Mads Dyrmann

Peter Christiansen

AUTOMATED CLASSIFICATION OF SEEDLINGS USING COMPUTER VISION

SOLIDUM PETIT IN PROFUNDIS · UNIVERSITAS ARHUSIENSIS ·

 Peter Christiansen[1] & Mads Dyrmann[2]
[1]repetepc@gmail.com
[2]madsdyrmannlarsen@gmail.com

Aarhus University
School of Engineering, Technical Information Technology
Finlandsgade 22
DK-8200 Århus N, Denmark
Phone +45 4189 3000, Fax +45 4189 3100
contact@ase.au.dk
www.iha.dk

1st edition
Date: June 20, 2014
International Standard Book Number (ISBN-13): 978-87-997533-0-7 (eBook-pdf)

Abstract

The objective of this project is to investigate the possibilities of recognizing plant species at multiple growth stages based on RGB images. Plants and leaves are initially segmented from a database through a partly automated procedure providing samples of 2438 plants and 4767 leaves distributed on seven different species. The segmentation process finds plant elements through a colour segmentation method combining excessive green and excessive red and the Plant Stem Emerging Point algorithm to separate leaves from plants. These plant elements are then described by 50 different feature descriptors or 261 single feature values jointly describing shape, contour and colour of the plant. The 50 features include many common features. Nevertheless, three of the most distinct features are; the proposed and better performing variation of the Distance Transform feature, a fully rotational variant Elliptic Fourier descriptor, and the proposed feature, that measures the distance between the adjacent Fourier approximations of contours. A subset of the features are selected through different selection methods in order to improve the classification accuracy for three different classifiers; the Multivariate Gaussian classifier, the k-Nearest Neighbour classifier and the Support Vector Machine classifier. Finally, classifier fusion is performed by using Bayes Belief Integration to combine the classification for the whole plant with the individual classifications of the leaves of the plant in order to identify the most likely species of the given plant. By exploiting the discriminating advantages of plants and leaves an improved identification accuracy of 95.8% is achieved.

Table of Contents

Acknowledgement

We would like to express gratitude to our two supervisors Henrik Karstoft and Rasmus Nyholm Jørgensen for their contribution in this project. They have shown great enthusiasm and provided crucial criticism and feedback.
We would also like to express our appreciations for our cooperation with Henrik Skov Midtiby and Thomas Mosgaard Giselsson who have given feedback and provided us with the image database used in this project.
The report is part of the project "Graduering af fungicider og herbicider i kartofler og korn", journal number 3405-10-0162, founded by GUDP (Grønt Udviklings- og Demonstrations Program) under the Danish Ministry for Food, Agriculture and Fisheries.

Nomenclature

Table 1
THE TERMINOLOGY USED IN THIS REPORT

Acronym	Description	page
BBCH	**B**undesanstalt, **B**undessortenamt und **CH**emische Industrie	1
BBI	Bayes Belief Integration	114
CA	Classification Accuracy	76
DT	Distance Transform	56
EF	Elliptic Fourier	64
ExG	Excessive green	6
ExR	Excessive red	6
ExG-ExR	Excessive Green minus excessive red	6
FCM	Fuzzy C-means	15
GK-FCM	Gustafson-Kessel C-means	16
HSI	Hue-Saturation-Intensity	6
kNN	k-nearest neighbour classifier	78
MDA	Multiple Discriminant Analysis	95
MVG	Multivariate Gaussian	82
OSM	Optimal Segmenting Mask	10
PCA	Principal Components Analysis	21
PSEP	Plant Stem Emerging Point	24
RBF	Radial basis function kernel	79
RFE	Recursive Feature Elimination	101
RGB	Red-Green-Blue	6
ROI	Region of Interest	50
SVM	Support Vector Machine classifier	79
TV	Total variation	35

Introduction

As the world's population is steadily increasing, it is necessary to continuously improve the cultivation methods that are used in agriculture.
In order to improve the conditions for plants and thereby improve the yield of the harvest, the amount of weeds in the field has to be minimized. In farming today, the preferred method used for minimizing weeds is to apply pesticides to the field. The amount of pesticides used in the EU-15 countries is on average 3.88kg/Ha of active ingredients. However, in Belgium and the Netherlands pesticide usage exceeds 10kg/Ha[1][1]. This provides an effective removal of weeds, which is crucial for crop yield and thereby for the farmer's prospects for keeping up a profitable business. The overall loss to weeds without doing weed control is estimated to 34% by [2], though other research projects[3] document a loss of between 48% and 71% for tomatoes depending on the weeds and locations and 23% for wheat when there are five Canada thistles per square meter[4]. However, the usage of pesticides comes to the detriment of the environment.
In addition to this, there is a growing governmental pressure on farming, imposed through regulations and taxes to limit the usage of herbicides, because of the unwanted impact that the herbicides have on the environment. Therefore the farmer, according to EU directive 2009/128/EC [5], has to investigate which weeds are present in the fields in order to apply to his field as specific a mixture of pesticides as possible. This can be a time consuming task, especially in large scale farming, but this task could be overcome by deploying automatic weed detection. Moreover, the directive requires that the amount of herbicides used should be kept at the minimum level necessary to do the job.

At the same time, the global market for organic products is increasing [6], leading farmers to demand a better coefficient of utilization on the organic fields. If the individual weeds can be located, they can be controlled using e.g. water steam, oil, mechanics[7] or by using lasers [8].
When doing weed control it is still beneficial to distinguish the weeds from

[1] Based on 1998 numbers

each other. Some weeds are not harmful for the crops and could therefore be left in the field; thereby some of the land's biodiversity would be preserved. Furthermore, the turning of the soil that would also be avoided by leaving unharmful weeds in the field will also prevent more harmful weeds from emerging.

To make the environmental friendly farming and the organic farming more competitive to conventional farming, different weeding methods have been listed below:

- Optimal herbicides mixture: An inspection of the field to identify the different species and the amount of weeds in the whole field or in sections of the field. Based on the inspection, the optimal mixture of herbicides is sprayed traditionally in the field or in smaller sections, avoiding the use of unnecessary or excessive amounts of herbicides. By doing this, it it possible to reduce the used amount of herbicides with at least 40% for cereals[9], which also results in savings estimated to 154DKK/Ha for winter cereal and 54DKK/Ha for spring-sown cereals[9][2].

- Coarse precision spraying: The field is divided into sections and each section is thereby evaluated in order to determine the amount of weeds present in the field. After this, pesticides are applied traditionally within each section but adjusted to the amount necessary for the given section. Cf. [10], reductions between 20% and 72% are to be expected from following this approach.

- Mechanical weeding: A mechanical hoe is used to automatically remove unwanted weeds. A mechanical solution is expected to be slow and require lots of maintenance compared to spraying, but fully avoids the use of pesticides. The principle has been implemented in the Robovator made by F. Poulsen Engineering ApS[11]. However, the performance of this machine is limited in that it only distinguishes crops from weeds by size.

- Organic precision spraying: Weeds are located and controlled with organic materials such as water steam or oil.

- Precision spraying: This method deploys knowledge about plant and weed positions. The aim is to achieve an optimal utilization of pesticides, where crops are sprayed with fertilizer and weed with herbicides. Studies show that by doing precision spraying, the amount of pesticides can be reduced with up to more than 99% of the amount used in conventional spraying methods[12].

[2] Based on 2007 numbers

An implementation of the above methods in a robust and cost efficient system could revolutionize modern farming by making possible a new environmental friendly alternative or perhaps a cheaper production of organic products. Several of these methods have been tried out, but they lack a fast and robust method to distinguish weed from crops or recognize plants species in general, providing a strong argument for the relevance of this project[13].

Reading guidelines

The Chapters 1 to 6 presents the chronological stages in the system. The chapters are provided with an introduction and discussion enabling the chapters to be read individually, but should, for a better understanding, be read in chronologically. The remaining chapters *Results and discussion*, *Future work* and *Conclusion* in Chapter 7, 8 and 9 sum up and discuss the results achieved throughout the report.

In the project, we alternate between two domains; plant identification and pattern recognition in general. A *class* is typically used in pattern recognition to define a certain object that needs to be classified, equivalent to a plant species in the plant domain. *Plant elements* are used as a general term to describe the three elements that are in concern here: The plant (including its leaves and stem), its cotyledon leaves and its foliage leaves. A plant struct refers to a structure containing information about the three elements for a certain plant, may it be the image or the classification result of each plant element. The definition of a plant and the plant struct is easily confused, but there is a distinct difference. The classification of a whole plant is based on the features determined by the image of the plant, while the identification of a plant struct is a combination of the classification results of, respectively, a plant, the cotyledon leaves and the foliage leaves. For convenience, plant species are often referred to as class number 1-7 according to Table 1.1 in chapter 1.

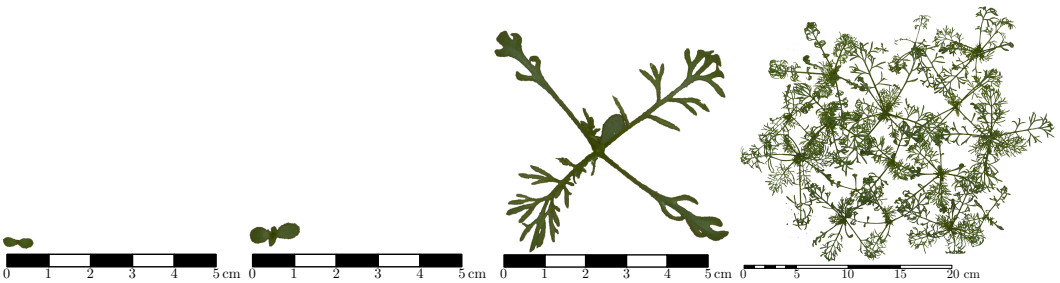
The project includes multiple MATLAB scripts to help others implement and achieve similar results faster. The directions to these scripts are referenced in footnotes throughout the documentation.

Project description

The project is based on the work of the authors' master thesis in Information Technology at Aarhus University School of Engineering, and is part of the ongoing project "*Sensorbaseret Graduering af fungicider og herbicider*

i kartofler og korn" (*Sensor-based modulation of fungicides and herbicides in potatoes and cereals*), which is a co-operation between Faculty of Agricultural Sciences and Department of Agroecology - Crop Health at Aarhus University under the leadership of Peter Kryger Jensen.

Problem definition The main problem in the domain of plant recognition is the variance within the same species. Plants are soft and sensitive to outer factors such as wind, light, nutrition and insects, which have impact on the colour, shape, texture and structure of the plant and its leaves. One species will also be hardly recognizable through its different growth stages as the plant will have small to none visual resemblance between the stages of, respectively, sprout, cotyledon plant and full grown plant as can be seen for scentless mayweed in Figure 1 and for the other species in Appendix O. Even at the same growth-stage, there can be a big variation in the appearance of the plants.



(a) BBCH 12 **(b)** BBCH 12 **(c)** BBCH 13 **(d)** BBCH 14

Figure 1: Example on how eg. scentless mayweed changes characteristics through different growth stages

To decrease the amount of herbicides needed, weeds have to be controlled at an early growth stage. This leads to other challenges, as the cotyledons are the first leaves to appear, but they are quite anonymous compared to the foliage leaves, that are the ones carrying most of the "identity" of the plant.

Contribution In the literature of plant recognition two approaches have been applied in order to identify plants using either the whole plant[14, 15, 16, 17] or its leaves[18, 19, 20]. The first approach, *describing the whole plant*, is the most applied procedure in recognition of crops for autonomous robotic weed control using images. The natural reason for this is that leaves are not naturally separated from plants in the field, whereby the whole plant

can be handled without extracting leaves from the images. The second approach, *describing the leaf*, is mostly applied under controlled conditions, where the leaf has been physically or manually extracted from the plant as the leaves are hardly extracted automatically. The advantage in a classification of leaves is that leaves roughly maintain their shape and appearance throughout the different growth stages, and that leaves are not so affected by damage from external impacts compared to whole plants. At the same time, the leaves will provide some redundancy, so e.g. a plant having one misshapen leaf could still be correctly identified, if it also has three well-shaped leaves. The drawback of using leaves in autonomous weed control, that identifies plants in the early growth stages, is that cotyledon plants have anonymous leaves in an early stage. The differences between plants are therefore presumably better at identifying species in the early stages as the variation between leaves is small.

Goal The aim of this report is to investigate and propose different methods identifying plants species based on raw RGB images. This approach seeks to segment a plant and its leaves in order to exploit the discriminating strength of both elements to provide an improved identification. In this process, methods for handling varying growth stages and difficulties related to the softness of plants will be investigated.

Project delimitations

In real-world applications of this technology,where the camera is mounted on a robot, driving in the field, further information is available, which will help the classification process. That information is for instance: which crops are in the field and when they have been sown and the approximate distance between them. With this information, the uncertainty is not only bound to the image-features, but e.g. the spatial location of the weed. An approach known as a row guidance system has achieved a high level of automation and some commercial success[3]. The concept, is that crops are positioned in rows and weeds are outside these rows. Plants are therefore not classified on their visible characteristics, but on the position in accordance to the row of the corps. In [21, 22] the row of crops are determined through a Hough Transform based procedure.
Another approach described in [3, 23] uses an RTK GPS position system to map the crop seeds during planting. Plants located far from the seeding points are then simply classified as weed.

Prior knowledge about where the crops have been planted/sown could also be combined with a vision system, so that the inter-row distance of the crops are known with some uncertainty, which helps determine the certainty that crops are observed in the images.

Studies also show that the non-visual plant reflectance can be useful in plant species identification for both imaging (cameras) and non-imaging sensors [24].

Though these different techniques have been applied in research, this project is delimited to plant recognition based on RGB images using features derived from visible characteristics.

The project treats the images of the provided dataset and will not directly access more machine vision related problems concerning images under e.g. less optimal light conditions. Furthermore, the project does not directly treat practical implementation issues such as computational cost and platform specific subjects.

Project outline

The project outline is shown in Figure 2 providing an overview of the major parts of the project. The first part "Data Acquisition" in Chapter 1 describes the collection of images of 12 different species at different growth stages. Each image contains only one species at a specific growth stage. The Data Acquisition has not been done within this project, but provides a good foundation for the proceeding parts of the project. The raw images are fed into the "Segmentation" component described in Chapter 2. The component must segment all plant-material by removing soil, gravel and other non-plant elements. The remaining elements in the image can now be divided into four groups; whole plants, overlapping plants, single leaves and non-plant material. Manual procedures transform these elements into whole plants and discard non-plant elements. The plants are then automatically divided into leaves and manually labelled as either cotyledon or foliage leaves. Hereby the "Segmentation" provides three plant elements to be recognized; the whole plant, cotyledon leaves and foliage leaves. The segmentation is only performed for 7 species of the 12 different species due to time constraints.

The project path then splits into three parallel sub-paths, each of which makes an individual recognition of one of the three plant elements. Each plant element is fed to the next stage "Feature Descriptors" documented in Chapter 3, which describes the process of extracting features using shape, contour and colour. All features for each plant element is then feed to the "Classifier" stage described in Chapter 4. The classification of the plant elements are made using three different classifiers; the multivariate Gaussian (MVG) classifier, the k nearest neighbour (kNN) classifier and the support

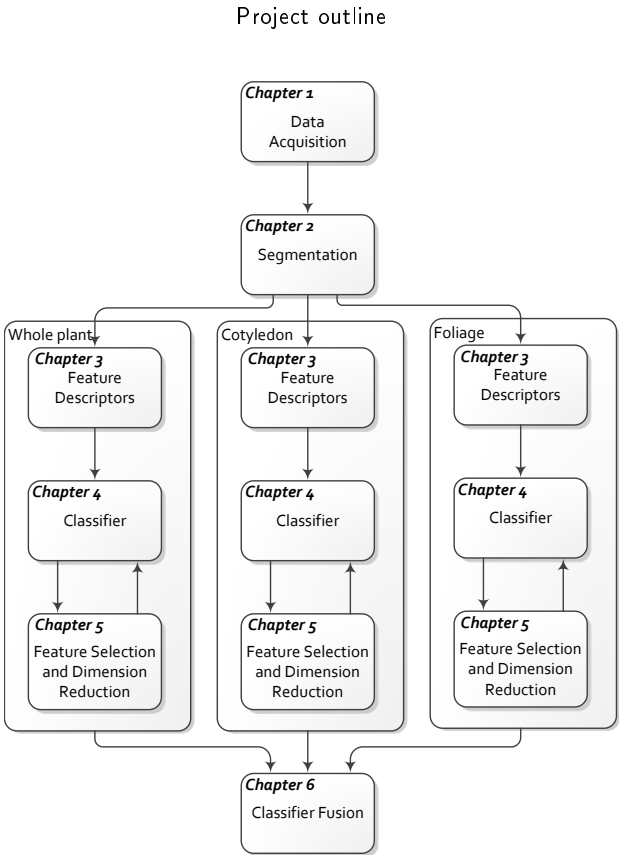


Figure 2: An overview of the different steps in the project.

vector machine (SVM) classifier. The next component called "Feature Selection and Dimension Reduction" described in Chapter 5 will reduce the number of features or dimensions to improve the classification accuracy. The features will have different discriminating power depending on whether they are tested on whole plants, cotyledons or foliages. The feature selection is therefore performed individually on the three plant elements to elect the most discriminating features for the given plant element. The result of "Feature Selection and Dimension Reduction" will be a set of features and a classifier targeted for each plant element, namely the whole plant, cotyledons and foliages. The final step "Classifier Fusion" described in Chapter 6 connects the three sub-paths by combining the classification of each plant element to perform a single identification of a plant struct.

