees were identified and 886.3 m^2 of area was obtained. The mean distance was 13.3 m and 41 trees were missing (Figure 12).

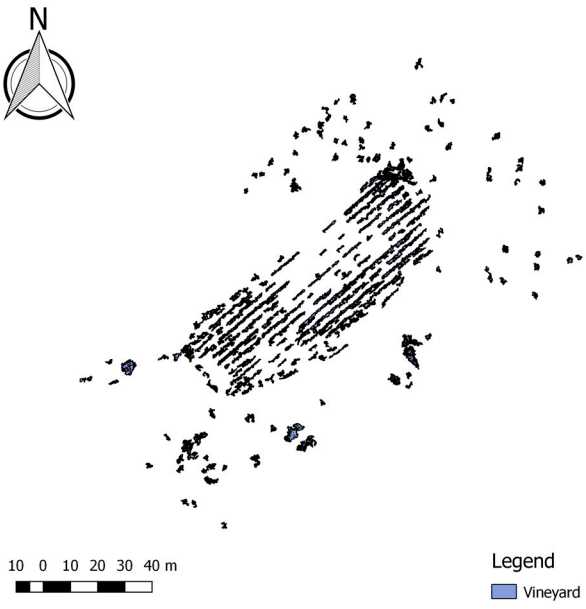


Figure 12. Vineyard crop result.

From the results obtained, testing the plugin with different plantations, some limitations were founded. These limitations could be overcome with the use of DSM, which can be generated from UAV imagery. Several studies proved that, with the DSM, it is possible to achieve a good performance in the extraction of individual tree metrics [46]. The plantation background is relevant to understand the accuracy of the results obtained with this plugin. As in olive crop some shrubs were wrongly identified as olives (Figure 9). In these cases, the DSM could help into distinguish tree crowns from the surrounding vegetation more reliably. In the future a new method should be implemented in this plugin integrating the DSM in the that is methodology proposed.

4.3. Processing Time

In order to compare the performance of the plugin developed, the processing time of the experimental tests were recorded (Table 2), for olive crop. These procedures were performed in the same environment: same computer, same CPU (Intel®, Santa Clara, CA, USA, Core™ i7-5500U CPU@ 2.40 GHz), same memory (8.00 GB) and same operating system (Windows 10 Home, Microsoft, Seattle, WA, USA). The objective of this analysis was to evaluate the performance of the plugin created comparing the processing time of it with the manual procedure.

Table 2. Processing time of manually procedure and plugin.

Procedure	Tool	Algorithm	Time (s)	
			Experimental Procedure	Plugin
OBIA segmentation	GRASS GIS 7	*i.segment* (default values)	1837	
	OTB	*Watershed segmentation*	319	
	GRASS GIS 7	*i.segment* (thresholding 0.05 until 0.65)	2017	
OBIA classification—supervised classification	OTB	*Compute second order statistics*	3	
	OTB	*TrainImageClassifier (svm)*	14	
	OTB	*Image Classification*	8	
OBIA classification—unsupervised classification	OTB	*Unsupervised kmeans image classification*	3	
Fractal Analysis	GDAL	*Raster Calculator*	8	
	GDAL	*Polygonize*	69	
TOTAL (with unsupervised classification)	Segmentation with *i.segment* (default values)		1917	2118
	Segmentation with *Watershed* algorithm		399	
	Segmentation with *thresholding*		2097	

Several conclusions can be drawn when considering the results that are presented in Table 2. The total processing time was calculated when considering three different segmentation methods. The *Watershed* algorithm took 6′39″. The segmentation with *i.segment*, defined with the default values, took 31′57″ and the segmentation with the thresholding takes 34′57″. The plugin processing time was also recorded and took 35′18″. From the manual procedures, the algorithm implemented in the plugin (with *thresholding* method) was the slowest method. However, the evaluation of the results that were obtained in this work proved that the *i.segment* with *thresholding* were the most satisfactory approach. The processing time was similar with the segmentation performed manually or with the plugin using the *i.segment* with *thresholding*. Even though the manual processing took less 21″ than the automatized processing implemented in the plugin, it must be considered that the manual processing was performed step by step and the plugin ran all of the algorithms automatically in one step. Furthermore, the fractal analysis including the estimation of the number of trees, the total area of the tree crown, the centroid of a tree, the perimeter and the total number of missing trees were implemented in the plugin automatizing this process. A method was defined where the missing trees are automatically detected, which improved the plugin capabilities, as this process can be very complex to perform without a specific script. When combining this with the high quality of the results obtained, this methodology proved to be the best one and was chosen to be implemented in the plugin.

The total processing time of the plugin for eucalyptus and vineyards resulted in 3 h 30′31′′ and 1 h 20′18′′, respectively. These results proved that the processing time depends on the crop used and the image size.

