A FRAMEWORK TO CALCULATE WINE PRODUCTION BASED ON HIGH RESOLUTION IMAGES, A CASE STUDY IN RIBERA DEL DUERO

By

Jorge Moyano

A DISSERTATION

Submitted to

The University of Liverpool

in partial fulfilment of the requirements

for the degree of

MASTER OF SCIENCE

01/06/2018

ABSTRACT

A FRAMEWORK TO CALCULATE WINE PRODUCTION BASED ON HIGH RESOLUTION IMAGES, A CASE STUDY IN RIBERA DEL DUERO

By

Jorge Moyano

This study aimed to provide an alternative mean to calculate yield prediction in a vineyard using high resolution images. The motivation for this study is the idea that there is a probability that if a grape cluster is surrounded by leaves, there is a high chance that part of this grape cluster might be occluded by these leaves, thus this parameter can be introduced in the calculation to improve the estimations. This study leveraged a pre-existing grape cluster image recognition algorithm developed by Smartrural (a start-up company located in Valladolid, Spain), and introduced the newly calculated parameter called *occlusion factor*. This parameter was calculated by inspecting the pixels surrounding the already recognized clusters and determining if they belong to a leaf or not.

This paper presents also an architecture for image processing and a leaf filtering algorithm based on image transformations and the histogram back projection algorithm. The mentioned architecture was designed to apply different configurable image filters and transformations in a sequential fashion to the set of available pictures, prior to extract and inspect the target pixels. The previously mentioned occlusion factor was calculated by dividing the number of pixels identified as leaves by the total number of pixels on the vicinity of the previously identified grape clusters.

Although it appears that the algorithm performed better by introducing the occlusion factor parameter, the dispersion of the results and the errors introduced by the lack of reliable measured weight data used for this study, prevent us from accepting the premise that initiated this study. However, it is expected that the present study serves as a foundation for future studies and enables new vineyard management techniques such as selective trimming or harvesting planning.

DECLARATION

I hereby certify that this dissertation constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions, or writings of another.

I declare that the dissertation describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

Jorge Moyano

“This dissertation contains material that is confidential and/or commercially sensitive. It is included here on the understanding that this will not be revealed to any person not involved in the assessment process.”

**Student, Supervisors and Classes:**

|  |  |
| --- | --- |
| Student name: | Jorge Moyano |
| Student ID number: | H00041124 |
| GDI name: | Taly Sharon |
| CRMT class ID: | LAUR-906-1 |
| DA name: | Taly Sharon |
| CAC class ID: | CKIT-702-2 |

ACKNOWLEDGEMENTS

First of all, I want to thank all the people that helped me to gather the necessary data that this dissertation required, special thanks to the people of Bodegas Valdemonjas and Smartrural, which were always willing to help me with any data requirement and explanation I needed for this paper.

I want to thank my partner Elena Medina as well as my parents for supporting me through the whole master’s process and who’s infinite patience deserve a big recognition here.

I would also like to express my gratitude to my advisor Taly Sharon, who helped me to shape and define the proposal and content of this dissertation.

**TABLE OF CONTENTS**

Page

[LIST OF TABLES 3](#_Toc512160938)

[LIST OF FIGURES 4](#_Toc512160939)

[Chapter 1. Introduction 5](#_Toc512160940)

[1.1 Motivation 5](#_Toc512160941)

[1.2 Approach and methodology 6](#_Toc512160942)

[1.3 Outcome 7](#_Toc512160943)

[1.4 Document Structure 7](#_Toc512160944)

[Chapter 2. Background and review of Literature 8](#_Toc512160945)

[2.1 Background 8](#_Toc512160946)

[2.2 Literature Review 9](#_Toc512160947)

[2.2.1 Fruit Detection 9](#_Toc512160948)

[2.2.2 Yield Estimation 12](#_Toc512160949)

[2.3 Theory 13](#_Toc512160950)

[2.4 Terms 14](#_Toc512160951)

[Chapter 3. Analysis and Design 15](#_Toc512160952)

[3.1 Datasets 15](#_Toc512160953)

[3.2 Design of the solution 17](#_Toc512160954)

[3.2.1 Creating the cluster surrounding masks 18](#_Toc512160955)

[3.2.2 Image transformations 20](#_Toc512160956)

[3.3 Alternative Solutions 25](#_Toc512160957)

[Chapter 4. Implementation (Realization) 26](#_Toc512160958)

[4.1 Design Approach 26](#_Toc512160959)

[4.1.1 Overall Design 26](#_Toc512160960)

[4.1.2 Application Classes 29](#_Toc512160961)

[4.1.3 Dependency Injection Module 30](#_Toc512160962)

[4.1.4 Runnables 32](#_Toc512160963)

[4.1.5 Application Services 33](#_Toc512160964)

[4.1.6 Image transformers 36](#_Toc512160965)

[4.1.7 Models 37](#_Toc512160966)

[4.2 Application scripts 38](#_Toc512160967)

[4.3 Testing 38](#_Toc512160968)

[4.4 Tools and Frameworks 39](#_Toc512160969)

[4.4.1 OS and platform 39](#_Toc512160970)

[4.4.2 Open CV 40](#_Toc512160971)

[4.4.3 scikit-learn 40](#_Toc512160972)

[4.4.4 Intellij IDEA 40](#_Toc512160973)

[4.4.5 SBT 41](#_Toc512160974)

[Chapter 5. Results and Evaluation 42](#_Toc512160975)

[5.1 Data sanitization & selection 42](#_Toc512160976)

[5.2 Evaluation 43](#_Toc512160977)

[5.2.1 Coefficient of determination 44](#_Toc512160978)

[5.2.2 Mean Squared Error 45](#_Toc512160979)

[5.2.3 Median Absolute Deviation 45](#_Toc512160980)

[5.3 Results 46](#_Toc512160981)

[Chapter 6. Conclusions 48](#_Toc512160982)

[6.1 Lessons Learned 48](#_Toc512160983)

[6.2 Academic Applications 49](#_Toc512160984)

[6.3 Business Application 50](#_Toc512160985)

[6.4 Recommendations / Prospects for Future Research 50](#_Toc512160986)

[REFERENCES CITED 52](#_Toc512160987)

[APPENDICES 57](#_Toc512160988)

LIST OF TABLES

[Table 1: Inference information 16](#_Toc512157044)

[Table 2: json cluster recognition extract 16](#_Toc512157045)

[Table 3: Runnable Interface 29](#_Toc512157046)

[Table 4: Image Reconstruction application Class 30](#_Toc512157047)

[Table 5: Dependency Injection Module 31](#_Toc512157048)

[Table 6: Service Injection Snippet 31](#_Toc512157049)

[Table 7: Image Reconstruction Runner 33](#_Toc512157050)

[Table 8: Bounding Box Service 33](#_Toc512157051)

[Table 9: File Manager Service 35](#_Toc512157052)

[Table 10: Image Reconstruction Service 35](#_Toc512157053)

[Table 11: Inference Service 36](#_Toc512157054)

[Table 12: Image Transformer trait 36](#_Toc512157055)

[Table 13: Median Filter Transformer 36](#_Toc512157056)

[Table 14: Bounding Box Model Class 38](#_Toc512157057)

[Table 15: Result Joiner script 38](#_Toc512157058)

[Table 16: Unit Test Example 39](#_Toc512157059)

[Table 17: Prediction script 44](#_Toc512157060)

[Table 18: Estimation using Pixel count 46](#_Toc512157061)

[Table 19: Estimation using Pixel count and occlusion factor 46](#_Toc512157062)

LIST OF FIGURES

Page

[Figure 1:Raw Image 15](#_Toc512157063)

[Figure 2:Inference pictures 16](#_Toc512157064)

[Figure 3: Cluster recognition output image 17](#_Toc512157065)

[Figure 4: Inference reconstruction and resulting mask 18](#_Toc512157066)

[Figure 5: Inference reconstruction flowchart 19](#_Toc512157067)

[Figure 6: Image processing flowchart 21](#_Toc512157068)

[Figure 7: Histogram target image 23](#_Toc512157069)

[Figure 8: Histogram backprojection result image 24](#_Toc512157070)

[Figure 9: Image Reconstruction UML Design 27](#_Toc512157071)

[Figure 10: Image Transform UML Design 28](#_Toc512157072)

[Figure 11: Models UML Design 29](#_Toc512157073)

[Figure 12: Intellij IDEA development environment 41](#_Toc512157074)

[Figure 13: Tileo visualization tool 42](#_Toc512157075)

[Figure 14: Picture matching the highest occlusion value found 49](#_Toc512157076)

[Figure 15: Picture matching the lowest occlusion value found 51](#_Toc512157077)

# Introduction

Motivation

One of the goals of precision viticulture is to improve the vineyard management processes in order to improve performance by introducing techniques and technology that helps the farmer in the decision taking. Smartrural is a company specialized in precision agriculture, and they aim to create a yield estimation algorithm to predict the total weight of the vineyards harvest from pictures taken with autonomous vehicles and their grape cluster recognition algorithm.

Calculating yield on a vineyard was traditionally performed by costly, destructive and inaccurate techniques, as samples of the bunches had to be collected and measured and the result of the calculations were extrapolated to the rest of the plot (Nuske *et al.*, 2011).

Nowadays, there is a plethora of techniques developed that uses autonomous vehicles and image recognition to predict the yield of a vineyard: volumetric estimations, grape counting, pixel counting, cluster recognition techniques, etc… From these techniques, pixel counting is one of the most accurate ones (Liu, 2013), this work aims to improve the accuracy of this method by introducing a new occlusion factor in the calculations as suggested by Nuske *et al.*, (2011), which would be estimated from the pixels that surrounds the identified grape clusters.

While Rey-Camarés *et al.* (2015) found that the high leaf area values does not always correlated with increased yield output (yield in grapevines is also related to the exposed lead area, reaches a point at which it cannot continue increasing, so the leaves that are not exposed may become a sink for the nutrients and energy of the plant competing with the clusters for these resources), the presence of a higher number of leaves and its size should be related with a higher probability for a cluster to be hidden by a leaf.  The scope of this project is to develop a technique to identify the cluster surrounding leaf percentage in order to feed a yield estimation algorithm with a variable representing the occlusion factor in order to improve the accuracy of the weight prediction. The project aim to provide a solution to prove the validity for the hypothesis that originated this dissertation, namely:

H1. That it is possible to calculate and predict the grape harvesting volume using data mining over the sensors gathered data and processed high resolution pictures.

H2. That is possible to improve the vineyard management by using machine learning algorithms created to process image data.

H3. That is possible to provide new inputs for visual model representation of the agriculture data using the information produced by the algorithm.

Approach and methodology

The research methods and image processing techniques used are based on the ones found while doing the exploratory research to conduct the literature review. A leaf detection algorithm was created to count the number of leaf pixels that appears surrounding the recognized clusters on the images. The proposed solution generates an occlusion factor by extracting the percentage of green pixels surrounding the cluster in a configurable radius (measured in number of pixels). Both variables grape pixel count and occlusion factor were used as independent variables on a multiple linear regression to calculate the expected throughput of the vineyard.

A prototype of the algorithm used to detect leaves was originally created based on the one proposed by Berenstein *et al.*, (2010) but it was found that the filtering based strictly on the RGB values of the image, yields poor results and appropriate boundaries for leaf detection are difficult to configure. A more performant solution was established following similar methods to the ones cited by Aquino *et al.* (2017) who used colour histograms, RGB filtering and fuzzy clustering to detect grape pixels. A similar approach is described by Fernandez *et al.* (2013) who also used RGB filtering to detect white grapes. As per validation, the datasets were divided 2/3 to build the model and 1/3 to test it, a common approach taken by researchers like Ivorra *et al.*, (2015) or Berenstein *et al.*, (2010).

The actual yield prediction uses linear regression, a common practice used by other researchers like Lopes *et al.* (2016), Nuske *et al.* (2011), Serrano *et al.* (2005) or Herrero Huerta (2015).

Outcome

Along with the report presented here, a framework designed to process images to detect the presence of leaves surrounding the previously identified grape clusters has been created. Using the information yield by Smartrural’s grape detection algorithm (consisting of a set of two colour images containing the identified clusters as well as files containing information about their position in the original picture), the framework is capable of identifying the percentage of the grape cluster that is surrounded by leaves and use this information to predict yield in a vineyard.

Document Structure

Chapter 2 starts with a review of the state of art in the fruit recognition systems, focused primarily in grape recognition techniques presented in the literature review. Chapter 3 presents the motivations, available datasets and design principles behind the design of the IT artefact develop to predict grape yield. A more detailed view on this artefact, architecture considerations and specifics on the implementation (as well as testing carried out) is given in Chapter 4. Chapter 5 focus on the results and evaluation criteria of the IT artefact and gives a comprehensive overview on the validity of the presented solution. Finally, Chapter 6 presents an outcome of this study, and presents recommendations to possible lines of future investigation on this field.

# Background and review of Literature

Background

There are three main components which can explain weight variance as noted by Clingeleffer *et al.* (2001): variance in number of clusters per vine (60%), variance in number of berries per cluster (30%) and variance in berry size (10%). Several studies had attempt to come up with a reliable yield estimator considering the morphological characteristics of the clusters: Ivorra *et al.*, (2014), mack *et al.*, (2017) or Herrero-Huerta (2014) to name a few. While these studies provide an accurate method to predict vine yield, the setup needed to deploy the required hardware makes it less ideal for field conditions.

Smartrural S.A. has developed a method to capture vineyard images using autonomous vehicles and mounted high-resolution cameras. By eliminating duplication and overlapping images, their cluster recognition algorithm provides reliable information about the number of clusters and their size tied to their geographical information. The aim of the current work is to provide and algorithm to estimate total yield throughput by using the output of Smartrural’s recognition algorithm as well as the information extracted from inspecting the images taken, to introduce a factor describing how likely the cluster is to be occluded.

Literature Review

Precision agriculture is not a new area of research, neither the use of image recognition and sensor data as input parameters. There is an extensive literature about fruit yield estimation written for several types of crops. We can find studies that demonstrate the feasibility of using image analysis and machine learning techniques to predict yield throughput for melons (Zhao *et al.*, 2017), orchard apples (Bargoti and Underwood, 2017), potatoes (Akhand *et al.*, 2016) or mango (Payne *et al.*, 2013). Furthermore, a more general approach was taken by Pothen and Nuske (2016), who generalized an algorithm to detect round fruits leveraging the gradual variation of intensity and gradient orientation on the surface of the fruit on high resolution images, or the research conducted by Shakoor *et al.* (2017) who tested several techniques to predict yield on different agricultural crops (Aus, Aman, Boro, Wheat, Jute and Potato) using multiple variables such as crop’s yield per hectare (M.Ton), average of minimum and maximum temperature, rainfall, year range, and location.

In precision viticulture, there are two main categories of related work as noted by Debadeepta *et al.* (2012):

* Fruit detection, including canopy and foliage reconstruction
* Yield estimation

Although as noted by Whalley and Shanmuganathan (2013) other applications such as quality estimation, disease detection or grape phenology identification are equally important.