Chapter 1

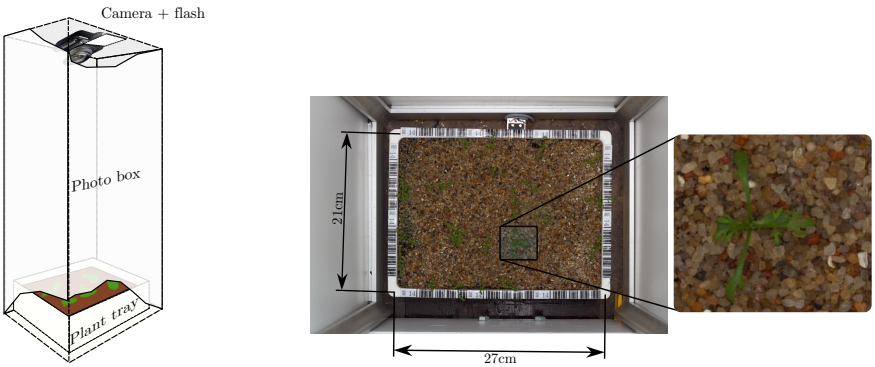
Data Acquisition

The data material for this project is based upon an image database created by Thomas Mosgaard Giselsson as part of the project "*Graduering af fungiciders og herbicider i kartofler og korn*" (*Graduation of fungicide and herbicide in potato and cereals*). The material consists of images of 12 plant species taken over two weeks and mainly covering the growth stages from BBCH 10-14. The BBCH-scale is a coding scheme to specify the growth stage of a plant[25, 26]. Of these 12 species, only the first seven are used in this project. Originally 14 species were sown, but Redshank and Field Pansy did not grow well and are therefore not present in this study. The rest of the species are listed in Table 1.1 and samples for each species can be found in Appendix O.

#	**Species**	**Growth stages (BBCH)**
1	Maize (*Zea mays*)	10-14
2	Wheat, winter (*Tricicum aestivum*)	10-15
3	Sugar beet (*Beta vulgaris*)	12-14
4	Scentless mayweed (*Tripleurospermum perforatum*)	12-14
5	Chickweed (*Stellaria media*)	12-23
6	Shepherd's-purse (*Capsella bursa-pastoris*)	12-20
7	Cleavers (*Galium aparine*)	12-33
9	Charlock (*Sinapis arvensis*)	12-33
10	Fat Hen (*Chenopodium album*)	12-14
11	Cranesbill (*Geranium pusillum*)	12-19
13	Black-grass (*Alopecurus myosuroides*)	10-15
14	Loose Silky-bent (*Apera spica-venti*)	10-15

Table 1.1: The 12 species in the database listed with the growth stages for which they are present

For each of the 12 species, the plants have been sown in four plant trays measuring $21cm \times 27$cm. These plant trays have been covered by fine, light gravel and the plants have grown under comparable conditions.
All images are taken by a Canon 600D RGB camera using ISO100 to decrease the amount of colour noise. The camera set-up is illustrated in Figure 1.1a. The set-up consists of the plant trays, a photo box and a camera with a diffused flash. The plant trays are placed at the bottom of the box, and the camera is mounted at the top, facing downwards. The photo box is white, which helps spread the light from the flash and thereby make the plants well-illuminated, while still avoiding strong shadows. With this set-up, all images are taken from the same height with the same background and the same illumination. These conditions help in the segmenting of the plants, as the variations in the colours of the background and plants are small. When dealing with plants in the field, the conditions will not be as ideal as in this case; the soil will have different colours and the nutrition levels are non-static, which increases the variation in the colours and sizes of the plants. However, illumination in the field can somewhat be controlled by shielding the camera and plants from sunlight [16, 27]. In addition to this, no visible marks from insects have been found in the dataset, which would also make the images less ideal.



(a) Camera setup. **(b)** Database sample of scentless mayweed

Figure 1.1: The camera is placed in the top of the photo box and the plant tray is placed at the buttom. the box is closed, so that the only light source is the camera flash, which ensures the same illumination in all images.

All images are taken from the same height, meaning that the scale of the plants relative to their actual sizes is close to identical for all plants. However, the leaves are not always perpendicular to the camera lens, and this introduces some variation in the appearance of the leaves in the images.

Chapter 2

Segmentation

In image pattern recognition - as well as in the field of plant recognition - segmentation is often necessary to isolate the elements that are to be classified.

Segmentation was initially thought of as a minor process in which plant elements could be found by simply extracting connected green components from an image, but as our knowledge and understanding of the domain has grown it became obvious that creating a good (enough) database requires a lot of effort. Firstly, the database consists of many GB of data that must be handled in a smart and fast way. Secondly connected green elements are not always a single plant, but may also be a single leaf or overlapping plants. Therefore, it is necessary to approach the segmentation in several steps: Leaves must be connected to whole plants, overlapping plants must be segmented into single plants and finally leaves should be extracted from whole plants. The segmentation process consists of seven steps, which can either be performed by automated algorithms or through a MATLAB developed segmentation tool, that combines automated segmentation with manual verification of the plants to ensure that labelling of plants, leaves, growth stages and other information is correct. The different steps are shown in Figure 2.1.

1. The process takes an RGB image and returns a binary mask of plant elements as illustrated in the figure. The process is described in Section 2.2.

2. The process connects green objects that belong to the same plant and store the individual plants in structs containing an RGB image of the plants without background.

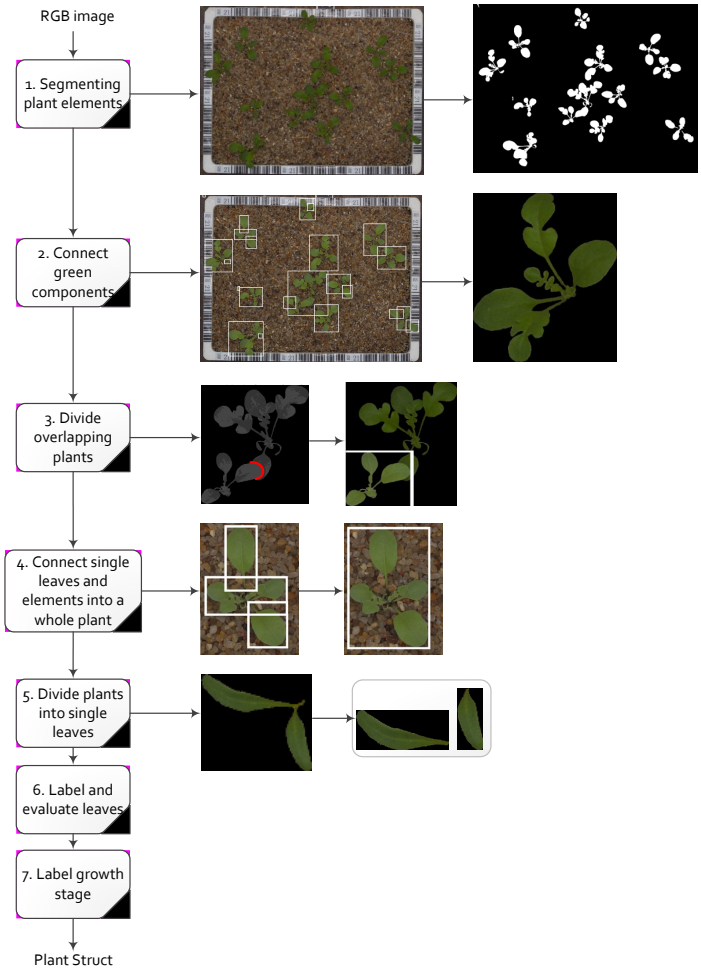


Figure 2.1: An overview of the different steps in the segmentation process.

3. Overlapping plants are divided into single plants through the segmentation tool described in Section 2.5. The process stores whole plants and discards the remaining parts.
4. This step consists of a manual process, connecting leaves or pieces of plants into whole plants.
5. Plants are divided into single leaves. Two methods for doing this are investigated in Section 2.3.
6. The cotyledon and foliage leaves are labelled and evaluated manually.
7. The growth stages of each plant are labelled manually.

The segmentation process returns a struct that will contain an RGB, greyscale and a binary image of the whole plant and the leaves within the plant. The struct also contains information about the growth stage and the species of the plant. Leaves are also labelled as either a cotyledon or a foliage leaf. The process is only performed on 7 of the 12 species provided in the data acquisition.

2.1. Related work in the field of plant segmentation

Different recording and segmentations methods can be used to extract plants from the background in images. One branch of segmentation uses RGB images, which cover part of the vision light spectrum (400nm to 700nm) [28]. By performing a sufficient colour transformation of the image, green colours will stand out whereby green elements can be found by thresholding the image[28, 29, 30, 31]. The second branch of segmentation uses non-visual light or hyper spectral imaging, which exploits the fact that plants also reflect other electromagnetic waves than visible light, as shown in Figure 2.2. In [32] images are recorded in the near-infrared spectrum (IR) (770-1.150nm) and in the visible red spectrum (610-670nm).

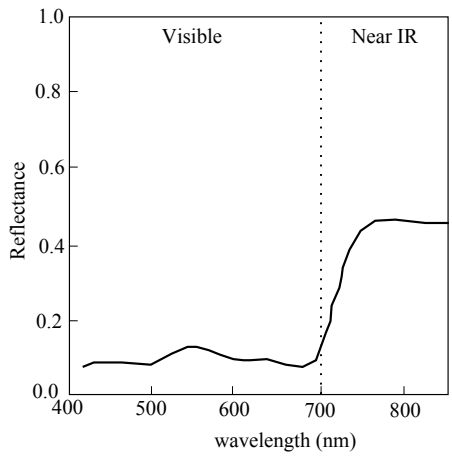


Figure 2.2: Reflectance of green leaves (after [33, p. 149])

Adjusting brightness of both images and subtracting them (IR-VIS) enables a robust segmentation of plants. In another study [34], the reflectance properties of the volunteer potato plant and the sugar beet were measured at multiple wavelengths in a band from 450 to 1650 nm. The ten most optimal wavebands were determined and used in the classification of the two plants.

The method cannot be used directly when segmenting plants, but it shows that plants emit non-visible electromagnetic radiation in the near IR area that can be used in segmentation.

Both segmentation branches perform well, but in a practical design the combination of visible and non-visual light will provide a more robust solution, as the colour of a plant is dependent on light condition, soil and other conditions. However, due to the fact that the database in this project only consists of RGB images, and that these images only show green plants that have been recorded under good conditions concerning constant lighting, same/constant background, no-shadows and same growth conditions, a colour transformation is used.

Lastly, a set-up using additional cameras to capture non-visible frequencies will also add an additional cost to the set-up, and it will also require a form of alignment/registration of the different cameras.

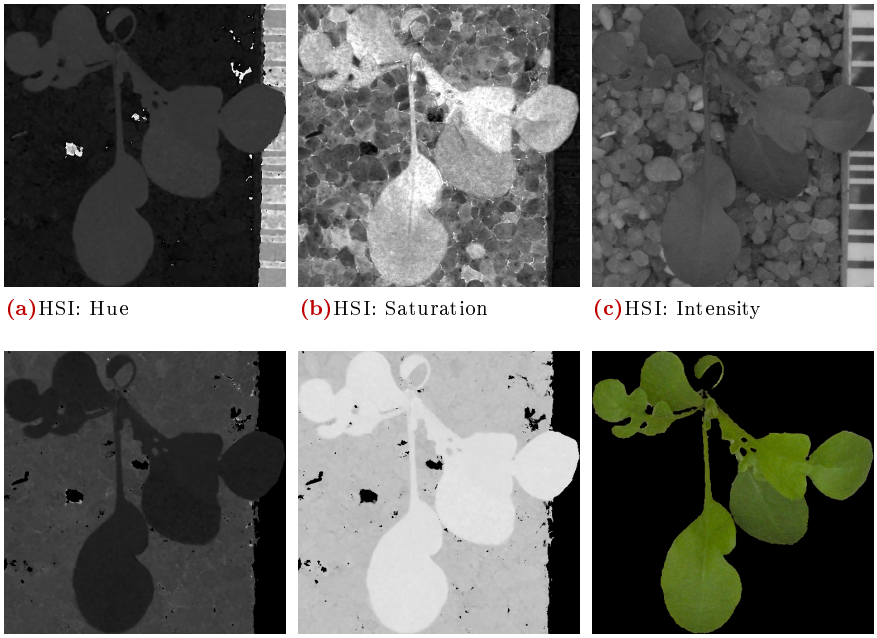
2.2. Segmenting Plants using Colour

A classical colour transformation used in image processing is the hue, saturation and intensity (HSI) colour model, as the transformation decouples the intensity from the colour information (hue and saturation)[28, p. 407]. Figures 2.3a, 2.3b and 2.3c show a regular transformation of an RGB into respectively hue, saturation and intensity on a sample.
A problem with the HSI model is that Hue is unstable around dark intensities ($R = G = B = 0$), and when the saturation is low ($R = G = B$)[29]. To improve the stability, pixels with a saturation below 0.2 and dark intensity below 0.1 are removed as shown in Figure 2.3d. Green material is segmented using a broad green criteria by thresholding hue at $\pm 60°$ around the centre of green (120°). In Figure 2.3e the hue values are shifted 120° so green pixels will present an intensity of 1, while non-green pixels are placed around 0. The shifted hue image is thresholded creating a binary mask showing the plant shape. The mask is used in Figure 2.3f to get the RGB information from only the plant.

A colour transformation called excessive green (ExG) described in [30] shows good results without the same stability problems. This method also has the advantage of being less computationally expensive than HSI.
ExG is defined as:

$$ExG = 2 \cdot G - R - B \tag{2.1}$$

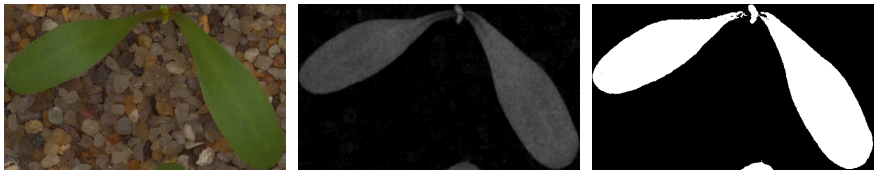
The global threshold is found with Otsu's method[35] using the MATLAB build-in function graythresh(). Intensities above the threshold are defined



(a) HSI: Hue **(b)** HSI: Saturation **(c)** HSI: Intensity

(d) Hue (dark and low saturated pixels removed) **(e)** Hue shifted around green **(f)** Gravel removed from image

Figure 2.3: Segmenting green with the HSI colour model.



(a) RGB image. **(b)** ExG. **(c)** Threshold using Otsu thresholding (threshold = 0.1412).

Figure 2.4: Segmenting green using the ExG method with Otsu threshold.

as green.

$$PlantMaterial = ExG > OtsuThreshold \tag{2.2}$$

The result of ExG is shown in Figure 2.4.

In [31] a combination of excessive green and excessive red ($ExG - ExR$) with a threshold of zero outperforms ExG with Otsu threshold and the method called Normalized Difference Vegetation Index (NDI) with Otsu

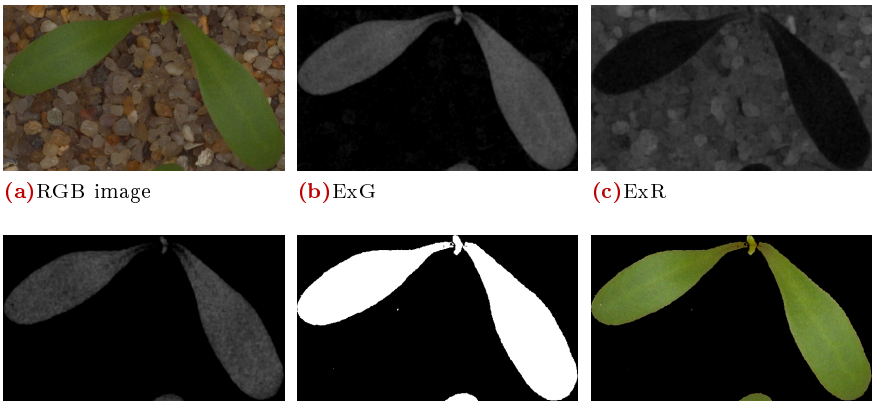
threshold.
ExR is defined as:

$$ExR = 1.4 \cdot R - G \tag{2.3}$$

The value of 1.4 is defined by human perception of colour. "*ExR takes into account relative proportions of rod and cone sensitivity for red and physiological green*"[36]. Green elements in $ExG - ExR$ are simply defined as values above zero when ExR is subtracted from ExG:

$$PlantMaterial = ExG - ExR > 0 \tag{2.4}$$

Figure 2.5a shows a plant sample with a gravel background. Figure 2.5b and 2.5c show respectively how ExG and ExR highlight green material and non-green material. Subtracting ExR from ExG provides a natural threshold around zero as shown in Figure 2.5d and 2.5e. The final result in Figure 2.5f shows a plant without gravel background.

