



NTNU – Trondheim
Norwegian University of
Science and Technology

Automatic visual Weed Recognition

Detection and Classification of Weed in Row

Cultures combining Machine Vision and

Artificial Intelligence

Øystein Grændsen

Master of Science in Cybernetics and Robotics

Submission date: June 2014

Supervisor: Jan Tommy Gravdahl, ITK

Co-supervisor: Trygve Utstumo, Adigo AS

Norwegian University of Science and Technology
Department of Engineering Cybernetics



Master project TTK4900

Student: Øystein William Grændsen
Program: M.Sc. Engineering Cybernetics

Title: **Automatic visual weed recognition**
Detection and classification of weed in row cultures combining machine vision and artificial intelligence

Adigo is developing an autonomous robot for weed control in row crops, Asterix. A vision system classifies weeds and crop, which form the basis for a spray map. The microsprayer target each weed avoiding soil and crop. The robot is set up with differential steering, and a camera enabling visual odometry to complement the navigation system.

Assignment:

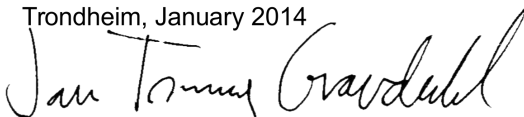
- Perform a literature study of earlier work in classification of weed in row crops
- Plan an architectural design of the Asterix project
- Plan an architectural design of the classifier and the training software by UML
- Implement the computer vision components for object segmentation, connected components analysis and feature extraction.
- Implement a GUI for creating training sets.
- Investigate different options of classification
- Use different classifiers and evaluate their performance based on the training sets.

Advisor: Jan Tommy Gravdahl
Professor, Dept. of Engineering Cybernetics

External advisor: Trygve Utstumo
Ph.D. candidate, Dept. of Engineering Cybernetics
MSc. Engineering Cybernetics, Adigo AS

Project assigned: January 2014
To be handed in by: June 9, 2014

Trondheim, January 2014



Jan Tommy Gravdahl
Professor, Dept. Engineering Cybernetics

Preface

This thesis concludes my years of study towards a Master of Science degree in the field of Engineering Cybernetics at the Norwegian University of Science and Technology.

First and foremost, I want give my deepest gratitude to my supervisor Jan Tommy Gravdahl. After all my strides, you always gets me back on track. I consider myself very lucky to have had you as my supervisor during the last year of my studies, thank you.

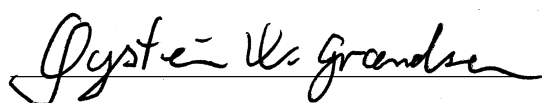
I also want to thank Trygve Utstumo, my external supervisor. Through Adigo AS you have given me interesting and challenging assignments. During the last year, and especially during the last month, you have helped me to get my work done.

During all my years in Trondheim, I always knew my family was there if I needed them. Ingrid and Kristine, my sisters, your are the best. Jan Erik and Elisabeth, my parents, I would not have been who I am today if it had not been for the two of you.

My friends during these years are the main reason for the good times in Trondheim. The friends I have made from Regi, my second home for many years. Haidar, my roomie for 5 years and my best friend for a decade, you are so important to me. And nothing would have been the same without this gang: Kristian, Hedvik, Hilde, Marianne, Signe og Ylva. You are awesome!

And finally, Ann-Kristin, thank you for who you are to me.

Trondheim, June 9, 2014

A handwritten signature in black ink, reading "Øystein W. Grændsen". The signature is written in a cursive style with a horizontal line underneath the name.

Øystein William Grændsen

Abstract

This thesis was motivated by the use of machine vision and artificial intelligence in agricultural robotics. More efficient agricultural production is needed as the world's population grows. It is also important, both from an economical and environmental point of view, to reduce the amount of applied herbicidal products in agriculture. The use of machine vision and artificial intelligence are already a part of the modern agriculture and will play an important role in the technology to come.

The aim of this thesis is to implement a program for leaf detection in row crops and investigate the use of different classifiers to detect weed. This work will form the base of the weed detection part in a bigger project, Asterix, owned by Adigo AS.

The program uses images, in this thesis from a carrot field, to detect leaves by segmentation and connected components analysis. From each leaf, ten features are extracted to be used in the classification process. By a graphical user interface, a user can label leaves into given classes to create training sets.

By the use of a resulting training set, leaf-objects were classified by six different basic classifiers. By this classification, a success rate of over 89 % total correct classified leaves was reached. Furthermore, by raising the lower boundary of the leaf size and using votes from all the classifiers to determine the class, total classification success rate surpassed 95%.

Sammendrag

Denne masteroppgaven ble skrevet for å undersøke bruk av datasyn og kunstig intelligens i landbruksrobotikk. Mer effektiv landbruksproduksjon trengs ettersom jordens befolkningen øker. Det er også viktig, både fra et økonomisk og et miljøvennelig perspektiv, å redusere bruksmengden av sprøytemidler. Datasyn og kunstig intelligens benyttes allerede i dagens jordbruk og vil ha en viktig rolle i fremtidens teknologi.

Målet med denne masteroppgaven er å implementere et program som detekterer blader i radkulturer og å undersøke bruken av forskjellige klassifikatorer for å kjenne igjen ugress. Dette arbeidet vil danne fundamentet for ugressdeteksjonsdelen i et større prosjekt, Asterix, som eies av Adigo AS.

Programmet bruker bilder til å detektere blader ved bruk av segmentering og sammenkoblede komponenters analyse¹. I denne oppgaven er dette blitt gjort i en gulrotåker. Ti attributter brukes for å beskrive hvert blad i klassifiseringsprosessen. Gjennom et grafisk brukergrensesnitt kan en bruker spesifisere hvilken klasse hvert enkelt blad tilhører. Dette gjøres for å danne et treningssett.

Ved bruken av treningssett ble blad-objekter klassifisert av seks forskjellige enkle klassifikatorer. Denne klassifiseringen førte til en suksessrate på over 89 % riktig klassifiserte blader totalt. Ved å øke grensen for hvor stort areal et blad må ha for å detekteres og innføre bruken av stemming ble en suksessrate på over 95 % nådd.

¹Connected components analysis

Contents

Preface	i
Abstract	iii
Sammendrag	v
1 Introduction	1
1.1 Agricultural robotics	2
1.2 Weed control and Precision Agriculture	3
1.3 Machine vision in agriculture	4
1.4 Classification of leaves	4
1.5 Project outline	6
1.6 Contributions of this project	6
2 Theory	7
2.1 Machine vision	7
2.1.1 OpenCV	7
2.1.2 RGB and HSV color space	8
2.1.3 Segmentation	10
2.1.4 Connected components analysis	11
2.1.5 Feature extraction	12
2.1.6 Skeleton analysis	13
2.2 Classification by the use of Artificial Intelligence	16
2.2.1 Linear and Quadratic Discriminant Analysis	17

CONTENTS

2.2.2	Naive Bayes Classifier	19
2.2.3	Decision Tree	21
2.3	Conceptual architecture	23
3	Implementation	27
3.1	System architecture	27
3.1.1	System architecture of the Asterix project	27
3.1.2	Software architecture of visual weed recognition	29
3.2	Machine vision	30
3.2.1	Image handling	30
3.2.2	Segmentation	32
3.2.3	Connected components analysis	34
3.2.4	Feature extraction	35
3.3	Training program	37
3.3.1	Functionality	37
3.3.2	Graphical user interface	38
3.4	Classification	41
4	Results	43
4.1	Object detection	44
4.2	Classification	44
5	Discussion	53
5.1	Architecture and implementation	53
5.2	Segmentation and object detection	54
5.3	Feature extraction and classification	55
6	Conclusion	59
7	Future work	61

Bibliography	62
List of tables	69
List of figures	75
A Acronyms	79
B Classification results	81
C Class diagrams	101
D Scatter plots	105
E DataHandler source code	113

Chapter 1

Introduction

Robots used in agriculture has of today become the symbol of modern agriculture. By the use of agricultural robotics one can greatly enhance the efficiency of agricultural processes. One of the main focus areas of Adigo AS is weed control in crops using agricultural robotics. Precision Agriculture (PA) is a concept in agricultural management based on observing, measuring and responding to the variety between fields and within a field. This can significantly reduce use of herbicide as well as human removal of weed. Agriculture is a complex process and numerous factors play in to the end yield of harvest. Information of these factors can aid better management decisions. Precision Agriculture span in scale from using satellite images to monitor drought and crop health, to initiatives focusing on individual plants. This project belongs in the latter category where we strive towards a ultra-precise weed management in row crops.

Other students from NTNU¹ have already written thesis for Adigo about precision agriculture, (Utstumo, 2011), (Lien, 2013) and (Klungerbo, 2013). The Asterix project, ”Automatic detection and control of weed in row crops” is a research project by Adigo AS in cooperation with NTNU Dept. of Cybernetics and Robotics and Bioforsk Plant Health and Plant Protection.

¹Norwegian University of Science and Technology



Figure 1.1: Asterix prototype robot frame (Adigo AS)

1.1 Agricultural robotics

Agriculture is one of the oldest and most important economic activity. With the worlds population exceeding 7 billion and still growing, there is an enormous challenge to produce enough food. The UN Sixty-fourth General Assembly clearly states that the world food production need to double by 2050 to meet the demand of the world’s growing population (64th General Assembly, 2009). As hand labor is costly and the sales price for crops are competitive, an automated agriculture may be economically feasible.

Robotic implements for harvesting, sowing and spraying are relatively common in modern agriculture and self-steering tractors have become an industry standard. As technology develops and become affordable for more farmers, the use of advanced sensors and actuators Grændsen (2013) have increased in agricultural robotics.

As the agricultural industry becomes more dependent of machine usage the demands for further development of agricultural robotics grows. As (Pota et al., 2007) presents, the need for future Precision Autonomous Framing Systems that deploy unmanned sensing and machinery systems with the possibility of centralized tele-supervision is needed.

Autonomous machinery will form the backbone of tomorrows farming systems.

1.2 Weed control and Precision Agriculture

Weed species grows in different rates and competes differently with crops with respect of minerals and water in the soil, space and sunlight. Weed control can therefore not be applied universally, but has to be adapted for each environment, growth-stage and species.

Precision Agriculture is a term commonly used for methods that exploit the variability and heterogeneity within a crop field, to better adapt the treatment of the field. The importance of weed control has been documented in a number of researches. The study by (Monaco et al., 1981) reports a loss associated with weed competition of 71%, 67%, 48% and 48% of four different tomato species. (Roberts et al., 1976) found that season-long competition from mixed stands of grass and broadleaf weeds at 64 weeds/m² resulted in a complete loss of marketable lettuce in England.



Figure 1.2: Asterix principal illustration where the white dots corresponds to herbicide droplets for weed removal (Adigo AS)

As the review article (Slaughter et al., 2008) presents, there are four core technologies (guidance, detection and identification, precision in-row weed control and mapping) re-

quired to successfully develop a general-purpose robotic system for weed control. Of the four, detection and identification of weeds under the wide range of conditions common to the agricultural fields remains the greatest challenge.

1.3 Machine vision in agriculture

A farmer can tell if his crops are ripe or not. This comes natural for a farmer, but for a robot it is not as simple. To make a robot distinguish ripe crops from non-ripe or weed from non-weed the use of optical sensors are applied, mostly as cameras. In the later years the computational power of CPUs has increased drastically, making robots able to use advanced machine vision functionality in real-time.

Today machine vision in agriculture is used mostly in six areas; sorting by color, quality assessment, detection of weeds, livestock identification, picking and machine guidance (Billingsley, 2011). Visual sensing surpass the human eye with the possibility to inspect object by light in the ultraviolet (UV), the near-infrared (NIR) and the infrared (IR) region with special camera and lenses. Information received from these light regions can be useful determining preharvest plant and vegetable maturity, variety, ripeness, disease, stress states and quality. Machine vision is also applied in both land-based and aerial-based sensing. (Chen et al., 2002)

Identification of individual crops and weed in the field and locating their exact position, is one of the most important tasks needed to further automate farming. Only with the technology to locate individual plants, can "smart" field machinery be developed to automatically and precisely perform treatments such as weeding, thinning, and chemical application (Tian et al., 2000).

1.4 Classification of leaves

Many tasks as fault diagnosis, pattern recognition and forecasting can be viewed as classification. Methods for dividing data into classes based on features (attributes) have become

a very attractive topic in later research. As the use of Artificial Intelligence have developed, classifiers have become one of the central aspects of many intelligent systems, in agricultural robotics as well.

As mentioned, identification of weed is the most difficult of the four core technologies in developing a general-purpose robotic system for weed control (Slaughter et al., 2008). For plant identification, color, shape and pattern as information features are normally used. A problem with plants is the huge variety of these features that each specie form. For the human eye, it is easy to distinguish species because humans does not need to linger to specific rules and can evaluate a huge amount of data. This information can be compared to knowledge of the species, experience and environment. For an artificial intelligent classifier problems occur when information does not follow patterns, but each case can differ in form.

Classification of plants are primarily done by analyzing its leaves. To be able to distinguish leaves it is crucial to have sufficient data features as classifier inputs. The goal of determining species can only be achieved with a huge set of training data and a number of individual feature parameters. A lot of research has been done in the area of feature parameters. Wang et al. (2003) used Centroid-Contour Distance, Moment Invariants and Angle Code Histogram as features on 93 Chinese medical plants which gave promising results. Other features as Leaf Vein Extraction, Fourier Coefficient(Fu et al., 2004), Aspect Ratios, Dissimilarity Measures(Lee and Chen, 2006), Color and Texture(Kadir et al., 2013) has also been used.

The most common classifiers today are Linear and Quadratic Discriminant Analysis (LDA and QDA), Decision Tree(DT), Naive-Bayes Classifier(NBC), Bayesian Network Classifier(BNC), Probabilistic Neural Network(PNN) and Artificial Neural Network(ANN). Classifiers uses data from a training set to predict what class the each input belong to. The success is not only based on feature quality, as mentioned, but strongly conditioned by the quantity and quality of the training set. (Yang et al., 2000) has already achieved a classification rate above 91 % by the use of ANNs and (Kadir et al., 2013) achieved above 93 % by the use of PNNs.

1.5 Project outline

Introduction: Presents some background information and accounts for the motivation for the project.

Theory: Gives the theory behind the image-processing part and the artificial intelligence part of the program. Also presents system architecture theory from (Grændsen, 2013).

Implementation: Explains how the system is designed and implemented with functionality and graphical user interface.

Results: Presents results from the segmentation and the classification.

Discussion: Here a discussion about the different aspects of the thesis will be given together with advantages, challenges and remarks in respect of the project goal.

Conclusion: The concluding part.

Future work: Suggest what the next steps would be.

1.6 Contributions of this project

1. Established an architectural plan for the Asterix project and a class diagram for the classification program.
2. Implemented an image handler that sets the framework for the program.
3. Implemented modules for object segmentation, connected components analysis and feature extraction.
4. Implemented graphical user interface for training.
5. Investigated and used different types of classifiers.

Chapter 2

Theory

2.1 Machine vision

Thirty years ago there was a lot of excitement about using visual sensing in robotics. Along with AI, people believed that this would be a technological revolution (Kak and DeSouza, 2002). But it took about twenty more years until the software and hardware were enough developed to make a real difference. As of today, visual sensing has been applied to a lot of robotic applications as well as automatic monitoring systems, guidance systems and game consoles to mention a few.

2.1.1 OpenCV

From (Grændsen, 2013):

As the use of machine vision increase, a lot of tools have been developed to support this. One of the biggest and most powerful tool in machine vision is OpenCV.

OpenCV is an open source computer vision library that originally is written i C. The later versions uses C++ and Python interfaces (Bradski and Kaehler, 2008). There is also ongoing development for Ruby, Matlab and other languages. The project was started by Intel Research in 1999, but officially released in 2006.

As (Bradski and Kaehler, 2008) presents, OpenCV was designed for computational

efficiency with a strong focus on real-time applications. As a result of an optimized C implementation, the library can take advantages of multi-core processors and lately the use of GPU¹.

As a result of OpenCV's great library, users are able to create sophisticated vision applications quite easily. It also includes machine learning functionality for computer vision tasks. The open source license results in a rapidly growing library making thousands of users contribute to functionality and web support.

2.1.2 RGB and HSV color space

The most common way to look at color images is in the RGB² space or CMY³ color space. The first is commonly used in images and displays and the latter in printing and reproduction. RGB is an additive representation while the CMY is a subtractive representation.

In image processing there are also other color spaces witch are used to analyze. The most common are HSV/HSB⁴ (hereby referred to as HSV) and HSL⁵. HSV was first used by painters because it was closer to their thinking and technique (Sonka et al., 2007). HSV decouples the intensity of a color where hue and saturation corresponds to the human perception. Using HSV as input in image processing algorithms will more easily maintain the human perception of an image when the RGB input would make the image look unfamiliar to the human eye.

According to (Morshidi et al., 2008), a transformation from the RGB to the HSV color space can be achieved with the following equations:

$$H_1 = \cos^{-1} \left[\frac{(R - G) + (R - B)}{2\sqrt{(R - G)^2 + (R - B)(G - B)}} \right] \quad (2.1)$$

¹Graphics processing unit

²Red-Green-Blue

³Cyan-Magenta-Yellow

⁴Hue-Saturation-Value/Hue-Saturation-Brightness

⁵Hue-Saturation-Lightness

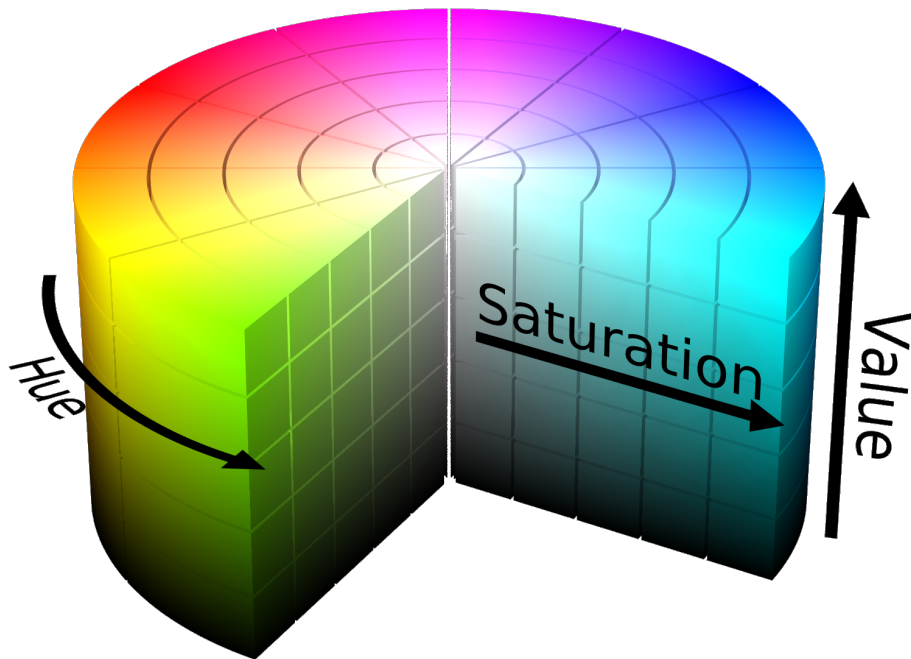


Figure 2.1: HSV cylinder (wikipedia.org)

$$H = \begin{cases} H_1, & \text{if } B \leq G \\ 360^\circ - H_1, & \text{if } B > G \end{cases} \quad (2.2)$$

$$S = \frac{\max(R, G, B) - \min(R, G, B)}{\max(R, G, B)} \quad (2.3)$$

$$V = \frac{\max(R, G, B)}{255} \quad (2.4)$$

Hue is, as presented in (Fairchild, 2013), the degree to which a stimulus can be described as similar to or different from stimuli that are described as red, green, blue, and yellow. A hue-value is independent of saturation and value(brightness). This it is important when looking for colors in images taken in changing light conditions and by different cameras. Saturation is the colorfulness relative to its own brightness (Fairchild, 2013). Colors with high saturation is considered as strong and clean while colors with low saturation value

considered closer to gray. Value is a measure of the strongest component in the RGB space. As the value increases from zero, the color changes from black to a color defined by hue and saturation.

2.1.3 Segmentation

Image segmentation is one of the most important steps leading to the analysis of processed image data. Its main goal is to divide an image into parts that have a strong correlation with objects or areas of the real world contained in the image (Sonka et al., 2007). The aim of segmentation can be either a *complete segmentation*, which gives a set of unique objects and regions, or a *partial segmentation* where the regions do not correspond directly to unique objects. A common use of partial segmentation is to separate foreground and background. A complete segmentation require more information about the image and its objects and is often achieved by advanced algorithms.

There is a whole class of partial segmentation problems that can be solved using simple algorithms only. These consist mostly of separating contrasted parts of the image by color, lightness or edges. The results from these algorithms are often used as the first step in image processing and are followed by more complex tasks. By segmenting with simple algorithms first, the following steps could then preformed faster as the regions are drastically decreased. Partial segmentation is also often used as a first step to acquire complete segmentation (Pal and Pal, 1993). From (Sonka et al., 2007):

A complete segmentation of an image R is a finite set of regions R_1, \dots, R_S ,

$$R = \bigcup_{i=1}^S R_i, \quad R_i \cap R_j = \emptyset, \quad i \neq j \quad (2.5)$$

Complete segmentation can result from thresholding in simple scenes. Thresholding is the transformation of an input image f to an outputbinary image (segmented) g as follows:

$$g(i, j) = \begin{cases} 1, & \text{for } f(i, j) \geq T, \\ 0, & \text{for } f(i, j) < T, \end{cases} \quad (2.6)$$

where T is the threshold, $g(i, j) = 1$ for image elements of objects, and $g(i, j) = 0$ for image elements of the background (or vice versa).

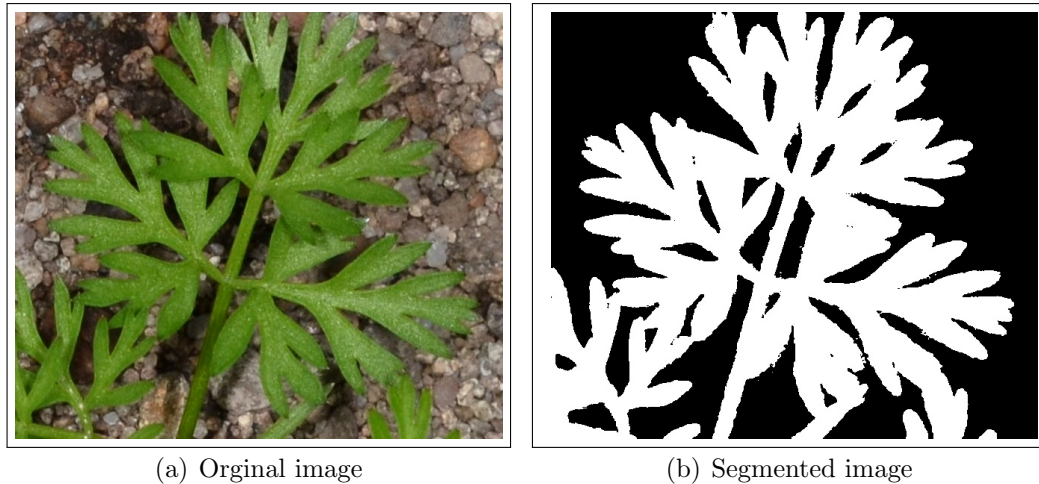


Figure 2.2: Segmentation of a carrot leaf (Adigo AS).

Thresholding is the most common used method for partial segmentation. As most images used in image processing has contrasted regions, a constant or a *threshold* can be determined to segment objects and background (Sonka et al., 2007). The threshold T (ref. eq. 2.6) could be set as a constant prior of image processes or could be adaptive by calculating independent threshold for each image. The most common of the adaptive algorithms are P-tile, Otsu and Simulated Annealing (Sezgin et al., 2004).

2.1.4 Connected components analysis

To be able to detect different objects in an image, Connected Components Analysis(CCA) is often applied. This is an algorithmic application of graph theory where subsets of an image are uniquely labeled based on a given heuristic (Westman et al., 1990). Merging predetermined regions or creating regions pixels-wise are the most common usages for this

analysis. As image processing algorithms often takes objects as input, CCA is often used as a preprocessing step. This subsection will look into CCA of binary segmented images.