### Fruit Detection

Several studies attempted to tackle the fruit recognition problem, Chamelat *et al.* (2006) applied Zernike and Support Vector Machine (SVM), Aquino *et al.* (2017) used images coming from a simple smartphone and a cardboard to increase the background contrast to feed an algorithm that were able to detect and count berry candidates by morphological filtering (it detected grape centers by using the camera flash reflection) and calculate berry descriptors to model the grapes and detect false positives on the algorithm by the use of a neural network first, and a Support Vector Machine later. Previous work of Aquino *et al.* (2015) used mathematical morphology and pyramidal decomposition transformations to identify flower in grapevine in inflorescences and a single variety-independent linear model to predict flower count.  Pérez *et al.* (2017) used a Scale Invariant Feature Transform (SIFT) to extract features to build a descriptor to feed a SVM to detect grapevine buds.

The recognition algorithms can have other applications such as plant disease detection, which was explored by Arivazhagan *et al.* (2013) who developed a framework to detect and classify several plant diseases in four steps: color transformation on an RGB image, filtering of uninteresting pixels (the healthy ones), and then a segmentation process using texture statistics extracted from with a Colour-Co-Occurrence method to finally classify these features using a neural network. This framework utilized contextual picture information such as camera trajectories, vehicle position, fruit type being scanned, distance from camera to the tree, sun relative position to determine the illumination incident angle to train the fruit recognition algorithm. Pujari *et al.* (2015) developed an image processing framework for early detection of fungal disease symptoms on different plant crops. In their paper, they use several methods (K-means clustering, feature selection, nearest neighbour, SVM, Mahalnobis distance and others) to identify diseases in fruit, vegetable, cereals and commercial crops. Singh and Misra (2017) created an algorithm for image segmentation and classification to be used for automatic detection and identification of plant leaf diseases using K-Mean clustering

Several attempts to create 3D cluster models has been also undertaken to explain volume and therefore weight and yield. Herrero-Huerta *et al.* (2015) used close-range photogrammetry to reconstruct grape clusters in 3D to estimate features and determine productivity parameters of the vineyard such as volume, number of berries and mass of the bunch using linear regression. Ivorra *et al.* (2015) also constructed a 3D model of the grape cluster using stereo vision to predict grape yield with feature descriptors to predict compactness and other morphological components that affects yield using also SVM. Mack *et al.* (2017) created a phenotyping classification pipeline using 3D reconstruction to categorize grape bunches using SVM algorithms. Debadeepta *et al.* (2012) leveraged also 3D modelling combined with colour detection and spatial smoothing of results using a conditional random field (CRF) and SVM to create a framework to classify points in a grapevine picture and classify them into berries, foliage or branches.

Berenstein *et al.* (2010) used an autonomous vehicle and machine learning algorithms (decision trees, pixel comparison and shape comparison) to differentiate leaves and grape clusters in order to selectively spray hormones and pesticides on the vines. Cubero *et al.* (2015) used morphological analysis of RGB images for feature extraction and applied a bayesian classifier which fed a predictive partial least squares (PLS) model to estimate grape bunch compactness in several grape varieties.

Fernández *et al.* (2013) used an autonomous vehicle equipped with a camera that captured spectral samples in the visible and the near-infrared range and the result pixels were classified using K-means clustering to differentiate different parts of the plant such as leaves, stems, branches, fruits and background. Other experiments using autonomous vehicles in the vineyard for image collection to classify phenotyping features for early detection of plant diseases were carried by Kicherer *et al.* (2015). A different autonomous vehicle (a drone) equipped with a multispectral camera was used by Rey-Camarés et al. (2015) to calculate the vegetation spectral indices to study their relationship with the spatial distribution and their relationships with grapevine vegetative, yield, and berry composition. Liu and Whitty (2015) presented a method to detect bunch areas in images using a limited feature space and a K-nearest neighbour classification algorithm. Murillo-Bracamontes *et al.* (2012) created an automatic vision system to detect grape clusters using the Hough transform and histogram equalization to enhance the image features (although an orange cardboard was needed to increase the contrast of the background in the images).

Rahman and Hellicar (2014) were able to identify mature grape bunches in a vineyard using Hough transformations in order and SVM to detect mature and immature grape bunches. Roscher *et al.* (2014) also used Hough transformation and CRF to classify the berry candidates detected on RGB images taken in field conditions with an algorithm that was able to learn about the grape features autonomously. Biswas *et al.* (2015), made a comparison between the Support Vector Machine (SVM) and back-propagation neural network (BPNN) algorithms for image classification concluding than the SVM performs better and faster than the BPNN.

### Yield Estimation

The feasibility of grape yield estimation using colour segmentation and regression analysis was demonstrated by Dunn and Martin as early as 2004, followed by studies (Serrano *et al.*, 2005) to establish the relation between the bunch volume measured for different grape varieties and the real weight during harvesting at different maturity stages. Researches on the differences in yield, taking into account the regional and continental differences and their effect on yield estimation were conducted by Taylor *et al.* (2005), finding that spatial yield patterns were fairly consistent while yield level was not.

Diago *et al.* (2012) used image analysis for canopy feature extraction, leaf area estimation and yield forecast using the Mahalanobis distance to determine class membership of the pixels in the images. De la Fuente *et al.* (2015) compared several models to predict grape yield at different stages of growing season. Hung *et al.* (2015) created a general framework which generalised the yield prediction for different crops using image processing of autonomous vehicle pictures using CRF and a logistic regression classifier. Lopes *et al.* (2016) proposed a solution that uses an autonomous vehicle and image processing to estimate yield based on the berry pixel count using linear regression. Kushwaha and Bhattachrya (2015) presented a paper in which they discuss a general algorithm for yield estimation using large datasets and the Hadoop distributed platform for computation.

Liu, Marden and Whitty (2013) gathered a summary of the automated image processing methods for fruit weight estimation and studied the contribution of the bunch parameters to the total weight of the bunch. Nuske *et al.* (2011) developed an algorithm to predict yield in a vineyard using radial symmetry to detect and count berries, removing the false positives by considering only points that has at least 5 neighbors (thus forming a cluster), and finally estimating yield using linear regression, this algorithm was enhanced and tested in field conditions in a later work (Nuske *et al.* 2014). Biswas *et al.* (2015), made a comparison between the Support Vector Machine (SVM) and back-propagation neural network (BPNN) concluding than the SVM performs better and faster than the BPNN.

## Theory

Early studies from Chamelat *et al.* (2006) showed a method to detect grape clusters in vines which transformed RGB image into a Hue, Saturation Value (HSV) colour space, in order to extract colour features that would later be used to feed a support machine vector to identify clusters, which is similar to the approach proposed by Berenstein *et al.*(2010), who described four machine vision algorithms to identify both leaf and grape pixels as a way to reduce pesticide consumption in vineyards, although their leaf detection algorithm was based purely on RGB channel value filtering. A newer study carried out by Lopes *et al.* (2016) used autonomous vehicles to gather images to feed a cluster detection algorithm which calculated the total yield as a linear function of the detected cluster area. This approach was able to explain up to 80% of the cluster weight variability and it is similar in approach to Smartrural’s one.

Nuske *et al.* (2014) proposed two approaches to increase the yield prediction accuracy of the existing algorithms: the first one considered the convex hull (smallest polygon for which a set of points is either on the verge or the interior of the polygon) of the detected clusters and estimating that the yield is a function of the density, average thickness of the grape cluster area and depth. The second approach tried to predict the cluster size, estimating the occluded cluster volume by approximating the shape of the cluster with a convex hull and ellipsoid model.

Assuming that the level of occlusion has a direct relation in the yield prediction, the work proposed here aims to provide an alternative way to detect cluster occlusions by inspecting the immediate boundaries of the cluster to detect leaves that hide a part of the cluster and introduce this factor on the final throughput calculations.

Terms

|  |  |
| --- | --- |
| **Term** | **Meaning** |
| RGB | Colour model consisting of Red, Green & Blue channels |
| HSV | Colour model consisting on Hue, saturation & Value channels |
| PLS | predictive partial least squares |
| SVM | Support Vector Machine |
| ML | Machine Learning |
| CRF | conditional random field |
| SIFT | Scale Invariant Feature Transform |
| BPNN | back-propagation neural network |
| MSE | Mean squared error |
| MAD | Median absolute deviation |

# Analysis and Design

The following section explains the methods and engineering decisions taken to create the framework that extracts the picture information features needed. The first section would focus on the datasets used, with subsequence sections detailing the image processing techniques applied and the resulting datasets created.

Datasets

Smartrural provided the following datasets for our study:

1. A set of 20157 raw (untreated) images
2. 75686 images containing information about the shape of the detected grape clusters
3. 12 text files describing the position of these inference images in the original pictures
4. A JSON file with information of the raw picture geolocation, number of grape pixels and number of clusters for each picture

Examples of these datasets are shown below for clarification:



Figure 1:Raw Image

https://lh3.googleusercontent.com/hiZW3-NJQgTkPyfp37h9CbIty6Gijd_GbC7eNEClnlcIKvXo_t5DRt0FoC7xinjIu6Y5kiUVBYUm7mnQN8mU-Ife-M77ZTnPWWfMJd2ogd10MYoQ3BnhhWx7g8YbebnCw1EPqxEhhttps://lh5.googleusercontent.com/UUNDRi70GhEsff5qfPuSK13J7BH2Kt73X2iC5Fvq3pzUzxrn5GSQpohyGJTpwaaiiwpZvU_bZNXGWYazAj2mXsHV0PtYH27Q-O4PxSnpTZiuJV91dMNU1UMQxS9OESjo9RWIaDKAhttps://lh5.googleusercontent.com/3crjKBahGyaE01qwlpoV3h8zzQmsdhW3BHL8H07Zf1t9nU8FmMCTQXngJF2e1ZRWA43l9pe9CuZLGVYYWvecQOjBxo-W-0hcpJdTK_7kEThPdEfisk4geaqvKL5H5wppumrg7NzVhttps://lh5.googleusercontent.com/DiMgeFi45l90HQzkCBpQlDpATKr4PjD-CJMM4n3222kMqDVCJqZp7N279pJI1MC92y9XaGqcXj-o0Wv42Id0wUwy42BSUeZlXrDE-TR_COfYbW6UBPQCs7JRBjtDYL3uMGITsf_dhttps://lh5.googleusercontent.com/WzOwCo-nlUo2ba_m8u8ojTaSPQrFANknAH-7noZvEs1xzifbHsj9y_WOT67F59p0su-bvgI8-UfQOl0CIzrOZdb-SHpyPnZBRa2Z0-KNr2dkGXYddWu9dMM92vpcj4EJvIpOE_a9

Figure 2:Inference pictures

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Image Name | Inference type | inference probability | column min | row min | column max | row max |
| Name of the original image | Type of inference detected (only clusters) | probability of the inference being of the type described in the “Inference type column | Y coordinate of the bottom left corner of the inference picture within the original image | X coordinate of the bottom left corner of the inference picture within the original image | Y coordinate of the upper right corner of the inference picture within the original image | X coordinate of the upper right corner of the inference picture within the original image |
| z-img-000-000004.jpg | cluster | 0.994097 | 371.372 | 1349.73 | 468.226 | 1481.06 |

Table 1: Inference information

The following element was extracted from the JSON described before. The information was given in a single line, but it has been reformatted here for displaying purposes:

|  |
| --- |
| {  **"type"**:"Feature",  **"geometry"**:{  **"type"**:"Point",  **"coordinates"**:[                -4.283807,                41.623104             ]          },  **"properties"**:{  **"url"**:"https://s3.amazonaws.com/tileo-datasets/quadvine/2017/valdemonjas/valdemonjas-2017-09-13-inferences/valdemonjas-2017-09-13\_04/z-img-000-000869.jpg",  **"width"**:2148,  **"height"**:2048,  **"pixels"**:"236",  **"clusters"**:"2"          }       } |

Table 2: json cluster recognition extract

The picture below has been included to help visualize the result of Smartrural’s cluster recognition process, the area enclosed by the red lines corresponds to a positive cluster match which position information is contained on the inference files previously described and appears surrounded by a blue line in the image.



Figure 3: Cluster recognition output image

Design of the solution

The solution to extract the feature pixels (leaf pixels) was divided in 2 stages: The first step was to reconstruct the overall inference images from the partial inference pictures and the inference information contained on the text files. The result image that this process yields was passed to an algorithm that detects the cluster surrounding pixels in order to create a picture which was later used as a pixel mask. The second step was to process and filter the original images to extract pixels belonging to leaves, and apply the masks mentioned earlier, so only the pixels belonging to the cluster surroundings were considered in the final calculation.

### Creating the cluster surrounding masks

The first step of the designed solution reconstructed the binary images from the partial inference pictures and the inference information available. An example of the output of this process can be seen on the left image below (although this image is not stored due to performance and disk optimization reasons):