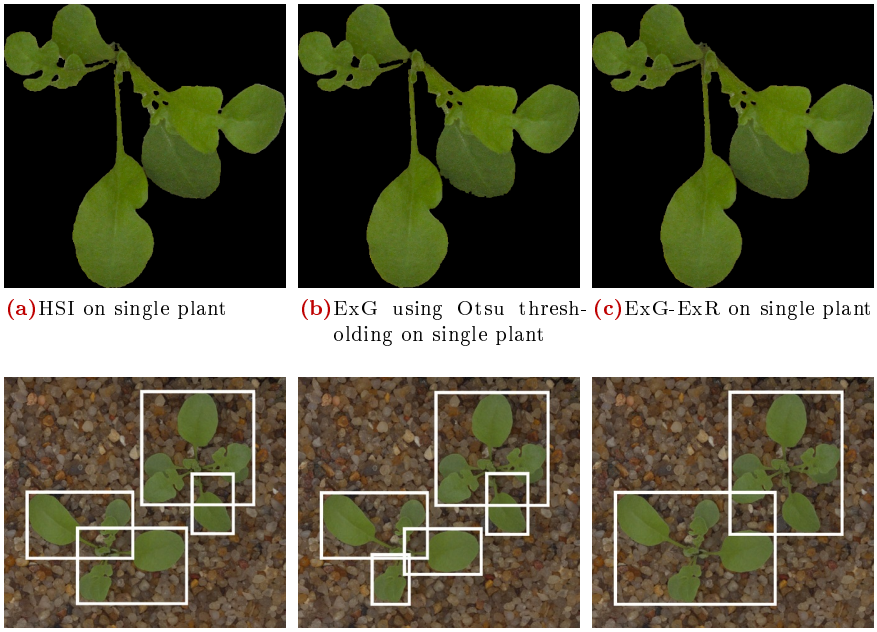


(a) RGB image **(b)** ExG **(c)** ExR

(d) ExG-ExR (only showing positive values) **(e)** Positive values are thresholded **(f)** Green material without background

Figure 2.5: Segmenting green using the ExG-ExR method.

Comparison on HSI and Excessive Green-Excessive Red An empirical experimentation was made with HSI, ExG and ExG-ExR all performing well. Figure 2.6 shows how the green material has been segmented for both a single plant and a whole image sample from the database by using HSI, ExG and ExG-ExR. ExG-ExR selects larger areas of green - especially in the stem area - compared to the two others as seen in 2.6a, 2.6b and 2.6c.
A broader selection of green colours will enhance the detection of the stem, and thereby make sure that more leaves are connected in joint plants. In Figure 2.6d, 2.6e and 2.6f it is seen that a single plant more often is divided



(a) HSI on single plant

(b) ExG using Otsu thresholding on single plant

(c) ExG-ExR on single plant

(d) HSI

(e) ExG using Otsu thresholding

(f) ExG-ExR

Figure 2.6: Segmenting green with respectively HSI, ExG and ExG-ExR. The methods are highly correlated. The difference it though notable around the stem of the plant in Figure (a), (b) and (c). Green segments are marked with a bounding box in Figure (d), (e) and (f) showing how the ExG-ExR connects more plant elements into whole plants.

into smaller connected elements by deploying those methods using HSI and ExG than by using the ExG-ExR method. Each of the three methods are implemented in MATLAB[1].

Based on this simple test, ExG-ExR is chosen as the method to be used for colour segmentation, as it is better at combining plant elements into whole plants. However, a more thorough research on the different segmentation examples is required to fully evaluate the different colour segmentation methods. That step has been omitted from this study due to the following reasons:

1. Each method must be evaluated with accordance to the ground truth segmentation. This is a very time consuming procedure as the ground truth segmentation must be performed manually by hand in a range of images to show robustness with different types of backgrounds.

[1] The MATLAB script is located in: `Matlab/Segmenting/SegmentingTool/simpleSegmenting.m`

2. Thorough research has already been performed in [31]; in this study, ExG-ExR is selected as the optimal method.

2.2.1 The ExG-ExR method

The method of ExG-ExR provides a good solution for robustly segmenting green material. An improvement of the method is therefore not an urgent lack of the whole system. However, the following analysis will provide an exploration of colour segmentation in general, and, more specifically, it shall examine the constants used in ExG-ExR as well as the ExG and ExR constants. The expression of ExG-ExR can easily be simplified into only one expression containing a factor of the R, G and B component.

$$ExG - ExR = 2G - R - B - (1.4R - G) = -2.4R + 3G - B \tag{2.5}$$

This expression can be rewritten to a more general form

$$y_{GreyScaleImage} = r \cdot R + g \cdot G + b \cdot B, \tag{2.6}$$

or as the inner product of the colour vector $\mathbf{x}_{rgb}$ and the fixed weighting vector $\mathbf{w}$

$$y_{GreyScale} = \begin{bmatrix} R \\ G \\ B \end{bmatrix}^T \cdot \begin{bmatrix} r \\ g \\ b \end{bmatrix} = \mathbf{x}_{rgb}^T \cdot \mathbf{w}, \tag{2.7}$$

where $\mathbf{x}_{rgb} = \begin{bmatrix} R \\ G \\ B \end{bmatrix}$ and $\mathbf{w} = \begin{bmatrix} r \\ g \\ b \end{bmatrix}$.

The equation describes a whole branch of colour segmentation methods that extracts a colour by only using a linear combination of the R, G and B components. In other words, the $\mathbf{w}$ vector projects the RGB values onto a line. A threshold on that line can now be used to separate foreground and background. The vectors for ExG, ExR and ExG-ExR are shown below:

$$\mathbf{w}_{ExG} = \begin{cases} r = -1 \\ g = 2 \\ b = -1 \end{cases}, \mathbf{w}_{ExR} = \begin{cases} r = 1.4 \\ g = -1 \\ b = 0 \end{cases}, \mathbf{w}_{ExG-ExR} = \begin{cases} r = -2.4 \\ g = 3 \\ b = -1 \end{cases}$$

In Figure 2.7 the vectors of ExG, ExR and ExG-ExR are shown in an RGB colour space. If all colours in the RGB colour space are projected onto the ExG vector, green colours will have a high value, while colours on the opposite side of the colour cube (magenta) will have low values. Adding a threshold will segment green colours while removing other colours in the image; especially magenta. On the other hand, the ExR method will result in high values around red colours and low values around cyan. The ExG-ExR vector is comparable to the ExG vector, but it will more aggressively remove the colour red.

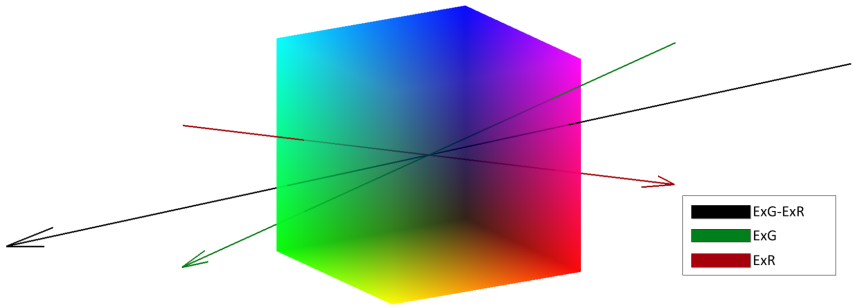


Figure 2.7: Lines showing the ExG, ExR and ExG-ExR

This project will present and investigate three colour segmentation methods that determine a vector $\mathbf{w}$ like those of ExG, ExR and ExG-ExR, but modified to segment colour for a given data set in an optimal way. The first method is presented in the project to demonstrate the main idea of the three methods, the two others can be found in Appendix A. The advantage of the three methods is that they allow for adjustments to be made to the colour, that is to be segmented, and they can be used for other colours than green. Each method requires an RGB image and a binary mask of a desired segmentation. The desired elements in the image are defined as the foreground and must - like leaves - have a distinct colour. The undesired elements are defined as the background. The desired segmentation is made manually by hand and is defined as the **o**ptimal **s**egmentation **m**ask (OSM). An optimization problem can now be set up by using the OSM to find the optimal linear combination $\mathbf{w}$ of R, G, and B that best separates the fore- and background of the image. The $\mathbf{w}$ vector can either be determined on behalf of one image or a set of images presenting a broad spectrum of the possible images that are to be recorded for an automated system.

The first approach uses linear regression and defines the cost function below. The RGB image is of dimensions $m \times n \times 3$ and the OSM is image of the dimensions $m \times n \times 1$.

$$J(r,g,b) = \|r \cdot \mathbf{R} + g \cdot \mathbf{G} + b \cdot \mathbf{B} - \mathbf{OSM}\|^2 = \left\| \begin{bmatrix} \mathbf{R} \\ \mathbf{G} \\ \mathbf{B} \end{bmatrix}^T \begin{bmatrix} r \\ g \\ b \end{bmatrix} - \mathbf{OSM} \right\|^2 \tag{2.8}$$

The RGB image is reshaped to a 2D vector with three rows for $\mathbf{R}$, $\mathbf{G}$ and $\mathbf{B}$, respectively, and a column for each of the $m \cdot n$ pixels in the image. The OSM mask is reshaped to $\mathbf{OSM}$, a 1d vector of the length $m \cdot n$. The problem is a set of linear equations (one for each pixel) that forms a classical linear least

squares problem

$$\|\mathbf{A} \cdot \mathbf{x} - \mathbf{b}\|^2 \tag{2.9}$$

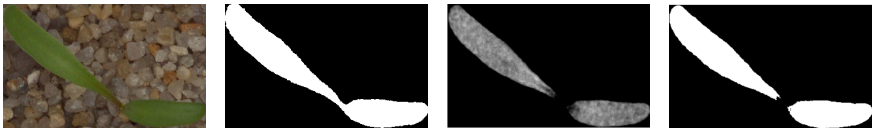
The solution to $\mathbf{x}$ is:

$$\mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b} \tag{2.10}$$

Where

$$\mathbf{A} = \begin{bmatrix} R_{1,1} & G_{1,1} & B_{1,1} \\ R_{2,1} & G_{2,1} & B_{2,1} \\ \vdots & \vdots & \vdots \\ R_{m,1} & G_{m,1} & B_{m,1} \\ R_{1,2} & G_{1,2} & B_{1,2} \\ \vdots & \vdots & \vdots \\ R_{m,n} & G_{m,n} & B_{m,n} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} r \\ g \\ b \end{bmatrix}, \mathbf{b} = \begin{bmatrix} OSM_{1,1} \\ OSM_{2,1} \\ \vdots \\ OSM_{m,1} \\ OSM_{1,2} \\ \vdots \\ OSM_{m,n} \end{bmatrix} \tag{2.11}$$

The method is used on an RGB image as shown in Figure 2.8a. The image is hand segmented to achieve the OSM mask in Figure 2.8b. In the linear approach the OSM mask is set to 1 in the desired segments and set to -1 in undesired segments. The least square solution of r, g and b is found to be[2] -2.4, 3.22 and -1.36 comparable to ExG-ExR values of -2.4, 3 and -1. The greyscale image showing green is made with Eq. 2.6, see Figure 2.8c. Finally the image is thresholded around 0, which results in 2.8d.



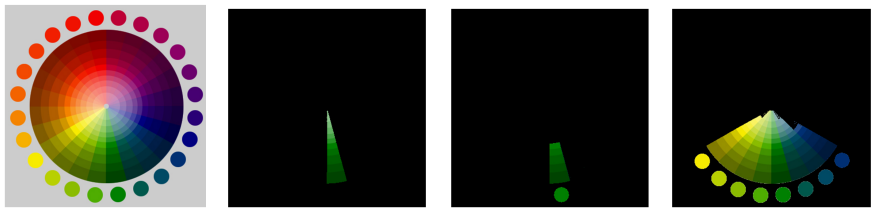
(a) RGB image **(b)** Optimal segmented mask, OSM. **(c)** The optimal combination of r, g, and b. **(d)** The segmented mask.

Figure 2.8: The procedure of the linear approach.

The result is comparable to the result given by ExG-ExR. The advantage of this procedure - as with the 2 other methods - is that a narrow or broader colour criteria can be set depended on what colour is being selected in the OSM. In Figure 2.9a, an RGB image of a colour spectrum is shown. A narrow green area is now selected from the colour spectrum to be used as OSM, see Figure 2.9b. The segmentation difference between the new method and ExG-ExR can easily be seen in Figure 2.9c and Figure 2.9d, respectively.

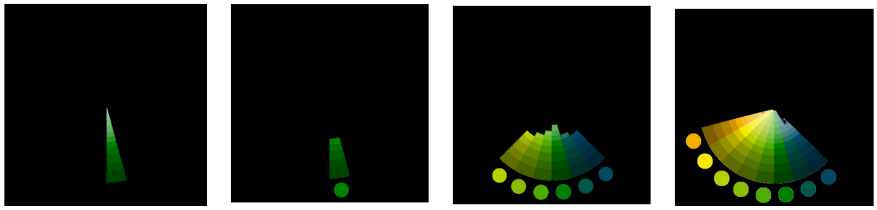
Additionally, the procedure allows the user to adjust how strictly the colour segmentation is performed by adjusting the OSM by multiplying a

[2] values are scaled to make $r = -2.4$ like in ExG-ExR



(a) RGB image **(b)** Optimal segmented mask, OSM **(c)** Result of new method. **(d)** Result of ExG-ExR.

Figure 2.9: The result of the new colour segmentation method.



(a) Optimal segmented mask, OSM. **(b)** Weight = 1. **(c)** Weight = 10. **(d)** Weight = 50.

Figure 2.10: The dependency of weight.

factor to the desired segments. Figure 2.10 shows the segmented area with different weights. The procedure is implemented in MATLAB[3].

The second colour segmentation method also uses a cost function which is related to the first method, it introduces a threshold in Eq. 2.8 for values larger than 0, which are set to 1, and othervise are set to 0, which provides a more correct solution

$$J(r,g,b) = \left\| \left(\begin{bmatrix} \mathbf{R} \\ \mathbf{G} \\ \mathbf{B} \end{bmatrix}^T \begin{bmatrix} r \\ g \\ b \end{bmatrix} > 0 \right) - \mathbf{OSM} \right\|^2 \tag{2.12}$$

But, the equation becomes non-linear, making it hard (if not impossible) to solve analytically, and the problem must be solved using a numerical optimization method.

[3]The MATLAB script is located in:
`Matlab/Segmenting/SegmentingTool/NewColorSegment/Test2ColorMethods.m`

The final colour segmentation implementation uses the Fishers Linear Discriminant method to likewise determine the r, g and b component, but also adds a threshold value adding an extra degree of freedom.
The two methods are described in more details in Appendix A.

The first colour segmentation method is computational efficient, but the cost function optimizes towards the OSM without including a threshold and therefore only provides an approximation. However, this adds a freedom to the method allowing the user to define how strict a colour must be segmented. The second method includes a threshold in the cost function and defines a more mathematically correct segmentation, but as the solution can not be solved analytically, the solution is found through a seemingly computationally insensitive numerical optimization. The third approach adds a dynamic threshold adding extra freedom to the expression, it is computational efficient and determines the optimal projection of the RGB colour space for a discrimination of the fore- and background.

2.3. Segmenting Leaves

One of the aims in this project is to compare how easy plants species can be determined from the appearance of whole plants as well as the appearance of their leaves. In this section, two methods for leaf segmentation will be tested; a method that segments leaves using fuzzy clustering by looking at pixel distances and colour information of the leaf and a method that segments leaves based on changes in the curvature of the plant.

2.3.1 Segmenting Leaves - Fuzzy Clustering of leaves

This section describes the implementation and an algorithm for extracting individual "good" leaves from images of plants with multiple leaves. The method is based on [37] in which the pixels of the plant are first divided into small clusters using the Gustafson-Kessel C-means method and afterwards, they are grouped in individual, good leaves using a genetic algorithm.