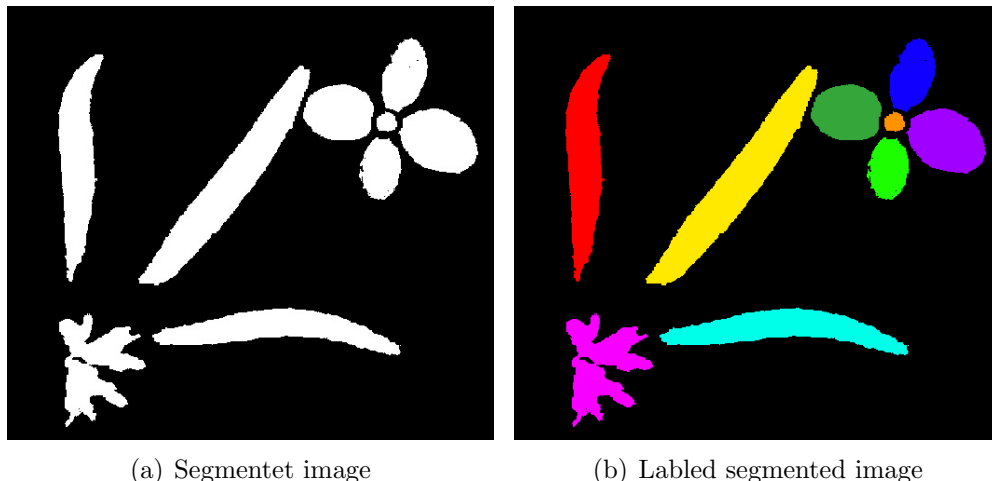


Figure 2.3: Connected components analysis where each color represents an ID (Adigo AS).

After a partial segmentation process with a binary image as output, CCA will effectively distinguish objects as illustrated in fig. 2.3. This could be done by applying algorithms as Graph traversal, Union find and Flood-fill. The Flood-fill algorithm is commonly used in image processing and is applied in most image editing softwares today. By labeling and flooding neighbours pixels with 4- or 8-connectivity recursively, its a robust and efficient algorithm. The Flood-fill algorithm is presented in alg. 2.1.1.

2.1.5 Feature extraction

Feature extraction is a fundamental part of classifying objects of any type in pattern recognition or in image processing. Instead of sending raw data into a classifier, a transformation called dimensionality reduction is applied. In an image processing perspective, the run-time speed of a classifier is minimized compared to using images as input. This also makes it possible to select specific qualities and attributes of an object. This can also make the process more robust to environmental changes.

Algorithm 2.1.1 Flood-fill

1. Let R_i be the i -th element of a set of image pixels R line wise, F be a set of foreground pixels, B be a set of background pixels and H be a set of objects. Each object H_j consists of a set of pixels where $H_j \subseteq F$. Initially, $H = \emptyset$. Let $S(R_i)$ be the set of pixels in neighbors in 4-connectivity of R_i .
 2. Iterate though R until $R_i \in F$ and $R_i \notin H$. If iteration is done, go to step 7.
 3. Create a new object H_j in H .
 4. If $R_i \in F$ and $R_i \notin H$, add R_i to H_j .
 5. Perform step 4 on all elements of $S(R_i) \notin H$ recursively.
 6. Continue iteration in step 2.
 7. H is now all objects in R .
-

In image processing there are no limits to what kind of features that can be extracted. This only depends of the knowledge and creativity of the developer. It is however important to choose features carefully with respect of the desired usage. As mentioned in section 1.4, features as Centroid-Contour Distance, Moment Invariants, Angle Code Histogram(Wang et al., 2003) Leaf Vein Extraction, Fourier Coefficient(Fu et al., 2004), Aspect Ratios, Dissimilarity Measures(Lee and Chen, 2006), Color and Texture(Kadir et al., 2013) has already been proven to be efficient in classification of leaves, but also basic features as area, circumference and density are widely applied. Both (Søgaard, 2005) and (Persson and Åstrand, 2008) extracted advanced shape models as features with a classification success from 65% to over 90%, while (Tang et al., 2003) based her features on Gabor wavelets.

2.1.6 Skeleton analysis

Skeleton analysis based on geometric and topological properties of a shape. The analysis corresponds significantly curving points of a region boundary to graph nodes. Its main focus is to present a description for how far points are from the shape edge and their

internal distance from each other (Sonka et al., 2007). A skeleton structure can be represented in different ways. This subsection will look at a graph representation and a distance transformation.

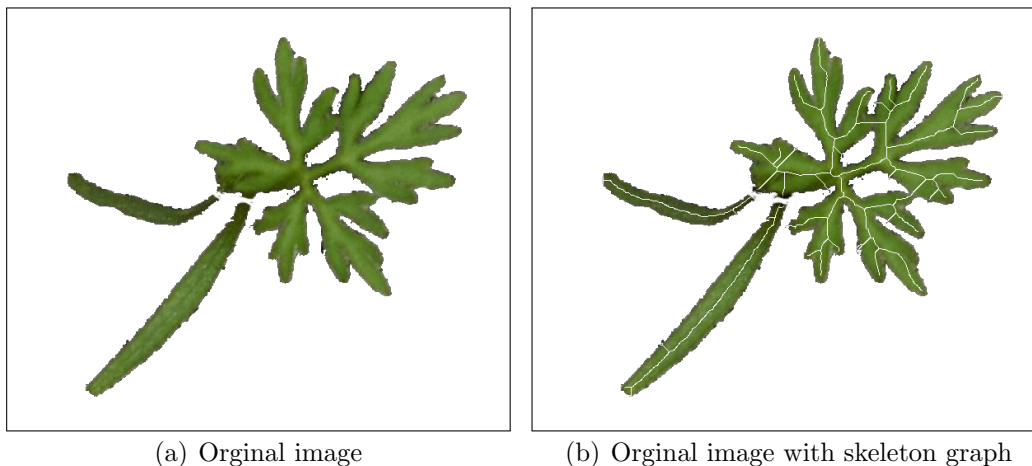


Figure 2.4: Illustration of a skeleton graph on a leaf (Adigo AS).

Skeleton analysis by machine vision has been applied in such as optical character recognition, fingerprint recognition, visual inspection, pattern recognition, binary image compression, and protein folding (Abeyasinghe et al., 2008). The skeleton graph is easily recognized by simple matching algorithms making classifying efficient.

The most used algorithm to find a graph representation of a skeleton structure is by *thinning*. This is done by repeatedly removing boundary elements until a pixel set with maximum thickness of 1 or 2 is found. The steps of this algorithm is presented in alg. 2.1.2. A common problem with thinning algorithms is the run-time. As seen in the algorithm, step 2 is repeated until the skeleton structure is achieved. If the objects are big and the distance from center to the edge is large, step 2 is repeated d times where d is the greatest distance from one point to the closest edge.

A distance transform of an image is a derived representation of an image. This is also known as distance map or distance field (Pudney, 1998). A distance representation is often used to measure the distance from obstacle objects and is most used in motion planning in robotics and path-finding. A skeleton distance image is an image transform

Algorithm 2.1.2 Skeleton by thinning (Sonka et al., 2007)

1. Let R be the set of region pixels, $H_i(R)$ its inner boundary, and $H_0(R)$ its outer boundary. Let $S(R)$ be a set of pixels from the region R which have all their neighbors in 8-connectivity either from the inner boundary $H_i(R)$ or from the background — from the residuum of R . Assign $R_{old} = R$.

2. Construct a region R_{new} which is a result of one-step thinning as follows

$$R_{new} = S(R_{old}) \cup [R_{old} - H_i(R_{old})] \cup [H_0(S(R_{old})) \cap R_{old}]. \quad (2.7)$$

3. If $R_{new} = R_{old}$, terminate the iteration and proceed to step 4. Otherwise assign $R_{old} = R_{new}$ and repeat step 2.

4. R_{new} is a set of skeleton pixels, the skeleton of the region R .

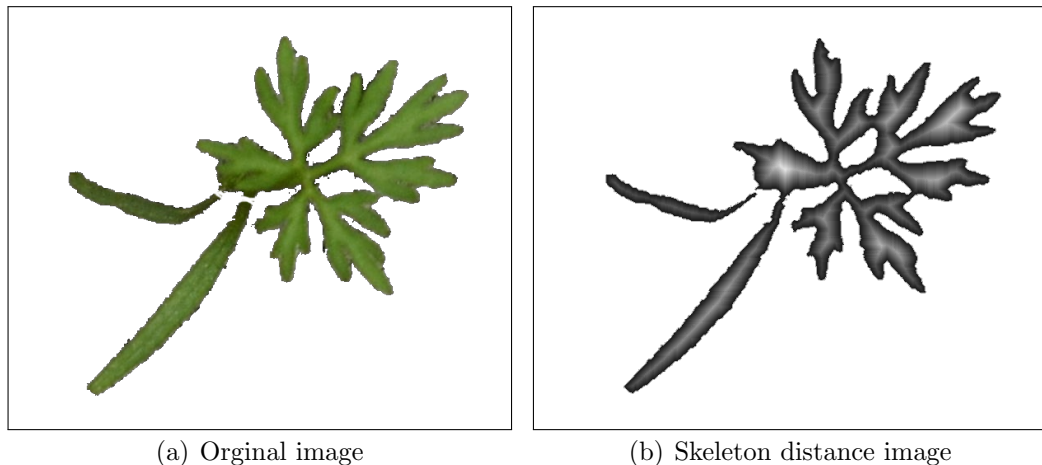


Figure 2.5: Illustration of a skeleton distance image where the distance is indicated with a lighter color (Adigo AS).

used to measure the distance from a pixel of an object to its closest edge. An illustration of this can be seen in figure 2.5 where the value of the pixel represents its distance to the closest edge. The algorithm is presented in alg. 2.1.3. This form of distance image is mostly used to analyze the shape of an object for further analysis and classification. Key values for classification are the highest skeleton distance value and the sum of all distances.

Algorithm 2.1.3 Transformation to skeleton distance image

1. Let R_i be the i -th element of the set of region pixels R line wise and B its outer boundary. All elements of R are initialized with the value of -1 . Let $S(R_i)$ be the set of pixels in neighbors in 8-connectivity of R_i that are not -1 .
2. Iterate though all elements of R as follows

$$R_i = \begin{cases} \min(S(R_i)) + 1, & \text{if } S(R_i) \cap B = \emptyset, \\ 1, & \text{otherwise,} \end{cases} \quad (2.8)$$

3. Repeat step 2, but reverse iteration order.
 4. R is now transformed to a distance image where R_i is the chessboard distance to the nearest edge.
-

2.2 Classification by the use of Artificial Intelligence

A classifier is a combination of the input values. In the AI literature, discrimination and classification are described as supervised learning techniques; together, they are also referred to as class prediction. (Izenman, 2008)

We assume the population P is portioned into K unordered classes which we denote as $\Pi_1, \Pi_2, \dots, \Pi_K$. Each item is classified into one (and only one) of those classes. Item measurements from a training set of size m are to be used to help assign future unclassified items to one of the designated classes. The random r -vector \mathbf{X} , given by

$$\mathbf{X} = (X_1, \dots, X_r)^T \quad (2.9)$$

represents the r measurements on an item (i.e., $\mathbf{X} \in \mathcal{R}^r$). The variables X_1, X_2, \dots, X_r are likely to be chosen because of their suspected ability to distinguish between the K classes. The variables in eq. 2.9 are called discrimination or feature values (Izenman, 2008).

2.2.1 Linear and Quadratic Discriminant Analysis

One of the most simple and robust classifiers is the Linear Discriminant Analysis classifier. Even if its old and simple, it is still one of the best classifiers in complex data sets. (Fisher, 1936) presented the LDA classifier and since then this method has been expanded to also handle multiple class problems. The LDA classifier have the assumption that the measurements from each class are normally distributed. However if this assumption cannot be made, the QDA classifier have to be applied.

The following equations for the LDA classifier in this section are based on (Li et al., 2006), (Balakrishnama and Ganapathiraju, 1998) and (Izenman, 2008).

For a LDA classifier the scatter matrix for all classes are given as

$$S_i = \sum_{\mathbf{X} \in K_i} (\mathbf{X} - \bar{X}_i)(\mathbf{X} - \bar{X}_i)^T \quad (2.10)$$

where the mean \bar{X}_i for each class is given by $\bar{X}_i = \frac{1}{m_i} \sum_{\mathbf{X} \in \Pi_i} \mathbf{X}$ and m_i is the number of samples in Π_i . Hence the total intra-class scatter matrix is given by

$$\hat{\Sigma}_w = S_1 + \dots + S_K = \sum_{i=1}^K \sum_{\mathbf{X} \in \Pi_i} (\mathbf{X} - \bar{X}_i)(\mathbf{X} - \bar{X}_i)^T \quad (2.11)$$

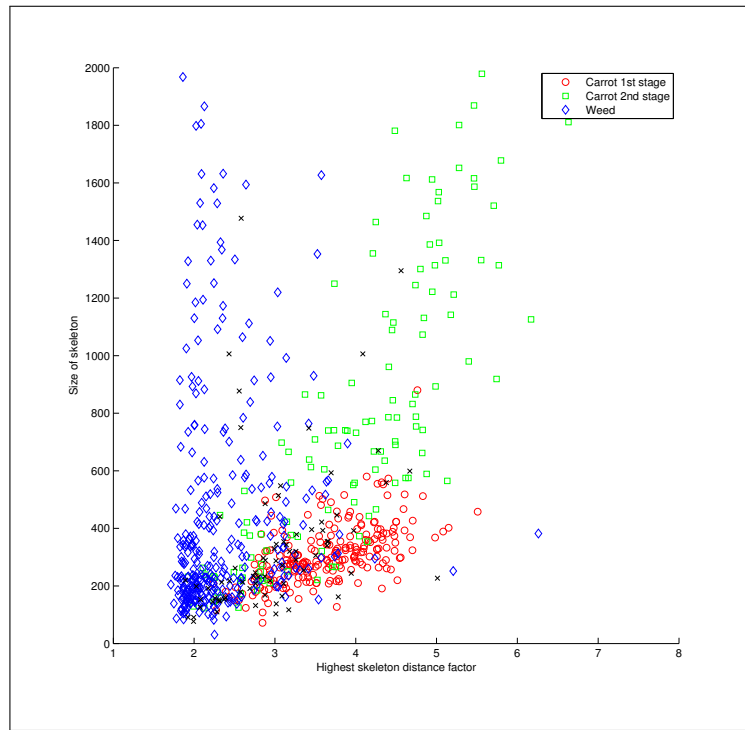
and the inter-class scatter matrix given by

$$\hat{\Sigma}_b = \sum_{i=1}^K m_i (\bar{\mathbf{X}} - \bar{X}_i)(\bar{\mathbf{X}} - \bar{X}_i)^T \quad (2.12)$$

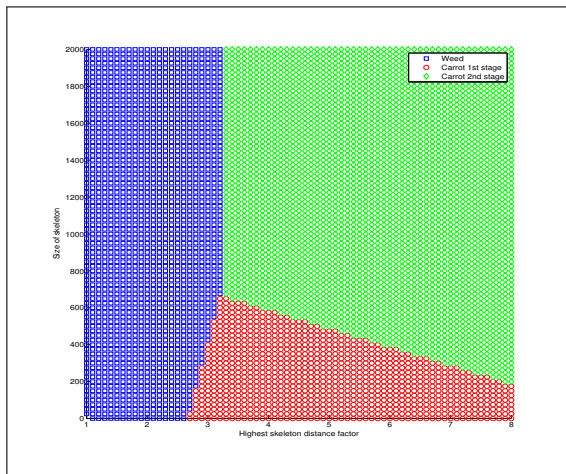
where $\bar{\mathbf{X}}$ is the total mean vector given by $\bar{\mathbf{X}} = \frac{1}{m} \sum_{i=0}^K m_i \bar{X}_i$. Then the linear transformation Φ is found to maximize the so-called Rayleigh quotient. This coefficient is the ratio of the determinant of the inter-class scatter matrix to the intra-class scatter matrix of the projected samples:

$$\mathcal{J}(\Phi) = \frac{|\Phi^T \hat{\Sigma}_b \Phi|}{|\Phi^T \hat{\Sigma}_w \Phi|} \quad (2.13)$$

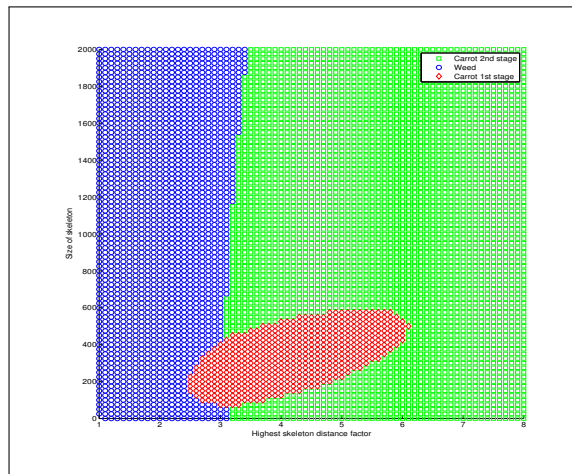
2.2. CLASSIFICATION BY THE USE OF ARTIFICIAL INTELLIGENCE



(a) Scatter plot of three classes



(b) Corresponding LDA classified area



(c) Corresponding QDA classified area

Figure 2.6: Illustration of resulting classification areas of LDA and QDA with the same input (from results presented in section 4.2).

It can be shown that the transformation Φ can be obtained by solving the generalized eigenvalue problem:

$$\hat{\Sigma}_b \Phi = \lambda \hat{\Sigma}_w \Phi \quad (2.14)$$

It is easy to prove that the upper bounds of the rank of $\hat{\Sigma}_w$ and $\hat{\Sigma}_b$ are respectively $m - K$ and $K - 1$.

Once the transformation Φ is given, the classification is then preformed in the transformed space based on some distance metric, such as Euclidean distance $d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2}$ or cosine measure $d(\mathbf{x}, \mathbf{y}) = 1 - \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2} \sqrt{\sum_i y_i^2}}$. Then upon the arrival of the new instance \mathbf{z} , it is classified to

$$\arg \min_k d(\mathbf{z}\Phi, \bar{\mathbf{X}}_k) \quad (2.15)$$

where $\bar{\mathbf{X}}_k$ is the centroid of the k -th class.

QDA is not really that much different from LDA except that you do not assume that the inter-class scatter matrix is the same for all classes, thus the scatter matrix $\hat{\Sigma}_b$ in QDA is to found separately for each class Φ_k , $k = 1, 2, \dots, K$ (Wu et al., 1996). The eq. 2.12 then stays quadratic.

2.2.2 Naive Bayes Classifier

Naive Bayes is one of the most efficient and effective inductive learning algorithms for machine learning and data mining. Its competitive performance in classification is surprising, because the conditional independence assumption it is based on, is rarely true in real-world applications (Zhang, 2004).

Naive Bayes is the simplest form of Bayesian network where the class nodes has one parent and no children. All attributes are independent given the value of the class variable. A simple example is illustrated in fig. 2.7. The following equations are from (Chen et al., 2009) and (John and Langley, 1995):

Bayes rule of x_i belonging to a class Π_j is

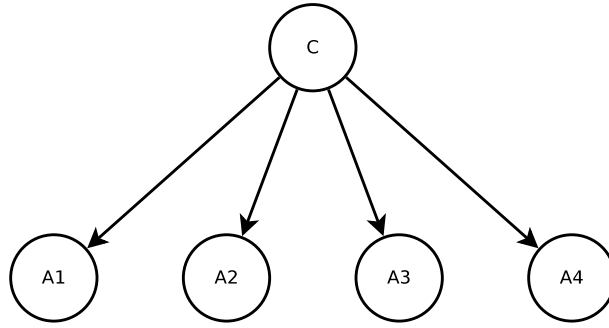


Figure 2.7: An example of naive Bayes (Zhang, 2004)

$$P(\Pi_j|x_i) = \frac{P(x_i|\Pi_j)P(\Pi_j)}{P(x_i)} \quad (2.16)$$

As $P(x_i)$ is the same for all classes, then $label(x_i)$, the class label of x_i , can be determined by

$$label(x_i) = \arg \max_{\Pi_j} \{P(\Pi_j|x_i)\} = \arg \max_{\Pi_j} \{P(x_i|\Pi_j)P(\Pi_j)\} \quad (2.17)$$

To simplify the calculation, the Naive Bayes assumption is made: In an element x_i the probability of a class Π_j is independent of what class the other elements in \mathbf{X} belongs to.

Given a training set, the probability $P(\Pi_j)$ from eq. 2.17 is estimated as

$$P(\Pi_j) = \frac{1 + n_j}{K + m} \quad (2.18)$$

where n_j is the number of elements in class Π_j , K the number of classes and m the number of all elements.

When dealing with continuous data, two common methods will be presented, Gaussian distribution and Kernel density estimation. As for Gaussian, a common assumption not intrinsic to the naive Bayesian approach, is that the values of numeric attributes are normally distributed within each class (John and Langley, 1995). One can represent such a distribution in terms of its mean and standard deviation. For continues attributes a Gaussian distribution is given by

$$P(x_i|\Pi_j) = g(x_i; \mu_{\Pi_j}, \sigma_{\Pi_j}), \quad (2.19)$$

where

$$g(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (2.20)$$

given the mean μ_{Π_j} and variance $\sigma_{\Pi_j}^2$ of class Π_j .

The kernel density does not require a strong assumption such as a Gaussian distribution and can be used in cases where the distribution of a feature may be skewed or have multiple peaks or modes. For each feature you model with a kernel distribution, the Naive Bayes classifier computes a separate kernel density estimate for each class based on the training data for that class. Even so, the kernel density estimation is based on a Gaussian distribution:

$$P(x_i|\Pi_j) = \frac{1}{n_j} \sum_{k=0}^{n_j} g(x_i; \mu_k, \sigma_{\Pi_j}) \quad (2.21)$$

To make the native Bayesian classifiers more effective, a prior calculation of g can be done before evaluating unseen elements.

2.2.3 Decision Tree

The use of Decision tree (DT) models in classification, called Classification tree, has not been used widely before the later years. This is because DT analysis is unlike other analyze methods (Lewis, 2000). A classification tree is a graph-tree and consist of nodes and leaves. The most common form is the binary tree where each node has two arcs that forms a statement. This statement could be either true/false or $x > c/x \leq c$. A leaf-node represents the the resulting class of an element. Fig. 2.8 illustrates a simple decision tree.

DT analysis is a form of binary recursive partitioning where each node can be split into two children nodes several times. Thus, each parent node can give rise to two child nodes, in turn, each of these nodes may themselves be split and from additional children. The

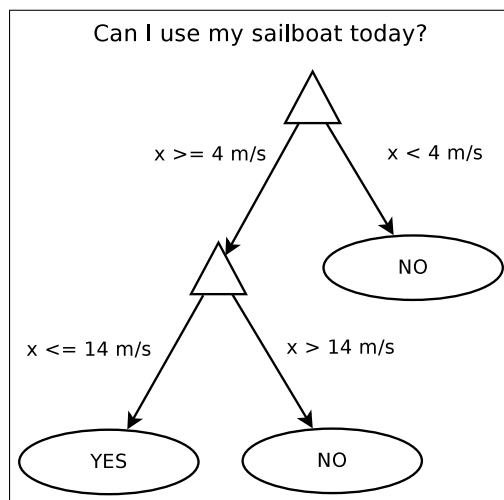


Figure 2.8: A simple decision tree with x being the wind speed.

term *partitioning* refers to the fact that the dataset is split into sections or partitioned (Lewis, 2000). Decision trees have the advantage of being very easy to interpret for the human eye which is not common for a classifier. It is therefore often used a lot for illustrating complex decisions, both in machine learning and the human brain.

Alg. 2.2.1 gives a brief presentation of creating a classification tree. This algorithm has two functions that needs a closer look. A splitting function is applied to find the best value of all feature variables in the current data set. In choosing the best *split*, the program seeks to minimize the average entropy (Steinberg and Colla, 2009). This could lead to one of the two sets gains entropy and the other reduce, but each set will be evaluated in the next depth. The other function is a function that decides if a node is qualified to be a class. This can be done by predefined conditions, utility or cost function. The later is well presented in (Lewis, 2000).

An issue with the presented algorithm is *overfitting*. Overfit is when a tree is too specific regarding the training set and will therefore misclassify future elements. This is when *tree pruning* is applied. Often the method of cost-complexity pruning are used to reduce the number of nodes in a classification tree. A pruned tree should perform better over time than the original tree.

The cost-complexity pruning is done by calculating the resubstitution error for the

Algorithm 2.2.1 Creating a classification tree based on a training set

1. Let N_i be a node containing a data set D_i and N_0 being the root node that contains the whole training set. The training set consists of elements where each element has the feature variables M and belongs to one of the classes in K . Let H be a set of nodes, l be the maximum allowed depth and $L(N_i)$ return the depth of N_i . Initially, add N_0 to H .
 2. Pull the first element of H , N_i . Find the value of M_j in D_i , if possible, that best splits D_i in two by the use of a splitting function. Copy each part of the splitted set into two new children nodes, N_{i+1} and N_{i+2} .
 3. Check both N_{i+1} and N_{i+2} if none of the following criteria are not met. If not, add that node to H :
 - There are only elements of one class K_k in D_{i+x} .
 - $L(N_{i+x}) > l$
 - The criteria for assigning class to node are met.
 4. Repeat step 2 and 3 until $H = \emptyset$.
 5. Calculate probability for each class in all nodes.
-

subsets of the original tree and then calculate the cross validation error rate for these subtrees. As the number of leaf nodes grows in the construction of the decision tree, the cross validation error rate decreases. But after a certain point, as more nodes are added, the cross validation error rate starts to increase. The best pruned tree has the number of leaf nodes as when the cross validation error rate are minimized.