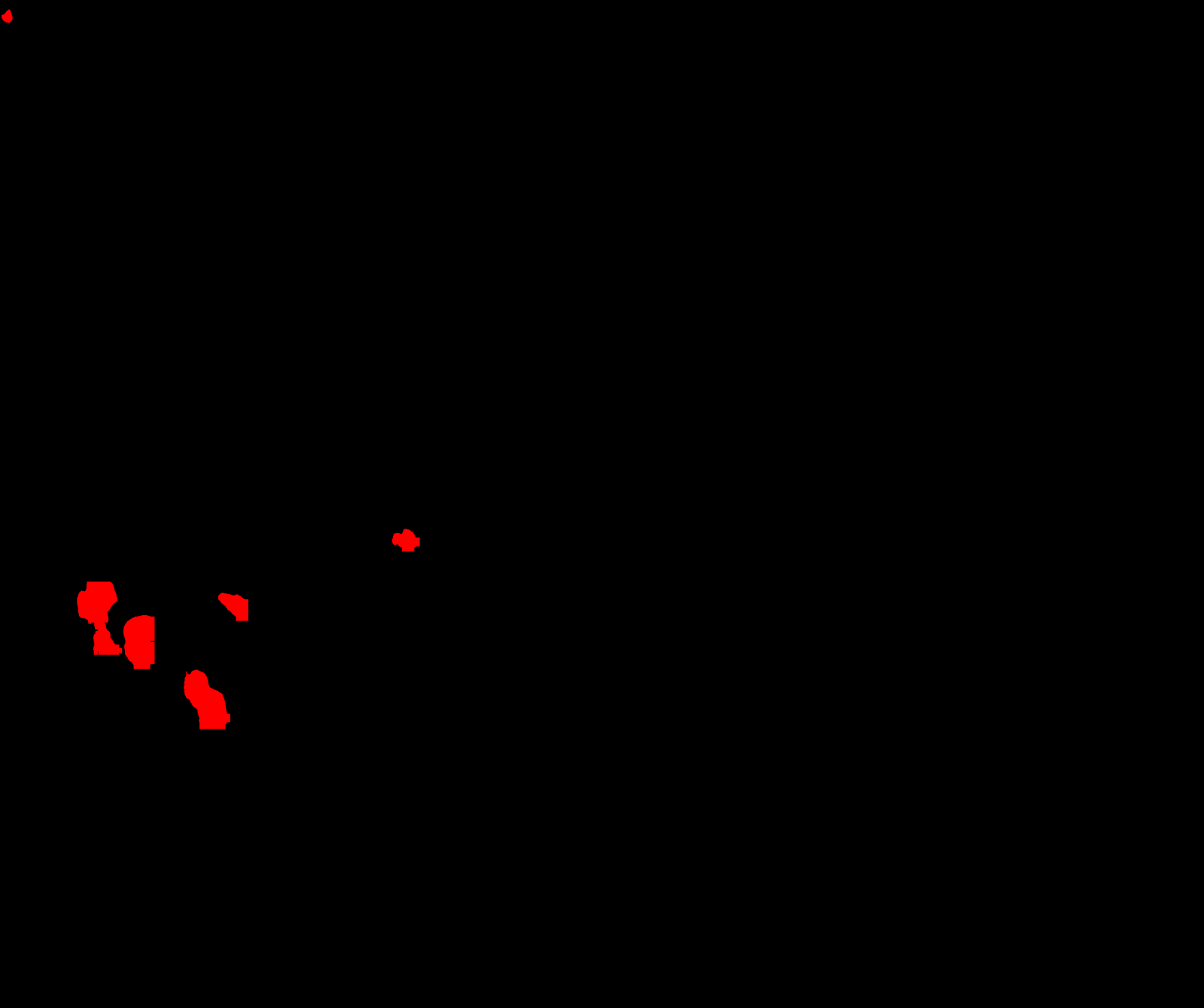
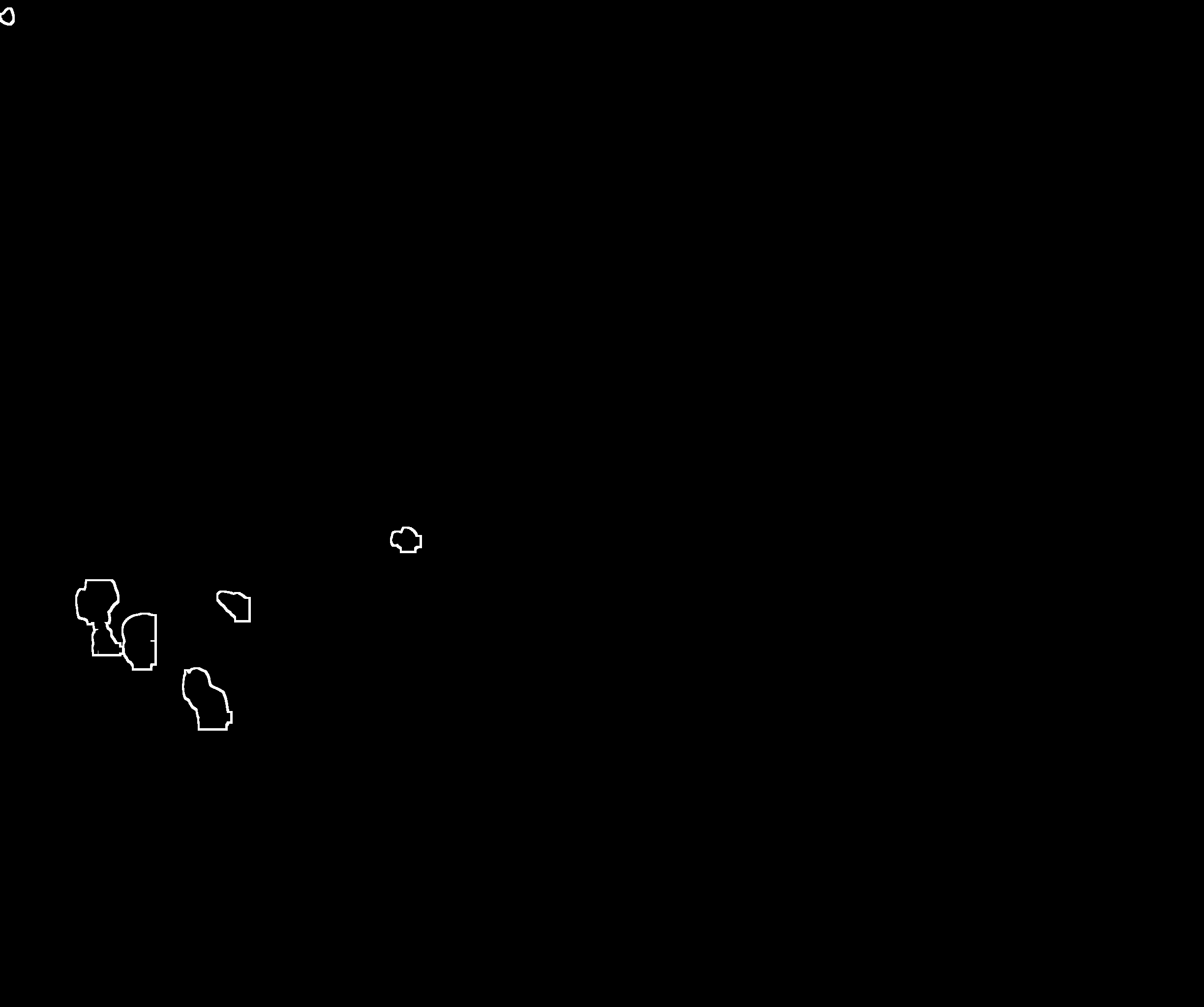
 

Figure 4: Inference reconstruction and resulting mask

An algorithm to detect the pixel surrounding the clusters is then applied to the image (right picture above), resulting in an image that would be used as a binary mask by the second step of the solution. The name of the images and their parent folder serves to uniquely identify the original images with its inference information and partial inference pictures. A diagram flow of the solution is displayed below:

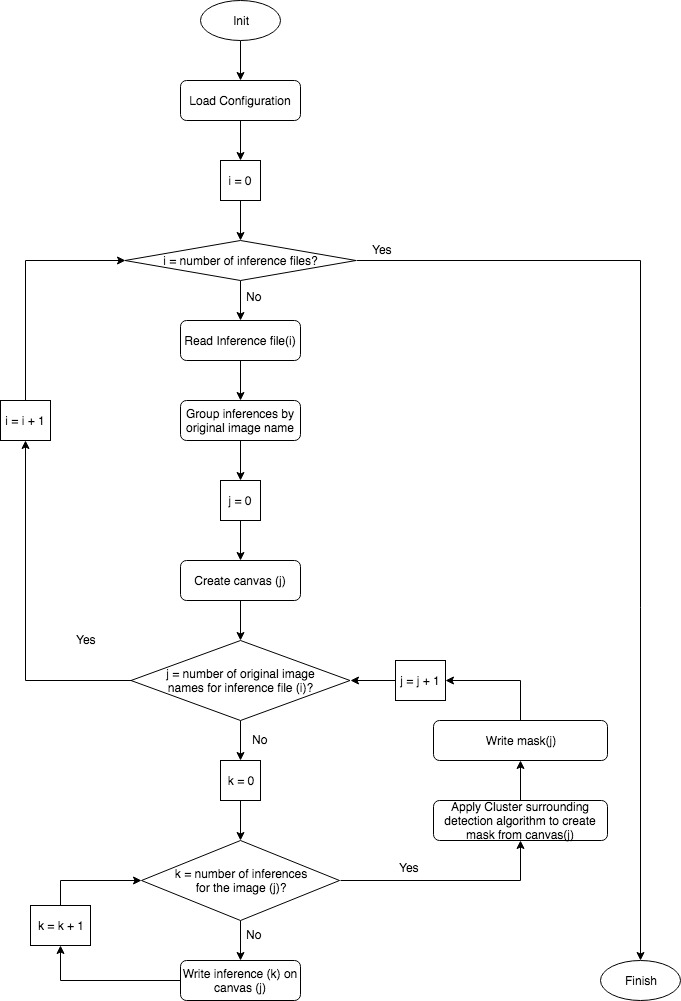


Figure 5: Inference reconstruction flowchart

The result mask image file has the same name and parent folder than the original image but different root path, so further steps can match the original image with its derived inference reconstruction image.

### Image transformations

The second stage on the solution architecture is a pipeline that applies several filters and transformations to the original images in order to detect the leaf pixels in the pictures. The last one of these filters is a bitwise\_and filter that uses the masks generated in the first step. The bitwise\_and operation removes all the pixels from the filtered image that does not have a correspondent value on the applied mask, it is similar to cropping an image with the non-zero value pixels of the mask image. To calculate the actual leaf percentage, we divide the number of pixels with a non zero value in the image coming from the last transformation between the number of pixels with a non zero value in the mask image. All the image transformations are done in memory to avoid filling the disk with intermediate images as well as slowing down the whole process due to I/O operations.

Using the image name as a join key, the application performs a lookup on the json file containing the cluster recognition information, to match the values of the cluster recognition process with the leaf percentage information, and writes this information to a file containing the following fields: the picture name, latitude and longitude where the picture was taken, number of pixels belonging to grape clusters detected, number of grape cluster detected and an occlusion percentage. The name of the file is also the same than the image name, but with a different root path and different extension. The implemented solution has the following flow diagram:

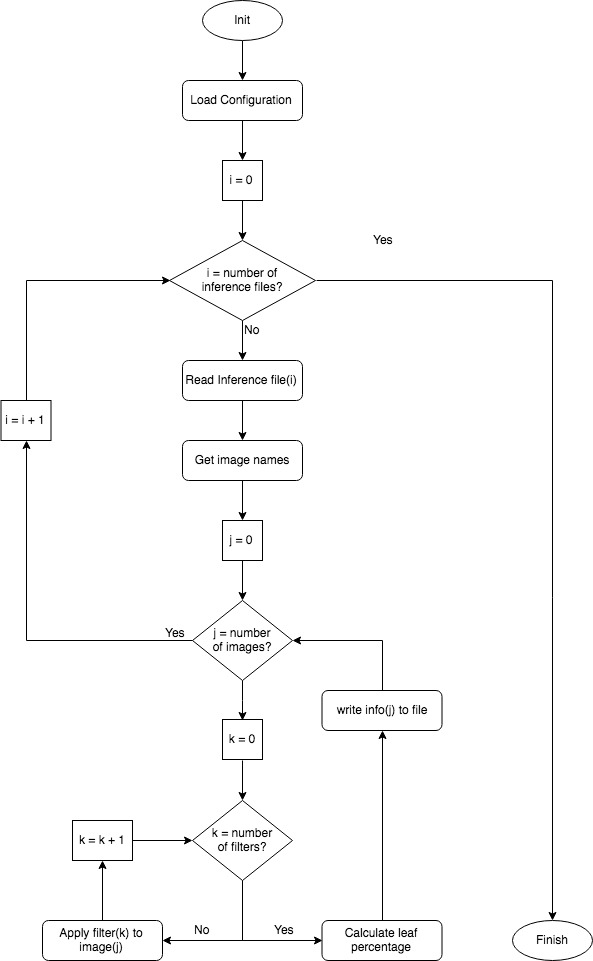


Figure 6: Image processing flowchart

As it can be seen, the process iterates over all the images, and applies a list of transformations sequentially, with the output image of one filter being the input of the next one.

#### Average Blur filter transformer

The filter aim is to reduce the noise by computing a pixel value as the average of the surrounding pixels in a radius of r which corresponds to the kernel size:

In the above formularepresents the pixel in coordinates x and y which is computed as a weighted sum of the input pixel values, represented here by , multiplied by the kernel factor at that distance, represented by .

#### Median filter transformer

The median filter computes the pixels value by ordering the values of the neighbour pixels and taking the middle value of the sorted list. As noted by Szeliski (2011) the median filter is not as effective as the gaussian filter because it selects a single input pixel to replace the original pixel. Although as noted by Szeliski (2011) locally adaptive histogram equalization which uses a set of neighbour pixel values in the vicinity of a pixel to determine the pixel’s final output value are a common technique for image noise reduction.

#### Gaussian filter transformer

The gauss filter uses a kernel that represents the shape of a gaussian bell, which is defined by the following equation:

The idea behind this filter is that the final value of the transformed pixel would be less affected by farther pixels and closer pixel would have greater weight on the value.

#### Histogram Backprojection filter

The backprojection algorithm was developed by Swain and Ballard (1990) this filter searches for a target for which a multidimensional histogram has been computed beforehand (noted with M).  Then a histogram of the image in which we would search the target object is computed (noted by I) as well as a third histogram (noted R) which is the relation between M divided by I. This R histogram is ‘backprojected’ on the image in a way that the values of the image are replaced by the values of R that they index. The resultant image is then convolved with a kernel, being the peak in this convolved image the expected location of the target image. The pixels with a probability less than the configured lower boundary are not presented in the resulting image. The target image (M) has been extracted cropping a leaf area using Apple Preview version 9.0



Figure 7: Histogram target image

The histogram of the image shown above was used to convolve all the images resulting from applying the median and gaussian filters to the original images, in order to extract only the leaf pixels from the pictures. An example of the result of applying this filter can be seen below, where the majority of the pixels not belonging to leaves were removed.



Figure 8: Histogram backprojection result image

As it can be seen, most of the non leaf pixels has been removed from the image, although there are some inner leaf areas that has been removed too, mostly due to the brightness produced by the camera flash.

#### Cluster surrounding detection filter

The cluster surrounding detection filter is composed by three operations: An average blur transformation, a bitwise\_not transformation and finally a bitwise\_and transformation. The transformation is applied over the results of the binary image reconstruction described on section 3.2.1. The idea behind is the following:

* The blur filter applied on the image containing the cluster detections, extends the border of the clusters on a radius r (defined as the blur radius by configuration)
* A bitwise\_not operator is applied to the original image (not blurred), inverting the value of the pixels. If we take the figure 4 as an example, the cluster detections would appear black, and the rest of the pixels would be red.
* A bitwise\_not operation is then performed, with one of the operands being the blurred image, and the second operand being the output image from the previous step. This means that the pixels belonging to the original detected clusters would be removed from the final image, resulting in an image that contains only the blurred cluster borders.

#### Enforced Pixel color filter

This filter is intended to be run after the cluster surrounding detection filter in order to highlight and unify the color value across all the detected cluster surrounding area. The filter iterates over all the pixels of the image, setting the color to white if the pixel has any color value set on it, and leaving it blank otherwise.

## Alternative Solutions

The current solution makes extensive use of the OpenCV library, the reason for this is not only due to availability of already implemented transformations, but instead due to performance reasons. A first approach to the problem of detecting leaves surrounding the identified clusters used an iterative approach to identify the pixels to look up as well as checking the color range of those pixels. The Algorithm utilized iterated over all the pixels of the picture, and for each non-black pixel encountered, it inspected the pixels on a predefined radius to search for green pixels. This solution was orders of magnitude slower than the selected one (the processing time varied from 3 to 10 minutes per picture compared to the 5 seconds on average the current process takes) and failed occasionally due to the memory shortages as result of having to store partial results in memory during the computation.