5. Conclusions

The automatic trees detection using a UAV image is a complex process. Different types of objects can be identified in the UAV images, so it is required a robust algorithm to detect the correct objects. UAV technology combined with GIS software proved to be efficient in the detection of tree crown, as it combines tools to visualize, manipulate, analyse and process geographic information data acquired by UAVs. The open source GIS software, as QGIS, provides the possibility of developing new plugins in order to automatize procedures and integrates several external algorithms, for instance, processing image algorithms. This functionality allowed for developig/creating a new plugin with the implementation of the more accurate methodology to obtain the parameters/metrics of tree crown. In this work, a new plugin was created automatizing the process of obtaining the parameters/metrics of tree crown. Different methodologies were tested and the more accurate was implemented in the plugin. To evaluate the performance of the plugin, the time processing of the set of procedures implemented was compared with the manual procedure. From the results that were obtained, it was concluded that the methodology implemented in the segmentation stage using *i.segment* with *thresholding* method presented the most satisfactory result in terms of processing time and accuracy. However, some limitations were founded in the code implemented, which were related to the precise identification of trees of a specific crop. Therefore, in the future, the plugin can be improved with a more efficient code and new methods. For instance, it can be improved when considering: (i) the correctly overlap of the bands; (ii) to use hyperspectral images instead of multispectral images, as the hyperspectral images presents continuous values allowing for a better distinction between the objects; (iii) to provide morphological operations, such as erosion to remove surrounding pixels and dilatation of the objects adding pixels to the surrounding areas; (iv) added supervised classification algorithms; and, (v) incorporation of DSM in the methodology. This plugin was created in the context of plantations, however the plugin presents a huge potential for other different areas such in small-crop delineations (cabbage fields, strawberries plantations, among others) or even non-environmental applications such as counting cars or buildings. The greater advantages of the developed plugin are the possibility of estimating the parameters with a simple and intuitive tool, not requiring the knowledge of the algorithms existent in QGIS and the possibility of the incorporation of algorithms and functions from others software. The first version of this tool is presented in this paper. The plugin is free, open source and available in http://www.fc.up.pt/pessoas/liaduarte/treecrown.rar.

Acknowledgments: The authors would like to thank Eye2Map Lda, Porto, Portugal, for the data provided.

Author Contributions: Ana Cláudia Teodoro conceived and designed the experiments; Pedro Silva and Lia Duarte performed the experiments; Ana Cláudia Teodoro and Lia Duarte analyzed the data; and Lia Duarte and Ana Cláudia Teodoro wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Santoro, F.; Tarantino, E.; Figorito, B.; Gualano, S.; D'Onghia, A.M. A tree counting algorithm for precision agriculture tasks. *Int. J. Digit. Earth* **2013**, *6*. [CrossRef]
2. Bazi, Y.; Malek, S.; Alajlan, N.; AlHichri, H. An Automatic Approach for Palm Tree Counting in UAV Images. Presented at the IEEE Geoscience and Remote Sensing Symposium IGARS, Quebec City, QC, Canada, 13–18 July 2014.
3. Pierrot-Deseilligny, M. Presentation, IGN/ENSG, France, Institut Geographique National. Available online: http://perso.telecomparistech.fr/~tupin/JTELE/PRES012009/Deseilligny.pdf (accessed on 10 March 2016).
4. Tian, J.Y.; Wang, L.; Li, X.J.; Gong, H.L.; Shi, C.; Zhong, R.F.; Liu, X.M. Comparison of UAV and WorldView-2 imagery for mapping leaf area index of mangrove forest. *Int. J. Appl. Earth Obs. Geoinform.* **2017**, *61*, 22–31. [CrossRef]