2.3 Conceptual architecture

From (Grændsen, 2013):

Figure 2.9 gives a superficial description of how an agent should be implemented, but details of what goes on in the different modules is a non general answer. As an agent becomes complex it is difficult to make a good approach without a plan, an architectural description of the agent.

2.3. CONCEPTUAL ARCHITECTURE

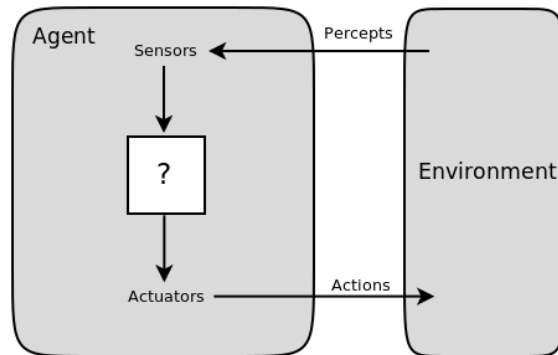


Figure 2.9: A brief schematic description of an agent

The perception module can consist of lots of different sensors and feedback inputs. These inputs need to be interpreted before deciding a plan of action. A plan does not only have to be based of the input of the sensor, knowledge from earlier experiences and knowledge given in implementation. Other factors could also be due to security measures or to failures.

Described functionality need to be implemented in a reasonable way. (Jensen et al., 2012) suggest that an introduction of an open conceptual architecture tailored to field robots. Implementing this architecture in a well supported framework will significantly decrease development time and resources required to preform precision agriculture and experiments due to efficient code reuse across field robot platforms and research groups. They stress that this conceptual architecture is not to be a obstacle, but a modular design so the researches can freely develop and experiment.

With the same simple description agent as in figure 2.9, they have created an expansion as a suggestion for an open conceptual architecture as shown in figure 2.10.

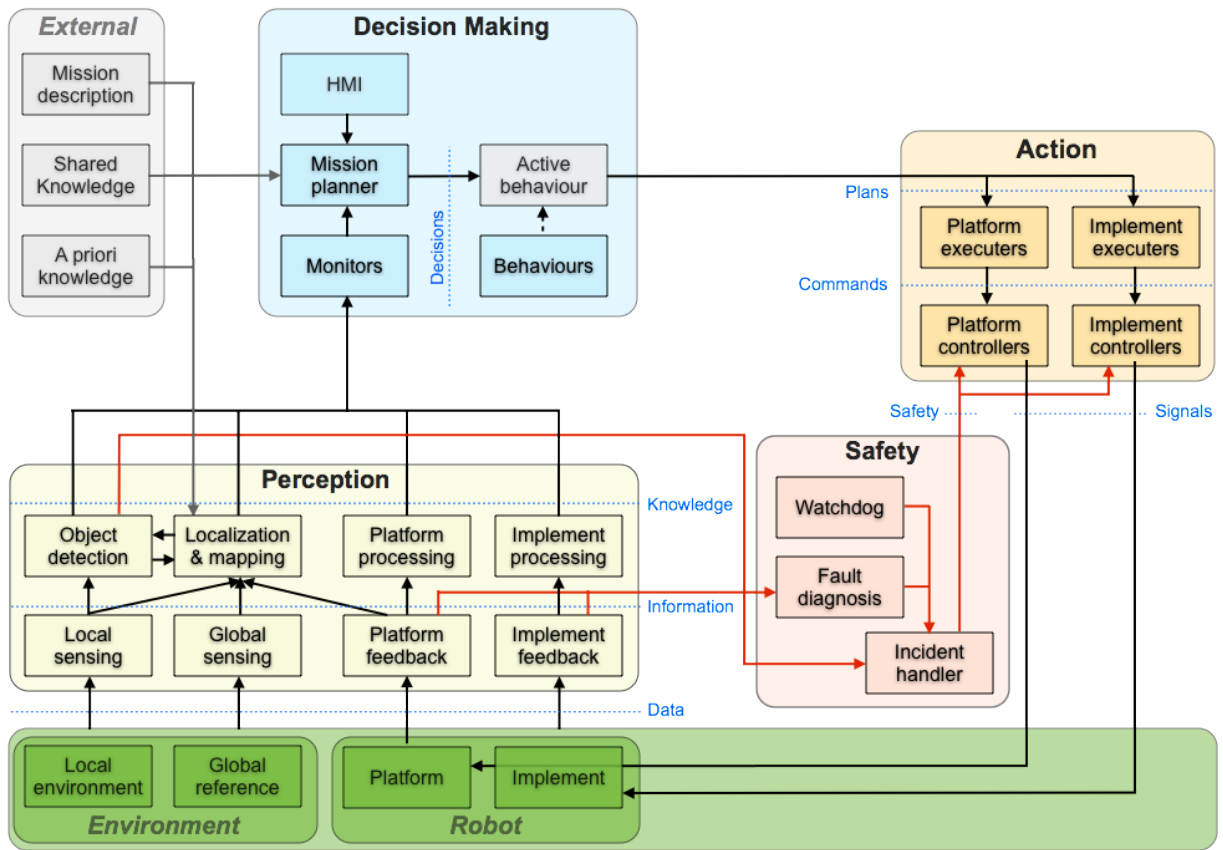


Figure 2.10: The FroboMind conceptual architecture (Jensen et al., 2012)

2.3. CONCEPTUAL ARCHITECTURE

Chapter 3

Implementation

3.1 System architecture

This section will present the overall system structure of the Asterix project and the software architecture of the weed recognition program.

3.1.1 System architecture of the Asterix project

Adigo AS looks to the conceptual architecture of (Jensen et al., 2012), presented in section 2.3, for development of the Asterix project. By applying this as the principal system architecture of the project the modularity will be similar to other agricultural robots. This will improve the overall shared development level and experience in this small industrial field.

The adapted conceptual architecture from (Jensen et al., 2012), see figure 2.10, to the architecture of the Asterix project can be seen in figure 3.1. This is how the system is planned at this stage. Because of its modularity, add-ons can easily be implemented as the development process continues.

The project can be separated into three main parts which are closely connected, as seen in figure 3.1 by the stippled lines; navigation (blue), weed detection (green) and precision herbicide spraying (red):

3.1. SYSTEM ARCHITECTURE

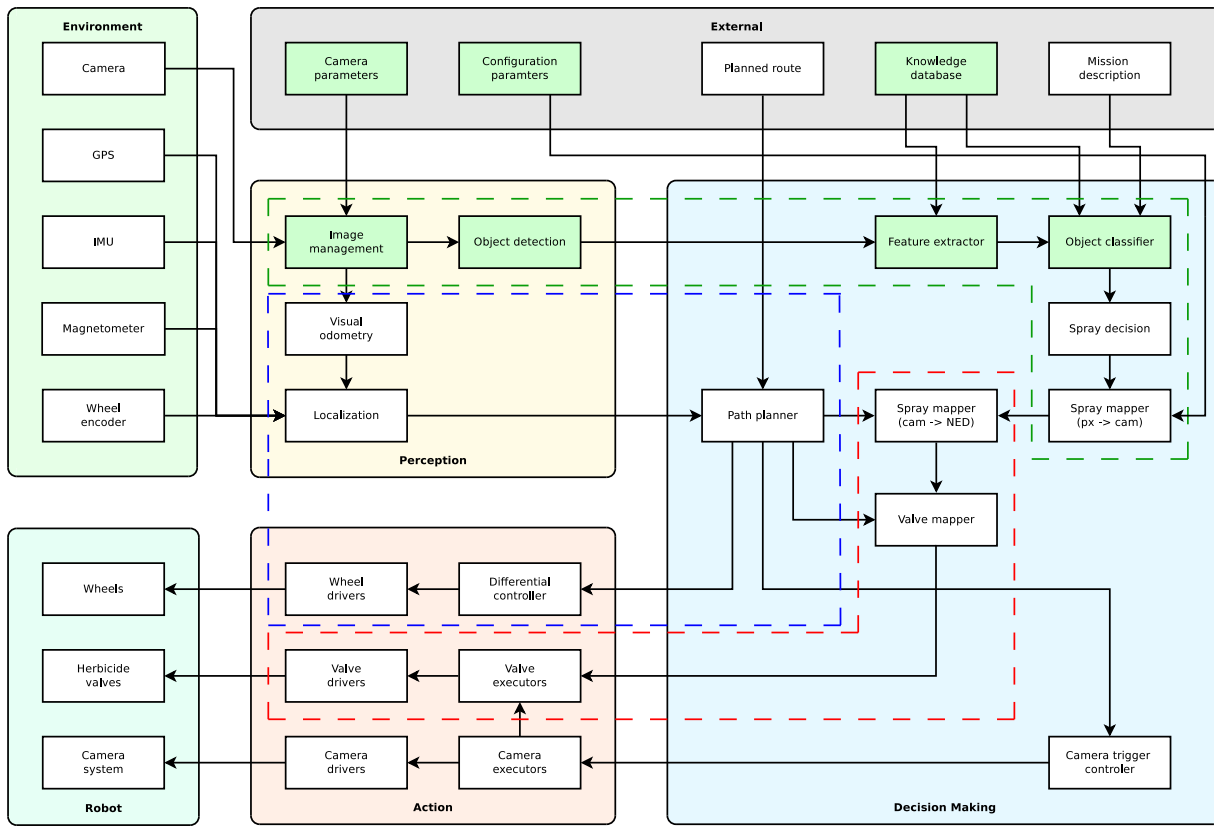


Figure 3.1: System architecture of the Asterix project. The stippled lines indicates the main three parts in the project and the green boxes are the ones that this thesis involves.

- The navigation part shall consist of a program that reads from positioning instruments as GPS, IMU, magnetometer and wheel encoder to localize where the robot is at all times. It will also use visual odometer (Grændsen, 2013) to enhance local positioning. A path planner program will also be applied as well as a differential controller to the wheel drivers.
- The weed detection part will consist of a program that uses images from a camera to detect and classify leaves into crops or weed. This program require knowledge from prior training to classify correctly. After classification, the program shall create a local spray map for where to spray the herbicide.
- The purpose of the Precision herbicide spraying part is to apply the herbicide by individual droplets. Spray maps and local localization are inputs to a valve mapper

that calculates where to shoot droplets to hit the desired weed leaves. A valve executor will then use the camera trigger and output of the valve mapper to precisely time the signal to the valve driver for releasing droplets.

The green boxes in figure 3.1 are the ones that this thesis involves. The next subsection will look closer into these modules.

3.1.2 Software architecture of visual weed recognition

An abstract class diagram of the program can be seen in figure 3.2. The First thing of notice is the program dual use. As mentioned in the last subsection, the classification process can not be executed without a trained classifier. This is why a graphical user interface is applied for off-line training, but this will be explained in section 3.3. For on-line spraying, the program will output a spray image as mentioned. A brief explanation will follow in the rest of this subsection, but will be explained in detail in section 3.2. The total class diagram of the implemented program can be seen in figure C.1 in appendix C.

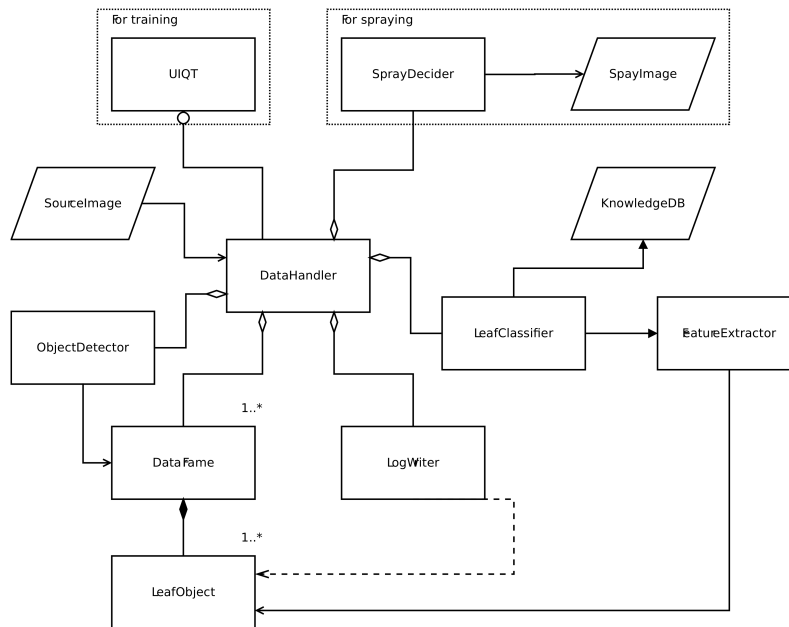


Figure 3.2: Abstract class diagram

As the program receives a `SourceImage` from either a camera (on-line) or a file (off-line), the `DataHandler` creates an unique `DataFrame`-object. This `DataFrame` process the image into several new images for different analyses. Each `DataFrame`-object holds a number of `LeafObjects` that corresponds to leaves in the `SourceImage`. All objects, except for `DataFrame` and `LeafObject`, are static. This means they only do work on other objects, and in this case these objects are `DataFrame` and `LeafObject`. The `ObjectDetector` distinguish leaves from background and creates unique `LeafObjects`. The `LeafClassifier` is responsible for the classification process by gather data by the use of the `FeatureExtractor` and classify each leaf by knowledge from `KnowledgeDB`. If the program are set up for training, the `LeafClassifier` will instead store data in the `KnowledgeDB`. The `LogWriter` documents all information from the objects as the program runs. The `UIQT` is running the graphical user interface for training. The `SprayDecider` is jet to be implemented, but this shall create a binary image, the `SprayImage`, of where to apply herbicide.

The source code of the implementation of the `DataHandler` is presented in appendix E.

3.2 Machine vision

All implementation in this part has been done using an `OpenCV`¹ environment. Pre-implemented functionality from the `OpenCV` library has been used in order to keep robustness of the program and will be stated when presented. This section will present a step-by-step solution of the image processing from source images to extracted features data of each leaf.

3.2.1 Image handling

As this program needs to handle multiple images as it runs, the `DataHandler` will take care of creating and deleting new `DataFrames` objects. It also holds and handle the other

¹`OpenCV` version 2.4.9

static objects, including the GUI. As mentioned above, the program can receive images from a camera as input in spraying mode, but will get a path to a folder of training images as input in training mode. The program stores a vector of all paths of the image files according to the input folder.



Figure 3.3: A SourceImage example (Adigo AS)

As a new DataFrame is created, a constructor sets an unique ID and adds the SourceImage to a vector of images. Figure 3.3 shows an example of a SourceImage. The constructor then calls an initialization function that creates all the images for further processing. These images are in RGB, HSV, binary and integer formats. The use of the different types of images will be explained later, as they are used. As this program is still in development, the DataFrame stores all images to a file when it is destroyed.

3.2.2 Segmentation

As explained in subsection 2.1.3, different methods can be used for segmentation. As the leaves are green, they are easy to distinguish from the background by the human eye. After some research and testing, the use of threshold for a partial segmentation was chosen. But using RGB-values for the thresholding process proved not to be robust. As (Kumar et al., 2012) suggest, the HSV color space is a good solution. The advantages of the HSV color space are explained in subsection 2.1.2.

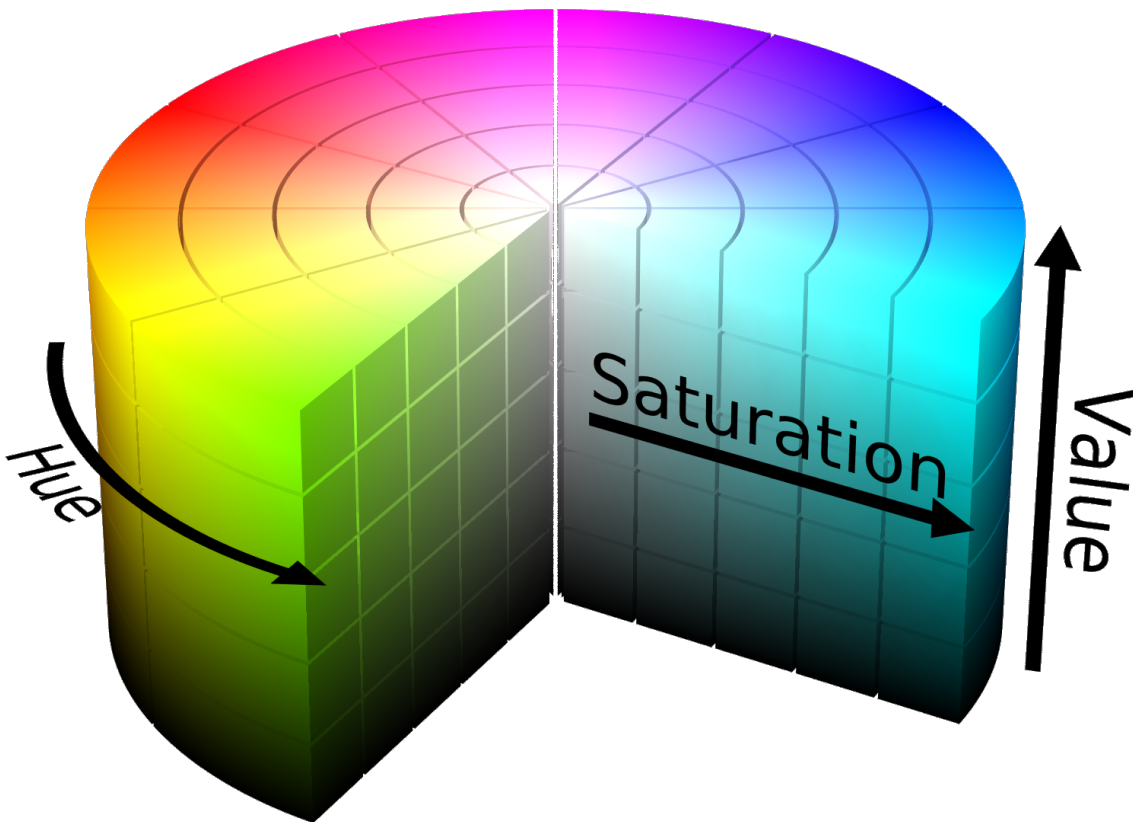


Figure 3.4: SourceImage converted into HSV color space

Figure 3.4 shows the SourceImage converted to HSV. The image is not intuitive, but the red channel represents value, the green represents saturation and the blue represents hue. In figure 3.5 the HSV image has been decomposed, this might ease the understanding.

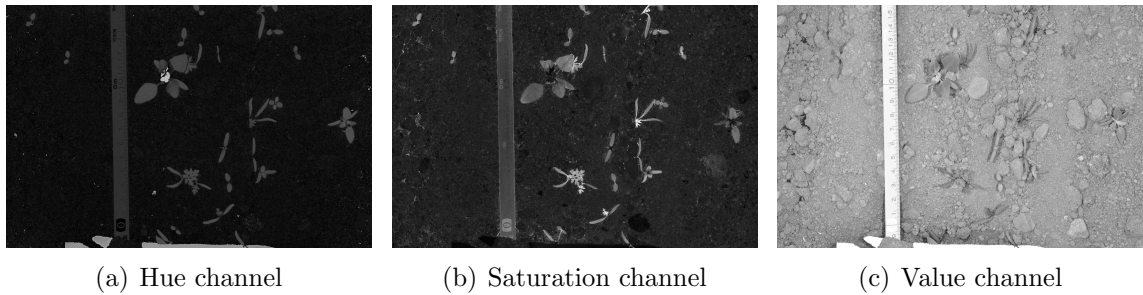


Figure 3.5: The decomposed image in figure 3.4.

As seen in figure 3.5(a), the hue channel shows the most clearly difference from the leaves and the background. This is why we set the hue values as the primary source of the thresholding. The threshold $T_{h_1} > 30$ and $T_{h_2} < 90$. As leaves have a strong saturated green color, it is necessary to distinguish these from background or other objects that might have the same hue value, but less saturated. The threshold $T_s > 58$ is then also applied. Lastly, dark areas can contain the same hue and saturation as the leaves, but setting the last threshold as $T_v > 50$ solves this issue. All these values are set by examining images as well as experimenting.

By iterating though the HSV image, a binary mask image is set by applying all four thresholds in respective channels. The resulting segmented binary image can be seen in figure 3.6 and will be used as a image mask for further processing. Even if the segmentation is good, there are some small areas or pixels in the image which is not part of an object. This is due to pixel error and noise.

This by applying the image mask to the SourceImage, a segmented RGB image is found. This image is used, instead of the SourceImage, from this moment for image processing in terms of RGB information. Because it only holds information of the objects of interest, it will make analysis more efficient. Such an image can be seen in figure 3.7.



Figure 3.6: Resulting segmented binary image

3.2.3 Connected components analysis

As the segmentation process is done, a connected components analysis is in order. This is for labeling each object with a unique id and create a `LeafObject` of them. The theory behind this subsection, can be found in 2.1.4.

The label image is created by cloning the binary image mask, where 0 is background and 1 is foreground, and changing the image format to an integer image. By use of the floodfill function in the OpenCV library, a resulting image with unique values from 2 and up for each object and still 0 for background is done. All points of a labeled object are stored in vectors. As each object is labeled, the object size is checked before creating a `LeafObject` of the labeled pixels. If the sum of pixels is less than a given value (in this case, 1000), the labeled pixels are marked as background and discarded as an object. Figure 3.8 illustrates different detected leaves. If comparing this figure with figure 3.6, small detected objects

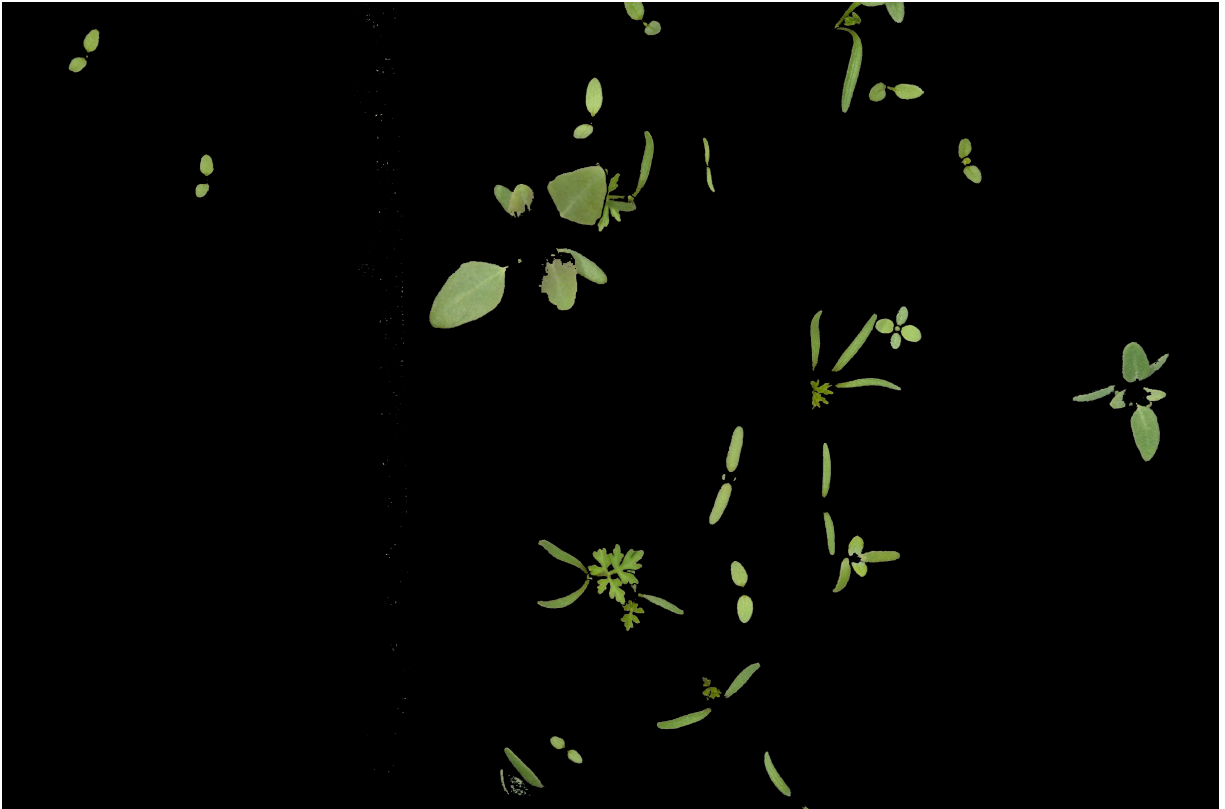


Figure 3.7: Resulting segmented image in RGB

are now removed.