# Implementation (Realization)

Design Approach

The implementation of the solution previously described was done bearing in mind that the application would need to run in a personal computer with limited capabilities, although these applications can be easily parallelized by means of dividing the input in several chunks, and assigning those chunks to different nodes in a cluster. The filter functionality was originally developed using iterative methods that made several passes over each one of the pixels of the pictures, but this solution was proven to be slow and inadequate for the available hardware. The final version of the code made extensive use of the OpenCV library (https://opencv.org/), released under a BDS license and supported in MacOS among other Operating systems. This library is written in C/C++ but has a java API which acts as a proxy to the C/C++ library.

### Overall Design

The UML diagram of the two applications used to extract the occlusion information, as well as the table model diagrams are displayed below. The application is divided in *layers,* which main purpose is the following:

1. Application classes: Wrapper to retrieve the configuration and instantiate and run the runnables
2. Runnable classes: Class that has several services and executes the business logic of the application, mainly iterative calculations
3. Service classes: Classes containing all the functionality related with a domain model
4. Transformer and utility classes: “Bricks” that contain the logic of the image transformation required to extract the leaf percentage and other general functions
5. Model classes: Representation of the business model domain

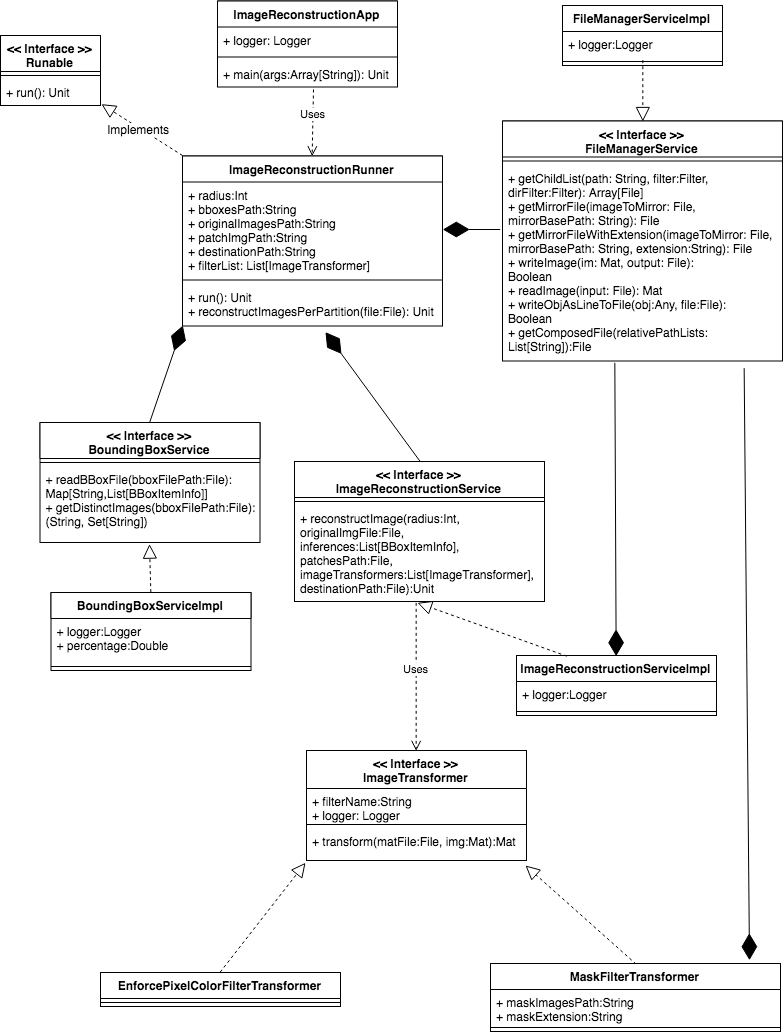


Figure 9: Image Reconstruction UML Design

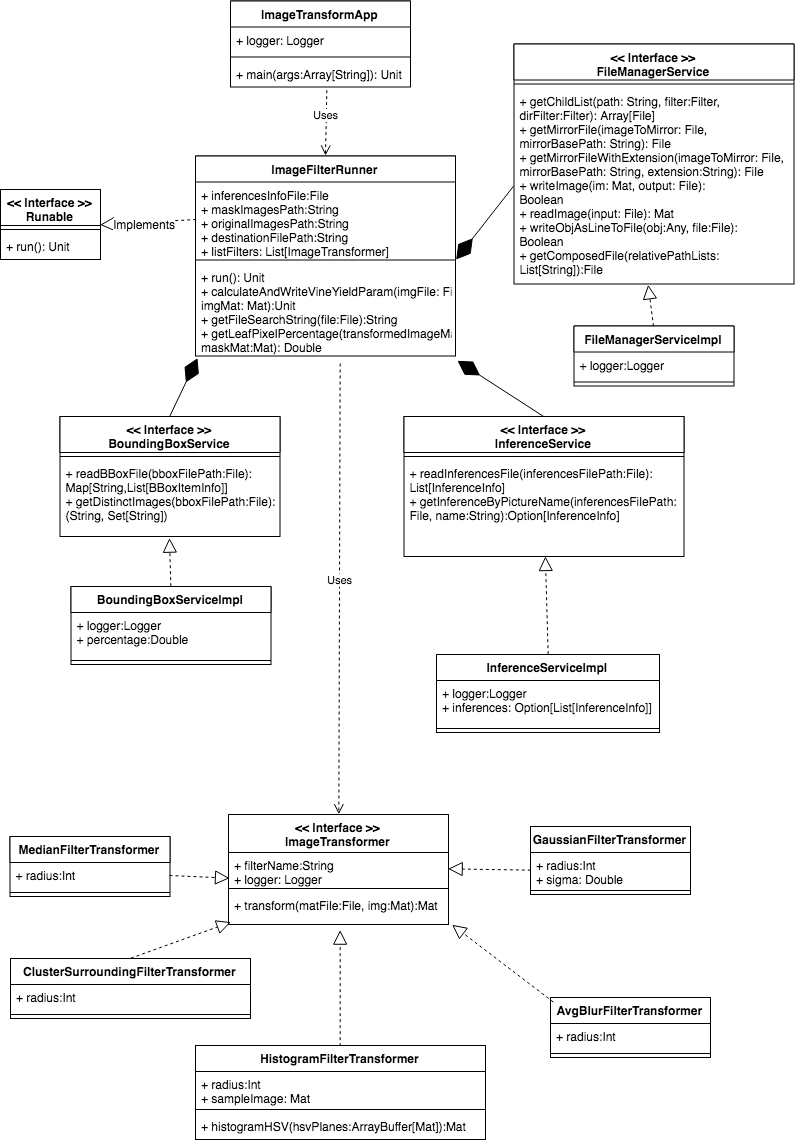


Figure 10: Image Transform UML Design

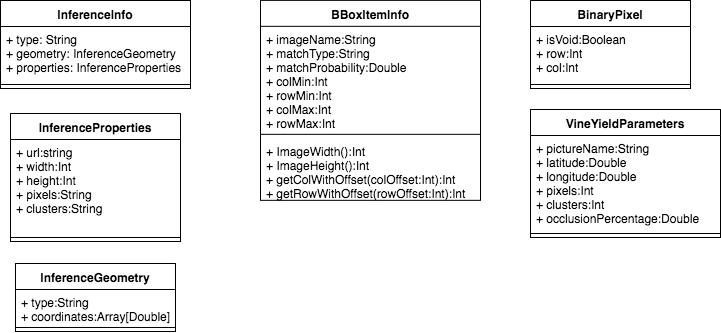


Figure 11: Models UML Design

### Application Classes

There are two classes that contains a main method (entry point for java or scala applications) named *ImageReconstructionApp* and *ImageTransformApp*. We will call this classes *Application classes* for the rest of the paper and they are defined inside the *com.smartrural.estimator* package. These application classes requires a single runtime argument specifying the location of the properties configuration file it uses. The configuration values contained in these properties files are validated and passed to an instance of the appropriate *Runnable class* that would execute the application logic, the application classes does not more than acting as a wrapper to retrieve the configuration and pass it to the appropriate *Runnable* class. *Runnable* is a java interface designed to be implemented by any class intended to be executed in a thread. It defines a single method:

|  |
| --- |
| **public** **interface** Runnable{  **public abstract void** run();  } |

Table 3: Runnable Interface

The *Application Class* instantiates the appropriate *Runnable class* which specific implementation depends on the type of application executed. It also instantiates a module (described in 4.1.2) to inject the different *Service classes* (described in 4.1.4) used by the *Runnable class* and starts the execution. The code of one of these application classes (the *PixelImageLocatorApp*) can be seen below, the other class has been omitted for brevity:

|  |
| --- |
| **object** ImageReconstructionApp {  **val** *logger* = LoggerFactory.*getLogger*(getClass)  **def** main(args:Array[String]):Unit = {  **if**(args.length!=1) {  *logger*.error(**"Invalid number of parameters. USAGE: 'com.smartrural.estimator.ImageReconstructionApp <Conf\_path>'"**)      System.*exit*(1)    }    System.*loadLibrary*(Core.NATIVE\_LIBRARY\_NAME)  **implicit val** imageReconstructionService = **new** YieldEstimatorModule  **val** properties = **new** Properties()    properties.load(**new** FileInputStream(**new** File(args(0))))  **val** radius = properties.getProperty(AppConstants.*PropertyRadiusPixelLocator*)  **val** bboxesPath = properties.getProperty(AppConstants.*PropertyBBoxesPath*)  **val** originalImagesPath = properties.getProperty(AppConstants.*PropertyOriginalImagePath*)  **val** patchImgPath = properties.getProperty(AppConstants.*PropertyPatchesPath*)  **val** dstPath = properties.getProperty(AppConstants.*PropertyMaskImagePath*)  **if** (*Some*(radius).isEmpty ||  *Some*(bboxesPath).isEmpty ||  *Some*(originalImagesPath).isEmpty ||  *Some*(patchImgPath).isEmpty ||  *Some*(dstPath).isEmpty ){  *logger*.error(**"Invalid set of parameters to run the Image reconstruction process. Please review the configuration"**)      System.*exit*(1)    }  **val** runner = **new** ImageReconstructionRunner(radius.toInt, bboxesPath, originalImagesPath, patchImgPath, dstPath)  **new** Thread(runner).start  }  } |

Table 4: Image Reconstruction application Class

### Dependency Injection Module

Dependency injection is an object oriented design pattern that enforces the assembly of an object’s internal dependencies by an external entity, releasing the object from the need of having to know the concrete implementations of the interfaces used, as well as creating those class instances. Dependency injection is a common pattern which allows custom “wiring” of the dependencies, thus decoupling the different layers that compose an application.

Scala’s dependency injection management is not as evolved as the java counterpart (with the Spring framework being a standard *de facto*) however there are libraries that are trying to feel the gap. One of those libraries is *Scaldi (http://scaldi.org)*, which provides a simple way to do dependency injection in Scala by the use of implicit parameters and runtime type checking. The idea behind this framework is that you define an implicit class extending the scaldi’s *Module* trait, and inside this class you define the specifics *bindings* between the trait definitions and their specific implementation. Below is the code of the Module defined and used across all the *Application classes* which is contained inside the  *com.smartrural.estimator.di* package:

|  |
| --- |
| **class** YieldEstimatorModule **extends** Module{    bind[BoundingBoxService] to **new** BoundingBoxServiceImpl    bind[ImageReconstructionService] to **new** ImageReconstructionServiceImpl()    bind[FileManagerService] to **new** FileManagerServiceImpl    bind[InferenceService] to **new** InferenceServiceImpl  } |

Table 5: Dependency Injection Module

This class is passed as an implicit parameter from the *Application class* to the *Runnable class* or any other class that makes use of the scaldi’s dependency injection as it has already been shown before. When the *Runnable class* is instantiated, Scaldi decides how to wire the specific instances of the services based on the class type annotations. This is better understood with an example:

|  |
| --- |
| *// Scaldi injects the specific implementation of the FileManagerService trait, if the*  *// YieldEstimatorModule above is used, an instance of FileManagerServiceImpl will be*  *// assigned to the fileManagerService value*  **val** *fileManagerService* = inject[FileManagerService] |

Table 6: Service Injection Snippet

Dependency injection eases testing classes, as you can choose to inject a mock implementation of the dependency with predefined responses, and test the expected behavior given those responses.

### Runnables

The *Runnable classes* are those which implements the *Runnable* interface previously described, and executes the business logic that performs the application task. These classes are defined inside the *com.smartrural.estimator.runner* package. The constructor parameters needed to execute are tailored to the functionality the *Runnables* carries out. An example of this can be seen below:

|  |
| --- |
| **class** ImageReconstructionRunner(radius:Int,                                bboxesPath:String,                                originalImagesPath:String,                                patchImgPath:String,                                destinationPath:String)(**implicit** inj:Injector) **extends** Injectable **with** Runnable{  */\*\**  *\* The image reconstruction service*  *\*/*  **val** *imageReconstructionService* = inject[ImageReconstructionService]  */\*\**  *\* The file manager service*  *\*/*  **val** *fileManagerService* = inject[FileManagerService]  */\*\**  *\* The bounding box service*  *\*/*  **val** *boundingBoxService* = inject[BoundingBoxService]  */\*\**  *\* The list of filters to apply*  *\*/*  **val** *filterList* = List(**new** ClusterSurroundingFilterTransformer(radius.toInt), **new** EnforcePixelColorFilterTransformer)  */\*\**  *\** **@inheritdoc**  *\*/*  **override def** run():Unit = *fileManagerService*  .getChildList(bboxesPath)      .foreach(reconstructImagesPerPartition)  */\*\**  *\* Reconstructs all the images described in the bbox file passed*  *\** **@param bboxFile** *the bbox file containing the image information*  *\*/*  **def** reconstructImagesPerPartition(bboxFile:File):Unit =  *boundingBoxService*.readBBoxFile(bboxFile)      .map({**case** (image, inferenceList) =>{  **val** partition = bboxFile.getParentFile.getName  *imageReconstructionService*.reconstructImage(radius,  *fileManagerService*.getComposedFile(*List*(originalImagesPath, partition, image)),          inferenceList,  *fileManagerService*.getComposedFile(*List*(patchImgPath, partition)),  *filterList*,  *fileManagerService*.getComposedFile(*List*(destinationPath, partition)))      }    })  } |

Table 7: Image Reconstruction Runner

As it can be seen the class *ImageReconstructionRunner* implements the Injectable trait (required by the scaldi framework to perform the dependency injection) and the Runnable interface. The class has three injected services which are then used by the class logic, defined in the *run* and *reconstructImagesPerPartition* methods.

### Application Services

The *Application Service classes* are designed to enclose all the functionality related with a specific domain model. Four services have been created which are described below and defined inside the *com.smartrural.estimator.service* package.