5. Senthilnath, J.; Kandukuri, M.; Dokania, A.; Ramesh, K.N. Application of UAV imaging platform for vegetation analysis based on spectral-spatial methods. *Comput. Electron. Agric.* **2017**, *140*, 8–24. [CrossRef]
6. Malehmir, A.; Dynesius, L.; Paulusson, K.; Paulusson, A.; Johansson, H.; Bastani, M.; Wedmark, M.; Marsden, P. The potential of rotary-wing UAV-based magnetic surveys for mineral exploration: A case study from central Sweden. *Lead. Edge* **2017**, *36*, 552–557. [CrossRef]
7. Suh, J.; Choi, Y. Mapping hazardous mining-induced sinkhole subsidence using unmanned aerial vehicle (drone) photogrammetry. *Environ. Earth Sci.* **2017**, *76*. [CrossRef]
8. McKenna, P.; Erskine, P.D.; Lechner, A.M.; Phinn, S. Measuring fire severity using UAV imagery in semi-arid central Queensland, Australia. *Int. J. Remote Sens.* **2017**, *38*, 4244–4264. [CrossRef]
9. Kang, J.; Wang, L.; Chen, F.; Niu, Z. Identifying tree canopy areas in undulating eucalyptus plantations using JSEG multi-scale segmentation and unmanned aerial vehicle near-infrared imagery. *Int. J. Remote Sens.* **2017**, *38*, 2296–2312.
10. Park, T.; Cho, J.; Lee, J.; Lee, W.; Choi, S.; Kwak, D.; Kim, M. Unconstrained approach for isolating individual trees using high-resolution aerial imagery. *Int. J. Remote Sens.* **2014**, *35*, 89–114. [CrossRef]
11. Hassaan, O.; Nasir, A.K.; Roth, H.; Khan, M.F. Precision Forestry: Trees Counting in Urban Areas Using Visible Imagery based on an Unmanned Aerial Vehicle. *IFAC-PapersOnLine* **2016**, *49–16*, 16–21. [CrossRef]
12. Nevalainen, O.; Honkavaara, E.; Tuominen, S.; Viljanen, N.; Hakala, T.; Yu, X.; Hyyppa, J.; Saari, H.; Polonen, I.; Imai, N.N. Individual Tree Detection and Classification with UAV-Based Photogrammetric Point Clouds and Hyperspectral Imaging. *Remote Sens.* **2017**, *9*, 185. [CrossRef]
13. Özcan, A.H.; Hisar, D.; Sayar, Y.; Ünsalan, C. Tree canopy detection and delineation in satellite images using probabilistic voting. *Remote Sens. Lett.* **2017**, *8*, 761–770. [CrossRef]
14. Koukoulas, S.; Blacknurn, G.A. Mapping individual tree location, height and species in broadleaved deciduous forest using airborne LIDAR and multi-spectral remotely sensed data. *Int. J. Remote Sens.* **2007**, *26*, 431–455. [CrossRef]
15. Katoh, M.; Gougeon, F.A. Improving the Precision of Tree Counting by Combining Tree Detection with Canopy Delineation and Classification on Homogeneity Guided Smoothed High Resolution (50 cm) Multispectral Airborne Digital Data. *Remote Sens.* **2012**, *4*, 1411–1424. [CrossRef]
16. Tanhuanpaa, T.; Saarinen, N.; Kankare, V.; Nurminen, K.; Vastaranta, M.; Honkavaara, E.; Karjalainen, M.; Yu, X.; Holopainen, M.; Hyyppa, J. Evaluating the Performance of High-Altitude Aerial Image-Based Digital Surface Models in Detecting Individual Tree Canopys in Mature Boreal Forests. *Forests* **2016**, *7*, 143. [CrossRef]
17. Thiel, C.; Schmullius, C. Comparison of UAV photograph-based and airborne lidar-based point clouds over forest from a forestry application perspective. *Int. J. Remote Sens.* **2017**, *38*, 2411–2426. [CrossRef]
18. Wei, T.; Lin, Y.; Yan, L.; Zhang, L. Tree species classification based on stem-related feature parameters derived from static terrestrial laser scanning data. *Int. J. Remote Sens.* **2016**, *37*, 4420–4440. [CrossRef]
19. Yin, D.; Wang, L. How to assess the accuracy of the individual tree-based forest inventory derived from remotely sensed data: A review. *Int. J. Remote Sens.* **2016**, *37*, 4521–4553. [CrossRef]
20. Rosell, J.R.; Sanz, R. A Review of Methods and Applications of the Geometric Characterization of Tree Crops in Agricultural Activities. *Comput. Electron. Agric.* **2012**, *81*, 124–141. [CrossRef]
21. Jiang, H.; Chen, S.; Li, D.; Wang, C.; Yang, J. Papaya Tree Detection with UAV Iamges Using a GPU-Accelerated Scale-Space Filtering Method. *Remote Sens.* **2017**, *9*, 721. [CrossRef]
22. Guerra-Hernández, J.; González-Ferreiro, E.; Monleón, V.J.; Faias, S.P.; Tomé, M.; Díaz-Varela, R.A. Use of Multi-Temporal UAV-Derived Imagery for Estimating Individual Tree Growth in Pinus pinea Stands. *Forests* **2017**, *8*, 300. [CrossRef]
23. Deng, G.; Li, Z.; Wu, H.; Zhang, X. Automated Extracting Tree Canopy from Quickbird Stand Image. In Proceedings of the 4th Conference on Computer and Computing Technologies in Agriculture (CCTA), Nanchang, China, 22–25 October 2010; IFIP Advances in Information and Communication Technology, AICT-344 (Part I); Springer: Berlin/Heidelberg, Germany, 2010; pp. 304–311.
24. Chianucci, F.; Disperati, L.; Guzzi, D.; Bianchini, D.; Nardino, V.; Lastri, C.; Rindinella, A.; Corona, P. Estimation of canopy attributes in beech forests using true colourdigital images from a small fixed-wing UAV. *Int. J. Appl. Earth Obs. Geoinform.* **2016**, *47*, 60–68. [CrossRef]
25. Panagiotidis, D.; Abdollahnejad, A.; Surový, P.; Chiteculo, V. Determining tree height and canopy diameter from high-resolution UAV imagery. *Int. J. Remote Sens.* **2017**, *38*, 2392–2410.