One thing worth mentioning in the terms of connected components analysis is the ability to separate leaves that are connected after segmentation. After several attempts with Sobel filters and graph cuts where both methods were inconclusive, the only way of advancing with the development was to assume the separation were complete. This was done by adding red lines between connected leaves before running the program.

3.2.4 Feature extraction

After successfully detecting, labeling and storing `LeafObjects` in a `DataFrame`, the last step in the machine vision part of the program is to extract features from each `LeafObject`. The `LeafClassifier` asks the `FeatureExtractor` for features from each `LeafObject`. The `FeatureExtractor` analyze each leaf with the respect of the given feature from the `Leaf-`

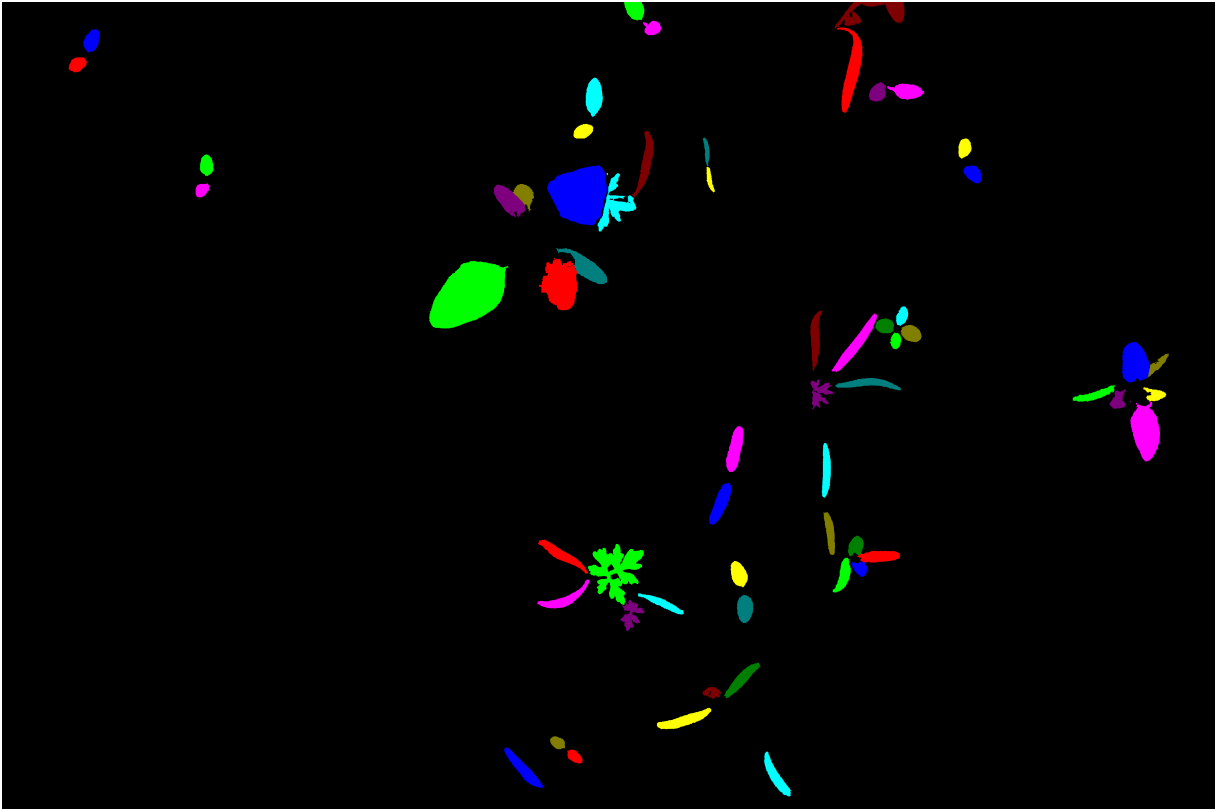


Figure 3.8: Labeled image with random colors indicating unique objects

Classifier. In total, there are ten different features that are being extracted, but some of them are combinations of other features:

Area equals the number of pixels included in the LeafObject.

Circumference equals the sum of all pixels in an edge of the LeafObject.

Density is the area divided by the circumference.

Highest skeleton distance is the greatest distance from a point to the edge of a LeafObject. This is based on a distance transformation presented in subsection 2.1.6.

Highest skeleton distance factor is the square root of the areal divided by the highest skeleton distance.

Skeleton size is the sum of all distances from all points to the closest edge of a LeafObject. This is based on a distance transformation presented in subsection 2.1.6.

Skeleton size factor is the area divided by the square root of the skeleton size.

Skeleton back is the number of pixels included in the skeleton graph of the LeafOb-

ject. This is based on the skeleton structure presented in subsection 2.1.6.

Hue is the average hue value of the LeafObject. The theory is presented in 2.1.2.

Saturation is the average saturation value of the LeafObject. The theory is presented in 2.1.2.

The *highest skeleton distance factor* and the *skeleton size factor* are found by analyzing relationships by regression in Matlab. Every feature of all LeafObjects are logged by the LogWriter to unique comma-separated values (CSV) files. When the program is in training mode, the LeafClassifier looks for CSV files that equals the DataFrame id. If found, it loads the values so they do not need to be extracted again.

3.3 Training program

This section will present how the program work in training mode, without focusing on the already covered machine vision functionality. If users are to be able to create a training set with this program, there have to be a user interface that is easy to understand and use. The program also need to be adaptable for other environments than, in this case, carrot fields.

3.3.1 Functionality

Before running the program, some parameters have to be set. A configuration file contains all the classes you want to label the leaves in. In this thesis, there are four classes; *Carrots 1st stage*, *Carrots 2nd stage*, *Weed* and *Other*. The *Other* class is for eventually wrongly detected objects or leaves in the edges that would damage the training set. As this program is not developed and hard coded for identifying solely carrots and weed, this kind of external configuration files are needed. An example of labeling six different classes are shown in figure 3.9.

As an input when running the program, the path of the folder where the training images are located, are to be given. As mentioned earlier, the program finds all images of the



Figure 3.9: An example of six different classes labeled.

Joint Photographic Experts Group format (JPEG/JPG) and stores them. As the program iterates through the image list, initiated by the user, the program stores all information of every leaf. This includes id, features and class. If the user relaunches the program with the same training folder, the program loads the previously stored information of all the images that has been opened before. This enables users to continue earlier work if wanted.

3.3.2 Graphical user interface

OpenCV has its own graphical user interface functions, but this proved to be too limited. Instead a GUI library called Qt² was chosen. Qt is the most powerful GUI library in the C++ world. It also has its own cross-platform IDE for development called QtCreator, which was partly used for this program. The purpose of this interface is to give the user

²Qt version 4.8.6

an easy way of labeling the leaves to their corresponding classes as well as iterating through the whole folder of training images.



Figure 3.10: The graphical user interface for training mode without overlay

After the the image processing is done for the current DataFrame, the SourceImage is displayed as seen in figure 3.10. To match the screen resolution of the user, the resolution is found and the image is scaled down so it covers 85 % of the screen height. A panel on the right side of the interface is set up with four static buttons; *Toggle*, *Prev*, *Next* and *Quit*.

If *Toggle* is clicked, a transparent overlay will be added on top of the image and covers all detected LeafObjects. This overlay is white for all objects that is not yet labeled (ref. subsection 3.3.1). As the user labels the objects, the overlay will change to the color that corresponds to the labeled class. This process will be explained later in this subsection. If *Toggle* is clicked again, the overlay will become invisible. The overlay is created from a copy of a binary version of the image shown in figure 3.8, but then filled with colors corresponding to its labeled class. When the LeafObject are created, it is labeled as class

3.3. TRAINING PROGRAM

zero, an invalid class. If the LeafObject belongs to an invalid class, the overlay is white. The figure 3.11 illustrates this.



Figure 3.11: The graphical user interface for training mode with overlay

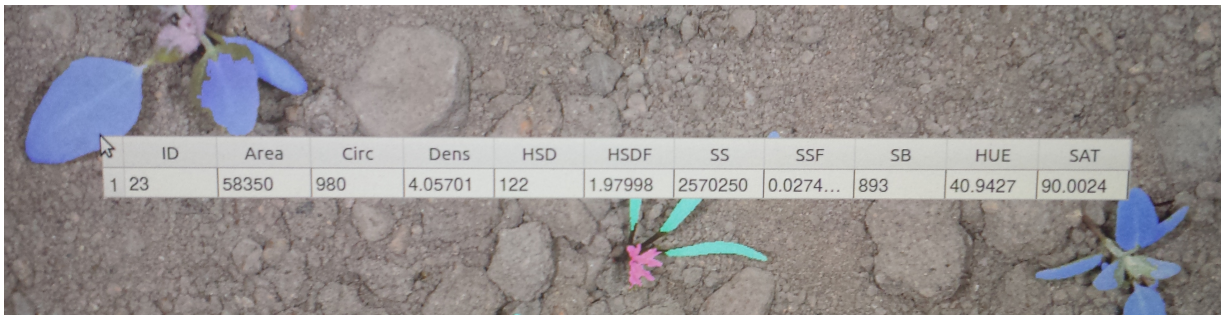
The goal of *Next* and *Prev* are intuitive, but their functionality needs explanation. As mentioned, the *DataHandler* holds a list of all images in the training folder. If one of these buttons are clicked, the *DataHandler* is notified. It then loads the next or previous image and processes it as presented in section 3.2 and then deletes the *DataFrame* that has been replaced. The GUI is then notified that the new image are loaded and processed, and will then display the new image. If *next* is clicked at the last image or *prev* is clicked at the first image, nothing happens.

If *quit* is clicked, the *DataHandler* is notified and will close the program in the correct manner by logging the current *DataFrame* before shutting down.

The four buttons with colored squares represents the classes that are to be used in labeling. These are loaded from the *LeafClassifier* dynamically. This means the number of buttons listed can change based on the input from the configuration file. If one of this

buttons are clicked it becomes toggled, and the user can then click on a leaf to label it into the desired class. The button is toggled until an other of the label-buttons are clicked. When a LeafObject is clicked, the GUI calculates the coordinates based on the scaling of the image. It then asks the corresponding DataFrame what id the LeafObject in the given coordinates has. If this is an LeafObject, the GUI informs the LeafClassifier of the class which sets it in the LeafObject.

All buttons have hotkeys connected to them. In example, the label buttons are connected to the numbers on the keyboard and *toggle* to the keyboard button "t".



	ID	Area	Circ	Dens	HSD	HSDf	SS	SSF	SB	HUE	SAT
1	23	58350	980	4.05701	122	1.97998	2570250	0.0274...	893	40.9427	90.0024

Figure 3.12: Table showing the features of a selected leaf

The last function to be mentioned is the listing of features that belongs to a selected LeafObject. If the SHIFT-button on the keyboard is pressed and the cursor is held over a LeafObject, a table with the id and all features are shown as in figure 3.12. This is mostly an feature for developing.

3.4 Classification

The classification has been done in Matlab (R2013b, 64-bit linux version) with the statistics toolbox. The procedure was inspired by an example given at the MathWorks web page³.

After the training process, a script which gathers all data from the training, sorts them into classes and stores each class in a separate file is run. Then this data is put into a $n \times m$

³<http://www.mathworks.se/help/stats/examples/classification.html>

matrix, where n is the number of elements and m is the number of feature measurements. A vector of size n is also created with classes corresponding to the rows of the matrix. To be able to train and test the classifiers, the matrix and the vector is split into two sets by selecting them even and odd. This two sets will form the training set and the test set.

The following classifiers have been tested:

LDA: Linear Discriminant Analysis, presented in subsection 2.2.1.

QDA: Quadratic Discriminant Analysis, presented in subsection 2.2.1.

DT: Decision tree, presented in subsection 2.2.2.

DTp: Decision tree, pruned, presented in subsection 2.2.2.

nbGau: Naive Bayes with Gaussian distribution, presented in subsection 2.2.3.

nbKd: Naive Bayes with Kernel density, presented in subsection 2.2.3.

To be able to analyze the quality and behaviour of the classifiers further, the classification process was run with different reduction of the size of the initial training set. All tests was performed with the same test set.

After looking at the results, presented in chapter 4, an experiment was done by creating a voting classifier that uses the results from all classifiers mentioned above. This is done by labeling each LeafObject as the class the majority of the classifiers have chosen. If two classes have the same amount of votes for a given object, it is labeled as a carrot.

By studying the scatter plots in chapter 4 and in appendix D, another experimental approach was done by setting raising the lower limit of the size of each LeafObject. By doing so, the small leaves were removed from the training set. The same classification process as before was then executed with the resulting training set.

Chapter 4

Results

This chapter will present results from the segmentation and the classification that was explained in chapter 3. The images used in this thesis are given by Adigo AS. All images were taken the same day in a carrot field in Østfold, the southeastern part of Norway, in the middle of June 2012.



Figure 4.1: Camera setup in field (Adigo AS)

For photographing, a Nikon D7000 camera mounted on a tripod, 54 cm above ground, was used. For making the light conditions as similar as possible, an umbrella was set up to shade the sunlight and a flash was used as the main light source. The camera setup can be seen in figure 4.1.

4.1 Object detection

As mentioned in section 3.2, the connective component analysis implemented in this thesis can not distinguish two overlapping leaves from another. But the detection of leaves in general are done solely by image processing. By respect of this, the figure 4.2 illustrates the presence of leaves by contours.



Figure 4.2: SourceImage with contours of the detected presence of leaves

4.2 Classification

The classification was done with a total set of 1351 objects from 25 different images, where 677 objects was used for training and 674 objects for testing.

The results from classifying all objects with the six classifiers and the use of voting, all presented in section 3.4, can be seen in table 4.1.

	LDA	QDA	DT	DTp	nbGau	nbKd	voted	total
Carrot 1st stage	206	211	197	167	212	207	211	223
Carrot 2nd stage	120	123	104	109	109	102	111	133
Weed	276	267	290	306	243	287	294	318
Total	602	601	596	577	564	596	616	674

Table 4.1: Classification success by number of unique LeafObjects correctly classified

In table 4.2 the results from table 4.1 have been presented by the percentage of total elements of each class.

	LDA	QDA	DT	DTp	nbGau	nbKd	voted
Carrot 1st stage	92.38 %	94.62 %	88.34 %	74.89 %	95.07 %	92.83 %	94.62 %
Carrot 2nd stage	90.23 %	92.48 %	78.20 %	81.95 %	81.95 %	76.69 %	83.46 %
Weed	86.79 %	83.96 %	91.19 %	96.23 %	76.42 %	90.25 %	92.45 %
Total	89.32 %	89.17 %	88.43 %	85.61 %	83.68 %	88.43 %	91.39 %

Table 4.2: Classification success in percent

Table 4.3 to 4.8 shows specified results for each of the six classifiers based on the total elements of their true class.

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	92.38 %	2.24 %	5.38 %
	Carrot 2nd stage	6.77 %	90.23 %	3.01 %
	Weed	11.01 %	2.20 %	86.79 %

Table 4.3: True vs. detected for Linear Discriminant Analysis

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	94.62 %	4.04 %	1.35 %
	Carrot 2nd stage	3.76 %	92.48 %	3.76 %
	Weed	13.52 %	2.52 %	83.96 %

Table 4.4: True vs. detected for Quadratic Discriminant Analysis

4.2. CLASSIFICATION

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	95.07 %	2.69 %	2.24 %
	Carrot 2nd stage	11.28 %	81.95 %	6.77 %
	Weed	22.01 %	1.57 %	76.42 %

Table 4.5: True vs. detected for Naive Bayes with Gaussian distribution

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	92.83 %	2.24 %	4.93 %
	Carrot 2nd stage	9.02 %	76.69 %	14.29 %
	Weed	7.55 %	1.89 %	90.25 %

Table 4.6: True vs. detected for Naive Bayes with Kernel density

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	88.34 %	2.69 %	8.97 %
	Carrot 2nd stage	11.28 %	81.95 %	6.77 %
	Weed	6.29 %	2.52 %	91.19 %

Table 4.7: True vs. detected for Decision tree

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	74.89 %	4.04 %	21.08 %
	Carrot 2nd stage	8.27 %	78.20 %	13.53 %
	Weed	3.14 %	0.63 %	96.23 %

Table 4.8: True vs. detected for Decision tree, pruned

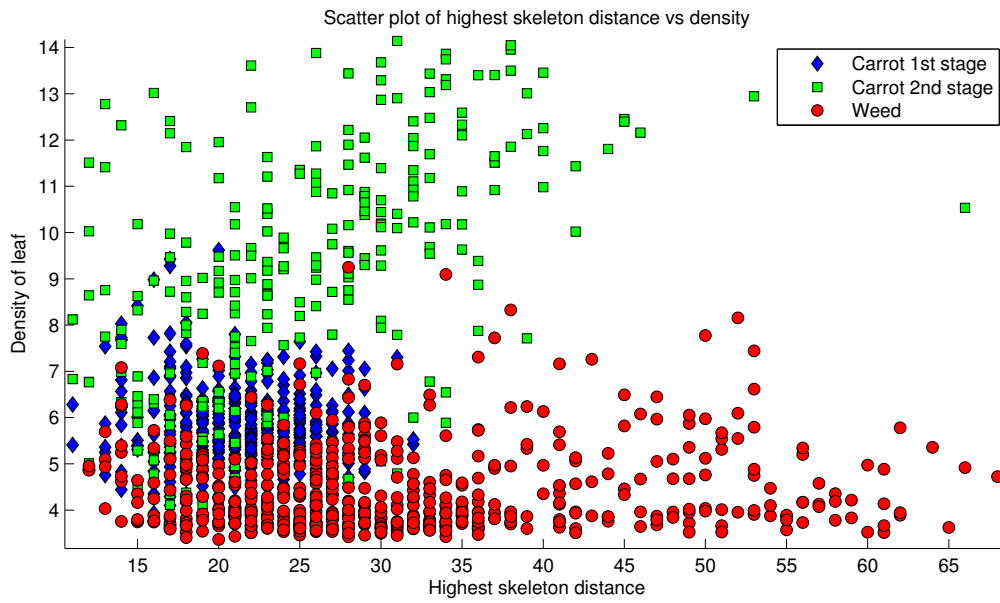


Figure 4.3: Scatter plot of density vs highest skeleton distance.

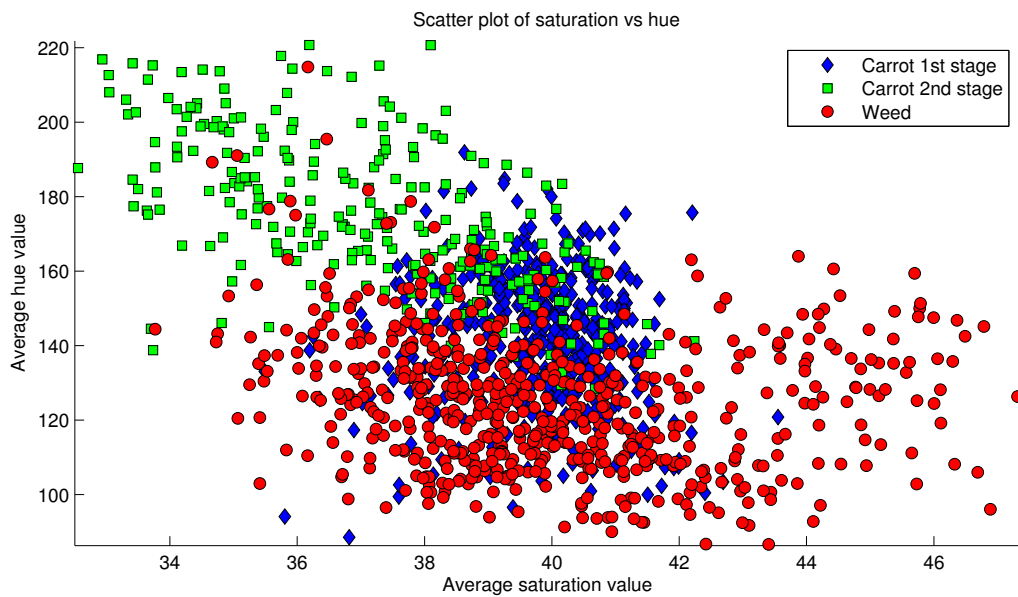


Figure 4.4: Scatter plot of average hue value vs average saturation value.

In figure 4.3 and 4.4 a selection of two scatter plots are given. More of these plots can be seen in appendix D.

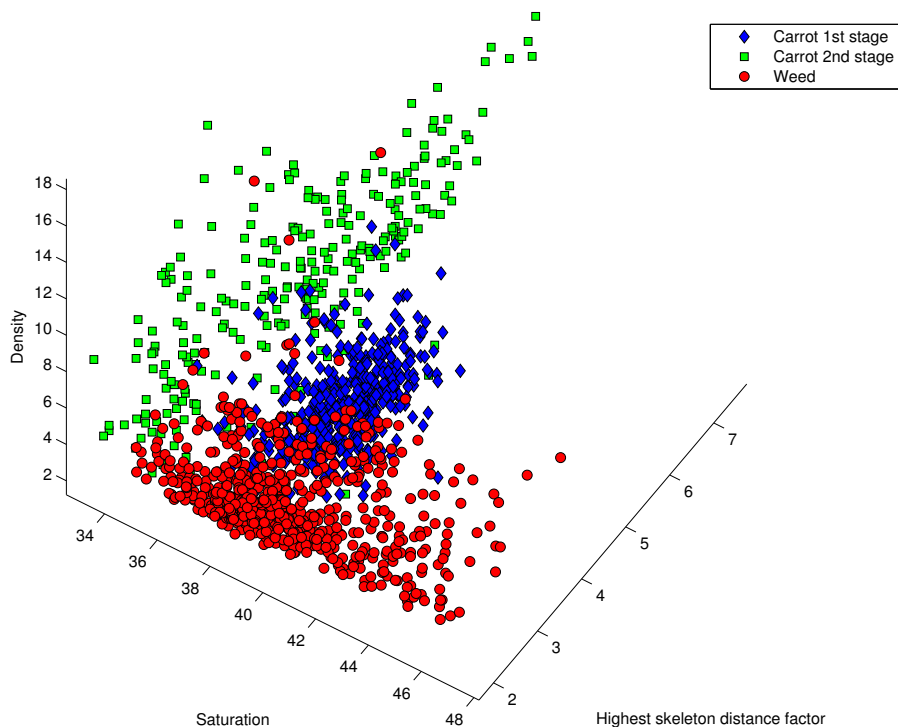


Figure 4.5: 3d scatter plot of density, saturation and highest skeleton distance factor.

Figure 4.5 show a three dimensional scatter plot of *Density*, *Saturation* and *Highest skeleton distance factor*.

The full decision tree after training can be seen in 4.6. This figure illustrates the complexity of the tree, rather than its values and node statements. The total number of leaf nodes is 45.

Figure 4.7 shows the decision tree in figure 4.6 after pruning. The number of leaf nodes is now reduced to 6.

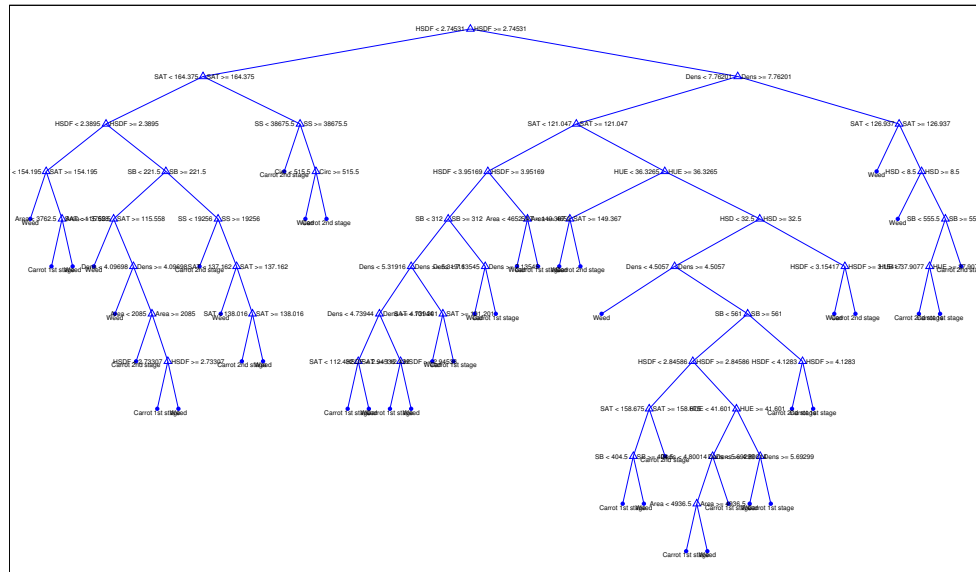


Figure 4.6: Resulting decision tree.