#### BoundingBoxService

Service that provides operations to parse and extract the information contained in the bounding boxes files. This service defines two methods:

|  |
| --- |
| /\*\*  \* Reads the bboxFile provided  \* **@param bboxFilePath** the file to read  \* **@return** the Inferences extracted from the file  \*/  **def** readBBoxFile(bboxFilePath:File):Map[String, List[BBoxItemInfo]]  /\*\*  \* Gets the list of distinct images contained in the file  \* **@param bboxFilePath** the file to read  \* **@return** the list of distinct images  \*/  **def** getDistinctImages(bboxFilePath:File):(String, Set[String]) =  (bboxFilePath.getParentFile.getName, readBBoxFile(bboxFilePath).keySet) |

Table 8: Bounding Box Service

#### FileManagerService

Service that deal with file Input/Output operations as well as path creation and traverse functionality:

|  |
| --- |
| **trait** FileManagerService {  **private val** *filter* = **new** IOFilterDSStore(*IOfilter*)  */\*\**  *\* Returns the list of files contained in the folder passed*  *\** **@param path** *the path to the folder*  *\** **@param filter** *the filename filter*  *\** **@param dirFilter** *the dir filename filter*  *\** **@return** *the list of child files*  *\*/*  **def** getChildList(path: String, filter:IOFileFilter = *filter*, dirFilter:IOFileFilter = *filter*): Array[File]  */\*\**  *\* Gets a file that is mirroring the passed one with a different root path*  *\** **@param imageToMirror** *the image to mirror*  *\** **@param mirrorBasePath** *the base path to construct the mirror image*  *\** **@return** *the File representing the mirror file*  *\*/*  **def** getMirrorFile(imageToMirror: File, mirrorBasePath: String): File  */\*\**  *\* Gets a file that is mirroring the passed one with a different root path*  *\** **@param imageToMirror** *the image to mirror*  *\** **@param mirrorBasePath** *the base path to construct the mirror image*  *\** **@param extension** *the extension that the image should have*  *\** **@return** *the File representing the mirror file*  *\*/*  **def** getMirrorFileWithExtension(imageToMirror: File, mirrorBasePath: String, extension:String): File  */\*\**  *\* Wrapper of the ImageIO static write function, so it is possible to mock this call*  *\**  *\** **@param im***a* <**code**>*RenderedImage*</**code**> *to be written.*  *\** **@param output***a* <**code**>*File*</**code**> *to be written to.*  *\** **@return** <**code**>*false*</**code**> *if no appropriate writer is found.*  *\*/*  **def** writeImage(im: Mat, output: File): Boolean  */\*\**  *\* Wrapper of the ImageIO static read function, so it is possible to mock this call*  *\**  *\** **@param input** *the file to read from*  *\** **@return** *the mat*  *\*/*  **def** readImage(input: File): Mat  */\*\**  *\* writes the given object into the destination file*  *\** **@param obj** *the obj to write as a line*  *\** **@param file** *the destination file*  *\*/*  **def** writeObjAsLineToFile(obj:Any, file:File):Boolean  */\*\**  *\* Gets a composition of the relative paths into a File*  *\** **@param relativePathLists** *the paths list*  *\** **@return** *the file composed*  *\*/*  **def** getComposedFile(relativePathLists:List[String]):File = {  **val** reversedRelativePathLists = relativePathLists.reverse  **def** getComposedFileAux(relativePaths:List[String]):File = relativePaths **match**{  **case** filePath *:: Nil* => **new** File(filePath)  **case** filePath *::* elementList => **new** File(getComposedFileAux(elementList), filePath)  **case** \_ => **throw new** IllegalArgumentException(**"Unrecognized collection"**)    }    getComposedFileAux(reversedRelativePathLists)  }  */\*\**  *\* Filter class to skip Mac filesystem files*  *\** **@param filter** *the default filter to call*  *\*/*  **private** [FileManagerService] **class** IOFilterDSStore(filter:IOFileFilter) **extends** IOFileFilter {  **override def** accept(file: File): Boolean = **if** (file.getName == **".DS\_Store"**) **false else** filter.accept(file)  **override def** accept(dir: File, name: String): Boolean =  **if** (name == **".DS\_Store"**) **false**  **else** filter.accept(dir, name)  }  } |

Table 9: File Manager Service

#### ImageReconstructionService

Service that provides the functionality to construct the binary images from the partial inference pictures and the bounding boxes file. Defines a single method:

|  |
| --- |
| **trait** ImageReconstructionService {  */\*\**  *\* Creates a full binary image from the patches and the information contained in the bboxFile provided*  *\** **@param radius** *the radius of the cluster to consider*  *\** **@param originalImgFile** *the path to the original image*  *\** **@param inferences** *List of inferences for the image*  *\** **@param patchesPath** *the path to the patch images*  *\** **@param destinationPath** *the destination path to write the final image*  *\*/*  **def** reconstructImage(radius:Int,                       originalImgFile:File,                       inferences:List[BBoxItemInfo],                       patchesPath:File,                       imageTransformers:List[ImageTransformer],                       destinationPath:File):Unit  } |

Table 10: Image Reconstruction Service

#### InferenceService

Service that provides operations to parse and extract the information contained in the inference information file. This service defines two methods:

|  |
| --- |
| **trait** InferenceService {  /\*\*  \* Reads the inferences file provided  \* **@param inferencesFilePath** the file to read  \* **@return** the Inferences extracted from the file  \*/  **def** readInferencesFile(inferencesFilePath:File):List[InferenceInfo]  /\*\*  \* Gets the inference corresponding to the picture name passed  \* **@param inferencesFilePath** the file to read  \* **@param name** the picture name  \* **@return** the option containing the inference  \*/  **def** getInferenceByPictureName(inferencesFilePath:File, name:String):Option[InferenceInfo] =  readInferencesFile(inferencesFilePath).find(inf => inf.properties.url.endsWith(name))  } |

Table 11: Inference Service

### Image transformers

The image transformers are classes that acts as wrappers to the transformations defined in the ‘Image Transformations’ section. These classes are contained in the *com.smartrural.estimator.transformer.impl* package and implements the *com.smartrural.estimator.transformer.ImageTransformer* trait, which has a single method:

|  |
| --- |
| */\*\**  *\* Transform an imagen according to the internal implementation of the Transformer*  *\** **@param img** *the image to transform*  *\** **@return** *the transformed image*  *\*/*  **def** transform(img:Mat):Mat |

Table 12: Image Transformer trait

An example of a class of this type is given below:

|  |
| --- |
| **class** MedianFilterTransformer(radius:Int) **extends** ImageTransformer{  */\*\** **@inheritdoc** *\*/*  **override def** applyTransform(img:Mat):Mat = {  **val** dst = *getMat*()    Imgproc.*medianBlur*(img, dst, radius)    dst  }  } |

Table 13: Median Filter Transformer

The images processed are represented here by the Mat class, which is an openCV class, defined as “an n-dimensional dense numerical single-channel or multi-channel array. It can be used to store real or complex-valued vectors and matrices, grayscale or color images, voxel volumes, vector fields, point clouds, tensors, histograms.” As both the input and output type of the *applyTransform* operation is a *Mat*, several image transformers can be chained together in a pipeline.

### Models

The bottom layer of the solution is composed by the model classes, which provides a representation of the business domain. Scala provides a special type of classes called *case classes*, which makes available a set of functionality such as default equals and hashcode implementation, class constructors and pattern matching over the classes. The model classes are defined inside the *com.smartrural.estimator.model* package and provides a mapping between the output data formats and the class instances inside the program, the following is the class representation of the dataset presented in table 1. As it can be seen this classes might contain logic referred exclusively to the domain these classes represents.

|  |
| --- |
| */\*\**  *\* Model class for the BBox info file content*  *\** **@param imageName** *the image name*  *\** **@param matchType** *type of the match, this attribute is always "cluster"*  *\** **@param matchProbability** *the probability of the detected match*  *\** **@param colMin** *column min value*  *\** **@param rowMin** *row min value*  *\** **@param colMax** *column max value*  *\** **@param rowMax** *row max value*  *\*/*  **case class** BBoxItemInfo(imageName:String,                        matchType:String,                        matchProbability:Double,                        colMin:Int,                        rowMin:Int,                        colMax:Int,                        rowMax:Int){  */\*\**  *\* The width of the image*  *\*/*  **val** *ImageWidth*:Int = colMax - colMin  */\*\**  *\* The height of the image*  *\*/*  **val** *ImageHeight*:Int = rowMax - rowMin  */\*\**  *\* Method that returns the relative position of the column pass into the image this inference is located*  *\** **@param colOffset** *the column offset value*  *\** **@return** *the relative column value*  *\*/*  **def** getColWithOffset(colOffset:Int):Int = colMin + colOffset  */\*\**  *\* Method that returns the relative position of the row pass into the image this inference is located*  *\** **@param rowOffset** *the row offset value*  *\** **@return** *the relative row value*  *\*/*  **def** getRowWithOffset(rowOffset:Int):Int = rowMin + rowOffset  } |

Table 14: Bounding Box Model Class

## Application scripts

The Image transform application’s output is a set of comma separated text files which name is equal to the image from where the result comes from, but with a *txt* extension (i.e. example.jpg results would be stored in a file named example.txt). The reason behind this decision is that if the process fails or is stopped, it can skip already processing images just by checking if a file with the same name that the result file exists already saving a considerable amount of processing time.

This produces up to 7043 text files with a single line, that are combined through a simple bash script. The logic of the script is simple, it gets a sorted list of files and directories and then it copies the content of each one of those files to a single result file. The script is displayed below:

|  |
| --- |
| **#!/bin/bash**  **RESULT\_FILE='inferences\_result.csv**  **if [ -f $RESULT\_FILE ]; then**  **rm $RESULT\_FILE**  **fi**  **for dir in $(ls | sort)**  **do**  **for file in $(ls ${dir}/\*.txt |sort)**  **do**  **cat $file *>> $RESULT\_FILE***  **done**  **done** |

Table 15: Result Joiner script

## Testing

The functions contained in the service classes, the util classes and the runnables have been unit tested using the Junit, scalatest and scalamock frameworks. The fact that the class members are set through dependency injection allows for the different components to be replaced by mock implementations with predefined responses. The example below demonstrates this capability. The FileManagerService and the InferenceService were replaced by mock implementation with responses set in the method *setExpectations1.*

|  |
| --- |
| **val** *fileManager* = mock[FileManagerService]  **val** *inferenceService* = mock[InferenceService]  **def** setExpectations1():Unit= {    **val** result:MockParameter[Any] =  *VineYieldParameters*(**"test-classes/FilteredImage1.png"**, 1,-4.283807,41.623104,36,10, 0.5)  **val** mockResultFile:MockParameter[File] = *resultFile*  (*fileManager*.getChildList \_)    .expects(*rootPath*, TrueFileFilter.*INSTANCE*, TrueFileFilter.*INSTANCE*)    .returns(*Array*(*filteredImageFile1*)).anyNumberOfTimes()  (*fileManager*.readImage \_).expects(*filteredImageFile1*).returns(*filteredImageMat1*)  (*fileManager*.getMirrorImageFile \_).expects(\*, \*).returns(*inferenceImageFile*)  (*fileManager*.readImage \_).expects(*inferenceImageFile*).returns(*inferenceImageMat*)  (*fileManager*.writeObjAsLineToFile \_).expects(result, mockResultFile).returns(**true**).once  (*inferenceService*.getInferenceByPictureName \_)    .expects(*inferencesFile*, **"test-classes/FilteredImage1.png"**)    .returns(*Some*(*InferenceInfo*(  **"Feature"**, *InferenceGeometry*(**"Point"**, *Array*(-4.283807,41.623104)),  *InferenceProperties*(1, **"test-classes/FilteredImage1.png"**, 10, 10, 36, 1, 10))    )).once  } |

Table 16: Unit Test Example

## Tools and Frameworks

The software and libraries used to create the artifact are listed below:

* Programming language: Scala 2.11.8, python 3.0
* Integrated development environment: IntelliJ IDEA 2012.3.2
* Image processing library: OpenCV 3.4.0
* Test Framework: ScalaTest, ScalaMock, Junit
* Machine learning library: scikit-learn library 0.19.1

### OS and platform

All the described computations had been performed using a MacBook Pro (Retina, 15-inch, Mid 2014) with a 2.2 GHz Intel Core i7 processor, 16 GB 1600 MHz DDR3 memory and an Intel Iris Pro 1536 MB graphic card running on MacOS Sierra (v 10.12.6)

### Open CV

The image processing library used is Open Source Computer Vision Library (OpenCV) which was written in optimized C/C++, leveraging the hardware acceleration (GPUs) of the computer machine. Used extensively in the image transformations.

### scikit-learn

scikit-learn is an open software library for machine learning written in python and available in the three main operating systems (MacOs, Windows and Linux), and includes functionality to perform classification, regression, clustering or model selection.

### Intellij IDEA

The development work was carried out using the integrated development environment Intellij IDEA version 2016.3.2. The community version provides an Apache License 2.0 which lets the users create open source or commercial products with it. This integrated environment provides a set of easily installable plugins to augment the functionalities provided such as SBT, Junit or Git, which were used on the development of this project.