26. Blaschke, T. Object based image analysis for remote sensing. *ISPRS J. Photogr. Remote Sens.* **2010**, *65*, 2–16. [CrossRef]
27. Cheng, G.; Han, J. A survey on object detection in optical remote sensing images. *ISPRS J. Photogr. Remote Sens.* **2016**, *117*, 11–28. [CrossRef]
28. Teodoro, A.C.; Araújo, R. A Comparison of Performance of OBIA Techniques Available in Open Source Software (Spring and OTB/Monteverdi) considering Very High Spatial Resolution Data. *J. Appl. Remote Sens.* **2016**, *10*, 016011. [CrossRef]
29. Shahzad, N.; Ahmad, S.R.; Ashraf, S. An assessment of pan-sharpening algorithms for mapping mangrove ecosystems: A hybrid approach. *Int. J. Remote Sens.* **2017**, *38*, 1579–1599. [CrossRef]
30. GRASS GIS. 2017. The World's Leading Free GIS Software. Available online: http://grass.osgeo.org/ (accessed on 10 March 2017).
31. Orfeu Toolbox (OTB). 2017. Orfeo Toolbox Is Not a Black Box. Available online: https://www.orfeo-toolbox.org/ (accessed on 10 March 2017).
32. GNU Operating System. 2017. Stallman Four Freedoms. Available online: https://www.gnu.org/philosophy/free-sw.html (accessed on 12 March 2017).
33. QGIS. 2017. QGIS Project. Available online: http://www.qgis.org/ (accessed on 10 March 2017).
34. Grippa, T.; Lennert, M.; Beaumont, B.; Vanhuysse, S.; Stephenne, N.; Wolff, E. An Open-Source Semi-Automated Processing Chain for Urban Object-Based Classification. *Remote Sens.* **2017**, *9*, 358. [CrossRef]
35. Duarte, L.; Teodoro, A.C.; Moutinho, O.; Gonçalves, J.A. Open-source GIS application for UAV photogrammetry based on MicMac. *Int. J. Remote Sens.* **2016**, *38*, 8–10. [CrossRef]
36. Reed, B.C.; Brown, J.F.; VanderZee, D.; Loveland, T.R.; Merchant, J.W.; Ohlen, D.O. Measuring phenological variability from satellite imagery. *J. Veg. Sci.* **1994**, *5*, 703–714. [CrossRef]
37. Python. 2017. Python Programming Language. Available online: http://python.org/ (accessed on 9 March 2016).
38. QGIS API. 2017. QGIS API Documentation. Available online: http://www.qgis.org/api/ (accessed on 10 March 2017).
39. PyQt4 API. 2017. PyQt Class Reference. Available online: http://pyqt.sourceforge.net/Docs/PyQt4/classes.html (accessed on 10 March 2016).
40. Qt Designer. 2017. Qt Documentation. Available online: http://doc.qt.io/qt-4.8/designer-manual.html (accessed on 10 March 2017).
41. Momsen, E.; Metz, M. Grass Osgeo Manual i.segment. Available online: https://grass.osgeo.org/grass73/manuals/i.segment.html (accessed on 5 December 2016).
42. Awf-Wik. 2017. Forest Inventory Remote Sensing. Supervised Classification (Tutorial). Available online: http://wiki.awf.forst.uni-goettingen.de/wiki/index.php/Supervised_classification (accessed on 12 March 2017).
43. Awf-Wik. 2017. Forest Inventory Remote Sensing. Unsupervised Classification (Tutorial). Available online: http://wiki.awf.forst.uni-goettingen.de/wiki/index.php/Unsupervised_classification (accessed on 12 March 2017).
44. Klinger, Riccardo. Digital Geography, Unsupervised Classification in QGIS: Kmeans or Part Two. Available online: http://www.digital-geography.com/unsupervised-classification-in-qgis-kmeans-or-part-two/#.WT5ixbpFzIV (accessed on 13 August 2013).
45. GDAL/OGR. 2017. Geospatial Data Abstraction Library. Available online: http://www.gdal.org/ (accessed on 10 March 2017).
46. Adão, T.; Peres, E.; Pádua, L.; Hruska, J.; Sousa, J.J.; Morais, R. UAS-based hyperspectral sensing methodology for continuous monitoring and early detection of vineyard anomalies. In Proceedings of the UAS4ENVIRO—Small Unmanned Aerial Systems for Environmental Reasearch—5th Edition, Vila Real, Portugal, 28–30 June 2017.