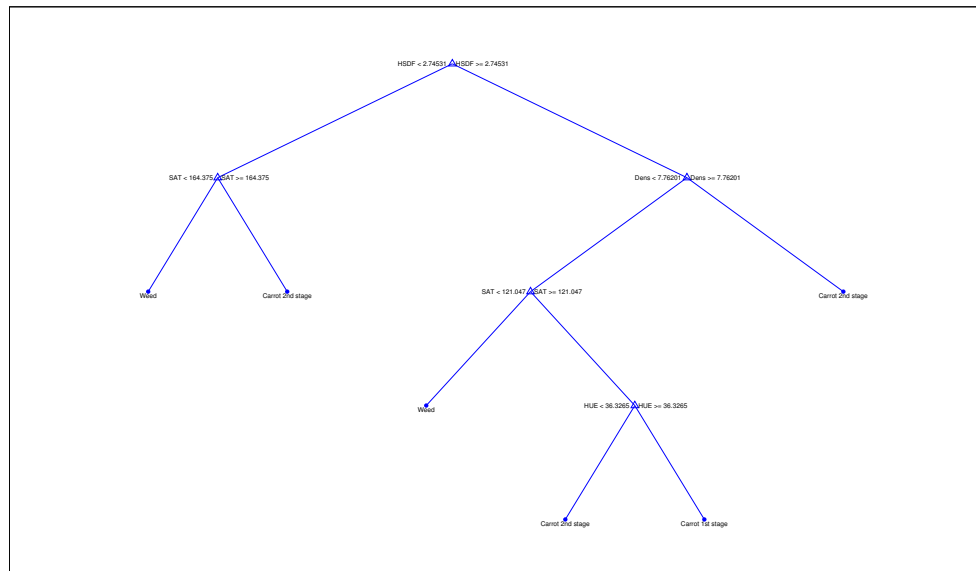


Figure 4.7: Resulting decision tree, pruned.

4.2. CLASSIFICATION

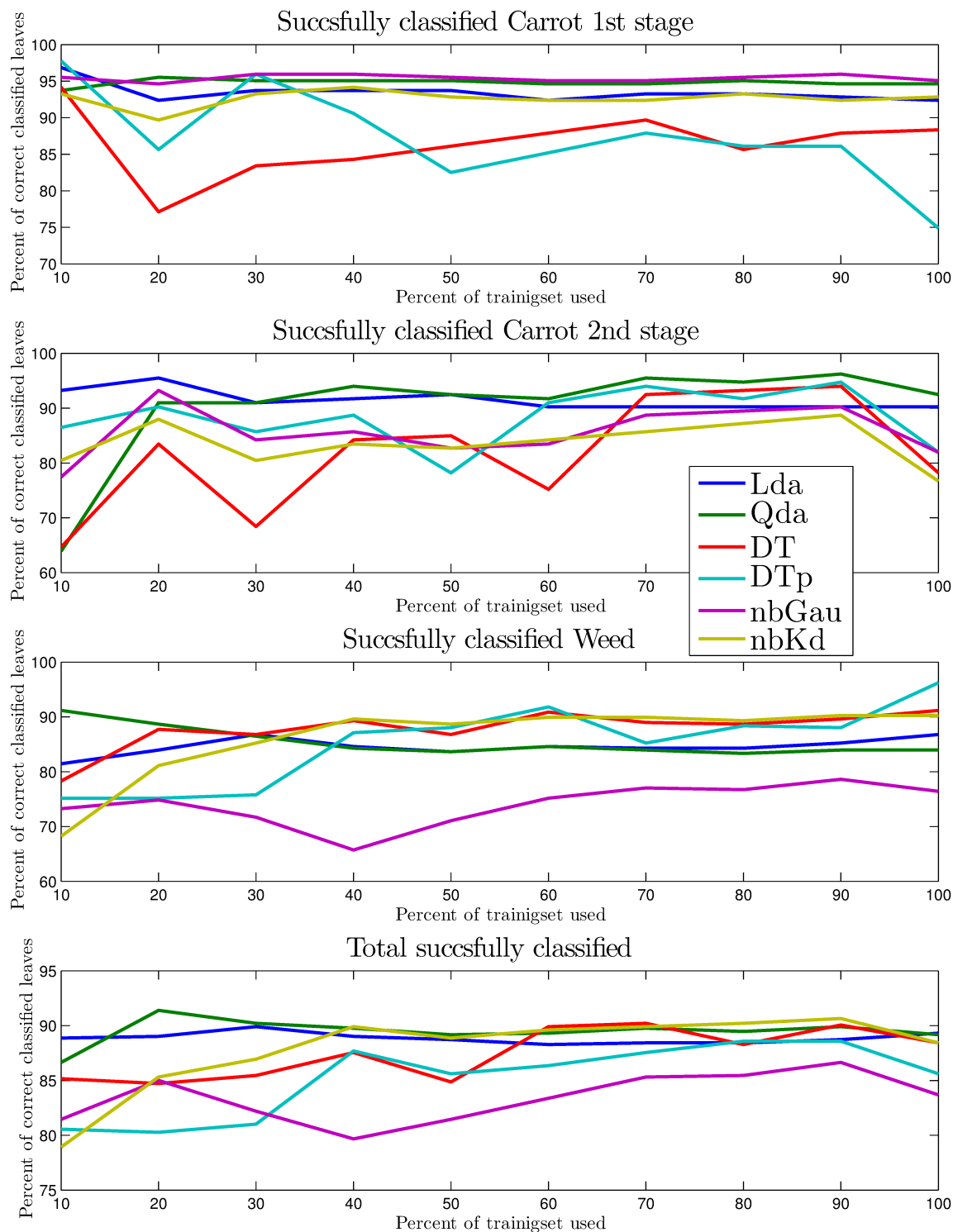


Figure 4.8: Plots of the different classifiers and their performance based on training size.

As described in section 3.4, the classification process was also tested with reduced training sets. Figure 4.8 shows the success rate of the of the different classifiers with respect of percentage of the full training set. All results of each individual test are provided in appendix B.

	Lda	Qda	DT	DTp	nbGau	nbKd	voted	total
Carrot 1st stage	147	144	137	140	148	147	148	152
Carrot 2nd stage	83	87	85	85	86	86	87	89
Weed	139	139	145	143	135	141	146	158
Total	369	370	367	368	369	374	381	399

Table 4.9: Classification success by number of unique LeafObjects correctly classified after setting the lower limit of areas to 5000 pixels.

	Lda	Qda	DT	DTp	nbGau	nbKd	voted
Carrot 1st stage	96.71 %	94.74 %	90.13 %	92.11 %	97.37 %	96.71 %	97.37 %
Carrot 2nd stage	93.26 %	97.75 %	95.51 %	95.51 %	96.63 %	96.63 %	97.75 %
Weed	87.97 %	87.97 %	91.77 %	90.51 %	85.44 %	89.24 %	92.41 %
Total	92.48 %	92.73 %	91.98 %	92.23 %	92.48 %	93.73 %	95.49 %

Table 4.10: Classification success in percent after setting the lower limit of areas to 5000 pixels.

An experiment was done by setting a lower limit of the area in each LeafObject, as presented in section 3.4. In this case, the limit was set to 5000 pixels. Table 4.9 and 4.10 shows the results of this experiment.

Chapter 5

Discussion

In this chapter the main topics of this thesis will be discussed; *Architecture and implementation*, *Segmentation and object detection* and *Feature extraction and classification*. In particular, the experience from the implementation in chapter 3 and the results in chapter 4 are to be evaluated and discussed.

5.1 Architecture and implementation

The system architecture presented in figure 3.1 has proved to be a promising approach at this stage of development. All though there is still major parts of the system to be developed, its modularity makes it easy to develop and understand. Furthermore, the design of the weed detection part of the Asterix project has been established and is presented in figure 3.2.

The implemented program works well and the training mode is fully functional. However, a classifier to be used in the spraying mode in the classification part of the system, presented in figure 3.2, is to be implemented. In this thesis, the use of MATLAB has been applied to examine the different classifiers. This is done in order to investigate the results and the performance before choosing a classifier and implementing it into the program. On the other hand, a framework for the program has been created and is easy to develop further.

As presented in section 3.3, a graphical user interface was implemented by the use of Qt. The choice of using Qt as GUI library has proven to be a good solution. By the huge amount of options in Qt, the GUI has become delicate and easy to use. Even though there are much data being processed, the program runs fluently during the labeling process. In addition, further functionality as simulation and evaluation of the spraying mode can be implemented without changing the already existing functionality.

The use of OpenCV has also proved to be well suited for this program. As there are a lot of already implemented functions in the library, the handling of images is easily controlled. When working with large images, the memory management is crucial to prevent memory leaks. The problem with memory leak was learned the hard way while training, but this was later solved. Another consequence of working with huge amounts of data, is that the run time of this program is not close to what it have to be if used on-line. On the other hand, the focus of this thesis is to investigate the possibility of implementing a program that preforms well in the terms of weed detection and classification. When this goal is fully reached, the runtime of the program can be properly addressed.

5.2 Segmentation and object detection

A big challenge when processing images from natural environments, is the variety it produce. This is certainly the case when it comes to images of leaves in a field. Every leaf is different from the next, even if they belong to the same species. They also unfold in different ways depending of weather, soil and specie.

The use of HSV color space in the segmentation process, see figure 3.4, has proven to be very robust. As seen in figure 4.2 almost all leaves has been successfully detected. The only problem in this case was a special weed specie¹ that becomes white in the center, just as the soil around it. This problem has not been addressed because the program are not implemented to handle specific species.

As the program is not able to separate overlapped leaves, the use of predefined separa-

¹Chenopódium album

tion was used. This was done in order to test and develop the rest of the system properly. As presented in subsection 3.2.3, the use of graph cut and Sobel filter was investigated and applied, but proved inaccurate. As the leaves often have a very similar color and the light source is mounted on the camera, the edge of the leaf is very hard to detect in the image, even for the human eye. This problem has to be solved if this program shall work in the field and classify on-line. A solution to this could be the use of stereo vision or a infra-red image to measure distance in height.

5.3 Feature extraction and classification

As seen in the table 4.2 the results from the classification are satisfiable based on the stage of development. Another thing to be noticed in the table is that there are no classifiers which clearly proves to be the best choice. However, there are two classifiers that scores significantly lower than the other four. The naive Bayesian classifier with Gaussian distribution and the pruned decision tree deviates from the other four by about 5 and 3 percent. The cause of this can be that the assumption of a Gaussian distribution of the data features is not valid and the pruning of the decision tree is too rough. But to be certain that these two classifiers are ill suited for this kind of problem, more testing on different unique data set are required.

By studying figure 4.8, some more remarks are to be discussed. The naive Bayesian Gaussian classifier preforms lower in general compared to the rest. In addition both the pruned and the regular decision tree are varying in their performance. The naive Bayesian kernel classifier (after 40 % of training set) and both of the discriminant analysis classifiers are stable and preform well.

It is important to note that even though there are three classes to be determined, the weed-class do not consist of only one species. This makes it very hard for any classifier to learn from a training set. As for the images used in this thesis, there are two main weed species that are frequent and more than ten that are not.

It does not matter what kind of classifier that is being used if the inputs are inadequate.

An example of this is trying to divide Weed from Carrot 2nd stage in figure 4.4. If all feature values for all classes were as inseparable as this, no classifier could be able to achieve a good result. By choosing features that can be used to find tendencies of pattern in a data set, and if the features are not directly correlated, then there can be a possibility for a good result. Figure 4.3 can be used to illustrate this. There are patterns to be seen between the three classes, but not enough to divide them adequately. On the other hand, if we include saturation as a new dimension (see figure 4.5), a more clear pattern can be seen. If all classifiers had ten dimensional feature data to use, as they have in this thesis, the ability to distinguish classes with good results is more likely.

In this thesis, six of the features are based on skeleton analysis. Even though the results are satisfying, introducing new and more uncorrelated features could make the classifiers perform even better. Examples of this are texture, aspect ratios, moment invariants and more as mentioned in subsection 2.1.5.

When studying the scatter plots of the feature values correlated with the area of each LeafObject, it can be seen clearly that in the area closer to zero, the classes are more disorganized. As the leaves get smaller, humans can also have problems distinguishing different species. From a farmers perspective, the smaller weeds do not present a threat at that stage. By applying a limit for size in the classification, the results became significantly better, as seen in 4.10. If restrictions can be made that enhances the classification without a negative effect of the weed control, there are reason to apply them.

An experiment done during the classification process is the use of voting. As seen in table 4.2 and 4.10, this performs better in the terms of total success than any other classifier does on its own. This does not prove that the best solution would be using as many classifiers as possible and use voting instead of the best classifier. On the other hand, the possibility to use three well adapted classifiers at the same time, can make the classification process more robust.

Lastly, a farmers view of classification has to be taken into account before making any conclusions. From a theoretical perspective, the total success of a classifier would be the determining result, but it is not the same for a farmer. A worst case scenario for a farmer is

spraying the crops, rather than not spraying the weed. As seen in table 4.2, the quadratic discriminant analysis classifier performs superior to the rest in terms of classifying carrots. The details of this can be seen in table 4.4. The table shows that carrots of 1st and 2nd stage has been classified as weed in 1.35 and 3.76 percent of the cases. In contrast, the naive Bayesian kernel classifier (see table 4.6) which scores almost the same as the quadratic discriminant analysis classifier in total, has classified the carrots as weed in 4.93 and 14.29 percent of the cases.

Chapter 6

Conclusion

This master thesis was motivated by the use of machine vision and artificial intelligence in agricultural robotics. A program that segmented and detected leaves was implemented for feature extraction from each leaf.

By using six different classifiers, the classification of leaves has proven promising due to the testing and evaluation done in this thesis. Even though a final conclusion can not be made on what type of classifier that performs best, the quadratic discriminant analysis classifier shows assuring results. Total correct classified leaves with a success rate of over 89 % has been reached by basic classification. Furthermore, by raising the lower boundary of the leaf size and using votes from all the classifiers to determine the class, total classification success surpassed 95 %.

The use of the 10 different features based on basic shape parameters, color and skeleton analysis are the main reason for the classification success. By using features that describes each leaf in different ways, the classifiers are able to distinguish the leaves and label them as the right class with a high success rate.



Chapter 7

Future work

According to the objective and the results of this master thesis, there are still more to be done before the *weed detection* part in the Asterix project can be tested in the field. For future work, the following is proposed:

- Find and implement a method for separating overlapping leaves in the ObjectDetector module.
- Find more uncorrelated features to extract in the FeatureExtractor module.
- Continue the testing and the evaluation of the classifiers with more training sets.
- Implement a classifier in the LeafClassifier module.
- Implement a simulation module in the GUI class for evaluation of the classifiers performance.
- Optimize the program to meet the real-time demands of the Asterix project.
- Implement the SprayDecider module from figure 3.2.



Bibliography

- 64th General Assembly, U. (2009). Food production must double by 2050 to meet demand from world's growing population. *Second Committee, Panel Discussion (AM) GA/EF/3242*, 64.
- Abeyasinghe, S., Ju, T., Baker, M. L., and Chiu, W. (2008). Shape modeling and matching in identifying 3d protein structures. *Computer-Aided Design*, 40(6):708–720.
- Ahmed, I., Islam, M., Shah, S. I. A., and Adnan, A. (2007). A real-time specific weed recognition system using statistical methods. *International Journal of Computer, Information & Systems Science & Engineering*, 1(4).
- Balakrishnama, S. and Ganapathiraju, A. (1998). Linear discriminant analysis-a brief tutorial. *Institute for Signal and information Processing*.
- Billingsley, J. (2011). Machine vision in agriculture. *Encyclopedia of Agrophysics*, pages 433–436.
- Chen, J., Huang, H., Tian, S., and Qu, Y. (2009). Feature selection for text classification with naïve bayes. *Expert Systems with Applications*, 36(3):5432–5435.
- Chen, Y.-R., Chao, K., and Kim, M. S. (2002). Machine vision technology for agricultural applications. *Computers and electronics in Agriculture*, 36(2):173–191.
- Fairchild, M. D. (2013). *Color appearance models*. John Wiley & Sons.
- Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

- Fu, H., Chi, Z., Feng, D., and Song, J. (2004). Machine learning techniques for ontology-based leaf classification. In *Control, Automation, Robotics and Vision Conference, 2004. ICARCV 2004 8th*, volume 1, pages 681–686. IEEE.
- Grændsen, Ø. W. (2013). Machine vision in agricultural robotics. Project thesis, Norwegian University of Science and Technology.
- Izenman, A. J. (2008). *Linear Discriminant Analysis*. Springer.
- Jensen, K., Nielsen, S. H., Larsen, M., Bøgild, A., Green, O., and Jørgensen, R. N. (2012). Frobomind, proposing a conceptual architecture for field robots. In *Proceedings of the International Conference of Agricultural Engineering (CIGR-AgEng), 5th Automation Technology for Off-road Equipment Conference (ATOE), Valencia, Spain*, pages 8–12.
- John, G. H. and Langley, P. (1995). Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc.
- Kadir, A., Nugroho, L. E., Susanto, A., and Santosa, P. I. (2013). Leaf classification using shape, color, and texture features. *arXiv preprint arXiv:1401.4447*.
- Kak, A. C. and DeSouza, G. N. (2002). Robotic vision: what happened to the visions of yesterday? In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 2, pages 839–847. IEEE.
- Kavdir, İ. (2004). Discrimination of sunflower, weed and soil by artificial neural networks. *Computers and electronics in agriculture*, 44(2):153–160.
- Klungerbo, A. T. (2013). Drop-on-demand i presisjonsjordbruk. Master’s thesis, Norwegian University of Science and Technology.
- Kumar, N., Belhumeur, P. N., Biswas, A., Jacobs, D. W., Kress, W. J., Lopez, I. C., and Soares, J. V. (2012). Leafsnap: A computer vision system for automatic plant species identification. In *Computer Vision—ECCV 2012*, pages 502–516. Springer.

- Lee, C.-L. and Chen, S.-Y. (2006). Classification of leaf images. *International Journal of Imaging Systems and Technology*, 16(1):15–23.
- Lewis, R. J. (2000). An introduction to classification and regression tree (cart) analysis. In *Annual Meeting of the Society for Academic Emergency Medicine in San Francisco, California*, pages 1–14. Citeseer.
- Li, T., Zhu, S., and Ogihara, M. (2006). Using discriminant analysis for multi-class classification: an experimental investigation. *Knowledge and information systems*, 10(4):453–472.
- Lien, T. A. (2013). Mobile robotics in precision agriculture. Master’s thesis, Norwegian University of Science and Technology.
- Monaco, T., Grayson, A., and Sanders, D. (1981). Influence of four weed species on the growth, yield, and quality of direct-seeded tomatoes (*lycopersicon esculentum*). *Weed science*, pages 394–397.
- Morshidi, M. A., Marhaban, M. H., and Jantan, A. (2008). Color segmentation using multi layer neural network and the hsv color space. In *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*, pages 1335–1339. IEEE.
- Naeem, A. M., Ahmad, I., Islam, M., and Nawaz, S. (2007). Weed classification using angular cross sectional intensities for real-time selective herbicide applications. In *Computing: Theory and Applications, 2007. ICCTA’07. International Conference on*, pages 731–736. IEEE.
- Pal, N. R. and Pal, S. K. (1993). A review on image segmentation techniques. *Pattern recognition*, 26(9):1277–1294.
- Persson, M. and Åstrand, B. (2008). Classification of crops and weeds extracted by active shape models. *Biosystems engineering*, 100(4):484–497.

- Pota, H., Eaton, R., Katupitiya, J., and Pathirana, S. (2007). Agricultural robotics: A streamlined approach to realization of autonomous farming. In *Industrial and Information Systems, 2007. ICIIS 2007. International Conference on*, pages 85–90. IEEE.
- Pudney, C. (1998). Distance-ordered homotopic thinning: a skeletonization algorithm for 3d digital images. *Computer Vision and Image Understanding*, 72(3):404–413.
- Roberts, H., Bond, W., and Hewson, R. (1976). Weed competition in drilled summer cabbage. *Annals of Applied Biology*, 84(1):91–95.
- Sezgin, M. et al. (2004). Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*, 13(1):146–168.
- Siddiqi, M. H., Ahmad, I., and Sulaiman, S. B. (2009). Weed recognition based on erosion and dilation segmentation algorithm. In *Education Technology and Computer, 2009. ICETC'09. International Conference on*, pages 224–228. IEEE.
- Slaughter, D., Giles, D., and Downey, D. (2008). Autonomous robotic weed control systems: A review. *Computers and electronics in agriculture*, 61(1):63–78.
- Søgaard, H. T. (2005). Weed classification by active shape models. *Biosystems Engineering*, 91(3):271–281.
- Sonka, M., Hlavac, V., and Boyle, R. (2007). *Image processing, analysis, and machine vision*. Thomson-Engineering.
- Steinberg, D. and Colla, P. (2009). Cart: classification and regression trees. *The Top Ten Algorithms in Data Mining*, pages 179–201.
- Tang, L., Tian, L. F., and Steward, B. L. (2003). Classification of broadleaf and grass weeds using gabor wavelets and an artificial neural network. *Transactions of the ASAE*, 46(4):1247.
- Tian, L., Slaughter, D., and Norris, R. (2000). Machine vision identification of tomato seedlings for automated weed control. *Transactions of ASAE*, 40(6).

- Utstumo, T. (2011). Attitude estimation in agricultural robotics. Master's thesis, Norwegian University of Science and Technology.
- Wang, Z., Chi, Z., and Feng, D. (2003). Shape based leaf image retrieval. In *Vision, Image and Signal Processing, IEE Proceedings-*, volume 150, pages 34–43. IET.
- Westman, T., Harwood, D., Laitnen, T., and Pietikanen, M. (1990). Color segmentation by hierarchical connected components analysis with image enhancement by symmetric neighborhood filters. In *Pattern Recognition, 1990. Proceedings., 10th International Conference on*, volume 1, pages 796–802. IEEE.
- Wu, W., Mallet, Y., Walczak, B., Penninckx, W., Massart, D., Heuerding, S., and Erni, F. (1996). Comparison of regularized discriminant analysis linear discriminant analysis and quadratic discriminant analysis applied to nir data. *Analytica Chimica Acta*, 329(3):257–265.
- Yang, C., Prasher, S., and Landry, J. (2002). Weed recognition in corn fields using back-propagation neural network models. *Canadian Biosystems Engineering*, 44:7–15.
- Yang, C., Prasher, S. O., Landry, J., and DiTommaso, A. (2000). Application of artificial neural networks in image recognition and classification of crop and weeds. *Canadian agricultural engineering*, 42(3):147–152.
- Zhang, H. (2004). The optimality of naive bayes. *A A*, 1(2):3.