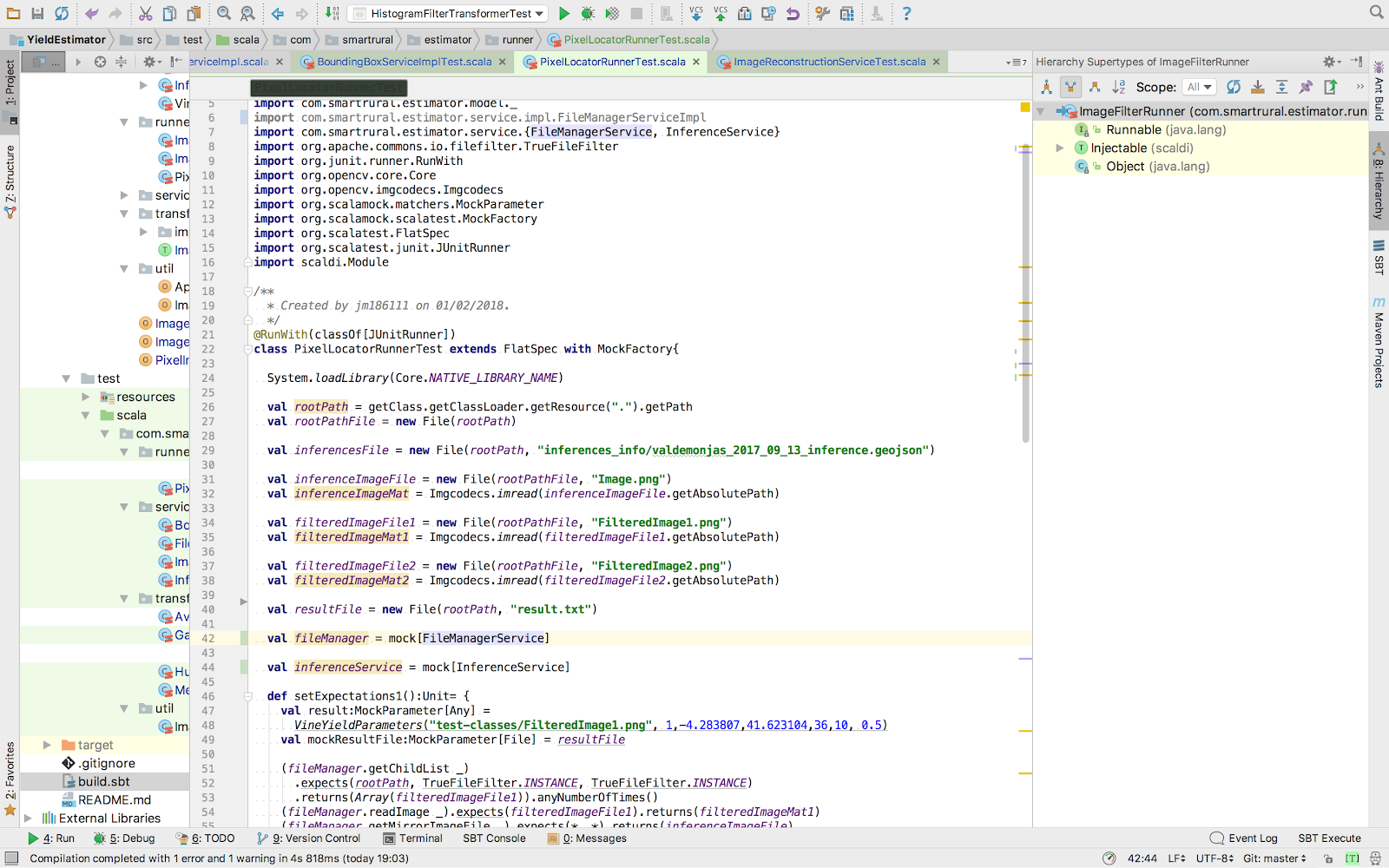


Figure 12: Intellij IDEA development environment

### SBT

SBT (<https://www.scala-sbt.org/>) is an interactive build tool for java and scala developed and maintained by Lightbend Inc. This tool provides comprehensive functionality to create build packages, run automated tests and perform dependency management. IntelliJ IDEA provides a visual plugin to seamlessly integrate SBT within the IDE, displaying buttons to access the functionality provided.

# Results and Evaluation

Data sanitization & selection

The set of images available contains unsanitized or irrelevant data that had not been used in the final artifact evaluation. The reason is that the autonomous vehicle used in the experiment, takes pictures at regular intervals (1 m) no matter it is travelling within vine rows or in a path where there is no plants at all. In order to extract the final dataset used for the evaluation of the model, a manual selection step had to be performed to decide the set of images that would be used in the final evaluation. The selection process is requiring the help of a visual web portal named Tileo (<http://tileo.co>) to assist on the identification of the images. Tileo is a platform that helps to visualize geolocalized data using google maps and enables analytics over the selected data. A capture of the tool can be seen below:

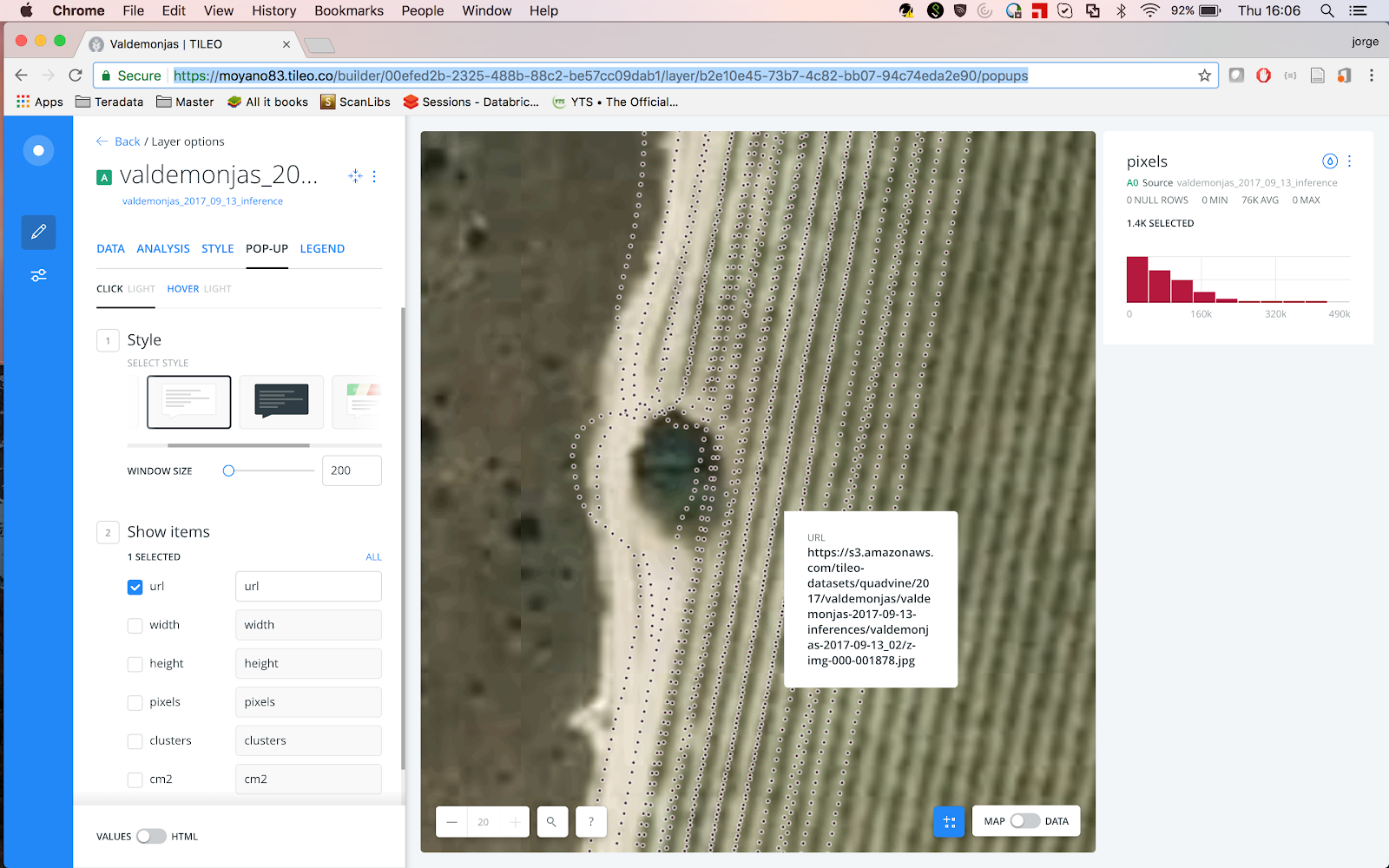


Figure 13: Tileo visualization tool

The set of images displayed in the map (each one represented by a dot in the picture above) have to be correlated with the available yield data captured by Valdemonjas winery.

The camera was mounted on the right side of the autonomous vehicle and each of the vine rows was photographed from both of its sides, (south to north and north to south) to maximize the grape detection and mitigate the possible occlusion of the clusters. The recognition process discards one every two images to avoid double counting of grapes due to overlapping of the images. The names of the pictures taken are sequential in the form <Date>/z-img-000-<Sequence-number>.jpg.

To identify the pictures that belongs to a vine row, it is necessary to determine the start and the end of the sequence of pictures as well as the direction of the vehicle (the vehicle passes twice in between the vine rows) with the help of Tileo. Clicking on the dots at the beginning of a vine row displays a popup with the name of the image that would be used as either the beginning or the end of the sequence of images. Doing the same at the end of the vine row gives you the boundaries of the picture names that would be considered for the estimation.

There is a total of 30 vine rows that were photographed from both sides, the rest of the rows with partial information were discarded. The yield information available for the study is not atomized by plant but by rows, therefore for each of the rows, a sum of the total number of pixels belonging to clusters and the average value for the occlusion factor was computed. The final dataset used for the artifact evaluation consists of 30 rows with 4 columns: Vine row identifier, pixel count, average occlusion and total yield. This corresponds approximately to a total of 1.44 hectares with more than 4100 plants.

Evaluation

With the information described in the previous step, a linear regression was performed using the pixel information and the average occlusion factor. Another linear regression was also performed using the pixel information only for comparison. Below is the python script used in the calculations:

|  |
| --- |
| **import pandas as pd**  **from sklearn import datasets, linear\_model**  **from sklearn.model\_selection import train\_test\_split**  **from sklearn.metrics import mean\_squared\_error as mse, r2\_score as r2, median\_absolute\_error as mad**  **import matplotlib.pyplot as plt**  **import sys**  ***# Loads the dataset passed by parameter when the script is called***  **df = pd.read\_csv(sys.argv[1])**  ***# Columns to use in the prediction***  **x = df[['nr\_pixels', 'occlusion']]**  ***# Target value***  **y = df['weight']**  ***# The dataset is divided in training (70%) and test (30%) using random selection***  **x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.33, random\_state=42)**  **reg = linear\_model.LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=1, normalize=False)**  **reg.fit(x\_train, y\_train)**  ***# Prediction of the weight (w)***  **w = reg.predict(x\_test)**  ***# Explained variance score: 1 is perfect prediction, the mean squared error and median absolute error***  **print('Variance=[%.2f], MSE=[%.2f], MAD=[%.2f]' % (r2(w,y\_test), mse(w, y\_test), mad(w, y\_test)))**  ***# Plotting the results***  **fig, ax = plt.subplots()**  **ax.scatter(y\_test, w, edgecolors=(0, 0, 0))**  **ax.plot([y\_test.min(), y\_test.max()], [w.min(), w.max()], '--', lw=1)**  **ax.set\_xlabel('Measured')**  **ax.set\_ylabel('Predicted')**  **plt.show()** |

Table 17: Prediction script

The script used a function to split the original dataset into training and test using a randomize function to separate the data points. The training dataset is used then to fit the model, which is later tested with the test dataset. The measures used to evaluate the model are described in the following sections.

### Coefficient of determination

Denoted by , is a measure of the quality of the predictions the model can yield over future samples and measures the ratio of the explained variation between the total variation. The perfect model would give a value for of 1. The value is calculated given the following equation:

where:

: True measured value

: Predicted value

: Mean of the measured values

### Mean Squared Error

The mean squared is a commonly used error measure that provides a measure of the quality of an estimation by computing the average of the squared errors. The greater the MSE the less accurate the prediction is:

where:

n: Number of measures

### Median Absolute Deviation

The median absolute deviation provides a measure of how disperse the data is. It is computed using the formula. It can also be used to find outliers that affects negatively the prediction power of the proposed model:

Results

Below are presented the graphics and results obtained for both the estimation using only the pixel count and using the pixel count plus the occlusion factor.

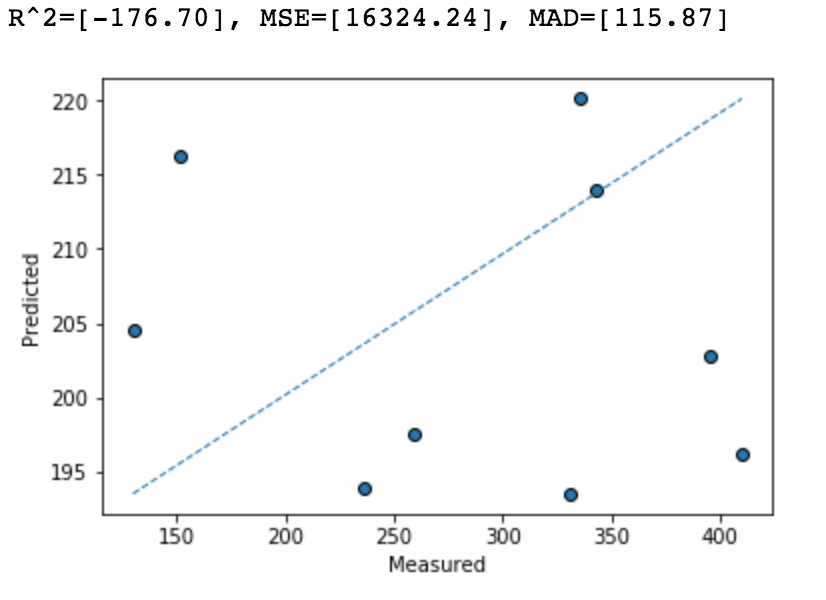
****

Table 18: Estimation using Pixel count

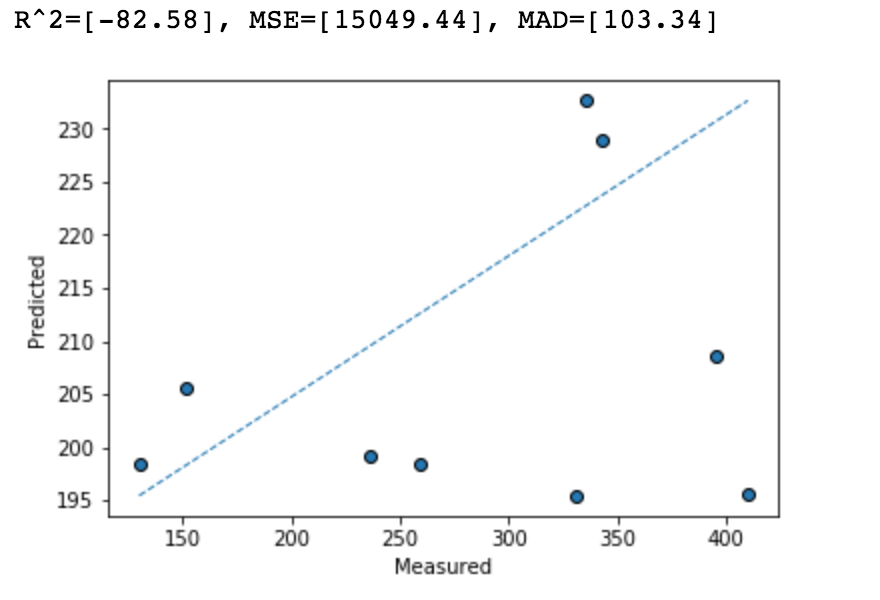
****

Table 19: Estimation using Pixel count and occlusion factor

As it can be seen, the , and performs better using the occlusion factor, reducing the error significantly, which can indicate that there exists a relation between the detected occluded factor and the final weight yield measured. However, the fact that there is a negative , indicates that the model performs worst than the horizontal line (null hypothesis), indicating that the model has no predictive value. As we know this is in fact not true because the number of pixels does have a relation with the detected weight as proved by Diago *et al.* (2012), Hung *et al.* (2015), Lopes *et al.* (2016) or Nuske *et al.* (2011) among others, the problem must be in the low number of data point used as well as the possible errors introduced both in the measurement techniques and the data categorization, providing an unreliable dataset to perform the study.