2.3.1.1 Plant Clustering

The clusters, which the pixels of the plant are divided into, are based upon the colour information and the spatial location of the pixels. The colour information could for instance be the RGB, HSI or the $ExG - ExR$-values. The drawback of using RGB or HSI is that they increase the computational complexity because of the three colour-channels instead of just one. A test and comparison of the clustering for different colour spaces are presented in

Section 2.3.1.3.

The notation in this section will be based on RGB-values, but for other colour-spaces, it should just be replaced with the respective colour. For a plant containing N pixels a data matrix, $\mathbf{X}_{N\times 5}$, is constructed in which each row consists of the processed colour (red, green, blue) and (row,column)-coordinates of each pixel within the plant.

$$\mathbf{X} = \begin{bmatrix} R_1 & G_1 & B_1 & r_1 & c_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ R_N & G_N & B_N & r_N & c_N \end{bmatrix}$$

To create the clusters, two methods have been tried out. One is the Fuzzy C-Means (FCM) algorithm and the other is the Fuzzy C-Means algorithm with the Gustafson-Kessel extension.

First the k-means algorithm is introduced, as it is the basis of the fuzzy C-means. The k-means algorithm is a hard assignment algorithm, in which pixels are labelled according to the minimum distances to a cluster-"mean" in a feature space consisting of pixel intensities and pixel coordinates. A clustering method based on k-means is described in [38, p. 415].
The two fuzzy algorithms shall be seen in contrast to this k-means algorithm, as each feature vector, $\mathbf{x}_n$, is not only assigned to one cluster, but have a degree of membership to all clusters depending on the distances to the means of the clusters [39]. The higher membership-value, $\mathbf{u}_{in}$, the larger association does $\mathbf{x}_n$ have to cluster i.

Fuzzy C-Means (FCM) In FCM, this membership is found by minimizing the cost function J_{FCM}, which is the sum of the distances from all data $\mathbf{x}_n$ to the cluster centers $\mathbf{v}_i$, multiplied with the corresponding degree of membership $\mathbf{u}_{in}$ of $\mathbf{x}_n$ to cluster i.

$$J_{FCM} = \sum_{i=1}^{C}\sum_{n=1}^{N} \mathbf{u}_{in}^m D_{in} \tag{2.13}$$

where D_{in} is the Euclidean distance defined by

$$D_{in} = \|\mathbf{x}_n - \mathbf{v}_i\|^2 \tag{2.14}$$

and C is the desired number of clusters. In the k-means algorithm each sample will be assigned to only one cluster, whereby you could say that the sum of all membership assignments $\mathbf{u}_{in}$ for a feature vector $\mathbf{x}_n$ should be 1. i.e. $\sum_{i=1}^{N} \mathbf{u}_{in} = 1$. In the fuzzy C-means algorithm, this is not the case, which is the reason why the exponent m is introduced. m determines how much weight are given to the nearest means, and thus it controls the "fuzziness" of the clusters. m can take values from 1 to ∞ [40]. The cluster centres are calculated as the weighted mean of $\mathbf{X}$.

$$\mathbf{v}_i = \frac{\sum_{n=1}^{N} \mathbf{u}_{in}^m \mathbf{X}}{\sum_{n=1}^{N} \mathbf{u}_{in}^m}, \tag{2.15}$$

and the elements of the membership matrix $\mathbf{U} = [\mathbf{u}_{in}]$ is found by using lagrange multipliers with the constraint that $0 \leq \mathbf{u}_{in} \leq 1$ and $\sum_i \mathbf{u}_{in} = 1$ whereby it can be calculated as [41]:

$$\mathbf{u}_{in} = \frac{1}{\sum_{l=1}^{C} \left(\frac{\|\mathbf{x}_n - \mathbf{v}_i\|}{\|\mathbf{x}_n - \mathbf{v}_l\|} \right)^{\frac{2}{m-1}}} \tag{2.16}$$

The optimal cluster means and relationships are calculated through an iterative process, which continues until the change of the relationship matrix is sufficiently small [40].

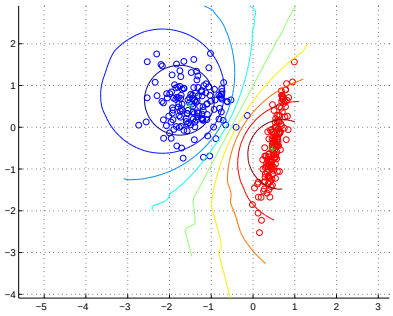
Figure 2.11a shows an example of the performance of the FCM algorithm on two different, Gaussian distributions, calculated using the fuzzy exponent $m = 2.0$. It should be noticed that the fitted clusters are identical, even though the data distributions are different from each other. Figure 2.11b shows the performance of the algorithm on a plant preprocessed with ExG-ExR, still with $m = 2.0$. This time the algorithm is set to find six clusters, and each pixel is labelled according to the nearest cluster[4]. As can be seen from the two figures, the clusters are spherical, which means that the performance will be even worse, if the axis are not normalized to each other, such that the colour and coordinates have the same range.

Gustafson-Kessel FCM (GK-FCM) As the FCM cost function (2.13) is minimized by minimizing the Euclidean distance between $\mathbf{x}_n$ and $\mathbf{v}_i$, the clusters will move towards a spherical shapes [39] with identical volumes. The Gustafson-Kessel extension compensates from this disadvantage by using a modified version of the Euclidean distance that is based on the covariance matrix of each cluster. With this extension ellipsoidal clusters can be built instead of only simple, spherical clusters [42], and different geometrical shapes in the data set can therefore be detected. This also means that the scaling of the axis is less important.

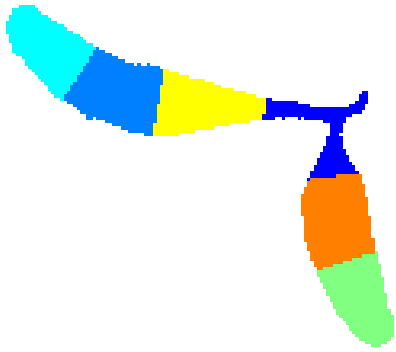
The cost function of the Gustafson-Kessel algorithm is defined in the same way as the cost function (2.13) from FCM.

$$J_{GK} = \sum_{i=1}^{C} \sum_{n=1}^{N} \mathbf{u}_{in}^m D_{in\mathbf{A}_i} \tag{2.17}$$

[4] MATLAB code for the test can be found in `Matlab/IndividualLeafExtraction/DemoafFCMogGK-FCM/testafFCMGK.m`



(a) Datafitting for random samples with the FCM algorithm, showing that the FCM algorithm finds the best spherical clusters for the data set.



(b) Datafitting with the FCM algorithm on sugar beet plant.

Figure 2.11: FCM performance on two random clusters and leaf a plant.

What Gustafson-Kessel changed was the distance function from $\mathbf{x}_n$ to $\mathbf{v}_i$, which now is given by:

$$D^2_{in\mathbf{A}_i} = (\mathbf{x}_n - \mathbf{v}_i)^T \mathbf{A}_i (\mathbf{x}_n - \mathbf{v}_i), \tag{2.18}$$

where $\mathbf{A}_i$ are variables used to make the clusters adapt to the given dataset. In the FCM distance function (2.14), $\mathbf{A}_i$ is simply the identity matrix. However, this could be changed to use the Mahalanobis norm, and thereby make the clusters hyper-ellipsoidal instead of spherical. Yet all clusters will still have the same shape [40, pp. 69].

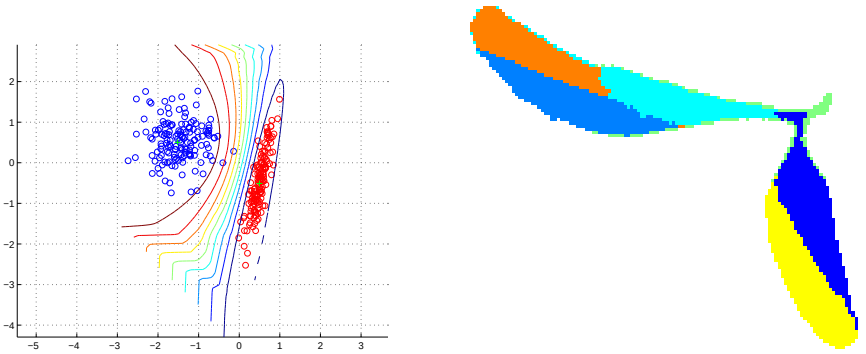
In the Gustafson-Kessel algorithm, the aim still is to minimize the cost function. However, as the cost function (2.17) is linear in $\mathbf{A}_i$, it is necessary to constrain $\mathbf{A}_i$, which is done by constraining the determinant of $\mathbf{A}_i$ [41]. The determinant of a diagonal matrix is simply the product of the entries on the diagonal, which have to be held constant and non-zero. The reason for doing this is that it will allow different scaling in each direction in the feature space, but without letting the distance measure grow unbounded. The determinant of $\mathbf{A}_i$ is called ρ_i

$$|\mathbf{A}_i| = \rho_i, \quad \rho_i > 0, \ \forall i \tag{2.19}$$

Where ρ_i is fixed for each i.

Such a constrained minimization problem is solved using Lagrange multipliers whereby $\mathbf{A}_i$ can be expressed as[41]:

$$\mathbf{A}_i = (\rho_i \cdot |\mathbf{\Sigma}_i|)^{1/d} \mathbf{\Sigma}_i^{-1}, \tag{2.20}$$



(a) GK-FCM performance on two random clusters, showing that the algorithm adapt to the shape of the clusters.

(b) GK-FCM performance on on a sugar beet plant.

Figure 2.12: Data fitting for the GK-FCM algorithm, showing that the GK-FCM algorithm are better at fitting the clusters to the data sets than the FCM algorithm, shown in Figure 2.11.

where d is the dimension of the feature space. Figure 2.12 shows how the Gustafson-Kessel extension performs on the same data set, as used in 2.11. It should be noticed that the two clusters are fitted much better to the data sets than in the standard FCM algorithm. The MATLAB implementation[5] is based on [43].

After the plant has been segmented, the segments, which belong to the same cluster, but are not spatially connected, are given different labels and are thereby considered new, independent segments. By doing this, edges are indirectly taken into account, as edges typically will be independent clusters because of their different colours, which divides other clusters in two.
After this processing, the image is filtered using a 3×3-median filter, to remove the segments consisting of only few pixels.

2.3.1.2 Cluster combination using the genetic algorithm

The idea of the cluster combination is to combine the clusters in whole, good leaves. To do this, the genetic algorithm is used.
The genetic algorithm is a global optimization algorithm, in which different cluster combinations are tried out, and the ones having the best fitness, i.e. provides the best results, have a higher probability of surviving to the next

[5] MATLAB code for the test can be found in
`Matlab/IndividualLeafExtraction/DemoafFCMogGK-FCM/testafFCMGK.m`

iteration of the algorithm[6].

The first part of the algorithm is the creation of a set of chromosomes. One chromosome is a vector representing a possible combination of segments to form a leaf and is constructed as follows[7]:

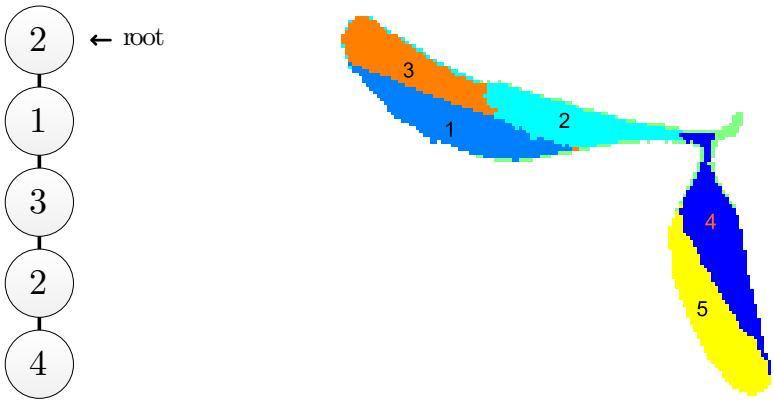
1. A symmetric matrix C with height and width equal to the number of labels is constructed as a look-up table of connected segments. Each row represent a segment, as does the columns. The entry at the intersection is then set to 1, if the segments represented by the row/column are neighbour segments, and set to 0 if they are not.

2. A random segment with an area larger than the average segment area is chosen as the root of the chromosome (=first entry of the vector).

3. A random neighbouring segment to the root segment is chosen for the second entry of the vector.

4. The third entry is then a random neighbouring segment to the segment from the second entry and so on...

The principle is illustrated in Figure 2.13, in which a chromosome consisting of five levels is constructed.



(a) A possible, five-level chromosome **(b)** Segmented and labelled sugar beet plant

Figure 2.13: Example on how a five-level chromosome could be made from a segmented leaf. Each level of the chromosome must be connected to the previous level of the chromosome

[6] The MATLAB script is located in:
`Matlab/IndividualLeafExtraction/geneticalgorithm.m`
[7] The MATLAB script is located in:
`Matlab/IndividualLeafExtraction/connectedRegions.m`

There are no upper bound of the number of chromosomes and the length of the chromosomes, but the more chromosomes, and the longer they are, the more computational power is required to find the optimal ones. Still the required length of the chromosomes are strongly related to the number of segments that makes up the leaf, as the length has to be at least as long as the number of segments. For convenience the population of chromosomes for iteration i is called $P(i)$.
When the first chromosomes have been constructed, the algorithm starts an iterative process, which goes as shown in algorithm 1. The idea in this algorithm is to continuously build new populations of chromosomes that are based upon the strongest chromosomes from the last iteration. I.e. the chromosomes that makes up the best leaves. To make sure that the algorithm does not get stuck in a local minimum, mutation and cross-over of the chromosomes introduces new chromosomes that could help avoiding such a situation.

Algorithm 1: Genetic algorithm

for $i \leftarrow 1$ **to** N_{iter} **do**
- **[Calculate fitness]: foreach** *chromosome* p_n *in* $P(i)$ **do** calcFitness(p_n);
- **[Mating pool]:** Create mating pool $M(i)$ with chromosomes from $P(i)$, the higher fitness of p_n, the larger probability of being selected to $M(i)$;
- **[Crossover]:** Repeatedly take out two parent chromosomes from $M(i)$, and with a crossover probability cross over the two chromosomes to form two offspring to put back in $M(i)$. If no crossover was performed, the offspring will be an exact copy of the parents;
- **[Mutation]: foreach** chromosome p_n in $M(i)$: with a mutation probability, change entries in p_n to random segments.
- **[New population]:** $P(i+1) = M(i)$

end

As the segments, that make up one leaf, have to be connected, a constraint for the crossover has been made to form whole leaves. It says that the crossover point has to be at a place that do not violate the initial structure of the chromosomes, in which the entries represent neighbouring segments. The fitness of the chromosomes is calculated using Regionprops-solidity in MATLAB. This calculates the fraction of the area of the segments divided by the convex area of the segments as illustrated in Figure 2.14.

With this objective for at good leaf, it is also clear that some leaves will be discarded, if they bow or have indented margins like, for instance a stinging nettle leaf.

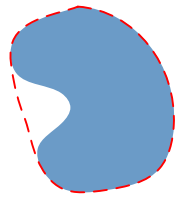


Figure 2.14: The fitness parameter is calculated as $Area_{blue}/Area_{red}$

When a good leaf has been found, it is removed from the image, and the algorithm starts over to find the second-best leaf. Area and solidity are used as stopping criteria for the algorithm, so that if only a small fraction of the original area is left or the best leaf in the remaining image has a low solidity, the algorithm stops.

2.3.1.3 Test of Gustafson-Kessel algorithm performance on different, preprocessed images

In this section, the performance of the algorithm will be tested for different colour inputs[8]. In the initial plant segmentation in Section 2.2, ExG-ExR has proven to be a good feature to segment the plant from the soil. This test will show, if this also is a good feature for the Gustafson-Kessel FCM algorithm, where clusters are based on the colour information, too.
In addition to this, six other colour spaces will be tested as well, so that input images based on the following colour spaces are tested:

- RGB
- ExG-ExR
- Greyscale
- Hue-Saturation-Intensity (HSI)
- Histogram equalized RGB
- First principal component of RGB (PCA)

As seen from Figure 2.15, the segmentation based on ExG-ExR (b)(h) is the one that performs the worst. This can be seen in the area with the overlapping leaves, where it makes clusters that overlap leaf margins. The clusters from RGB (a)(g) and greyscale (c)(i) images are both able to make clusters that follow the leaf margins, but both also have edges that they miss. It is also seen that the greyscale-based segmentation only have few segments

[8] The MATLAB script is located in:
`Matlab/IndividualLeafExtraction/IndividualLeafExtraction.m`

compared to the others, which will reduce the complexity of finding the optimal combination of clusters.