List of Tables

4.1	Classification success by number of unique LeafObjects correctly classified	45
4.2	Classification success in percent	45
4.3	True vs. detected for Linear Discriminant Analysis	45
4.4	True vs. detected for Quadratic Discriminant Analysis	45
4.5	True vs. detected for Naive Bayes with Gaussian distribution	46
4.6	True vs. detected for Naive Bayes with Kernel density	46
4.7	True vs. detected for Decision tree	46
4.8	True vs. detected for Decision tree, pruned	46
4.9	Classification success by number of unique LeafObjects correctly classified after setting the lower limit of areas to 5000 pixels.	51
4.10	Classification success in percent after setting the lower limit of areas to 5000 pixels.	51
B.1	Classification success by number of unique LeafObjects correctly classified (90 % of full traing set)	81
B.2	Classification success in percent (90 % of full traing set)	82
B.3	True vs. detected for Linear Discriminant Analysis (90 % of full traing set)	82
B.4	True vs. detected for Quadratic Discriminant Analysis (90 % of full traing set)	82
B.5	True vs. detected for Naive Bayes with Gaussian distribution (90 % of full traing set)	82

LIST OF TABLES

B.6 True vs. detected for Naive Bayes with Kernel density (90 % of full training set) 82

B.7 True vs. detected for Decision tree (90 % of full training set) 83

B.8 True vs. detected for Decision tree, pruned (90 % of full training set) 83

B.9 Classification success by number of unique LeafObjects correctly classified (80 % of full training set) 84

B.10 Classification success in percent (80 % of full training set) 84

B.11 True vs. detected for Linear Discriminant Analysis (80 % of full training set) 84

B.12 True vs. detected for Quadratic Discriminant Analysis (80 % of full training set) 84

B.13 True vs. detected for Naive Bayes with Gaussian distribution (80 % of full training set) 84

B.14 True vs. detected for Naive Bayes with Kernel density (80 % of full training set) 85

B.15 True vs. detected for Decision tree (80 % of full training set) 85

B.16 True vs. detected for Decision tree, pruned (80 % of full training set) 85

B.17 Classification success by number of unique LeafObjects correctly classified (70 % of full training set) 86

B.18 Classification success in percent (70 % of full training set) 86

B.19 True vs. detected for Linear Discriminant Analysis (70 % of full training set) 86

B.20 True vs. detected for Quadratic Discriminant Analysis (70 % of full training set) 86

B.21 True vs. detected for Naive Bayes with Gaussian distribution (70 % of full training set) 86

B.22 True vs. detected for Naive Bayes with Kernel density (70 % of full training set) 87

B.23 True vs. detected for Decision tree (70 % of full training set) 87

B.24 True vs. detected for Decision tree, pruned (70 % of full training set) 87

B.25 Classification success by number of unique LeafObjects correctly classified (60 % of full traing set)	88
B.26 Classification success in percent (60 % of full traing set)	88
B.27 True vs. detected for Linear Discriminant Analysis (60 % of full traing set)	88
B.28 True vs. detected for Quadratic Discriminant Analysis (60 % of full traing set)	88
B.29 True vs. detected for Naive Bayes with Gaussian distribution (60 % of full traing set)	88
B.30 True vs. detected for Naive Bayes with Kernel density (60 % of full traing set)	89
B.31 True vs. detected for Decision tree (60 % of full traing set)	89
B.32 True vs. detected for Decision tree, pruned (60 % of full traing set)	89
B.33 Classification success by number of unique LeafObjects correctly classified (50 % of full traing set)	90
B.34 Classification success in percent (50 % of full traing set)	90
B.35 True vs. detected for Linear Discriminant Analysis (50 % of full traing set)	90
B.36 True vs. detected for Quadratic Discriminant Analysis (50 % of full traing set)	90
B.37 True vs. detected for Naive Bayes with Gaussian distribution (50 % of full traing set)	90
B.38 True vs. detected for Naive Bayes with Kernel density (50 % of full traing set)	91
B.39 True vs. detected for Decision tree (50 % of full traing set)	91
B.40 True vs. detected for Decision tree, pruned (50 % of full traing set)	91
B.41 Classification success by number of unique LeafObjects correctly classified (40 % of full traing set)	92
B.42 Classification success in percent (40 % of full traing set)	92
B.43 True vs. detected for Linear Discriminant Analysis (40 % of full traing set)	92

LIST OF TABLES

B.44 True vs. detected for Quadratic Discriminant Analysis (40 % of full traing set)	92
B.45 True vs. detected for Naive Bayes with Gaussian distribution (40 % of full traing set)	92
B.46 True vs. detected for Naive Bayes with Kernel density (40 % of full traing set)	93
B.47 True vs. detected for Decision tree (40 % of full traing set)	93
B.48 True vs. detected for Decision tree, pruned (40 % of full traing set)	93
B.49 Classification success by number of unique LeafObjects correctly classified (30 % of full traing set)	94
B.50 Classification success in percent (30 % of full traing set)	94
B.51 True vs. detected for Linear Discriminant Analysis (30 % of full traing set)	94
B.52 True vs. detected for Quadratic Discriminant Analysis (30 % of full traing set)	94
B.53 True vs. detected for Naive Bayes with Gaussian distribution (30 % of full traing set)	94
B.54 True vs. detected for Naive Bayes with Kernel density (30 % of full traing set)	95
B.55 True vs. detected for Decision tree (30 % of full traing set)	95
B.56 True vs. detected for Decision tree, pruned (30 % of full traing set)	95
B.57 Classification success by number of unique LeafObjects correctly classified (20 % of full traing set)	96
B.58 Classification success in percent (20 % of full traing set)	96
B.59 True vs. detected for Linear Discriminant Analysis (20 % of full traing set)	96
B.60 True vs. detected for Quadratic Discriminant Analysis (20 % of full traing set)	96
B.61 True vs. detected for Naive Bayes with Gaussian distribution (20 % of full traing set)	96

B.62 True vs. detected for Naive Bayes with Kernel density (20 % of full training set)	97
B.63 True vs. detected for Decision tree (20 % of full training set)	97
B.64 True vs. detected for Decision tree, pruned (20 % of full training set)	97
B.65 Classification success by number of unique LeafObjects correctly classified (10 % of full training set)	98
B.66 Classification success in percent (10 % of full training set)	98
B.67 True vs. detected for Linear Discriminant Analysis (10 % of full training set)	98
B.68 True vs. detected for Quadratic Discriminant Analysis (10 % of full training set)	98
B.69 True vs. detected for Naive Bayes with Gaussian distribution (10 % of full training set)	98
B.70 True vs. detected for Naive Bayes with Kernel density (10 % of full training set)	99
B.71 True vs. detected for Decision tree (10 % of full training set)	99
B.72 True vs. detected for Decision tree, pruned (10 % of full training set)	99

List of Figures

1.1	Asterix prototype robot frame (Adigo AS)	2
1.2	Asterix principal illustration where the white dots corresponds to herbicide droplets for weed removal (Adigo AS)	3
2.1	HSV cylinder (wikipedia.org)	9
2.2	Segmentation of a carrot leaf (Adigo AS).	11
2.3	Connected components analysis where each color represents an ID (Adigo AS).	12
2.4	Illustration of a skeleton graph on a leaf (Adigo AS).	14
2.5	Illustration of a skeleton distance image where the distance is indicated with a lighter color (Adigo AS).	15
2.6	Illustration of resulting classification areas of LDA and QDA with the same input (from results presented in section 4.2).	18
2.7	An example of naive Bayes (Zhang, 2004)	20
2.8	A simple decision tree with x being the wind speed.	22
2.9	A brief schematic description of an agent	24
2.10	The FroboMind conceptual architecture (Jensen et al., 2012)	25
3.1	System architecture of the Asterix project. The stippled lines indicates the main three parts in the project and the green boxes are the ones that this thesis involves.	28
3.2	Abstract class diagram	29

LIST OF FIGURES

3.3	A SourceImage example (Adigo AS)	31
3.4	SourceImage converted into HSV color space	32
3.5	The decomposed image in figure 3.4.	33
3.6	Resulting segmented binary image	34
3.7	Resulting segmented image in RGB	35
3.8	Labeled image with random colors indicating unique objects	36
3.9	An example of six different classes labeled.	38
3.10	The graphical user interface for training mode without overlay	39
3.11	The graphical user interface for training mode with overlay	40
3.12	Table showing the features of a selected leaf	41
4.1	Camera setup in field (Adigo AS)	43
4.2	SourceImage with contours of the detected presence of leaves	44
4.3	Scatter plot of density vs highest skeleton distance.	47
4.4	Scatter plot of average hue value vs average saturation value.	47
4.5	3d scatter plot of density, saturation and highest skeleton distance factor.	48
4.6	Resulting decision tree.	49
4.7	Resulting decision tree, pruned.	49
4.8	Plots of the different classifiers and their performance based on training size.	50
C.1	Total class diagram	101
C.2	Presentation of the classes; DataHandler, LeafClassifier, FeatureExtractor, ObjectDetector and LogWriter	102
C.3	Presentation of the classes; DataFrame and LeafObject	103
C.4	Presentation of the classes; UIQT	104
D.1	Scatter plot of circumference vs highest skeleton distance.	105
D.2	Scatter plot of circumference vs sum of skeleton distance.	106
D.3	Scatter plot of circumference vs size of skeleton.	106
D.4	Scatter plot of density vs sum of skeleton distance.	107

D.5	Scatter plot of density vs highest skeleton distance factor.	107
D.6	Scatter plot of density vs sum of skeleton distance factor.	108
D.7	Scatter plot of density vs size of skeleton.	108
D.8	Scatter plot of highest skeleton distance factor vs sum of skeleton distance factor.	109
D.9	Scatter plot of size vs circumference.	109
D.10	Scatter plot of size vs highest skeleton distance.	110
D.11	Scatter plot of size vs sum of skeleton distance.	110
D.12	Scatter plot of size vs size of skeleton.	111
D.13	Scatter plot of sum of skeleton distance vs size of skeleton.	111
D.14	Scatter plot of size of skeleton vs highest skeleton distance factor.	112
D.15	Scatter plot of size of skeleton vs sum of skeleton distance factor.	112

LIST OF FIGURES

Appendix A

Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Network
CMY	Cyan Magenta Yellow
CPU	Central processing unit
CSV	Comma Separated Value
DB	Data Base
DT	Decision Tree
GPS	Global Positioning System
GPU	Graphics Processing Unit
GUI	Graphical user interface
HSV	Hue Saturation Value
IDE	Integrated Development Environment
IMU	Inertial Measurement Unit
IR	Infra Red
LDA	Linear Discriminant Analysis

NBC	Naive Bayesian Classifier
nbGau	Native Bayesian Gaussian Classifier
nbKd	Native Bayesian Kernel Classifier
NIR	Near Infra Red
NTNU	Norwegian University of Science and Technology
PA	Precision Agriculture
PNN	Probabilistic Neural Network
QDA	Quadratic Discriminant Analysis
RGB	Red Green Blue
UN	United Nations
UV	Ultra-Violet

Appendix B

Classification results

	LDA	QDA	DT	DTp	nbGau	nbKd	voted	total
Carrot 1st stage	207	211	196	192	214	206	210	223
Carrot 2nd stage	120	128	125	126	120	118	126	133
Weed	271	267	285	280	250	287	288	318
Total	598	606	607	597	584	611	624	674

Table B.1: Classification success by number of unique LeafObjects correctly classified (90 % of full traing set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted
Carrot 1st stage	92.83 %	94.62 %	87.89 %	86.10 %	95.96 %	92.38 %	94.17 %
Carrot 2nd stage	90.23 %	96.24 %	93.98 %	94.74 %	90.23 %	88.72 %	94.74 %
Weed	85.22 %	83.96 %	89.62 %	88.05 %	78.62 %	90.25 %	90.57 %
Total	88.72 %	89.91 %	90.06 %	88.58 %	86.65 %	90.65 %	92.58 %

Table B.2: Classification success in percent
(90 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	92.83 %	1.79 %	5.38 %
	Carrot 2nd stage	7.52 %	90.23 %	2.26 %
	Weed	12.58 %	2.20 %	85.22 %

Table B.3: True vs. detected for Linear Discriminant Analysis
(90 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	94.62 %	4.04 %	1.35 %
	Carrot 2nd stage	3.01 %	96.24 %	0.75 %
	Weed	11.95 %	4.09 %	83.96 %

Table B.4: True vs. detected for Quadratic Discriminant Analysis
(90 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	95.96 %	1.79 %	2.24 %
	Carrot 2nd stage	6.77 %	90.23 %	3.01 %
	Weed	18.55 %	2.83 %	78.62 %

Table B.5: True vs. detected for Naive Bayes with Gaussian distribution
(90 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	92.38 %	1.35 %	6.28 %
	Carrot 2nd stage	5.26 %	88.72 %	6.02 %
	Weed	7.55 %	1.89 %	90.25 %

Table B.6: True vs. detected for Naive Bayes with Kernel density
(90 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	87.89 %	2.69 %	9.42 %
	Carrot 2nd stage	1.50 %	94.74 %	3.76 %
	Weed	5.35 %	5.03 %	89.62 %

Table B.7: True vs. detected for Decision tree
(90 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	86.10 %	3.14 %	10.76 %
	Carrot 2nd stage	4.51 %	93.98 %	1.50 %
	Weed	8.18 %	3.77 %	88.05 %

Table B.8: True vs. detected for Decision tree, pruned
(90 % of full traing set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted	total
Carrot 1st stage	208	212	191	192	213	208	210	223
Carrot 2nd stage	120	126	124	122	119	116	125	133
Weed	268	265	282	281	244	284	285	318
Total	596	603	595	597	576	608	620	674

Table B.9: Classification success by number of unique LeafObjects correctly classified (80 % of full training set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted
Carrot 1st stage	93.27 %	95.07 %	85.65 %	86.10 %	95.52 %	93.27 %	94.17 %
Carrot 2nd stage	90.23 %	94.74 %	93.23 %	91.73 %	89.47 %	87.22 %	93.98 %
Weed	84.28 %	83.33 %	88.68 %	88.36 %	76.73 %	89.31 %	89.62 %
Total	88.43 %	89.47 %	88.28 %	88.58 %	85.46 %	90.21 %	91.99 %

Table B.10: Classification success in percent (80 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	93.27 %	1.79 %	4.93 %
	Carrot 2nd stage	7.52 %	90.23 %	2.26 %
	Weed	13.21 %	2.52 %	84.28 %

Table B.11: True vs. detected for Linear Discriminant Analysis (80 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	95.07 %	4.04 %	0.90 %
	Carrot 2nd stage	3.01 %	94.74 %	2.26 %
	Weed	13.21 %	3.46 %	83.33 %

Table B.12: True vs. detected for Quadratic Discriminant Analysis (80 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	95.52 %	2.24 %	2.24 %
	Carrot 2nd stage	7.52 %	89.47 %	3.01 %
	Weed	20.13 %	3.14 %	76.73 %

Table B.13: True vs. detected for Naive Bayes with Gaussian distribution (80 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	93.27 %	1.79 %	4.93 %
	Carrot 2nd stage	6.02 %	87.22 %	6.77 %
	Weed	8.18 %	2.20 %	89.31 %

Table B.14: True vs. detected for Naive Bayes with Kernel density (80 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	85.65 %	4.93 %	9.42 %
	Carrot 2nd stage	3.76 %	91.73 %	4.51 %
	Weed	6.29 %	5.03 %	88.68 %

Table B.15: True vs. detected for Decision tree (80 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	86.10 %	3.14 %	10.76 %
	Carrot 2nd stage	4.51 %	93.23 %	2.26 %
	Weed	8.18 %	3.46 %	88.36 %

Table B.16: True vs. detected for Decision tree, pruned (80 % of full training set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted	total
Carrot 1st stage	208	211	200	196	212	206	208	223
Carrot 2nd stage	120	127	123	125	118	114	125	133
Weed	268	267	283	271	245	286	284	318
Total	596	605	608	590	575	606	617	674

Table B.17: Classification success by number of unique LeafObjects correctly classified (70 % of full training set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted
Carrot 1st stage	93.27 %	94.62 %	89.69 %	87.89 %	95.07 %	92.38 %	93.27 %
Carrot 2nd stage	90.23 %	95.49 %	92.48 %	93.98 %	88.72 %	85.71 %	93.98 %
Weed	84.28 %	83.96 %	88.99 %	85.22 %	77.04 %	89.94 %	89.31 %
Total	88.43 %	89.76 %	90.21 %	87.54 %	85.31 %	89.91 %	91.54 %

Table B.18: Classification success in percent (70 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	93.27 %	1.79 %	4.93 %
	Carrot 2nd stage	7.52 %	90.23 %	2.26 %
	Weed	12.89 %	2.83 %	84.28 %

Table B.19: True vs. detected for Linear Discriminant Analysis (70 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	94.62 %	4.04 %	1.35 %
	Carrot 2nd stage	2.26 %	95.49 %	2.26 %
	Weed	12.89 %	3.14 %	83.96 %

Table B.20: True vs. detected for Quadratic Discriminant Analysis (70 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	95.07 %	2.69 %	2.24 %
	Carrot 2nd stage	8.27 %	88.72 %	3.01 %
	Weed	20.13 %	2.83 %	77.04 %

Table B.21: True vs. detected for Naive Bayes with Gaussian distribution (70 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	92.38 %	2.24 %	5.38 %
	Carrot 2nd stage	6.77 %	85.71 %	7.52 %
	Weed	7.55 %	2.20 %	89.94 %

Table B.22: True vs. detected for Naive Bayes with Kernel density
(70 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	89.69 %	3.14 %	7.17 %
	Carrot 2nd stage	3.76 %	93.98 %	2.26 %
	Weed	6.92 %	4.09 %	88.99 %

Table B.23: True vs. detected for Decision tree
(70 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	87.89 %	2.69 %	9.42 %
	Carrot 2nd stage	4.51 %	92.48 %	3.01 %
	Weed	9.12 %	5.66 %	85.22 %

Table B.24: True vs. detected for Decision tree, pruned
(70 % of full training set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted	total
Carrot 1st stage	206	211	196	190	212	206	210	223
Carrot 2nd stage	120	122	100	121	111	112	117	133
Weed	269	269	289	292	239	286	286	318
Total	595	602	606	582	562	604	613	674

Table B.25: Classification success by number of unique LeafObjects correctly classified (60 % of full traing set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted
Carrot 1st stage	92.38 %	94.62 %	87.89 %	85.20 %	95.07 %	92.38 %	94.17 %
Carrot 2nd stage	90.23 %	91.73 %	75.19 %	90.98 %	83.46 %	84.21 %	87.97 %
Weed	84.59 %	84.59 %	90.88 %	91.82 %	75.16 %	89.94 %	89.94 %
Total	88.28 %	89.32 %	89.91 %	86.35 %	83.38 %	89.61 %	90.95 %

Table B.26: Classification success in percent (60 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	92.38 %	1.79 %	5.83 %
	Carrot 2nd stage	7.52 %	90.23 %	2.26 %
	Weed	13.21 %	2.20 %	84.59 %

Table B.27: True vs. detected for Linear Discriminant Analysis (60 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	94.62 %	4.04 %	1.35 %
	Carrot 2nd stage	3.01 %	91.73 %	5.26 %
	Weed	12.89 %	2.52 %	84.59 %

Table B.28: True vs. detected for Quadratic Discriminant Analysis (60 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	95.07 %	2.69 %	2.24 %
	Carrot 2nd stage	10.53 %	83.46 %	6.02 %
	Weed	22.64 %	2.20 %	75.16 %

Table B.29: True vs. detected for Naive Bayes with Gaussian distribution (60 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	92.38 %	2.24 %	5.38 %
	Carrot 2nd stage	7.52 %	84.21 %	8.27 %
	Weed	7.55 %	2.20 %	89.94 %

Table B.30: True vs. detected for Naive Bayes with Kernel density
(60 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	87.89 %	3.14 %	8.97 %
	Carrot 2nd stage	3.01 %	90.98 %	6.02 %
	Weed	5.97 %	3.14 %	90.88 %

Table B.31: True vs. detected for Decision tree
(60 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	85.20 %	2.24 %	12.56 %
	Carrot 2nd stage	12.03 %	75.19 %	12.78 %
	Weed	7.86 %	0.31 %	91.82 %

Table B.32: True vs. detected for Decision tree, pruned
(60 % of full training set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted	total
Carrot 1st stage	209	212	192	184	213	207	207	223
Carrot 2nd stage	123	123	113	104	110	110	116	133
Weed	266	266	276	280	226	282	282	318
Total	598	601	572	577	549	599	605	674

Table B.33: Classification success by number of unique LeafObjects correctly classified (50 % of full training set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted
Carrot 1st stage	93.72 %	95.07 %	86.10 %	82.51 %	95.52 %	92.83 %	92.83 %
Carrot 2nd stage	92.48 %	92.48 %	84.96 %	78.20 %	82.71 %	82.71 %	87.22 %
Weed	83.65 %	83.65 %	86.79 %	88.05 %	71.07 %	88.68 %	88.68 %
Total	88.72 %	89.17 %	84.87 %	85.61 %	81.45 %	88.87 %	89.76 %

Table B.34: Classification success in percent (50 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	93.72 %	1.79 %	4.48 %
	Carrot 2nd stage	5.26 %	92.48 %	2.26 %
	Weed	13.84 %	2.52 %	83.65 %

Table B.35: True vs. detected for Linear Discriminant Analysis (50 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	95.07 %	4.04 %	0.90 %
	Carrot 2nd stage	3.76 %	92.48 %	3.76 %
	Weed	13.84 %	2.52 %	83.65 %

Table B.36: True vs. detected for Quadratic Discriminant Analysis (50 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	95.52 %	2.69 %	1.79 %
	Carrot 2nd stage	11.28 %	82.71 %	6.02 %
	Weed	26.42 %	2.52 %	71.07 %

Table B.37: True vs. detected for Naive Bayes with Gaussian distribution (50 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	92.83 %	2.24 %	4.93 %
	Carrot 2nd stage	9.02 %	82.71 %	8.27 %
	Weed	8.49 %	2.52 %	88.68 %

Table B.38: True vs. detected for Naive Bayes with Kernel density (50 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	86.10 %	3.14 %	10.76 %
	Carrot 2nd stage	10.53 %	78.20 %	11.28 %
	Weed	10.06 %	3.14 %	86.79 %

Table B.39: True vs. detected for Decision tree (50 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	82.51 %	2.24 %	15.25 %
	Carrot 2nd stage	12.03 %	84.96 %	3.01 %
	Weed	6.60 %	5.35 %	88.05 %

Table B.40: True vs. detected for Decision tree, pruned (50 % of full training set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted	total
Carrot 1st stage	209	212	188	202	214	210	213	223
Carrot 2nd stage	122	125	112	118	114	111	120	133
Weed	269	268	284	277	209	285	287	318
Total	600	605	590	591	537	606	620	674

Table B.41: Classification success by number of unique LeafObjects correctly classified (40 % of full traing set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted
Carrot 1st stage	93.72 %	95.07 %	84.30 %	90.58 %	95.96 %	94.17 %	95.52 %
Carrot 2nd stage	91.73 %	93.98 %	84.21 %	88.72 %	85.71 %	83.46 %	90.23 %
Weed	84.59 %	84.28 %	89.31 %	87.11 %	65.72 %	89.62 %	90.25 %
Total	89.02 %	89.76 %	87.54 %	87.69 %	79.67 %	89.91 %	91.99 %

Table B.42: Classification success in percent (40 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	93.72 %	1.79 %	4.48 %
	Carrot 2nd stage	6.02 %	91.73 %	2.26 %
	Weed	13.21 %	2.20 %	84.59 %

Table B.43: True vs. detected for Linear Discriminant Analysis (40 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	95.07 %	4.04 %	0.90 %
	Carrot 2nd stage	2.26 %	93.98 %	3.76 %
	Weed	13.21 %	2.52 %	84.28 %

Table B.44: True vs. detected for Quadratic Discriminant Analysis (40 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	95.96 %	2.69 %	1.35 %
	Carrot 2nd stage	9.77 %	85.71 %	4.51 %
	Weed	32.39 %	1.89 %	65.72 %

Table B.45: True vs. detected for Naive Bayes with Gaussian distribution (40 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	94.17 %	1.35 %	4.48 %
	Carrot 2nd stage	8.27 %	83.46 %	8.27 %
	Weed	8.18 %	1.89 %	89.62 %

Table B.46: True vs. detected for Naive Bayes with Kernel density (40 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	84.30 %	4.04 %	11.66 %
	Carrot 2nd stage	3.76 %	88.72 %	7.52 %
	Weed	7.23 %	3.46 %	89.31 %

Table B.47: True vs. detected for Decision tree (40 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	90.58 %	3.59 %	5.83 %
	Carrot 2nd stage	3.76 %	84.21 %	12.03 %
	Weed	10.38 %	2.52 %	87.11 %

Table B.48: True vs. detected for Decision tree, pruned (40 % of full training set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted	total
Carrot 1st stage	209	212	186	214	214	208	215	223
Carrot 2nd stage	121	121	91	114	112	107	109	133
Weed	276	275	276	241	228	271	280	318
Total	606	608	576	546	554	586	604	674

Table B.49: Classification success by number of unique LeafObjects correctly classified (30 % of full traing set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted
Carrot 1st stage	93.72 %	95.07 %	83.41 %	95.96 %	95.96 %	93.27 %	96.41 %
Carrot 2nd stage	90.98 %	90.98 %	68.42 %	85.71 %	84.21 %	80.45 %	81.95 %
Weed	86.79 %	86.48 %	86.79 %	75.79 %	71.70 %	85.22 %	88.05 %
Total	89.91 %	90.21 %	85.46 %	81.01 %	82.20 %	86.94 %	89.61 %

Table B.50: Classification success in percent (30 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	93.72 %	1.79 %	4.48 %
	Carrot 2nd stage	6.77 %	90.98 %	2.26 %
	Weed	11.64 %	1.57 %	86.79 %

Table B.51: True vs. detected for Linear Discriminant Analysis (30 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	95.07 %	3.59 %	1.35 %
	Carrot 2nd stage	3.76 %	90.98 %	5.26 %
	Weed	11.64 %	1.89 %	86.48 %

Table B.52: True vs. detected for Quadratic Discriminant Analysis (30 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	95.96 %	2.24 %	1.79 %
	Carrot 2nd stage	10.53 %	84.21 %	5.26 %
	Weed	26.42 %	1.89 %	71.70 %

Table B.53: True vs. detected for Naive Bayes with Gaussian distribution (30 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	93.27 %	0.90 %	5.83 %
	Carrot 2nd stage	9.77 %	80.45 %	9.77 %
	Weed	7.23 %	6.92 %	85.22 %

Table B.54: True vs. detected for Naive Bayes with Kernel density (30 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	83.41 %	4.04 %	12.56 %
	Carrot 2nd stage	6.02 %	85.71 %	8.27 %
	Weed	9.43 %	3.77 %	86.79 %