# Conclusions

Lessons Learned

The original design of the solution proposed was based on the leaf detection algorithm developed by Berenstein *et al.*, (2010), however as indicated before, this algorithm yield poor results due to the difficulty of setting the right boundaries for the channels (even after transforming the image to HSV), as well as the presence of noise pixels in areas belonging to leaves. The histogram back-projection algorithm used was more suitable for the task of leaf identification, but the sample image utilised as reference had to be small and it has to be constrained to an inner area of a single leaf to output acceptable results.

The iterative solutions described in 3.4 where not suitable for a large-scale image transformation solution. The computational time needed to transform the set of 7043 images available in a single machine needed more than a month to process all the images, making it unsuitable as a general solution. The OpenCV library used in the final artefact takes advantage of the multiple cores available in modern computers and is hardware-optimized, although the installation of the underlying software and the configuration of it to be used in the selected IDE is cumbersome.

Using aggregated results to evaluate the performance of the artefact has demonstrated to be ineffective, hiding the possible gains of inspecting individual clusters and apply the occlusion factor to them, although weighting the grape yield of individual plants during the harvesting is impractical. A solution to this problem could be to generate a model based on samples gathered for this only purpose and apply this model to the whole vineyard under study.

Academic Applications

The present study showed a new technique to provide a grape yield estimation using grape pixel count and a newly calculated occlusion factor based on the detection of leaves in the surrounding of the clusters. The artefact developed provided a reference architecture for image processing and filtering which is performant as it uses less disk space and I/O time due to the in-memory computation nature of the solution.



Figure 14: Picture matching the highest occlusion value found

This architecture can also be used in distributed computing as the non-dependant nature of the data used makes it a good candidate for parallelization and multithreading.

Business Application

There is an obvious reason to desire to know the estimated yield of a vineyard plot, as it helps the farmers to plan in advance resource utilization, staff augmentation and even plan the industrial processes required to make wine.

From a managerial perspective, Valdemonjas S.A. has been implementing nocturnal harvesting to collect the grapes at a lower temperature (between 3º to 15º C) to improve quality and have a controlled fermentation process (Europapress, 2011). The occlusion factor calculated here can be used to plan in advance which areas or plants can be harvested at later times of the day, because partially occluded clusters would be more resilient to overheat due to the effect of direct sunlight that completely uncovered ones.

Recommendations / Prospects for Future Research

The present work showed a slight improvement on the results of the model created to predict grape yield, although the dispersion of the data showed a very weak correlation and thus unsuitable to extract definitive conclusions. The algorithm created to filter leaf elements on the pictures can be improved with the use of RGB-D cameras, that provides information about the depth of the elements in the picture. With this information, it will be possible to eliminate the leaves that appears surrounding the cluster but does not occlude it as they are behind it (which the current algorithm does not differentiate).

With this setup it would also be possible to detect other elements that might occlude the clusters such as dry leaves which are not detected by the current algorithm due to the change of color in the leaves.



Figure 15: Picture matching the lowest occlusion value found

It would also be necessary to gather more data points to validate the model with feasible data by plant, as individual values provide better insight of the performance of the model than average values of weight and occlusion.

REFERENCES CITED

Akhand, K., Nizamuddin, M., Roytman, L., Kogan, F. (2016) ‘Using remote sensing satellite data and artificial neural network for prediction of potato yield in Bangladesh’, in Proceedings of SPIE - The International Society for Optical Engineering. Proceedings of SPIE - The International Society for Optical Engineering, SPIE. doi: [10.1117/12.2237214](https://doi.org/10.1117/12.2237214).

Aquino, A., Diago M. P., Millán, B., Tardáguila, J. (2017) ‘A new methodology for estimating the grapevine-berry number per cluster using image analysis’, Biosystems Engineering, 156(Supplement C), pp. 80–95. doi: [10.1016/j.biosystemseng.2016.12.011](https://doi.org/10.1016/j.biosystemseng.2016.12.011).

Aquino, A., Millán, B., Gutiérrez, S., Tardáguila, J. (2015) ‘Grapevine flower estimation by applying artificial vision techniques on images with uncontrolled scene and multi-model analysis’, Computers and Electronics in Agriculture, 119, pp. 92–104. doi: [10.1016/j.compag.2015.10.009](https://doi.org/10.1016/j.compag.2015.10.009).

Arivazhagan, S., Shebiah, S., Ananthi, S., Varthini, S. (2013) ‘Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features’, Agricultural Engineering International: CIGR Journal, 15(1), pp. 211–217.

Bargoti, S. and Underwood, J. P. (2017) ‘Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards.’, Journal of Field Robotics, 34(6), pp. 1039–1060.

Berenstein, R., Edan, Y., Shahar, O.B., Shapiro, A., (2010) ‘Grape clusters and foliage detection algorithms for autonomous selective vineyard sprayer’, Intelligent Service Robotics, 3(4), pp. 233–243. doi: [10.1007/s11370-010-0078-z](https://doi.org/10.1007/s11370-010-0078-z).

Biswas, S. K. and Mia, M. M. A. (2015) ‘Image Reconstruction Using Multi Layer Perceptron (MLP) And Support Vector Machine (SVM) Classifier And Study Of Classification Accuracy’, International Journal of Scientific & Technology Research, 4(2), pp. 226–231.

Chamelat, R., Rosso, R., Choksuriwong, A., Rosenberger, C., Laurent, J., Bro, P. (2006) ‘Grape Detection By Image Processing’, IECON 2006 - 32nd Annual Conference on IEEE Industrial Electronics, IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on, p. 3697. doi: [10.1109/IECON.2006.347704](https://doi.org/10.1109/IECON.2006.347704).

Clingeleffer, P., Dunn, G., Krstic, M., & Martin, S. (2001). Crop development, crop estimation and crop control to secure quality and production of major wine grape varieties: A national approach. Technical report, Grape and Wine Re- search and Development Corporation, Australia.

Cubero, S., Diago, M. P., Blasco, J., Tardáguila, J., Prats-Montalbán, J. M. (2015) ‘A new method for assessment of bunch compactness using automated image analysis.’, Australian Journal of Grape & Wine Research, 21(1), p. 101.

de la Fuente, M., Linares, R., Baeza, P., Miranda, C., Lissarrague, J. R. (2015) ‘Comparison of different methods of grapevine yield prediction in the time window between fruitset and veraison’, OENO One, 49(1), pp. 27–35.

Debadeepta, D., Lily, M. and Rahul, S. (2012) ‘Classification of plant structures from uncalibrated image sequences’, 2012 IEEE Workshop on the Applications of Computer Vision (WACV), Applications of Computer Vision (WACV), 2012 IEEE Workshop on, p. 329. doi: [10.1109/WACV.2012.6163017](https://doi.org/10.1109/WACV.2012.6163017).

Diago, M. P., Correa, C., Millán, B., Barreiro, P. Valero, C., Tardáguila, J., (2012) ‘Grapevine Yield and Leaf Area Estimation Using Supervised Classification Methodology on RGB Images Taken under Field Conditions.’, Sensors (14248220), 12(12), pp. 16988–17006.

Dunn, G. M. and Martin, S. R. (2004) ‘Yield prediction from digital image analysis: A technique with potential for vineyard assessments prior to harvest’, Australian Journal of Grape and Wine Research, 10(3), pp. 196–198.

Europapress (23/09/2011) ‘La bodega Lynus de Ribera de Duero realiza por primera vez la vendimia nocturna para garantizar calidad y sostenibilidad’ [online, accessed 01/04/2018] available at:

<http://www.europapress.es/castilla-y-leon/noticia-bodega-lynus-ribera-duero-realiza-primera-vez-vendimia-nocturna-garantizar-calidad-sostenibilidad-20110923150743.html>

Fernández, R., Montes, H., Salinas, C., Sarria, J., Armada, M. (2013) ‘Combination of RGB and multispectral imagery for discrimination of cabernet sauvignon grapevine elements.’, Sensors (Basel, Switzerland), 13(6), pp. 7838–7859. doi: [10.3390/s130607838](https://doi.org/10.3390/s130607838).

Herrero-Huerta, M., González-Aguilera, D., Rodriguez-Gonzalvez, P., Hernández-López, D. (2015) ‘Vineyard yield estimation by automatic 3D bunch modelling in field conditions’, Computers and Electronics in Agriculture, 110(Supplement C), pp. 17–26. doi: [10.1016/j.compag.2014.10.003](https://doi.org/10.1016/j.compag.2014.10.003).

Hung, C. Underwood C., Nieto, J., Sukkarieh, S. (2015) A feature learning based approach for automated fruit yield estimation. Springer Verlag (Springer Tracts in Advanced Robotics). doi: [10.1007/978-3-319-07488-7\_33](https://doi.org/10.1007/978-3-319-07488-7_33).

Ivorra, E., Sánchez, A. J., Camarasa, J. G., Diago, M. P., Tardaguila, J. (2015) ‘Assessment of grape cluster yield components based on 3D descriptors using stereo vision’, Food Control, 50(Supplement C), pp. 273–282. doi: [10.1016/j.foodcont.2014.09.004](https://doi.org/10.1016/j.foodcont.2014.09.004).

Kicherer A., Herzog K. and Töpfer R. (2015) ‘High-throughput phenotyping for trait detection in vineyards’, BIO Web of Conferences, Vol 5, p 01018 (2015), p. 01018. doi: [10.1051/bioconf/20150501018](https://doi.org/10.1051/bioconf/20150501018).

Kushwaha, A. K. and Bhattachrya, S. (2015) ‘Crop yield prediction using Agro Algorithm in Hadoop’, International Journal of Computer Science and Information Technology & Security (IJCSITS), 5.

Liu, S., Marden, S. and Whitty, M. (2013) ‘Towards automated yield estimation in viticulture’, in Proceedings of the Australasian Conference on Robotics and Automation, Sydney, Australia.

Liu, S. and Whitty, M. (2015) ‘Automatic grape bunch detection in vineyards with an SVM classifier’, Journal of Applied Logic, 13(Part 3), pp. 643–653.

Lopes, C., Graça, J., Sastre, J., Reyes, M., Guzman, R. (2016) ‘Vineyard yield estimation by vinbot robot -preliminary results with the white variety viosinho’. doi: [10.13140/RG.2.1.3912.0886](https://doi.org/10.13140/RG.2.1.3912.0886).

Mack, J., Lenz, C., Teutrine, J., Steinhage, V. (2017) ‘High-precision 3D detection and reconstruction of grapes from laser range data for efficient phenotyping based on supervised learning’, Computers and Electronics in Agriculture, 135, pp. 300–311. doi: [10.1016/j.compag.2017.02.017](https://doi.org/10.1016/j.compag.2017.02.017).

Murillo-Bracamontes, E. A., Martinez-Rosas, M., Miranda-Velasco, M., Martinez-Reyes, H., Martinez-Sandoval, J., Cervantes-de-Avila, H. (2012) ‘Implementation of Hough transform for fruit image segmentation’, Procedia Engineering, 35, pp. 230–239.

Nuske, S., Achar, S., Bates, T., Narasimhan, S., Singh, S. (2011) ‘Yield estimation in vineyards by visual grape detection’, in. IEEE, pp. 2352–2358. doi: [10.1109/IROS.2011.6095069](https://doi.org/10.1109/IROS.2011.6095069).

Nuske, S., Gupta, K., Narasimhan, S., Singh, S. (2014) ‘Modeling and Calibrating Visual Yield Estimates in Vineyards’, in Yoshida, K. and Tadokoro, S. (eds) Field and Service Robotics. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 343–356. doi: [10.1007/978-3-642-40686-7\_23](https://doi.org/10.1007/978-3-642-40686-7_23).

Nuske, S., Wilshusen, K., Achar, S., Yoder, L., Narasimhan, S., Singh, S. (2014) ‘Automated Visual Yield Estimation in Vineyards: Automated Visual Yield Estimation’, Journal of Field Robotics, 31(5), pp. 837–860. doi: [10.1002/rob.21541](https://doi.org/10.1002/rob.21541).

OpenCV (n.d.) ‘Mat class reference’ Online, accessed [04/03/2018] available at: <https://docs.opencv.org/3.4.0/d3/d63/classcv_1_1Mat.html>

Payne, A. B., Walsh, A. B., Subedi, P. P., Jarvis, D. (2013) ‘Estimation of mango crop yield using image analysis – Segmentation method’, Computers and Electronics in Agriculture, 91(Supplement C), pp. 57–64. doi: [10.1016/j.compag.2012.11.009](https://doi.org/10.1016/j.compag.2012.11.009).

Pérez, D. S., Bromberg, F. and Diaz, C. A. (2017) ‘Image classification for detection of winter grapevine buds in natural conditions using scale-invariant features transform, bag of features and support vector machines’, Computers and Electronics in Agriculture, 135, pp. 81–95.

Pothen, Z. S. and Nuske, S. (2016) ‘Texture-based fruit detection via images using the smooth patterns on the fruit’, in Proceedings - IEEE International Conference on Robotics and Automation. Proceedings - IEEE International Conference on Robotics and Automation, Institute of Electrical and Electronics Engineers Inc., pp. 5171–5176. doi: [10.1109/ICRA.2016.7487722](https://doi.org/10.1109/ICRA.2016.7487722).

Pujari, J. D., Yakkundimath, R. and Byadgi, A. S. (2015) ‘Image Processing Based Detection of Fungal Diseases in Plants’, Proceedings of the International Conference on Information and Communication Technologies, ICICT 2014, 3-5 December 2014 at Bolgatty Palace & Island Resort, Kochi, India, 46(Supplement C), pp. 1802–1808. doi: [10.1016/j.procs.2015.02.137](https://doi.org/10.1016/j.procs.2015.02.137).

Rahman, A. and Hellicar, A. (2014) ‘Identification of mature grape bunches using image processing and computational intelligence methods’, 2014 IEEE Symposium on Computational Intelligence for Multimedia, Signal and Vision Processing (CIMSIVP), Computational Intelligence for Multimedia, Signal and Vision Processing (CIMSIVP), 2014 IEEE Symposium on, p. 1. doi: [10.1109/CIMSIVP.2014.7013272](https://doi.org/10.1109/CIMSIVP.2014.7013272).

Rey-Caramés, C., Diago, M. P., Martín, M. P., Lobo, A., Tardaguila, J. (2015) ‘Using RPAS Multi-Spectral Imagery to Characterise Vigour, Leaf Development, Yield Components and Berry Composition Variability within a Vineyard’, Remote Sensing, Vol 7, Iss 11, Pp 14458-14481 (2015), (11), p. 14458. doi: [10.3390/rs71114458](https://doi.org/10.3390/rs71114458).

Roscher, R., Herzog, K., Kunkel, A., Kicherer, A., Töpfer, R., Förstner, W. (2014) ‘Automated image analysis framework for high-throughput determination of grapevine berry sizes using conditional random fields’, Computers and Electronics in Agriculture, 100, pp. 148–158.

Serrano, E., Roussel, S., Gontier, L., Dufourcq, T. (2005) ‘Early estimation of vineyard yield: Correlation between the volume of a vitis vinifera bunch during its growth and its weight at harvest’, FRUTIC, 5, pp. 311–318.