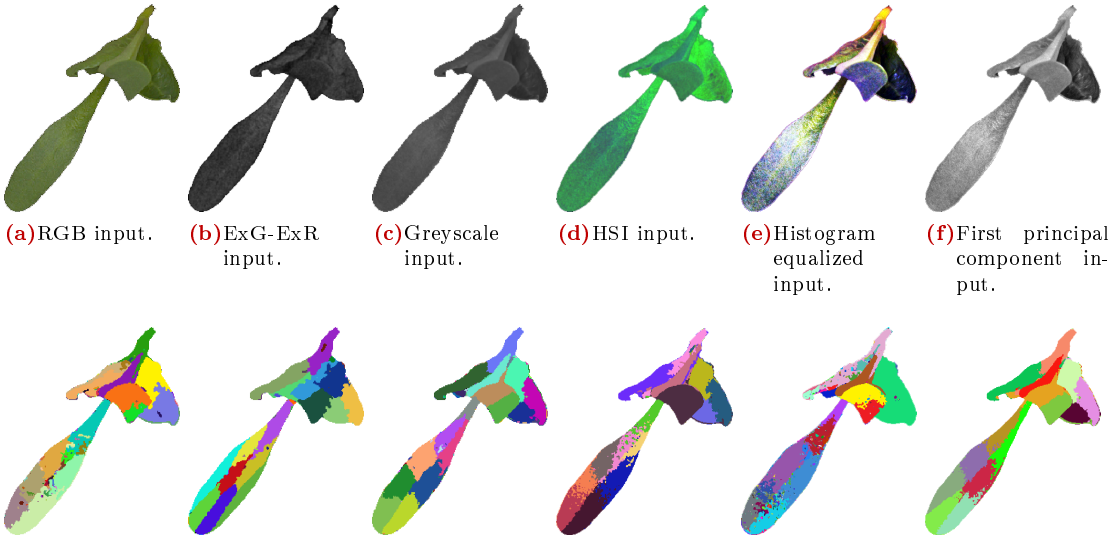
The clustering based on HSI (d)(j) seems to perform evenly good as the greyscale-based clusters regarding edges between leaves, though, it is noisy and the hue channel of the HSI image does not seem to add much to the clustering result and could therefore be omitted.

Figure 2.15e shows an RGB image that has been histogram equalized for all three colour channels. The reason for doing this is to utilize the full, dynamic spectrum and thereby push unrelated segments further apart before doing the clustering. From this segmented image (k), it is seen that the segments do not overlap leaf margins between the leaves. But the segments are very scattered in areas that appear flat in the RGB image, which will make the further processing even heavier, without adding useful information. The last test is made using principal component analysis (PCA) of the RGB image (f) (l), and use the Gustafson-Kessel algorithm on only the largest principal component. The purpose of using the largest principal component is that it will keep most of the variation of the colours, while only being having one colour channel. The result from this is very comparable to the others when finding leaf margins, but it also seem to be slightly better at finding the venation (skeleton) of the leaves. An explanation of Principal Component Analysis can be found in Appendix B. Based on this test, the greyscale and the first principal component input inputs seems to provide the best clustering for the given leaf.

2.3.1.4 Test of "Fuzzy Clustering of leaves"-algorithn

As stated earlier, this algorithm has solidity as the measure for a good leaf. This measure helps removing leaves that are odd oriented relative to the camera, or leaves that often bows. The algorithm is therefore not expected to perform well in these cases.
For all tests, the Gustafson-Kessel algorithm has been used to create the clusters, which have been combined using the genetic algorithm. Both the Gustafson-Kessel algorithm and the Genetic algorithm are computationally intensive algorithms, which are not suited for real-time implementation without some optimization. E.g. GPU implementation. Especially the Genetic algorithm is computationally intensive, as it requires an image to be generated to measure the solidity, which are done several times for each iteration. For instance an implementation with 80 chromosomes and 200 iterations requires $80 \times 200 = 16000$ images to be generated!

Therefore, to save computational power, the numbers of clusters and iterations are changed depending on the test.



(a) RGB input. (b) ExG-ExR input. (c) Greyscale input. (d) HSI input. (e) Histogram equalized input. (f) First principal component input.

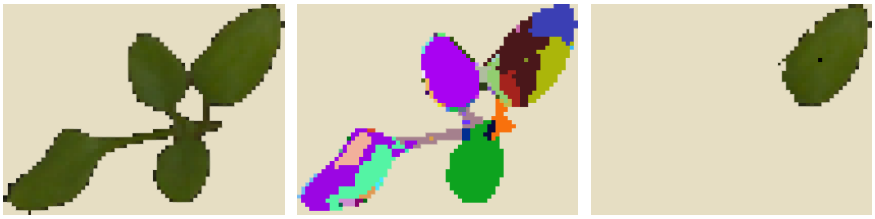
(g) RGB segmentation. (h) ExG-ExR segmentation. (i) Greyscale segmentation. (j) HSI segmentation. (k) Histogram equalized segmentation. (l) First principal component segmentation.

Figure 2.15: Example on the difference in segmentation of the Gustafson-Kessel algorithm for six kinds of input.

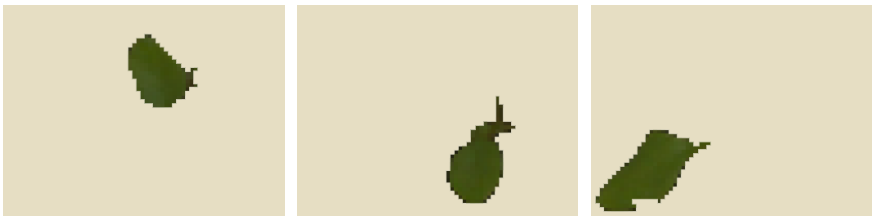
Round leaf with small overlap In this test, chickweed is used as input. The GK-FCM algorithm is set to have a fuzzy exponent of 2.0 and to find 15 clusters. The genetic algorithm has been set to have 80 chromosomes and to do 200 iterations for each leaf[9]. The result can be seen in Figure 2.16. What should be noticed is that all, four leaves are segmented even though two of the four leaves are overlapping each other.

Overlapping leaves An example of the ability to segment overlapping leaves is shown in Figure 2.17. This example shows a constructed image of two overlapping sugar beet leaves, where the upper leaf is segmented, and the two parts making up the lower leaf are segmented as well. It should be noticed that the leaf in the upper right corner miss the part close to the stem, which is one of the consequences of the fitness measure in this algorithm. This is because the dark, green area in the clustered image will add a large concave hull, which will result in a low solidity. In addition to that, small holes appear in the leaves, which probably are due to their small impact on the solidity measure.

[9] The MATLAB script is located in: `Matlab/IndividualLeafExtraction/IndividualLeafExtraction.m`



(a) RGB input. **(b)** Segmented and labelled plant. **(c)** Best "good" leaf.



(d) Second best "good" leaf. **(e)** Third best "good" leaf. **(f)** Fourth best "good" leaf.

Figure 2.16: Example on the performance of algorithm on chickweed (Stellaria media).

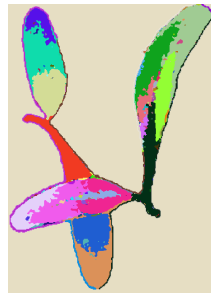
For this test, only the area of the remaining plant, has been used a stopping criteria, so that it is possible to see the order in which the algorithm categorizes good leaves.

2.3.2 Segmenting Leaves - The PSEP algorithm

Another method for finding leaves uses the general contour of the plant rather than the surface of it. The method is based on an algorithm called Plant Stem Emerging Point(PSEP)[44], which - as the title says - finds the stem emerging point of the plant. This project is not concerned with finding the stem point, but the initial steps in the algorithm involve extracting leaves from the plant. The leaf extraction exploits that the general shape of a plant consists of convex regions, i.e. the leaves of the plant, that are joint together by a thin stem to form the whole plant.
Leaf extraction consists of three steps:

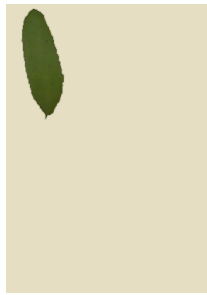
1. Extract the boundary of the plant mask. Make a sorted coordinate list of points from the boundary running in a clockwise direction.
2. Find leaf tip candidates by determining the most convex regions of the plant.
3. Find leaf cut. A search is begun from each leaf tip candidate to validate a leaf tip and to cut leaf from plant.

(a) RGB input.

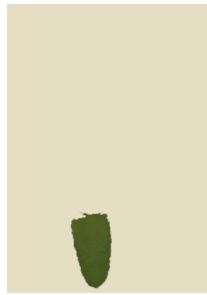


(b) Segmented and labelled plant.

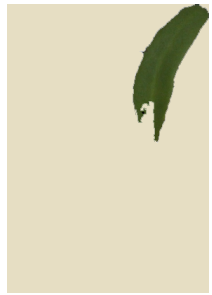
(c) Best "good" leaf.



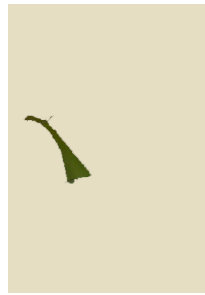
(d) Second best "good" leaf.



(e) Third best "good" leaf.



(f) Fourth best "good" leaf.



(g) Fifth best "good" leaf.

Figure 2.17: Example on the performance of algorithm on overlapping sugar beet leaves.

2.3.2.1 Extract boundary in a sorted list

The input of the algorithm is a binary mask of the plant as shown in Figure 2.18b. The contour is determined as a list of points running clockwise around the boundary. The boundary is shown in Figure 2.18c. A procedure has been implemented in MATLAB[10] as described in Appendix D but a built-in MATLAB function bwtraceboundary() provides the same functionality and is therefore used instead.

2.3.2.2 Leaf tip candidates

Leaf tip candidates are found by finding the curvature/change of direction around the boundary also known as the second derivative of the boundary. The sign of the curvature describes whether the leaf is concave (positive

[10] The MATLAB script is located in:
`Matlab/Segmenting/PSEP/PSEP_SortedEdgeList.m`

(a)RGB image **(b)**Plant binary mask **(c)**Plant boundary

Figure 2.18: Shows the RGB image, the binary mask and the extracted boundary of the plant.

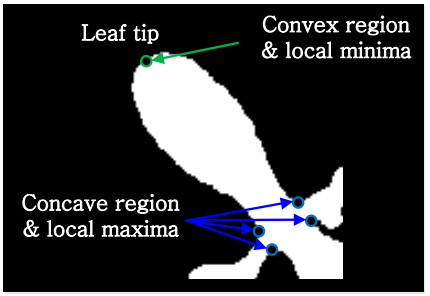
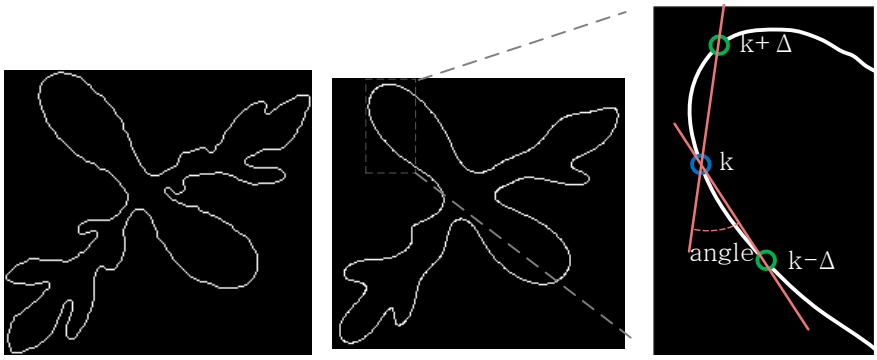


Figure 2.19: Convex and concave regions. Leaf tip candidates are defined as local minima of the curvature.

values) or convex (negative values). Leaf tip candidates are points of high convexity and are defined as local minima in the curvature, as illustrated in Figure 2.19. The n boundary pixels are sorted in a list of coordinates $\mathbf{z}_k$ where $k \in [1, \ldots, n]$ running in the clockwise direction around the boundary as described in [44]. An estimate of the second derivative of the boundary, defined as an estimate of the curvature, is determined. Firstly the list of coordinates is filtered individually over the x and y coordinates using a moving average filter with a size of $m = 31$ ($m = 5$ in [44]), as shown in Figure 2.20b. Finally, the curvature for each of the k in the coordinate list $\mathbf{z}_k$ is defined as the angle between the line that connects point $\mathbf{z}_{k-\Delta}$ to point $\mathbf{z}_k$ and the line that connects point $\mathbf{z}_k$ and point $\mathbf{z}_{k+\Delta}$ [44], as illustrated in Figure 2.20c. The value Δ is set to 24 ($\Delta = 12$ in [44]). A function has been implemented in MATLAB[11] to calculate the curvature. The MATLAB implementation adds additional $\Delta+m$ points in the beginning and in the end of the sorted coordinate list to allow the algorithm to calculate the curvature

[11]The MATLAB script is located in:
`Matlab/Segmenting/PSEP/PSEP_filterEdge.m`



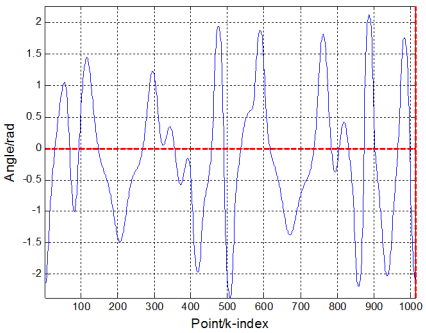
(a) Boundary image. **(b)** Filtered boundary image. **(c)** Determining the curvature.

Figure 2.20: Definition of curvature.

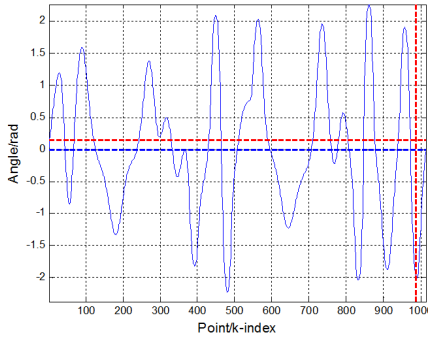
in the first point $\mathbf{z}_1$ and in the last point $\mathbf{z}_n$.

The curvature for the given sample is shown for each k in Figure 2.21a. The leaf tip candidates are determined by finding locale minima, which is handled in three steps [44]. The first step divides the curvature in concave and convex regions by looking at the sign of the curvature as described in [44]. A small adjustment (which was not mentioned in the original article) consists of doing an up shift and a left/right-shift of the curvature as shown in Figure 2.21b, where the previous origin is illustrated by two red, dashed lines.

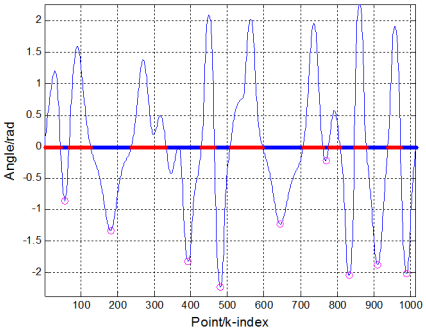
The up-shifting of the curvature is made by subtracting the mean value of the whole curvature to compensate for the fact that a closed shape will always be more concave than convex. The left/right-shift is made to avoid splitting a convex region in two, either at the beginning or at the end of the plot. Convex and concave regions are determined from the adjusted curvature in order to find candidate leaf tips. In step two, the minimum of the curvature is determined for each convex area by finding the most convex point within the convex region. In Figure 2.21c the blue colour along the x-axis marks convex regions, while concave regions are marked in red. The local minima are marked by pink circles. The third step simply performs thresholding by removing minima with a change of direction less than 0.5 radians (1 radian is used in [44]). In Figure 2.21d the local minima is marked on the boundary with a cross or circle. A circle indicates that a local minima is elected as a leaf tip candidate.



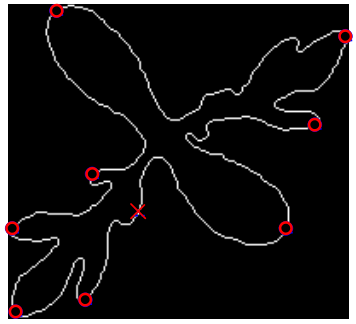
(a) Curvature of the boundary.



(b) Adjusted curvature by a horizontal and vertical shift. The shifting from Figure (a) is illustrated by marking the shifted origin by the crossing of the two dashed red lines.



(c) Shows convex areas blue and concave areas marked with red. Local minima marked with pink circle.



(d) Shows the local minima marked with circle/cross or leaf tip candidates marked with circle.

Figure 2.21: Show the curvature and the adjusted curvature by an up- and vertical-shift.