Table B.55: True vs. detected for Decision tree (30 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	95.96 %	2.24 %	1.79 %
	Carrot 2nd stage	21.80 %	68.42 %	9.77 %
	Weed	23.58 %	0.63 %	75.79 %

Table B.56: True vs. detected for Decision tree, pruned (30 % of full training set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted	total
Carrot 1st stage	206	213	172	191	211	200	206	223
Carrot 2nd stage	127	121	111	120	124	117	121	133
Weed	267	282	279	239	238	258	278	318
Total	600	616	571	541	573	575	605	674

Table B.57: Classification success by number of unique LeafObjects correctly classified (20 % of full traing set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted
Carrot 1st stage	92.38 %	95.52 %	77.13 %	85.65 %	94.62 %	89.69 %	92.38 %
Carrot 2nd stage	95.49 %	90.98 %	83.46 %	90.23 %	93.23 %	87.97 %	90.98 %
Weed	83.96 %	88.68 %	87.74 %	75.16 %	74.84 %	81.13 %	87.42 %
Total	89.02 %	91.39 %	84.72 %	80.27 %	85.01 %	85.31 %	89.76 %

Table B.58: Classification success in percent (20 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	92.38 %	4.04 %	3.59 %
	Carrot 2nd stage	1.50 %	95.49 %	3.01 %
	Weed	13.21 %	2.83 %	83.96 %

Table B.59: True vs. detected for Linear Discriminant Analysis (20 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	95.52 %	2.24 %	2.24 %
	Carrot 2nd stage	2.26 %	90.98 %	6.77 %
	Weed	9.43 %	1.89 %	88.68 %

Table B.60: True vs. detected for Quadratic Discriminant Analysis (20 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	94.62 %	3.14 %	2.24 %
	Carrot 2nd stage	3.76 %	93.23 %	3.01 %
	Weed	21.70 %	3.46 %	74.84 %

Table B.61: True vs. detected for Naive Bayes with Gaussian distribution (20 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	89.69 %	3.14 %	7.17 %
	Carrot 2nd stage	5.26 %	87.97 %	6.77 %
	Weed	7.55 %	10.69 %	81.13 %

Table B.62: True vs. detected for Naive Bayes with Kernel density
(20 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	77.13 %	9.42 %	13.45 %
	Carrot 2nd stage	4.51 %	90.23 %	5.26 %
	Weed	9.12 %	3.14 %	87.74 %

Table B.63: True vs. detected for Decision tree
(20 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	85.65 %	12.56 %	1.79 %
	Carrot 2nd stage	6.77 %	83.46 %	9.77 %
	Weed	18.55 %	6.29 %	75.16 %

Table B.64: True vs. detected for Decision tree, pruned
(20 % of full training set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted	total
Carrot 1st stage	216	209	210	218	213	208	218	223
Carrot 2nd stage	124	85	86	115	103	107	99	133
Weed	259	290	249	239	233	217	267	318
Total	599	584	574	543	549	532	584	674

Table B.65: Classification success by number of unique LeafObjects correctly classified (10 % of full traing set)

	LDA	QDA	DT	DTp	nbGau	nbKd	voted
Carrot 1st stage	96.86 %	93.72 %	94.17 %	97.76 %	95.52 %	93.27 %	97.76 %
Carrot 2nd stage	93.23 %	63.91 %	64.66 %	86.47 %	77.44 %	80.45 %	74.44 %
Weed	81.45 %	91.19 %	78.30 %	75.16 %	73.27 %	68.24 %	83.96 %
Total	88.87 %	86.65 %	85.16 %	80.56 %	81.45 %	78.93 %	86.65 %

Table B.66: Classification success in percent (10 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	96.86 %	1.79 %	1.35 %
	Carrot 2nd stage	4.51 %	93.23 %	2.26 %
	Weed	16.35 %	2.20 %	81.45 %

Table B.67: True vs. detected for Linear Discriminant Analysis (10 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	93.72 %	1.35 %	4.93 %
	Carrot 2nd stage	18.80 %	63.91 %	17.29 %
	Weed	8.49 %	0.31 %	91.19 %

Table B.68: True vs. detected for Quadratic Discriminant Analysis (10 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	95.52 %	3.14 %	1.35 %
	Carrot 2nd stage	16.54 %	77.44 %	6.02 %
	Weed	24.53 %	2.20 %	73.27 %

Table B.69: True vs. detected for Naive Bayes with Gaussian distribution (10 % of full traing set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	93.27 %	2.24 %	4.48 %
	Carrot 2nd stage	13.53 %	80.45 %	6.02 %
	Weed	11.01 %	20.13 %	68.24 %

Table B.70: True vs. detected for Naive Bayes with Kernel density
(10 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	94.17 %	1.35 %	4.48 %
	Carrot 2nd stage	3.76 %	86.47 %	9.77 %
	Weed	15.41 %	6.29 %	78.30 %

Table B.71: True vs. detected for Decision tree
(10 % of full training set)

		Detected		
		Carrot 1st stage	Carrot 2nd stage	Weed
True	Carrot 1st stage	97.76 %	0.45 %	1.79 %
	Carrot 2nd stage	25.56 %	64.66 %	9.77 %
	Weed	19.18 %	5.66 %	75.16 %

Table B.72: True vs. detected for Decision tree, pruned
(10 % of full training set)



Appendix C

Class diagrams

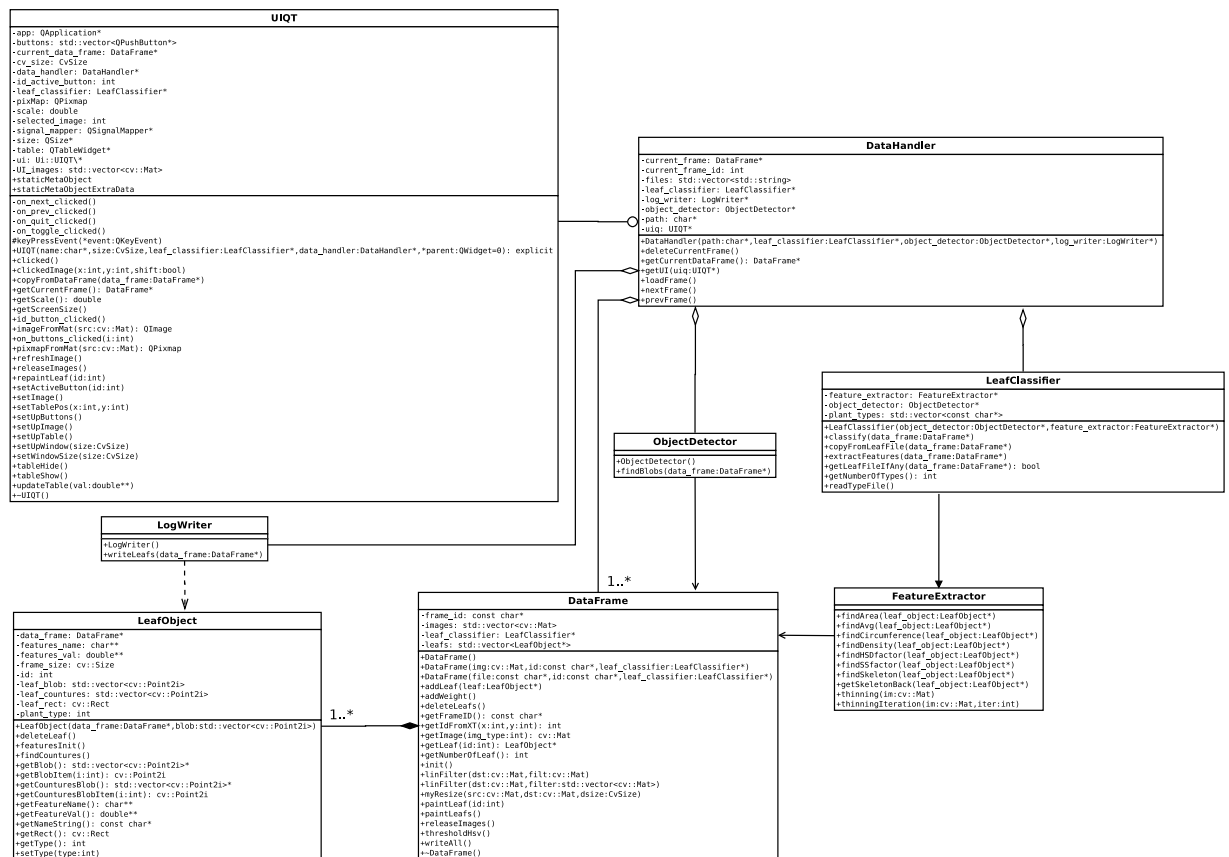


Figure C.1: Total class diagram

DataHandler
<pre> -current_frame: DataFrame* -current_frame_id: int -files: std::vector<std::string> -leaf_classifier: LeafClassifier* -log_writer: LogWriter* -object_detector: ObjectDetector* -path: char* -uiq: UIQT* </pre>
<pre> +DataHandler(path:char*,leaf_classifier:LeafClassifier*,object_detector:ObjectDetector*,log_writer:LogWriter*) +deleteCurrentFrame() +getCurrentDataFrame(): DataFrame* +getUI(uiq:UIQT*) +loadFrame() +nextFrame() +prevFrame() </pre>

LeafClassifier
<pre> -feature_extractor: FeatureExtractor* -object_detector: ObjectDetector* -plant_types: std::vector<const char*> </pre>
<pre> +LeafClassifier(object_detector:ObjectDetector*,feature_extractor:FeatureExtractor*) +classify(data_frame:DataFrame*) +copyFromLeafFile(data_frame:DataFrame*) +extractFeatures(data_frame:DataFrame*) +getLeafFileIfAny(data_frame:DataFrame*): bool +getNumberOfTypes(): int +readTypeFile() </pre>

FeatureExtractor
<pre> +findArea(leaf_object:LeafObject*) +findAvg(leaf_object:LeafObject*) +findCircumference(leaf_object:LeafObject*) +findDensity(leaf_object:LeafObject*) +findHSDfactor(leaf_object:LeafObject*) +findSSfactor(leaf_object:LeafObject*) +findSkeleton(leaf_object:LeafObject*) +getSkeletonBack(leaf_object:LeafObject*) +thinning(im:cv::Mat) +thinningIteration(im:cv::Mat,iter:int) </pre>

ObjectDetector
<pre> +ObjectDetector() +findBlobs(data_frame:DataFrame*) </pre>

LogWriter
<pre> +LogWriter() +writeLeafs(data_frame:DataFrame*) </pre>

Figure C.2: Presentation of the classes; DataHandler, LeafClassifier, FeatureExtractor, ObjectDetector and LogWriter

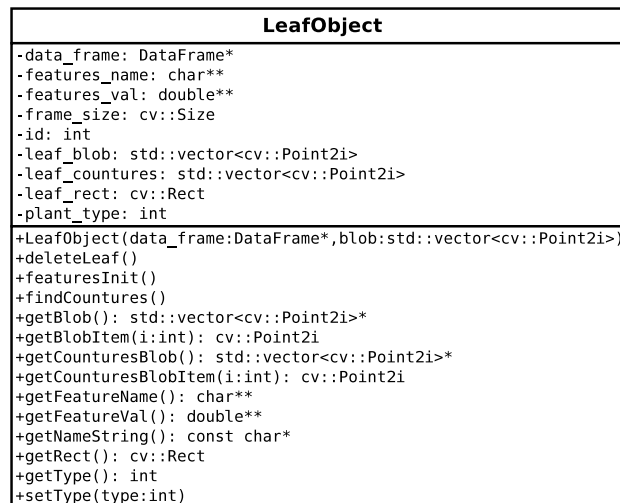
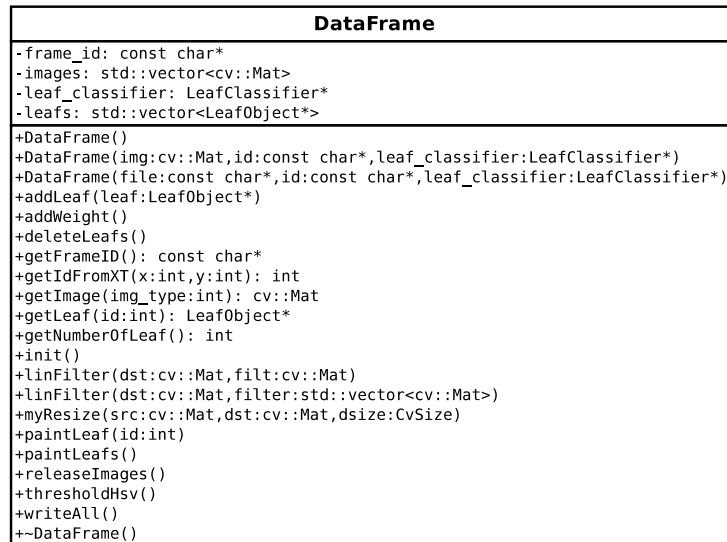


Figure C.3: Presentation of the classes; DataFrame and LeafObject

UIQT
<pre> -app: QApplication* -buttons: std::vector<QPushButton*> -current_data_frame: DataFrame* -cv_size: CvSize -data_handler: DataHandler* -id_active_button: int -leaf_classifier: LeafClassifier* -pixmap: QPixmap -scale: double -selected_image: int -signal_mapper: QSignalMapper* -size: QSize* -table: QTableWidget* -ui: Ui::UIQT* -UI_images: std::vector<cv::Mat> +staticMetaObject +staticMetaObjectExtraData </pre>
<pre> -on_next_clicked() -on_prev_clicked() -on_quit_clicked() -on_toggle_clicked() #keyPressEvent(*event:QKeyEvent) +UIQT(name:char*,size:CvSize,leaf_classifier:LeafClassifier*,data_handler:DataHandler*,*parent:QWidget=0): explicit +clicked() +clickedImage(x:int,y:int,shift:bool) +copyFromDataFrame(data_frame:DataFrame*) +getCurrentFrame(): DataFrame* +getScale(): double +getScreenSize() +id_button_clicked() +imageFromMat(src:cv::Mat): QImage +on_buttons_clicked(i:int) +pixmapFromMat(src:cv::Mat): QPixmap +refreshImage() +releaseImages() +repaintLeaf(id:int) +setActiveButton(id:int) +setImage() +setTablePos(x:int,y:int) +setUpButtons() +setUpImage() +setUpTable() +setUpWindow(size:CvSize) +setWindowSize(size:CvSize) +tableHide() +tableShow() +updateTable(val:double**) +~UIQT() </pre>

Figure C.4: Presentation of the classes; UIQT

Appendix D

Scatter plots

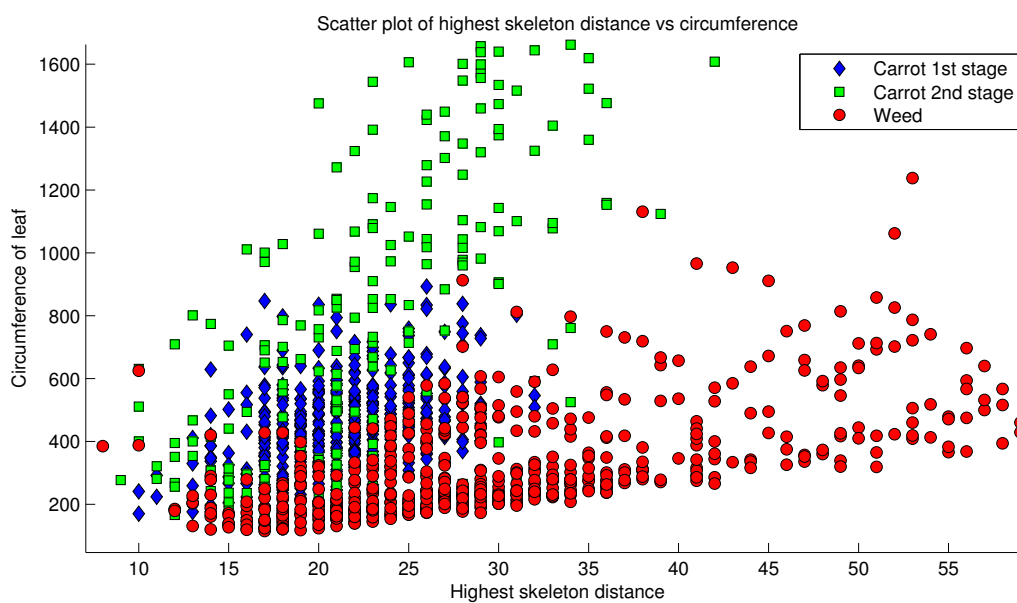


Figure D.1: Scatter plot of circumference vs highest skeleton distance.

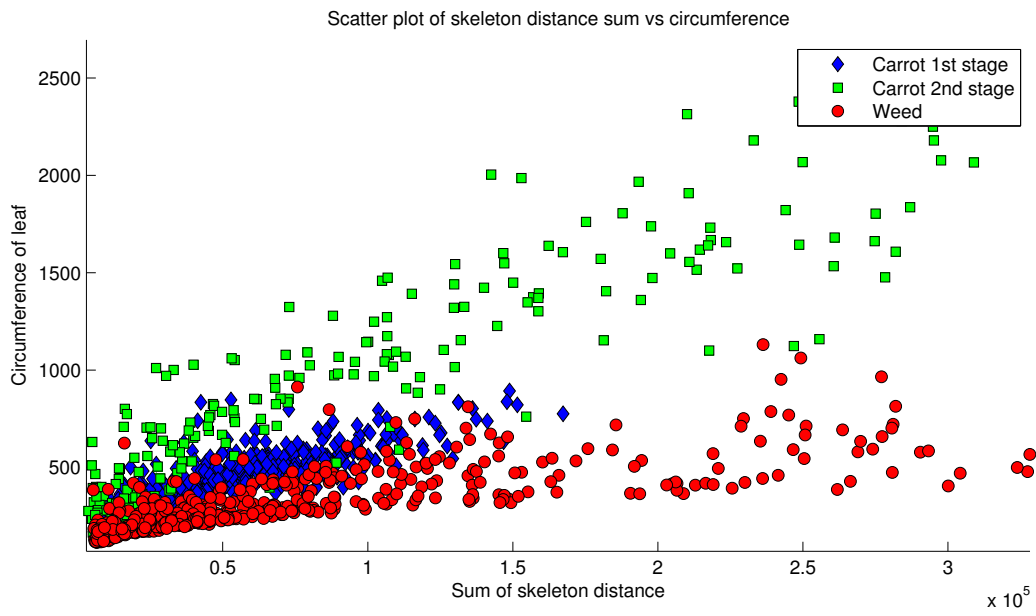


Figure D.2: Scatter plot of circumference vs sum of skeleton distance.

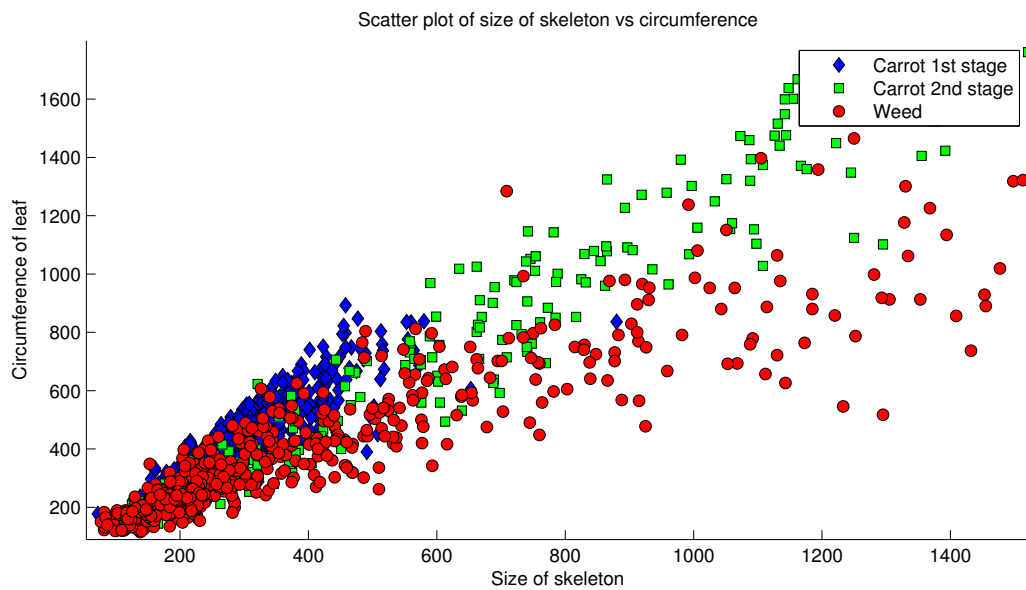


Figure D.3: Scatter plot of circumference vs size of skeleton.

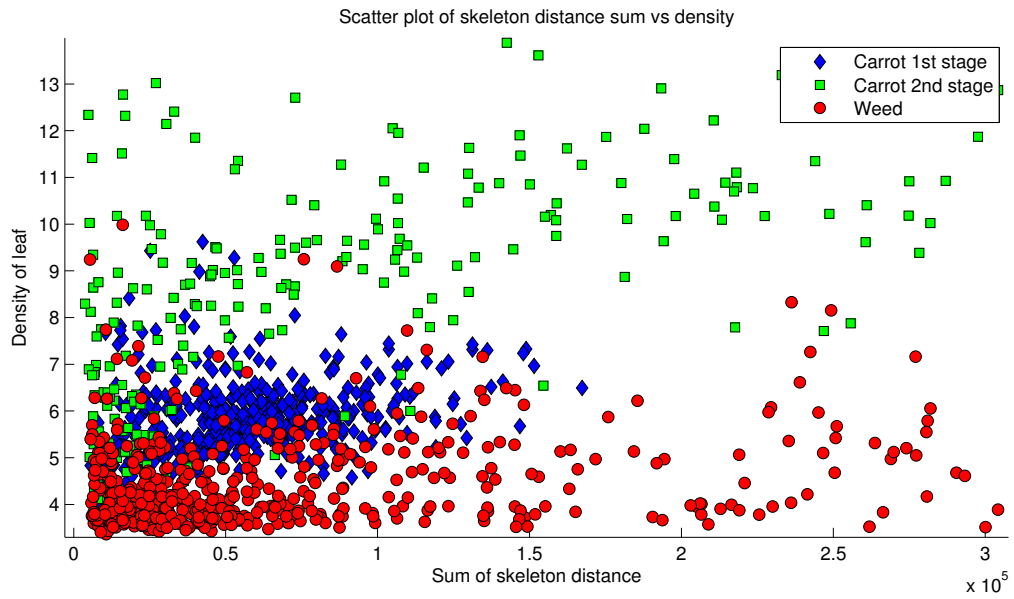


Figure D.4: Scatter plot of density vs sum of skeleton distance.

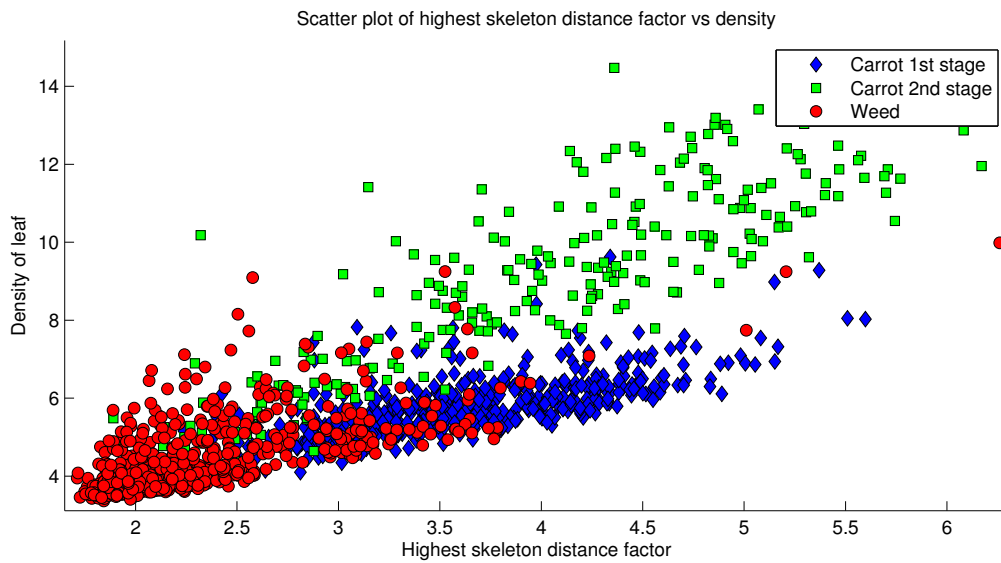


Figure D.5: Scatter plot of density vs highest skeleton distance factor.