Shakoor, T., Rahman, K., Rayta, S., Chakrabarty, A. (2017) ‘Agricultural production output prediction using Supervised Machine Learning techniques’, 2017 1st International Conference on Next Generation Computing Applications (NextComp), Next Generation Computing Applications (NextComp), 2017 1st International Conference on, p. 182. doi: [10.1109/NEXTCOMP.2017.8016196](https://doi.org/10.1109/NEXTCOMP.2017.8016196).

Singh, V. and Misra, A. K. (2017) ‘Detection of plant leaf diseases using image segmentation and soft computing techniques’, Information Processing in Agriculture, 4(1), pp. 41–49. doi: [10.1016/j.inpa.2016.10.005](https://doi.org/10.1016/j.inpa.2016.10.005).

Swain, M. J. and Ballard, D. H. (1990) ‘Indexing via color histograms’, Proceedings Third International Conference on Computer Vision, Computer Vision, p. 390. doi: [10.1109/ICCV.1990.139558](https://doi.org/10.1109/ICCV.1990.139558).

Szeliski, R. (2011) ‘Computer Vision’, London: Springer London (Texts in Computer Science). doi: [10.1007/978-1-84882-935-0](https://doi.org/10.1007/978-1-84882-935-0).

Taylor, J., Tisseyre, B., Bramley, R., Reid, A., Stafford, J. (2005) ‘A comparison of the spatial variability of vineyard yield in European and Australian production systems’, Precision agriculture, 5, pp. 907–914.

Whalley, J. L. and Shanmuganathan, S. (2013) ‘Applications of image processing in viticulture: A review’.

Zhao, T. ( 1 ) et al. (2017) ‘Melon yield prediction using small unmanned aerial vehicles’, in Proceedings of SPIE - The International Society for Optical Engineering. Proceedings of SPIE - The International Society for Optical Engineering, SPIE. doi: [10.1117/12.2262412](https://doi.org/10.1117/12.2262412).

APPENDICES

###### Computing Project Proposal

**Version 1 – December 2017**

**Student's Name**: Jorge Moyano

**Student's Number**: H00041124

**Student's Email Address**: jorge.moyano@online.liverpool.ac.uk

**Project Title**: A Framework to calculate wine production based on high resolution images, a case study in Ribera del Duero.

**Dissertation Deadline**: 1 August 2018

**Proposal Submission Date**: 25 November 2017

**Version Number of the Proposal**: 1

**DA Class ID**: CKIT-702-2-H00023684

**Name of DA**: Taly Sharon  
**RMT Class ID**: LAUR-906-1 - Computing Research Methods Training (Start **Date**: 26 October 2017)

**Name of GDI**: Taly Sharon

**Ethics Response Form completed**: Yes

**The Programme**: MSc in Web Sciences and Big Data

**Domain**:

|  |  |
| --- | --- |
| CKIT-525-1 | Big Data |
| CKIT-504-1 | Designing and Managing Databases |
| CKIT-503-1 | Programming the Internet |

**Sponsor's Details:**

SmartRural, Camino Viejo de Simancas, 4.5 km INEA Universidad Ingenieros Técnicos Agrícolas 47008 Valladolid, Spain.

Bodegas Valdemonjas, Carretera de Soria N-122, Km 322, 47360 Quintanilla de Arriba, Valladolid, Spain.

**Sponsor's Background:**

Smartrural is a company dedicated to improving the efficiency and cost effectiveness of farming operations using TIC. My contact here Sergio Rodriguez Gonzalez is an agronomist engineer that created the company.

‘Bodegas Valdemonjas’ winery in the Ribera del Duero denomination of origin is a small winery that produces red wine following organic farming principles. My contact here Alejandro Moyano is an agronomist engineer that created the company.

**Sponsor's Agreement:**

The sponsors have granted permission and availability in order to make this project viable.

***The Project Aims and Objectives:***

* That it is possible to calculate and predict the grape harvesting volume using data mining over the sensors gathered data and processed high resolution pictures.
* That is possible to improve the vineyard management by using machine learning algorithms created from provided historical data
* Production of an IT artefact that would be evaluated using the device gathered data and validated against the final data.

***In the table below, please state your hypothesis or hypotheses; the research methods you will use to guide the development of your IT artefact; the kind of IT artefact you will produce; and the means by which you will evaluate the IT artefact in the light of the hypothesis.***

|  |  |
| --- | --- |
| **Step** | **Short Description** |
| Hypothesis | A framework to predict vineyard yield based on the data obtained after processing high resolution images of vine clusters in Ribera del Duero (Spain) on the tempranillo variety, using data mining techniques.  This framework can:   * Predict the harvesting yield using data mining over the sensors gathered data and high-resolution pictures. * Improve the vineyard management by using machine learning models. * Provide new inputs for visual model representation of the agriculture data. |
| Research Methods | Computer simulation with high resolution picture data, weather measures and soil composition data. Literature research |
| IT Artefact | An artefact to predict the yield of the vineyard, test and verification suite. |
| Evaluation | Case study with simulation models with the gathered data as well as the use of historical data to test the results. |

**Project Outline:**

The literature survey will result in a categorization and feasibility of the distinct techniques used to predict yield estimations. The aim is to evaluate the different techniques and the most suitable ones to perform segmentation and clustering of the areas of the vineyard.

In the analysis and design phase the architecture, the approach to implement the decided algorithm would be decided, the tests cases and the validation scenarios would be defined. The artefact code would be produced.

In the evaluation phase, the artefact produced would be tested using the training, validation and test data.

Finally, the report would be written and it will contain the findings yielded from the evaluation phase, conclusions would be inferred from the results.

**Literature Survey / Resources’ List:**

Nuske et al. (2011) proposes an estimation based on linear regression of total berry count and actual crop weight.

Nuske et al. (2014) makes improvements of the yield estimation by using an algorithm to produce a yield count estimation function, and the result is multiplied by the average berry weight.

Herrero et al. (2015) uses a similar approach, which calculates the total yield as a product of number of berries multiplied by the average of the volume of berries and the average density of the bunches.

In the same line, Liu, Marden and Whitty (2013) used linear regression to estimate weight using berry number, bunch perimeter, bunch area and estimated bunch volume.

Lopes et al. (2016) uses the pixel count of the grape cluster surface in the pictures and applied a linear regression between the cluster surface and the cluster weight for the viosinho variety.

Akhand et al. (2016) used artificial neural network to develop a prediction model using nonlinear auto regressive with external input to predict potato yield in Bangladesh using satellite images.

Shakoor et al. (2017) describes various machine learning algorithms such as decission trees, supervised machine learning and iterative dichotomiser 3 to predict the yield production of different rice types, potato, jute and wheat crops.

**References**

Akhand k., Nizamuddin M., Roytman L. & Kogan F.

(2016) ‘Using remote sensing satellite data and artificial neural network for prediction of potato yield in Bangladesh’, in Proceedings of SPIE - The International Society for Optical Engineering. Proceedings of SPIE - The International Society for Optical Engineering, SPIE. doi: 10.1117/12.2237214.

Herrero-Huerta, M. González-Aguilera, D., Rodriguez Gonzalvez, P., Hernández-López, D., (2015) Vineyard yield estimation by automatic 3D bunch modelling in field conditions. doi: 10.1016/j.compag.2014.10.003.

Nuske, S., Achar, S., Bates, T., Narasimhan, S. and Singh, S., (2011) ‘Yield estimation in vineyards by visual grape detection’, in. IEEE, pp. 2352–2358. doi: 10.1109/IROS.2011.6095069.

Nuske, S., Wilshusen, K., Achar, S., Yoder, L., Narasimhan, S. and Singh, S., (2014) ‘Automated Visual Yield Estimation in Vineyards: Automated Visual Yield Estimation’, Journal of Field Robotics, 31(5), pp. 837–860. doi: 10.1002/rob.21541.

Liu, S., Marden, S. and Whitty, M., (2013) Towards automated yield estimation in viticulture. In Proceedings of the Australasian Conference on Robotics and Automation, Sydney, Australia (Vol. 24).

Lopes, C., Graça, J., Sastre, J., Reyes, M., Guzman, R., & Braga, R., Monteiro, A., Pinto, P., (2016) ‘Vineyard yield estimation by vinbot robot -preliminary results with the white variety viosinho’. Unpublished, p. doi: 10.13140/RG.2.1.3912.0886.

Shakoor T., Rahman K., Rayta S. & Chakrabarty A. (2017) ‘Agricultural production output prediction using Supervised Machine Learning techniques’, 2017 1st International Conference on Next Generation Computing Applications (NextComp), Next Generation Computing Applications (NextComp), 2017 1st International Conference on, p. 182. doi: 10.1109/NEXTCOMP.2017.8016196.

**Scholarly Contributions of the Project:** Evaluation of data mining methods to the problem of estimation of vine yield for the tempranillo variety in Ribera del Duero. The produced framework will use processed high-resolution image data taken with quads and drones to estimate the grape production yield. This work would improve the existing algorithms used for yield prediction using pixel data.

**Description of the Deliverables:**

At the end of this research project the following elements would be produced:

* Research dissertation: This document would content the description of the project as well as the test results and statistical analysis generated for the proposed solution
* An IT artefact modelling the problem.

**Evaluation Criteria:**Project research quality will also be ensured by the proposed literature and bibliography.

The mentioned winery would be used as a case example as it has detailed data about production atomized by zones and the owner has given permission to use his data. Multiple detailed harvesting data that can be used to train and validate the models. The model is expected to improve the accuracy of the existing methods. Using the work of Lopes, Graça, Sastre, Reyes, Guzman, Braga, Monteiro and Pinto (2016) as a baseline, the algorithm would be considered to be successful if it improves the 5.7% of error rate obtained by these researchers.

As stated before, the rule of thumb of 70 percent of the data would be used to train the model, 15 percent would be use to validate and 15 percent would be used to test it.

**Resource Plan:**

To complete this project, I will use my personal computer, and data generated by the drones, quads and other sensors by SmartRural for Bodegas Valdemonjas S.A. which will be use as a use case to extrapolate the results across the Rivera del Duero wine region in Spain. The project does not need additional external resources apart of the mentioned process data. Only free software like R commander, scala and Intellij IDE with GNU licensing will be used.

**Project Plan and Timing:**

The project would be divided in milestones which duration is estimated in about two months each, described below:

|  |  |
| --- | --- |
| **Phase 1 – Literature survey and review** | |
| December 10th-January 31st, 2018 | * Literature survey completed * Milestone: Literature survey chapter report completed and handed for review |
| **Phase 2 - Analyses/design of the solution. Implementation** | |
| February 1st - April 5th, 2018 | * Artefact design * Software technology evaluation * Artefact implementation * Milestone: Specification and design chapter handed for review. |
| **Phase 3 – Testing & Evaluation** | |
| April 16th - June 1st, 2018 | * Tests conducted * Project evaluation done * Milestone: Solution Evaluation chapter with the test reports handed for review |
| **Phase 4 – DS writing** | |
| June 2nd - June 30th, 2018 | * Compilation of the chapters, Abstract and conclusion chapters delivered for review * Milestone: First draft of the dissertation delivered for comments |
| **Phase 5 – Final DS Submission** | |
| July 1st – August 1st, 2018 | * Correct the dissertation after the DA comments * Milestone: Project deadline - Final dissertation delivered |

**Risk Assessment:**

1. Lack of availability of the sponsor: The sponsor of the project is a necessary actor as is the one that holds the data that I’ll use to create and validate the models for the project. The contingence plan is to get this data as soon as possible so I won’t be dependent on them after this.
2. Not being able to come up with an algorithm to calculate the volume of the harvesting of grapes based on the image recognition due to lack of statistical knowledge: As a fall-back plan, I’ll compare the different existing techniques and benchmark them with the gathered data.
3. Lack of software/licensing or hardware: To avoid costs only free software (Spark, SparkMlib, personal computer, google drive). All the files of my project would be stored in the cloud.

**Quality Assurance:**

The following points describe the steps that will be taken to ensure project quality.

* The DA will evaluate the validity of the hypothesis and the artefact produced.
* Ensure deadlines and intermediate deliverables with the DA.
* Validation of the artefact performance and prediction capabilities with final data collected at the farm that serves as a case study.

###### Aggregated Results For Yield Estimation

The following table contains the aggregated results used to calculate the yield estimation shown on section 5. The “Occlussion percentage” displayed is an average value for the identified row.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Row id | Number pixels | Number cluster | Occlusion percentage | Measured weight |
| N3 | 6555608 | 9.96969697 | 0.141848485 | 141 |
| N4 | 6471877 | 9.953125 | 0.156109375 | 128 |
| N5 | 6941297 | 9.526315789 | 0.162473684 | 103 |
| N6 | 8107822 | 10.38028169 | 0.187859155 | 99 |
| N7 | 9802453 | 10.62195122 | 0.173536585 | 124 |
| N8 | 10149083 | 10.35353535 | 0.182474747 | 128 |
| N9 | 10501477 | 9.211009174 | 0.159449541 | 155 |
| N10 | 15307038 | 9.714285714 | 0.175328571 | 243 |
| N11 | 20858068 | 10.07106599 | 0.18084264 | 259 |
| N12 | 19722884 | 9.922680412 | 0.17656701 | 410 |
| N13 | 24206155 | 9.495575221 | 0.189924779 | 298 |
| N14 | 28517054 | 9.488549618 | 0.216148855 | 215 |
| N15 | 38643102 | 10.62111801 | 0.195664596 | 395 |
| N16 | 31911005 | 9.961672474 | 0.195825784 | 370 |
| N17 | 27663504 | 9.648854962 | 0.190770992 | 370 |
| N20 | 25744286 | 9.647727273 | 0.181465909 | 331 |
| S5 | 11307603 | 12.13043478 | 0.194576087 | 117 |
| S6 | 12193422 | 10.80357143 | 0.167321429 | 130 |
| S7 | 13277636 | 11.96226415 | 0.208698113 | 130 |
| S8 | 11582626 | 10.88571429 | 0.187742857 | 117 |
| S9 | 11130426 | 10.24778761 | 0.189424779 | 96 |
| S10 | 12719501 | 10.93382353 | 0.169198529 | 140 |
| S11 | 15847776 | 11.88965517 | 0.201531034 | 152 |
| S12 | 13618913 | 9.682432432 | 0.190047297 | 236 |
| S13 | 22886309 | 12.03448276 | 0.162408867 | 152 |
| S14 | 38453296 | 12.34519573 | 0.190647687 | 369 |
| S15 | 34336545 | 11.57044674 | 0.163910653 | 401 |
| S16 | 36400287 | 12.44329897 | 0.221343643 | 336 |
| S17 | 35008924 | 11.79136691 | 0.224251799 | 343 |
| S20 | 33548165 | 11.46099291 | 0.168595745 | 350 |

###### Code Repository

The code displayed here is publicly accessible and can be found in the following GitHub repository:

<https://github.com/moyano83/YieldEstimator>

The mentioned repo was last accessed on 22/04/2018