2.3.2.3 Finding leaf cut

In the last step, a search is performed from each leaf tip candidate position by sending out two walkers around the boundary. One walker is sent in a clockwise and the other in a counter clockwise direction as illustrated in Figure 2.22. Every time a walker has moved a Euclidean distance of t_{th} away from the leaf tip candidate, the two walkers will measure the distance between them as shown in Figure 2.23a. The walkers are implemented as a state machine running in only two states. Initially the walkers are set in a *leaf-tip* state measuring the distance between them and keeping track of

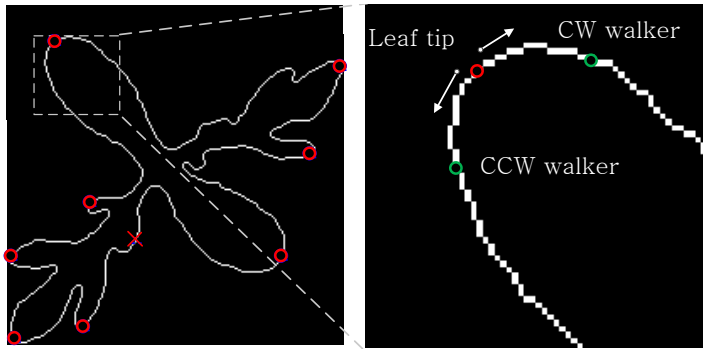
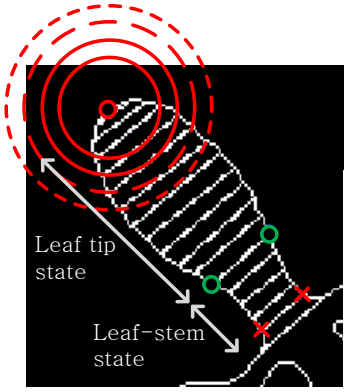
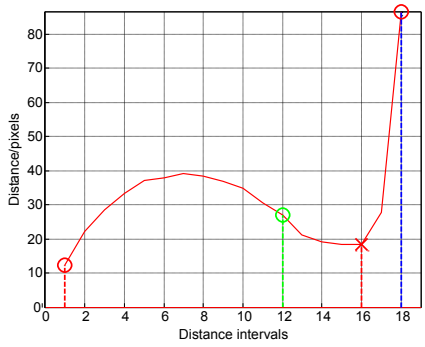


Figure 2.22: The walkers are sent in a clockwise and counter clockwise direction.



(a) Shows convex and concave areas and the local minima.

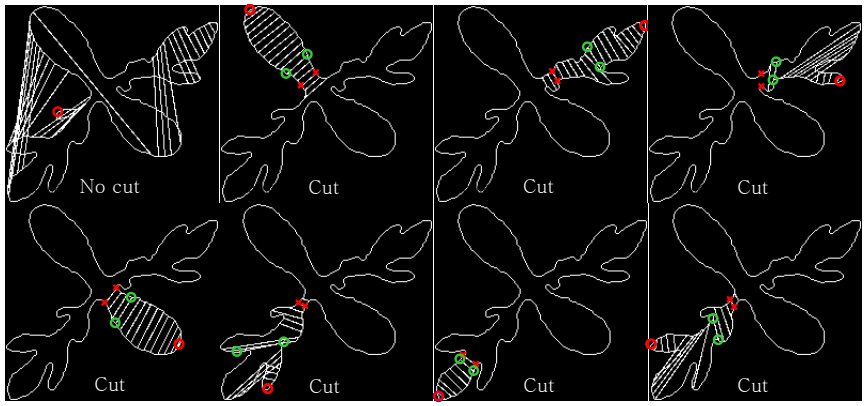


(b) Shows the local minima or leaf tip candidates.

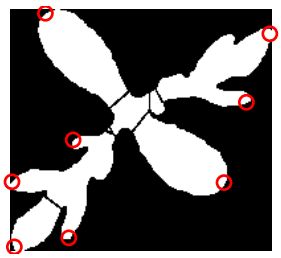
Figure 2.23: Shows the different states of the walkers.

the maximum distance as they walk along the boundary. When the walkers measure a distance of 75% (50% is used in [44]) of the maximum distance that they have measured along their way, the state will change to a *leaf-stem* state. In the leaf-stem state, the walker continues to measure distance, but will now keep track of the minimum distance. When the current distance exceeds the minimum distance by 250% (300% is used in [44]) the procedure will enter the *successfully-terminated* state, and the leaf will be cut at the point of the smallest distance recorded in the leaf-stem state. In Figure 2.23a and 2.23b the leaf tip is marked with a red ○, the changing of leaf-tip to leaf-stem state is marked with a green ○, and the leaf is cut between the red ×'s after which it can be extracted from the plant.

A leaf tip candidate, that is not actually a leaf tip, will often not enter the



(a) All cuts.



(b) All cuts and leaf tip candidates.

(c) Leaf tip candidates selects pieces to remake plant.

(d) Cuts are removed and connecting leaves.

Figure 2.24: Show how all the cuts removes away the stem only leaving non-connected leaves back.

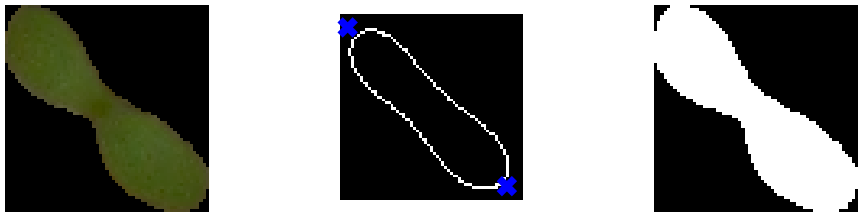
two states and therefore will not end in the successfully-terminated state. In this way invalid leaf tip candidates are removed and cuts are avoided. In opposition to the original paper, the walkers are not terminated when passing another leaf tip candidate. This termination state has been removed to allow foliage leaves, such as those shown in the figure above to be extracted as they contain multiple leaf tip candidates.

The cotyledons will be extracted as in the original paper. Yet, in this case, the idea is to let the foliage leaf perform multiple cuts and thereby actually cutting the foliage leaf and the whole plant into multiple pieces as shown in Figure 2.24a.

Some of these pieces are now selected by all valid leaf tip candidates to create a new plant, but as no leaf tip candidates are found in the stem, the stem is removed from the new plant leaving behind only leaves as shown in Figure 2.24b and 2.24c. The divided parts are now reconnected by reinserting cuts that connect the foliage leaves as shown in 2.24d.

Using this method produces good results on small plants having up to 4-leaves, but the robustness will fall rapidly for plants with more leaves, as they tend to overlap. In comparison with the procedure that was originally thought out in the article, leaves with multiple potential leaf tips (such as foliage leaves) will not be found (as foliage leaves). However, the robustness of the algorithm proposed in this project is lower as also invalid leaves or plant sections might be added.

Two examples of typical faulty leaf extractions are now presented: Figure 2.25 shows a plant at a very low growth stage. The boundary and the leaf tip candidate points are found, but walkers will never enter the successfully-terminated state, as the leaves are too slim compared to the stem, and the PSEP fails to extract the leaves of the plant.



(a) RGB image of plant.

(b) Filtered boundary and leaf tip candidates.

(c) No cuts are made.

Figure 2.25: An example of the PSEP not successfully extracting leaves.

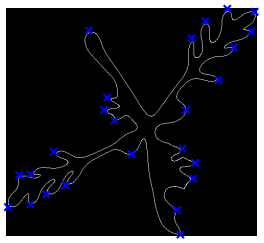
In Figure 2.26 a plant of a higher growth stage is presented. The boundary and leaf tip candidate points are found, and the walkers will provide different cuts to the plant. A plant of a higher growth stage will contain many leaves, and the walkers will therefore produce many cuts and thereby increasing the risk of creating invalid leaves leaves as those in Figure 2.26e and 2.26f. A manual procedure is therefore included to validate all leaves before they are included in the dataset.

2.3.3 Comparison of the "Fuzzy Clustering of leaves"-algorithm and the "PSEP"-algorithm'

As seen from the tests above, there are three areas where the two leaf-segmenting methods differ from each other. These three areas are:

- Segmentation of leaves with low solidity.
- Segmentation of overlapping leaves.

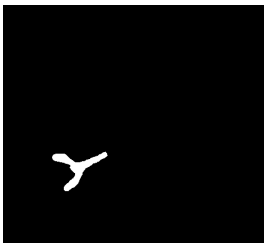
(a) RGB image of plant.



(b) Boundary and leaf tip candidate points.

(c) Mask showing the different cuts performed by the walkers.

(d) Leaf 1.



(e) Leaf 2.

(f) Leaf 3.

(g) Leaf 4.

(h) Leaf 5.

Figure 2.26

- Computational cost.

Both algorithms are quite good at segmenting round, convex leaves, but when it comes to leaves with low solidity, the *clustering based*-algorithm fails, due to its preference for convex leaves. One the other hand, the *PSEP based*-algorithm requires that all leaves are non-overlapping for it to be able to handle them. The PSEP-algorithm in the form described in [44] prefers convex leaves as well, but with the described modifications it is now able to handle non-round leaves like the scentless mayweed foliages that are shown in Figure 2.24.
Another not negligible detail is the computational cost of the two algorithms. While the PSEP-algorithm is working on one image the whole time, the Individual leaf extractions-algorithm has to create images over and over again in order to measure the fitness of the chromosomes that make up the leaves. For this reason, the clustering based-algorithm is not suitable for a real-time implementation, but could be used in an offline application for weed mixture estimation.

2.4. Leaf Straightening

In this section, a method to "straighten" or "normalize" leaves will be described. The problem with leaves is that they are soft and non-rigid, which leads to big variations in the appearance of leaves in the images.

The bigger variation in the appearance, the harder the classification process will be leading to more classification errors.

Thus, it is desirable to remove some of this variation and only keep the general shape information, by estimating the "straightened" leaf. A warping is then performed with the purpose of removing the bend of the leaf, thereby achieving the same shape of a pressed leaf or the image of a leaf in a photocopier.

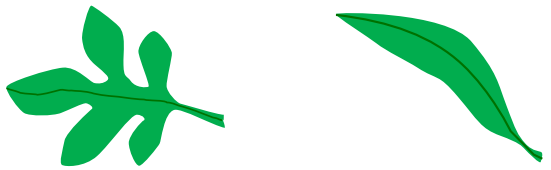
When doing this, it should be remembered that information is also removed. For instance it might be that some species have a larger tendency of bending than others, which would be a feature that is removed in the process.

In the case where the plant species is known beforehand, the shape of the leaf can be parametrized and straightened using active shape models as described in [45]. However, this is not possible, as the straightening process is used as a step in actually finding the species. Therefore, the method has to be generally applicable, which means that no prior knowledge about the leaf species can be used. This also means that there will be no ideal leaf for comparison and all information therefore have to be extracted from the single image.
The process of estimating the leaf structure could be eased, or could at least give a more precise result by having a 3D structure of the leaf. Such a structure is not accessible in the database, but in future applications, the structure could be achieved in multiple ways. For instance, by using structure-from-motion by taking the images from a moving vehicle or by simply using a stereo-camera. For some shiny leaves, the leaf-structure is seen in that way that reflections from the camera flash appear in the surface, almost perpendicular to the lens, but still the exact bending cannot be extracted from this reflection alone, as the amount of reflection depends on the shininess of the leaves.

To straighten the leaf, it is assumed that the centre vein of the straightened leaf forms a straight line and that the leaf is symmetrically shaped around this line.

The method described below requires that the leaf shape only has one edge that is perpendicular to the centre vein of the leaf, as shown in Figure 2.27

The first step in straightening the leaf is to find the two end-points of the leaf, which is done to separate the leaf-curvature on both sides of the leaf. The process of finding these endpoints is similar to the method described



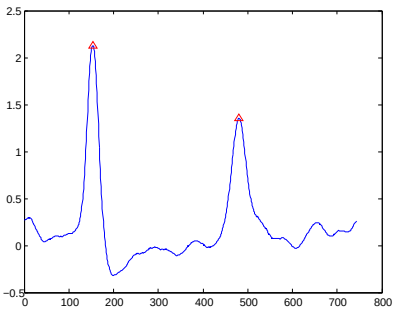
(a) Bad leaf shape for straightening

(b) Good leaf shape for straightening

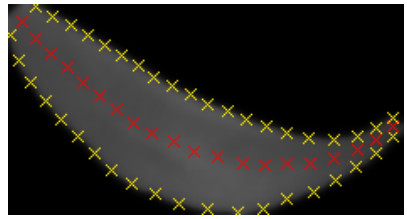
Figure 2.27: Good and bad leaf shapes for straightening using this method

in Section 2.3.2.2, where the two largest peaks indicate the end-points as illustrated in Figure 2.28a. When the end points have been found, each side of the leaf is divided into small sections. This is done by sampling fix-points on each side of the leaf which, because of the symmetric assumption, come in pairs that are placed on each side of the leaf. Again, because of the symmetric assumption, the line between these pairs will be orthogonal to the centre vein of the straightened leaf.

Because of the perspective of the leaf, the sampled fix-points cannot be assumed to be equidistant. However, at the moment, the fix points will be treated in this way, as the real, projected distances are unknown. An example, of such a sampled leaf can be seen in Figure 2.28b, where the leaf have been sampled 10 times on each side.



(a) Curvature of the leaf. The endpoints is marked with red triangles



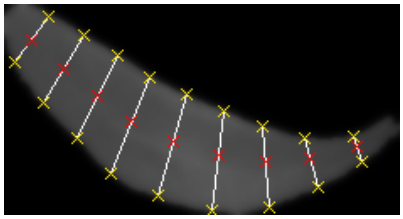
(b) Equidistant sampling. The estimated centre vein is marked with red crosses

Figure 2.28: Finding tip-points and sampling points on the leaf

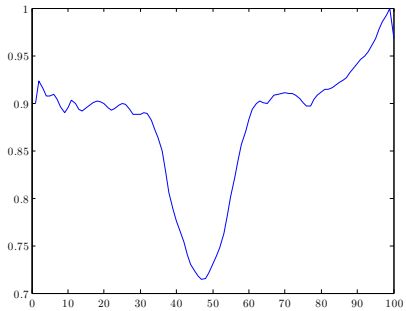
The next process is to estimate the centre line of the leaf. This is done in order to give a better estimation of the curvature and width of the leaf as the vein is assumed to be a straight line in the middle of the leaf.
To find the centre vein a line search is done on the line between each pair of samples. This line is made as a 100 points bicubic interpolation of the

pixels between the two sample points and the aim now is to find extrema corresponding to veins. A problem arises when the centre vein is unclear and covered in noise, which is shown in Figure 2.29c, where multiple points candidates to be the top point. A method to remove the noise while keeping the vein is to use total variation denoising, which is described in Appendix E. Total variation denoising is a denoising technique that tries to remove noise from the image while keeping edges. The method uses a regulation term which weighs how much the denoised image should look like the original image, and to what degree it should be flat. The result after doing such a denoiseing is that the image appears "patchy" with strong edges between the patches. However, by weighting the original image sufficiently high, gradients can be kept.

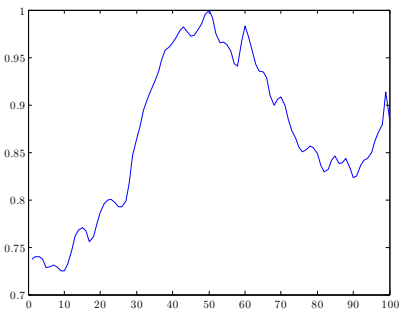
Vein Estimation When the leaf has been filtered from noise, the vein can be found by making a line search between the opposite sampled points as shown in Figure 2.29a. The line search is made by doing a 100 points



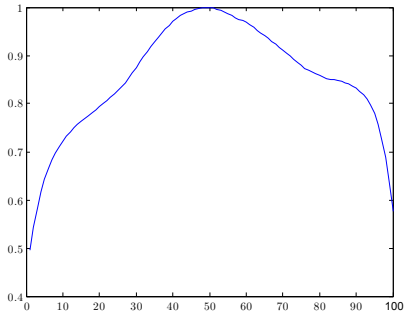
(a) Lines for vein searching



(b) Example on dark vein on sugar beet leaf without TV denoising



(c) Noisy leaf cross-section of sugar beet leaf



(d) Noisy leaf cross-section of sugar beet leaf filtered with TV denoising

Figure 2.29: cross sections of sampled sugar beet leaf

interpolation using the MATLAB function improfile, which returns a vector of the intensities along the line.

For some leaves, the vein is darker than the average colour of the leaf, whereas for other leaves, it is lighter. For some leaves, the intensity of the vein changes from being lighter than the average colour in some places of the leaf to be darker in others. Therefore, to take both cases into account, the vein is found at the largest difference from the mean intensity along the line. An example of a dark part of the vein is seen near the stem in Figure 2.29a, where the cross section is seen in Figure 2.29b. Likewise a crosssection at a light place is shown in Figure 2.29c.