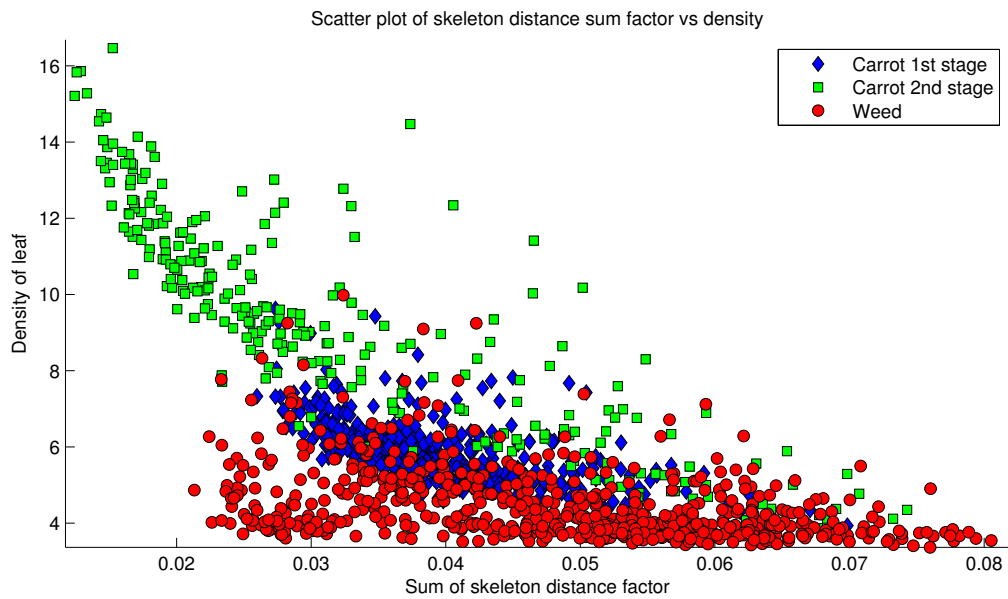


Figure D.6: Scatter plot of density vs sum of skeleton distance factor.

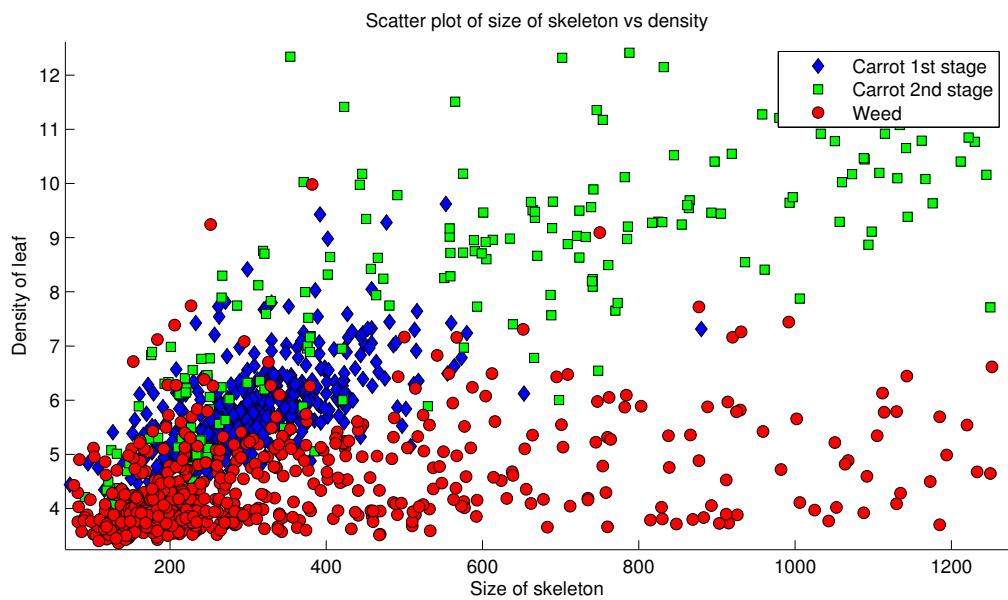


Figure D.7: Scatter plot of density vs size of skeleton.

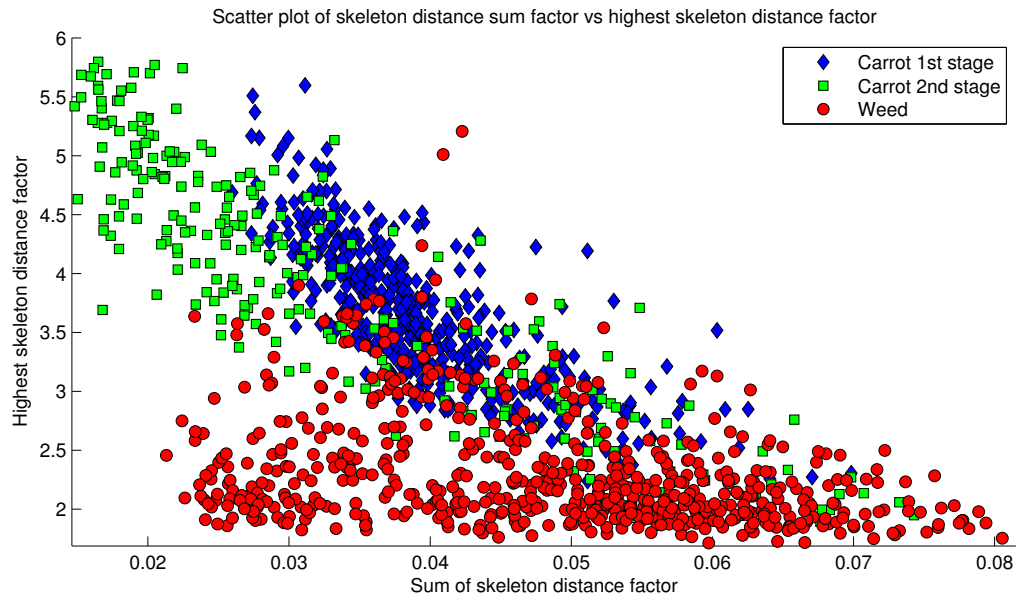


Figure D.8: Scatter plot of highest skeleton distance factor vs sum of skeleton distance factor.

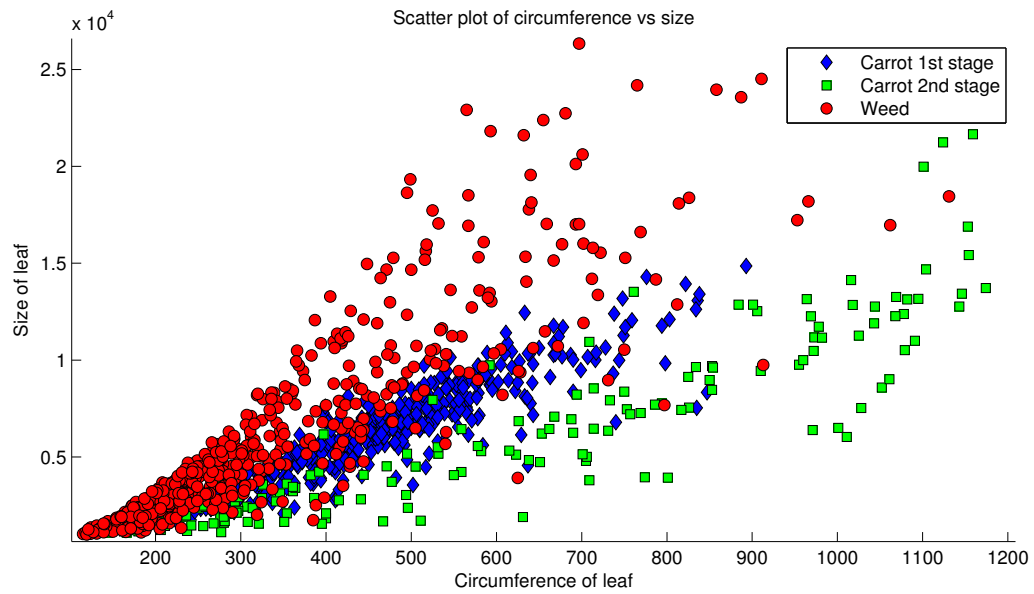


Figure D.9: Scatter plot of size vs circumference.

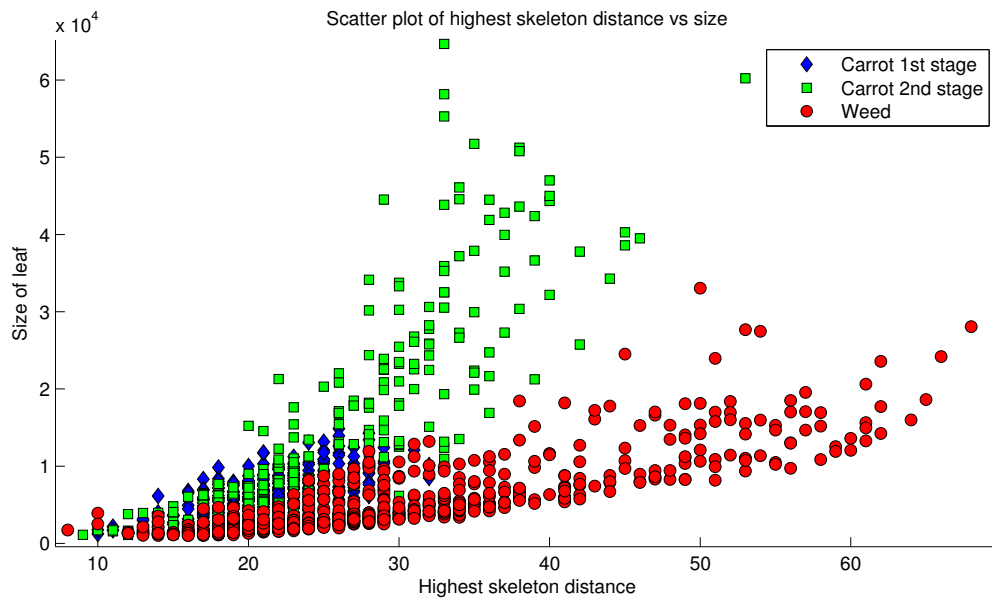


Figure D.10: Scatter plot of size vs highest skeleton distance.

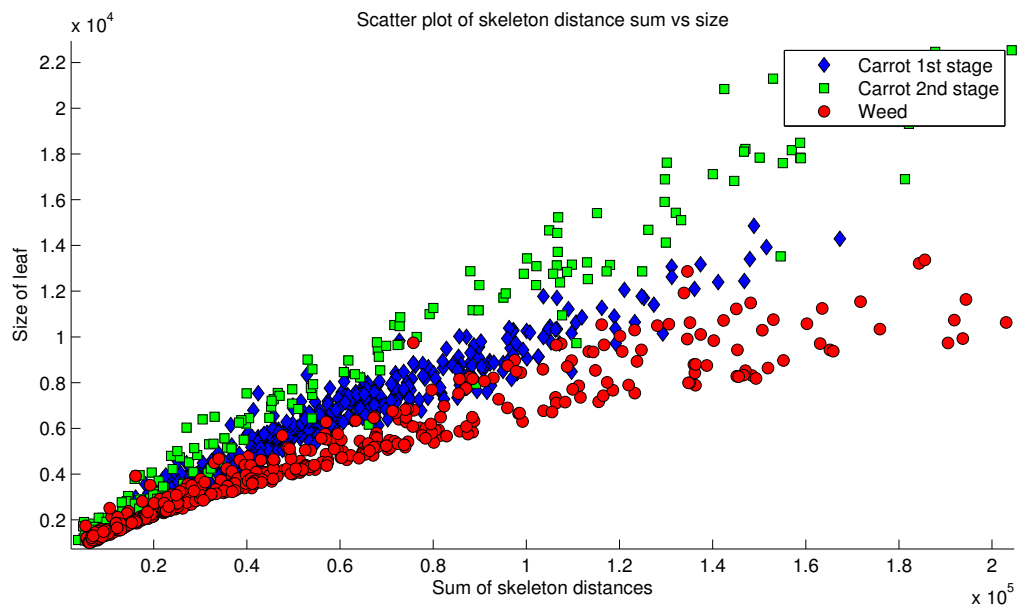


Figure D.11: Scatter plot of size vs sum of skeleton distance.

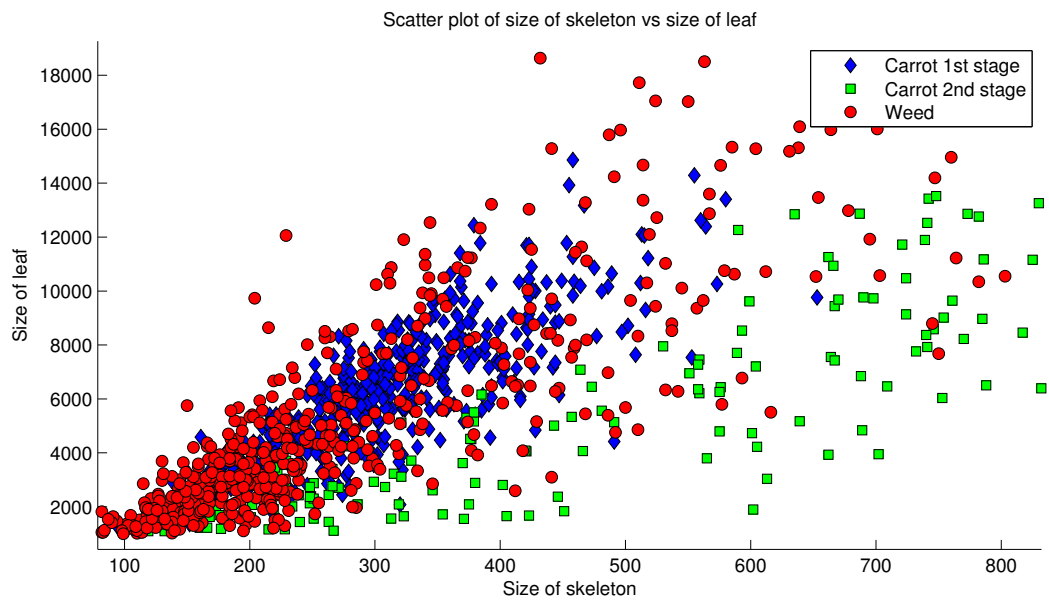


Figure D.12: Scatter plot of size vs size of skeleton.

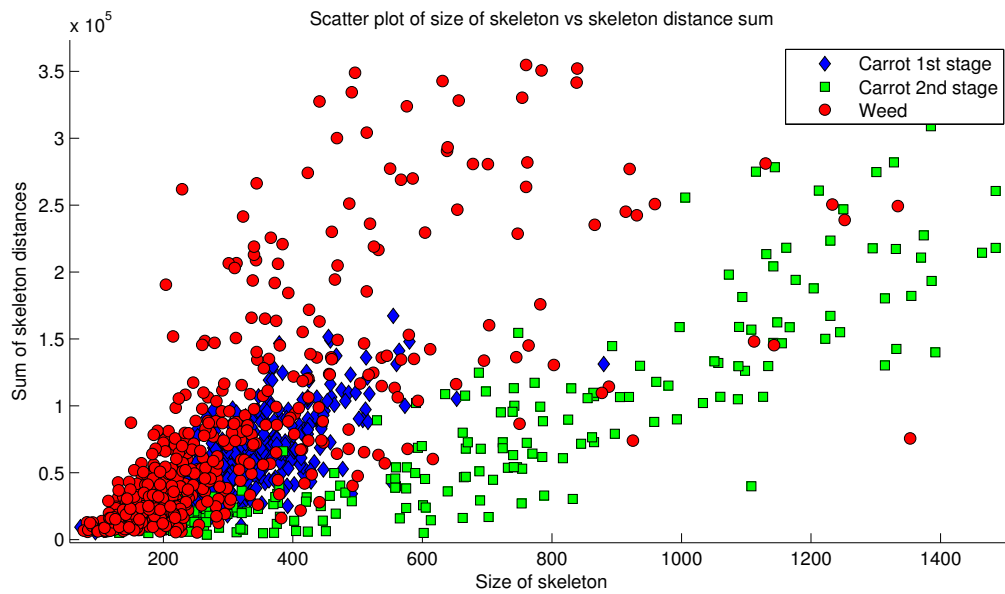


Figure D.13: Scatter plot of sum of skeleton distance vs size of skeleton.

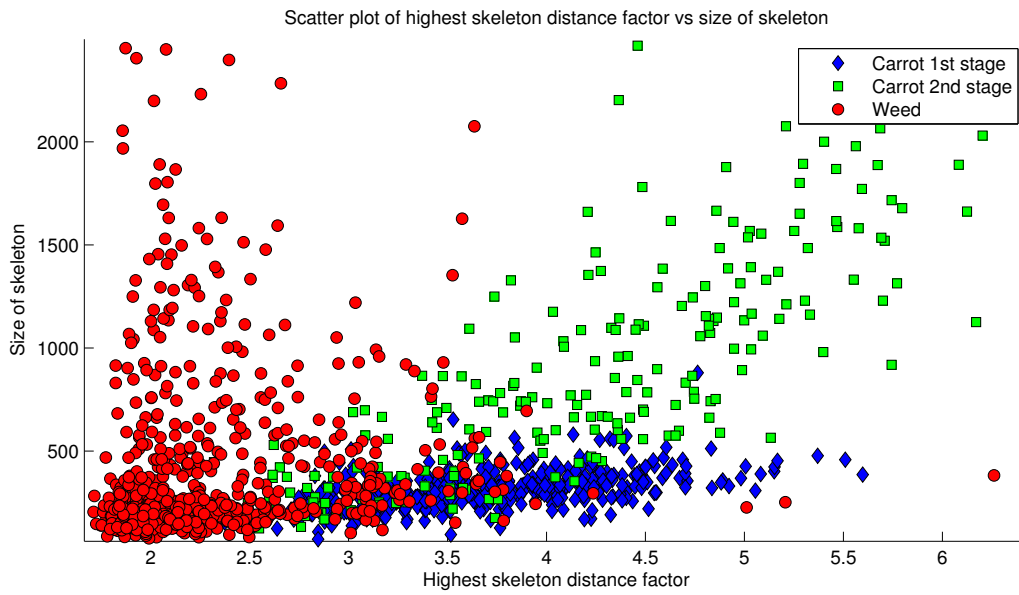


Figure D.14: Scatter plot of size of skeleton vs highest skeleton distance factor.

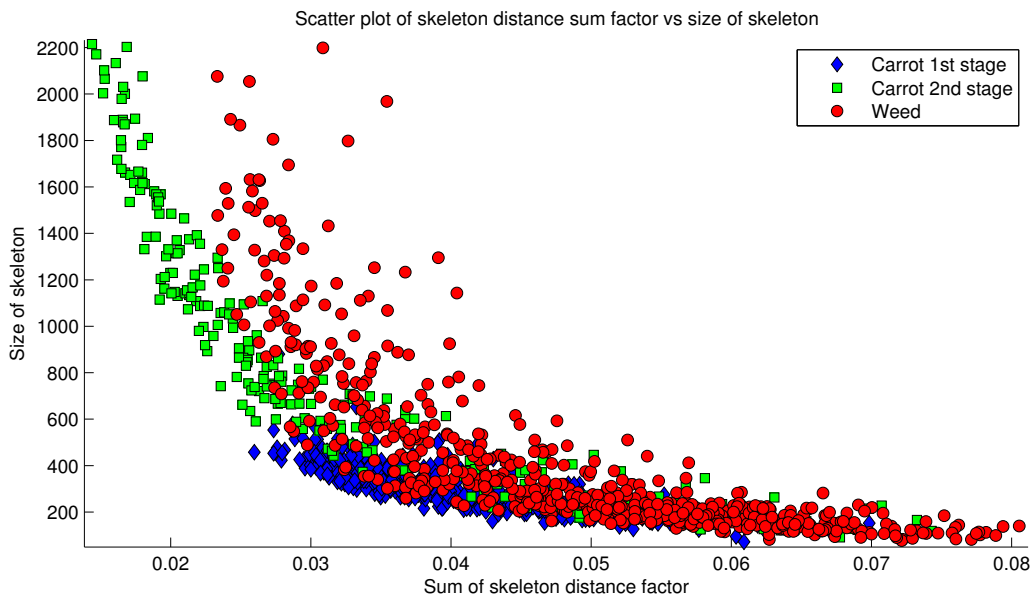


Figure D.15: Scatter plot of size of skeleton vs sum of skeleton distance factor.

Appendix E

DataHandler source code

This appendix presents the code from the DataHandler class, presented in chapter 3. This is given for the reader to see an example of the implementation.

```
1 #include <dirent.h>
2 #include <iostream>
3 #include <string>
4 #include <stdio.h>
5
6 #include "DataFrame.h"
7 #include "ObjectDetector.h"
8 #include "LogWriter.h"
9 #include "FeatureExtractor.h"
10 #include "LeafClassifier.h"
11 #include "DataHandler.h"
12 #include "UIQT.h"
13
14 DataHandler::DataHandler(char* path, LeafClassifier* leaf_classifier,
15     ObjectDetector* object_detector, LogWriter* log_writer){
16     this->path = path;
17     this->leaf_classifier = leaf_classifier;
18     this->object_detector = object_detector;
19     this->log_writer = log_writer;
20     this->current_frame = NULL;
```

```

20
21     DIR*     dir;
22     dirent*  pdir;
23
24     dir = opendir(path);      // open given directory
25
26     while ((pdir = readdir(dir)) { //find all images in directory
27         std::string filename = std::string(pdir->d_name);
28         if(filename.find(".jpg") != std::string::npos || filename.find(
29         ".JPG") != std::string::npos){
30             this->files.push_back(filename);
31         }
32     }
33     for(size_t i = 0; i < this->files.size(); i++){ //for information
34         while running
35         std::cout << this->files[i] << std::endl;
36     }
37     closedir(dir);
38     if(this->files.size() == 0){
39         perror("No images in folder");
40     }
41     this->current_frame_id = 0;
42     //this->loadFrame();
43 }
44
45 void DataHandler::getUI(UIQT* uiq){
46     this->uiq = uiq;
47 }
48
49 void DataHandler::nextFrame(){
50     if(this->current_frame_id != (int)this->files.size()-1){
51         this->current_frame_id++;

```



```
52 }
53 this->loadFrame();
54 }
55
56 void DataHandler::prevFrame(){
57     if(this->current_frame_id != 0){
58         this->current_frame_id--;
59     }
60     this->loadFrame();
61 }
62
63 void DataHandler::loadFrame(){ //this function is called for each new
64     image
65     DataFrame* prev_frame = NULL;
66
67     if(this->current_frame != NULL){
68         prev_frame = this->current_frame;
69     }
70
71     std::cout << "loading new data_frame" << std::endl;
72     std::cout << this->current_frame_id << std::endl;
73
74     char* file_id = new char[100];
75     strcpy (file_id, std::string(this->files[this->current_frame_id].
76         substr(0, this->files[this->current_frame_id].find(".JPG"))).c_str
77         ());
78     std::cout << "loading new data_frame" << std::endl;
79
80     std::cout << "opening image: " <<(std::string(this->path) + this->
81         files[this->current_frame_id]).c_str() << std::endl;
82     this->current_frame = new DataFrame((std::string(this->path) + this
83         ->files[this->current_frame_id]).c_str(),
84         file_id,
85         leaf_classifier); //Created new DataFrame
```

```

81
82 std::cout << (std::string(this->files[this->current_frame_id].
    substr(0,this->files[this->current_frame_id].find(".JPG")))).c_str
    () << std::endl;
83
84 std::cout << "created data_frame" << std::endl;
85
86 object_detector->findBlobs(this->current_frame); //connected
    component analysis
87 std::cout << "found blobs init" << std::endl;
88
89 leaf_classifier->classify(this->current_frame); //feature
    extraction
90 std::cout << "did classify" << std::endl;
91
92 this->current_frame->paintLeafs(); //for output in development
    process.
93 std::cout << "painted leafs" << std::endl;
94
95 uiq->copyFromDataFrame(this->current_frame); //copies the image
    from DataFrame to present in the GUI
96 std::cout << "copied from data_frame to ui" << std::endl;
97
98 this->current_frame->writeAll(); //stores all image for development
    analysis
99 std::cout << "wrote all images" << std::endl;
100
101 std::cout << "deleting old data_frame" << std::endl;
102 if(prev_frame != NULL){ //logs the last DataFrame if any. This
    includes feature values and labeled class
103     log_writer->writeLeafs(prev_frame);
104     std::cout << "written leafs" << std::endl;
105     delete prev_frame;
106 }

```

```
107
108 }
109
110 DataFrame* DataHandler::getCurrentDataFrame(){
111     return this->current_frame;
112 }
113
114 void DataHandler::deleteCurrentFrame(){ //logging and deleting
115     current image on quit
116     log_writer->writeLeafs(this->current_frame);
117     std::cout << "written leafs" << std::endl;
118     delete current_frame;
119     this->current_frame = NULL;
120 }
```

code/DataHandler.cpp