However, as illustrated in Figure 2.29c the top points are close to indistinguishable from each other. Especially when looking at the largest difference from the mean, which results in even more candidates.
Using TV denoising will provide a more easily distinguishable top point, as seen in 2.29d. It should also be notished that the leaf-boundary is forced towards zero as shown in Figure 2.29d. This has to do with the black background.

To make the estimate more robust, the estimation of the vein is made as a linear combination of the current estimate weighted by $2/3$ and the previous estimate weighted by $1/3$. This should be an acceptable estimate, as long as the sampling distance is reasonably small compared to the twist of the leaf[12].

Until now the edge has been sampled equidistantly. That is clearly not correct, as the perspective of the leaf is thereby omitted. The edge should therefore be re-sampled according to the real distances and not the projected distances that can be seen in the image. However, as the images of the plants are only taken from one direction, it is not possible to extract the correct bending of the leaf, which is necessary to find the correct re-sampling along the contour.

Vein rectification When the centre of the leaf has been found, it is time to rectify the leaf. The first step in this process is to divide the leaf into small quadrilaterals that will be processed individually. The quadrilaterals is made from two pairs of samples as shown in Figure 2.30a.

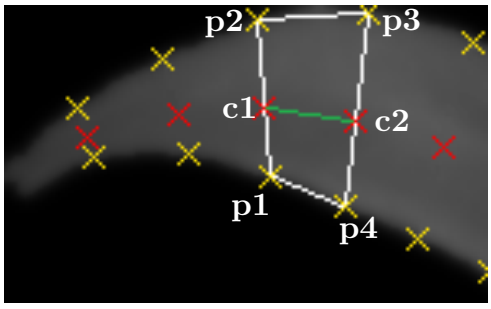
The reason for processing the leaf in small sections is that the leaf will be straighened using a homography transformation, in which straight lines are preserved. Therefore, the quadrilaterals have to be small enough to make this assumption valid. At the same time, it makes it easier to implement the rectification, using three general rules for how these shapes make up the straightened leaf shape. These rules are that:

[12]The MATLAB script is located in: `Matlab/Plantnormalization/findcenterline.m`

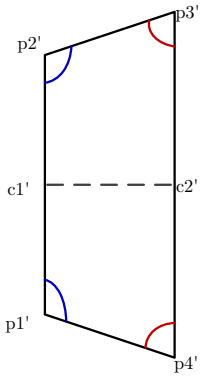
1. The leaf is symmetric around the centre vein,
2. The two cross sections of the leaf and the quadrilateral are parallel,
3. The quadrilateral is an isosceles trapezium,

where the last two rules are consequences of the former. Thereby the desired quadrilateral looks like Figure 2.30b.

As the real width of the leaf is unknown, the distance $|p1'p2'|$ in Figure 2.30b will be set to two times the largest of the distances from the two corners ($p1$ and $p2$) in the unrectified quadrilateral in Figure 2.30a to the centre line. The same procedure goes for the distance $|p3'p4'|$ in Figure 2.30b, where the distance will be the largest of the distances from $p3$ and $p4$ in Figure 2.30a to the centre line.
The width $|c1'c2'|$ of the trapezium will be set to the length of the centerline in the quadrilateral shown in Figure 2.30a



(a) Quadrilateral made from two sets of curvature samples



(b) Desired isosceles trapezium for the quadrilateral

Figure 2.30: Example on the desired rectification of the quadrilaterals from the leaf

To make the quadrilateral points $\mathbf{p}_i$ on the leaf in Figure 2.30a match the points $\mathbf{p}'_i$ on the desired isosceles trapezium in Figure 2.30b, we need to find the homography $\mathbf{H}$ that makes this transformation. The value i defines the index of a point, where $i = 1, \ldots, n$ for n points. The points $\mathbf{p}'_i$ can be estimated as [46, p. 78]:

$$\overset{\hat{\mathbf{p}}_i}{\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ \hat{w}_i \end{bmatrix}} = \overset{\mathbf{H}}{\begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}} \overset{\mathbf{p}_i}{\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}} \tag{2.21}$$

$$\overset{\mathbf{p}'_i}{\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix}} = \overset{\hat{\mathbf{p}}_i}{\begin{bmatrix} \frac{\hat{x}_i}{\hat{w}_i} \\ \frac{\hat{y}_i}{\hat{w}_i} \\ \frac{\hat{w}_i}{\hat{w}_i} \end{bmatrix}} \tag{2.22}$$

As can be seen from Eq. 2.21 and 2.22, the points are listed using homogeneous coordinates, which is necessary in order to handle the problem using simple multiplications.

From this we get:

$$x'_i = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \quad \text{and} \quad y'_i = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \tag{2.23}$$

$$\Downarrow$$

$$\begin{aligned} x'_i\left(h_{20}x_i + h_{21}y_i + h_{22}\right) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y'_i\left(h_{20}x_i + h_{21}y_i + h_{22}\right) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned} \tag{2.24}$$

$$\Downarrow$$

$$\overset{\mathbf{A}_{2n\times 9}}{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & n & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & n & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix}} \overset{\mathbf{h}_9}{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}} = \overset{\mathbf{0}_{2n}}{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}} \tag{2.25}$$

This defines a least squares problem [47], which is to minimize:

$$\min \|\mathbf{A}\mathbf{h} - \mathbf{0}\|^2 \tag{2.26}$$

As $\mathbf{h}$ is only defined up to scale [48], it has to be fixed. This can be done by setting the last entry $h_{22} = 1$. The solution is now achieved from the singular value decomposition of $\mathbf{A}$, and then set $\mathbf{h}$ equal to the singular vector corresponding to the smallest eigen value[13].

To make a homography transformation, at least four fix-points are required, $n \geq 4$. However, as the centre line should also be fixed, all six points

[13] The MATLAB script is located in:
`Matlab/Plantnormalization/hexRectify.m`

(a)Rectified parts of the leaf (b)Connected parts

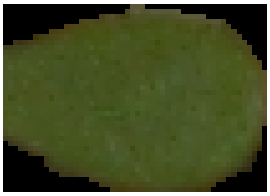
Figure 2.31: Leaf after it has been "straightened"

marked in Figure 2.30b must be used. On the two ends of the leaf, either (p1, p2 and c1) or (p3, p4 and c2) will be located at the same point, but this will not be a problem, as there are still four unique points.
When all quadrilaterals have been rectified, the only thing left is to combine the fractions. This is an easy step, as we know that the centre vein of the leaf is in the centre of the rectified quadrangles, whereby the pieces can be aligned at the centre. The pieces and a complete, straightened leaf is shown in Figure 2.31.

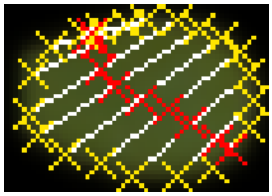
Evaluation of the leaf straightening method The method described above for straightening leaves is made with assumptions, which are not valid for all kinds of leaves. First of all the leaf is expected to have its two sharpest corners at the leaf tip and at the stem. For most small foliage leaves this can not be expected to be the case. Such a case, where the method fails, is when testing on a cotyledon of Scentless mayweed, where the size of the cut of the stem account for a big part of the total circumference of the leaf. This is shown in Figure 2.32. Conversely, one could argue, such a leaf might not require rectification.
Another problem arises when the leaf margin changes direction so that the cross sections of the leaf intersect with the margin at other points. One of these cases where the method fails is shown in Figure 2.33 of a Shepherd's-purse foliage leaf. The method therefore needs improvements to handle these cases as well, though a natural first step would be to only straighten leaves that are bending. To test whether leaves are bending, the area of the convex hull relative to the total area of the leaf (solidity) could be used as a measurement.

2.5. Segmentation Tool

To handle the step from raw RGB images to segmented and labelled plants and leaves, a segmentation tool has been developed. The different steps in the segmentation tool have not been documented in details though



(a) Input cotyledon with low curvature and big stem cut.



(b) Centre line of (a).

(c) Bad straightening of (a).

Figure 2.32: Cotyledon of Scentless mayweed after it has been "straightened". See how the method fails to find the leaf tip and stem, which result in undesirable behaviour.

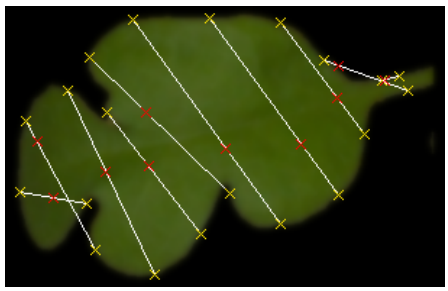
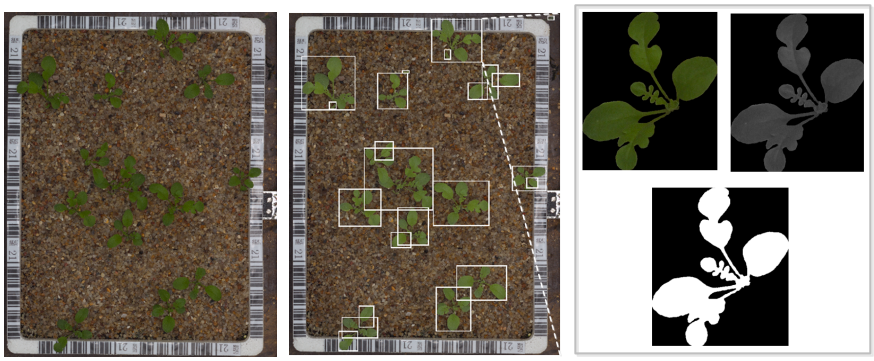


Figure 2.33: Foliage leaf of Shepherd's-purse that cannot be straightened due to intersecting cross sections

a great effort has been invested in this procedure. The first step is an automated procedure for colour segmentation. The tool uses ExR-ExG described in Section 2.2 in order to extract plants/green material from an image, thereby removing unwanted background. Afterwards, green materials are connected in structs. Each struct contains an RGB image, a greyscale image, a binary mask and a bounding box. Figure 2.34a shows an (cropped) RGB image from the database. Figure 2.34b presents the result of the colour segmentation, where each segment is encapsulated by a bounding box, and finally, the images stored for each plant struct in Figure 2.34c.
A function is implemented in MATLAB[14] to make a colour segmentation and to place plants into structs. Another function has been implemented in MATLAB to show the segmentation as seen in Figure 2.34b adding a bounding box around every segment and adding a little brightness to the plants for better visualization.

[14] The MATLAB script is located in:
`Matlab/Segmenting/SegmentingTool/simpleSegmenting.m`



(a) Cropped image from the database.

(b) Green elements are wrapped in a bounding box and highlighted with a little brightness.

(c) RGB, greyscale and the binary mask of a plant.

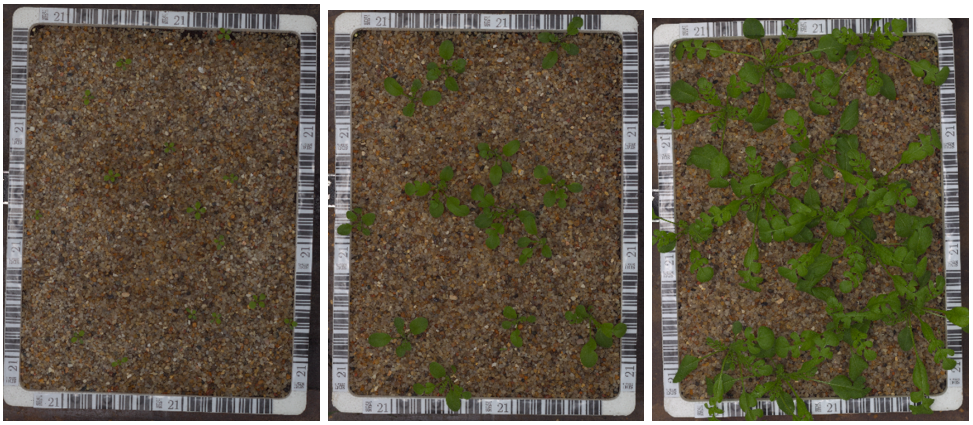
Figure 2.34: Green elements are divided in to plant structs. Each plant is shown with a bounding box and stored as an RGB image, a greyscale image and a binary mask.

The database consists of images of multiple species at different growth stages. For each species, four trays are filled with plants. Images are then taken in intervals of 1-4 days throughout the different growth stages. Initially all plants are well separated (also due to some pruning), but in the final growth stages, plants will start to overlap. In Figure 2.35 different growth stages are shown. The big overlap of plants in the final growth stage, should be noticed.

A script is therefore developed with the purpose of selecting overlapping plants and divide them into multiple plant structs. In Figure 2.36 the procedure for selecting and dividing plants has been illustrated. In Figure 2.36b overlapping plants are selected and highlighted in blue. In Figure 2.36c the selected overlapping plants are divided by manually drawing black lines to separate plants. Finally valid plant elements are selected and stored as shown in Figure 2.36c and 2.36d.

Another problem is that some leaves are not attached to a plant. A script is therefore needed to manually join individual leaves into a whole plant/struct. An image of all segments are shown in Figure 2.37a with a bounding box to show how plants are divided into segments. A left-click on multiple elements adds them into a single plant struct while colouring them red. A right-click will end the selection of leaves, colour them blue and start a new selection of leaves as shown in Figure 2.37a. By pressing *space*, the whole session ends and new structs are made as seen in Figure 2.37c.

The fourth step in the segmentation tool is made to manually discard



(a) Growthstage ≈ BBCH 14. Date: 2012-12-07. Small plants physically separated.

(b) Growthstage ≈ BBCH 17. Date: 2012-12-14. Medium plants start to overlap.

(c) Growthstage ≈ BBCH 33. Date: 2012-12-20. Almost all plants overlap.

Figure 2.35: An example of the different growth stages of shepherd's-purse in the provided database.

unwanted elements by pointing them out one by one, which turns them red as illustrated in Figure 2.38. Pressing *space* ends the session and the marked elements are removed (a plant is selected only to show that unwanted/red elements are removed). The elements that are removed are either green noise, which wrongly has been detected as a green element, or plants that are overlapped by other plants.

The steps described above are all implemented in the same MATLAB script[15]. Besides ways of redoing minor steps throughout the segmentation, a backup is also automatically stored at each of the four step to avoid restarting a whole session if something goes wrong.

All overlapping and discarded elements are stored to be on the safe side, but a minor MATLAB scripts[16] removes all overlapping and discarded plants from the backup.

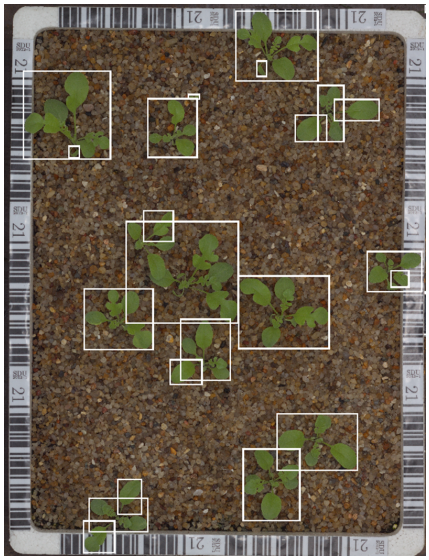
The next step seeks to divide plants into individual leaves.

The PSEP algorithm has been used in this stage, as it enables extraction of foliage leaves and non-ideal leaves in general, and because it performs must faster. The MATLAB script[17] runs through all leaves and stores the

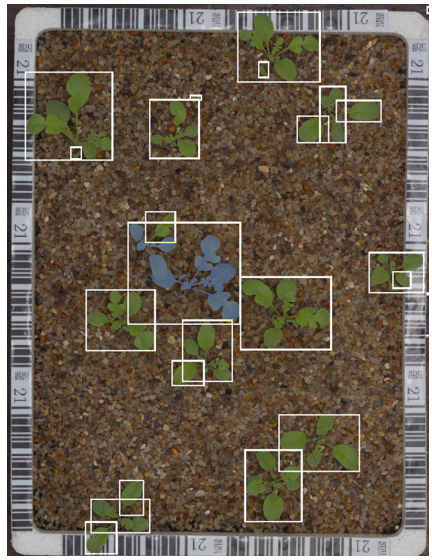
[15] The MATLAB script is located in:
`Matlab/Segmenting/SegmentingTool/segmenting1to4_MultiplePlants.m`

[16] The MATLAB script is located in:
`Matlab/Segmenting/SegmentingTool/segmenting5_MultiplePlants.m`

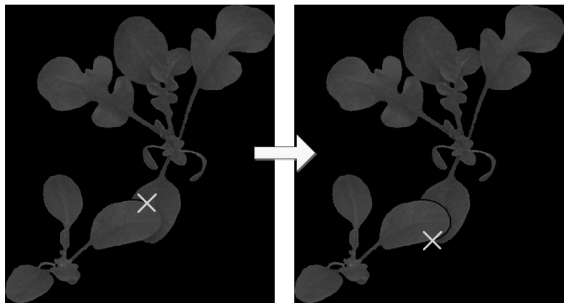
[17] The MATLAB script is located in:
`Matlab/Segmenting/SegmentingTool/segmenting6_PSEP_dividingPlantsInLeaves.m`



(a) All segments.



(b) All overlapping elements are selected manually.

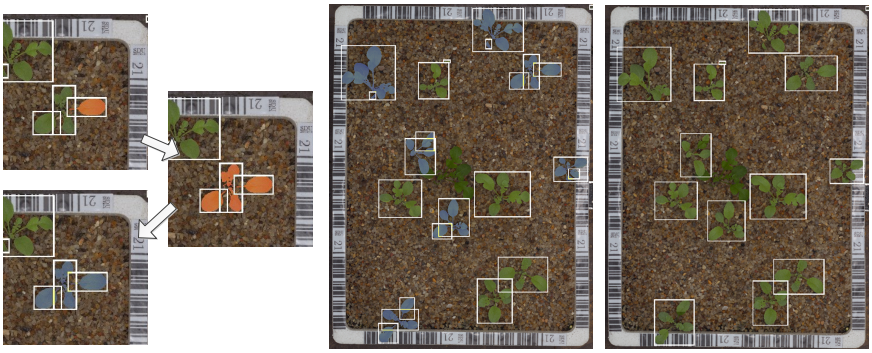


(c) Overlapping elements are divided manually by drawing black lines, dividing an element in two/more elements.

(d) Valid and whole plants are selected (marked by blue).

Figure 2.36: The procedure of selecting overlapping plants, dividing them and selecting whole plants.

bounding box and binary mask for each leaf found in the plant.
To make it possible to only describe a species within a certain growth stage, a segmentation step manually labels each plant within the different growth stages of either sprout, plant of only cotyledon leaves, a plant of both cotyledon and foliage leaves or a plant at a high growth stage. Each plant is



(a) Leaves are connected into plants. **(b)** All leaves have been selected. **(c)** Result after segmentation.

Figure 2.37: The procedure of selecting leaves and connecting them into a single plant.

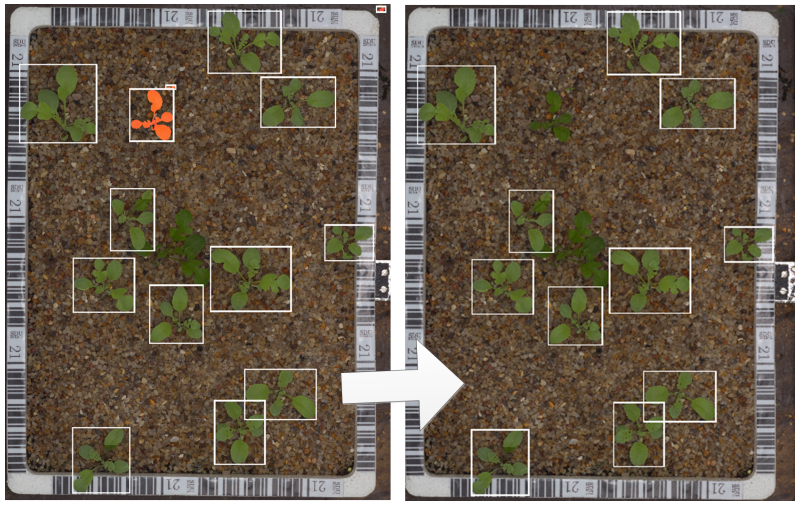


Figure 2.38: The procedure of selecting unwanted elements by manually pointing them out in the scene.

individually inspected and labelled in a MATLAB script[18]. The script shows an image of the whole scene making it possible to get a feel of the size of the plant sample, as illustrated in Figure 2.39. A number is then pressed to label the plant: 1) Discard element 2) Sprout element 3) Cotyledon 4) Cotyledon+foliage 5) Overgrowth. Finally a MATLAB script[19] is used to

[18] The MATLAB script is located in: `Matlab/Segmenting/SegmentingTool/segmenting7_dividingPlantsInGrowthstageTypes.m`

Figure 2.39: Select growth stage

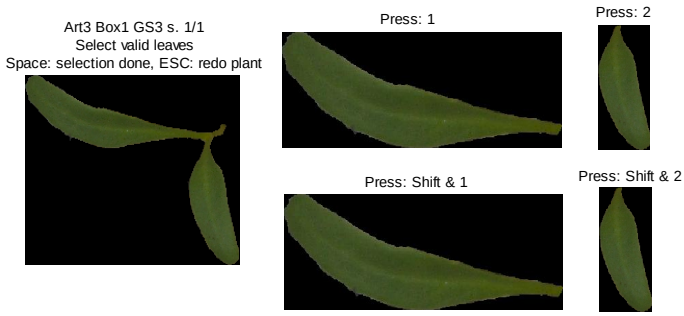


Figure 2.40: Valid leaves are selected. Choosing between two implementation of PSEP.

manually inspect each leaf labelling it either not-leaf, cotyledon or foliage. The procedure goes as follows: 1) Cotyledon leaves are selected pressing a number (with or without *shift* as the PSEP algorithm is implemented in two ways). 2) Selection is ended by *space*. 3) Foliage leaves are selected. 4) Selection is ended by *space*. 5) Each leaf is validated as either a bad, an OK or a good leaf by pressing 1,2 or 3 respectively.

2.5.1 Plant data structure

All the data, created by the segmentation tool, is provided in the raw database. It contains 7 different species with four plant trays for each species. These species are then photographed 7 to 9 times over the different growth stages, thereby providing a total of 236 images. These images contain around 2800 structs that all have to be treated in 8 segmenting stages. A data structure has therefore been used in order to provide easy and fast access to all the data. Firstly data is being stored for each segmentation step to

[19] The MATLAB script is located in:
`Matlab/Segmenting/SegmentingTool/segmenting8_EvaluateLeaves.m`

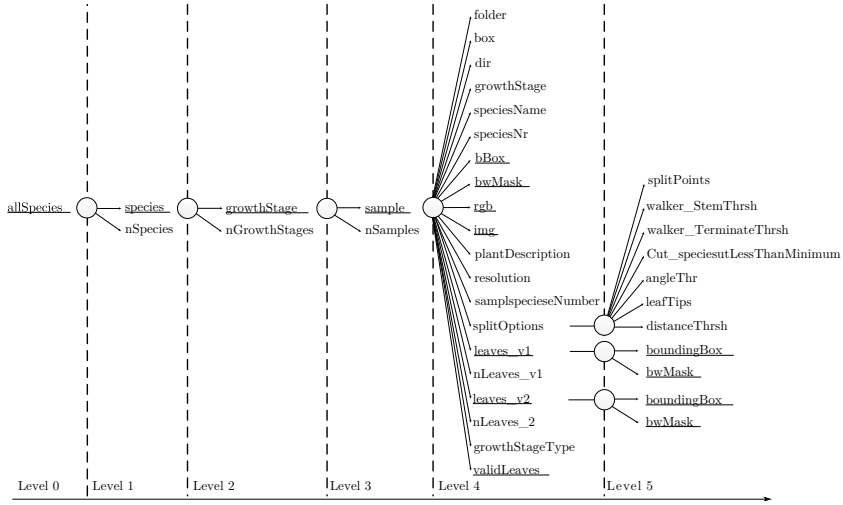


Figure 2.41: Numbers of labelled elements in the dataset for the different species

avoid redoing steps, if things have gone wrong. In each step, all plants are not stored in a single file, but in multiple .mat files - one for each species, growth stage and tray. This might seem messy, but the loading time is reduced significantly as fewer plant samples are loaded at the same time. A MATLAB function[20] is created to get one or multiple wanted plant samples, while only loading the necessary data files from the database. The function returns a struct containing lots of valuable (and less valuable) data for each plant. Figure 2.41 shows the structure and the provided information for each plant.

The structure is divided in six levels.

0. All plants are contained by the lowest level of the struct.
1. All species and the number of species (nSpecies) are accessible.
2. All growth stages and the number of growth stages (nGrowthStages) within the selected species are accessible.
3. All samples and the number of samples (nSamples) within the selected species and growth stage are accessible.
4. Information and image for the plant is accessible.
5. Information and the image of the extracted leaves are accessible.

Table 2.1 shows the number of plants structs for the 7 species including a count of the three plant elements; plants, cotyledons leaves and foliage

[20] The MATLAB script is located in:
`Matlab/DatabaseTools/getMultiplePlants.m`

leaves. As Maize does not have foliage leaves, and Winter Wheat (grass) does not have actual leaves theses count to zero.

#	Species	Plant structs	Whole plants	Cotyledons	Foliages
1	Maize	250	250	194	0
2	Wheat, winter	198	198	0	0
3	Sugar beet	462	325	832	23
4	Scentless mayweed	578	575	404	473
5	Shickweed	706	703	1199	309
6	Shepherd's-purse	275	255	386	322
7	Cleavers	344	130	394	231
total		2813	2436	3409	1358

Table 2.1: The number of plants and leaves for the seven species

2.6. Segmentation Discussion and Conclusion

In this chapter, the procedure for plant and leaf segmentation has been documented. Plants have been segmented from the background using ExG-ExR, whereafter these segments have been divided into individual leaves using a modified PSEP algorithm. A fuzzy clustering method to extract overlapping leaves has also been implemented. The last method is able to handle overlapping cotyledons. However, the method is computationally expensive and it does not perform well for non-convex-shaped leaves.
A method for straightening leaves has also been proposed in order to overcome some of the within-species variance of bending leaves. The method performs well for some leaves, but needs optimization for it to become more robust; either by a self-detection of whether the leaves are feasible for the straightening process or by extending the method to handle a bigger variation of species.

A segmentation tool sets a framework for handling the different segmentation steps for all plants. The tool includes automated steps of extracting plants and leaves, but it also provides some manual steps concerning the labelling of plants and leaves. Plants are labelled by their growth stage, while leaves are divided into either cotyledon and foliage leaves including a rating of how ideally-shaped these leaves are. For a fully automated system, the manual labelling step has to be automatized or avoided, but for testing purposes it makes it possible to detect the robustness for different degrees of non-ideal leaves.

Chapter 3

Feature Descriptors

With a good segmentation of both plants and leaves in hand, the next step is to extract features.

In pattern recognitions, features are values that describe certain characteristics of an object. A good feature provides discrimination between the objects or classes that are to be separated. This will, of course, depend on the given domain, but it is often desirable to have features that are invariant to translation, rotation and noise.

As plants will experience large transformation during their growth, a good feature in the plant domain will also have the quality of being growth invariant. In the literature of plant recognition, two feature branches are dominating. One branch describes the whole plant [14, 15, 16, 17] and the other branch describes the leaves of the plant [49, 18]. Features describing the whole plant are assumed to be very dependent on growth stages, as the plant will go from sprout to being a plant with more and more leaves. Based on this assumption, features describing only the leaves from a plant are presumed to be more invariant to growth. However, at low growth stages, the sprouts of different species have very similar leaves, which complicates the discrimination of the species.
Both branches will have advantages and features are therefore calculated for both whole plants and single leaves.

There are many aspects of the appearance of plants and leaves that can be considered when finding features. For botanists, shape, leaf margin and vein structure are the most useful features[50]. In image processing, shape and leaf margin/contour typically are easier to extract than the vein structure, because of greater contrast to the background, making contours an optimal feature for image processing. At the same time, working with the veins of the leaves requires a high image resolution, low blurring and high contrast, which is not the case for most cotyledon surfaces in the dataset.

3.1. Related work in the field of feature extraction

Features describing plants or leaves in computer vision are grouped in multiple categories [32]. One category describes the shape or region of the whole plant by using the mask of the element e.g. ratio between principal axes (elongation) and ratio between area and convex hull area (solidity). Another category, defined as the contour based features, uses a description of the boundary of the object. In elliptic Fourier the contour is approximated by a certain number of Fourier coefficients [49] and [18]. Elliptic variance [51] is also a contour based feature, providing a measure for the variation of the contour compared to an ellipse. The contour may also be described by active shape models [14, 52, 15]. A third category uses the reflectance properties for a plant. In the visible spectrum, this group is simply defined as colour features, but the plant may also be described in the near infrared spectrum, outside the visible spectrum. In [34] the reflectance properties of the volunteer potato plant and the sugar beet were used as features by measuring multiple wavelengths in a band from 450 to 1650 nm. A fourth feature category uses the texture of the plant as described in [53], using classical textural features.

3.2. Features Description

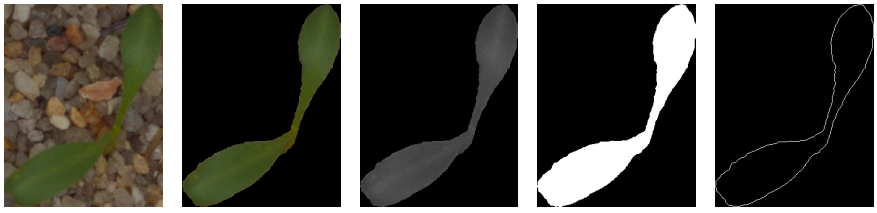
In the following section a broad range of features are documented and implemented. Some are general features used in pattern recognition, while others are targeted towards discriminating plants or leaves.
First, different presentations of a plant and the Region of Interest (ROI) are described in Section 3.2.1. The feature descriptors are then divided into the categories Shape Features, Contour Features, Colour Features and Other Features in Section 3.2.2, 3.2.3, 3.2.4 and 3.2.5, respectively.

Some features are merely included in the documentation for the sake of completeness, while the different variations of the distance transform features, the variation of elliptic Fourier, the variations of the distance elliptic Fourier and the stem thickness feature are more important contributions made in this project. To distinguish between a feature descriptor that output a single value (e.g Solidity) and the multiple values outputs by some feature descriptor (e.g Elliptic Fourier), the multiple values from a feature descriptor is defined as subfeatures.

A MATLAB script[1] has been made to run through the structs of all plants and leaves and calculate all features for these by using another MATLAB script[2].

3.2.1 Shape and Contour Definitions

Shape and contour features are calculated by using different representations of the plant, thereby including or removing information from the original RGB image, as illustrated in Figure 3.1a. All features in this project require the background (pixels not related to the specific plant) removed as shown in Figure 3.1b. In colour based features, the RGB image is required, while some features only rely on the greyscale image as in Figure 3.1c. Shape related features mostly require a binary image, which only contains the silhouette of the plant as shown in Figure 3.1d. This is also defined as the Region of Interest (ROI). Finally, the contour based features use the boundary of the plants, either as a binary image, Figure 3.1e, or as a sorted list containing all the points traced around the boundary in either the clockwise or counter clockwise direction. The boundary binary image and the sorted boundary list can be acquired by respectively using the built-in MATLAB functions called bwperim() and bwtraceboundary().



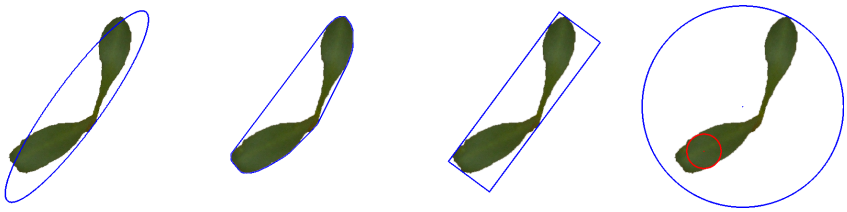
(a) Raw RGB image. **(b)** RGB image without background. **(c)** Greyscale image. **(d)** Binary plant mask image. **(e)** Binary plant boundary image.

Figure 3.1: The different representations of a plant.

Different features require a new representation of the ROI[54]. One representation defines an ellipse of the ROI as shown in Figure 3.2a. Orientation, centre, radius of the major and minor axis of the ellipse can be determined with the MATLAB built-in function regionprops(). The convex hull of the ROI is shown in Figure 3.2b and determined by the build-in MATLAB function regionprops(). The minimum bounding rectangle determines, as the

[1] The MATLAB script is located in:
`Matlab/Features/Demo_CalculateFeatures.m`
[2] The MATLAB script is located in:
`Matlab/Features/calculateFeature.m`

name implies, the rectangle with the minimum area that bounds the ROI as in Figure 3.2c. Finding the optimal solution is a computationally intensive procedure. An approximations is therefore determined using two different algorithms. The algorithms will not be described in details, but the first algorithm, rotating calipers [55], uses the convex hull of the ROI. The second algorithm [56] uses PCA to find the orientation of the ROI and afterward wraps a rectangle around the ROI in that direction. Both algorithms are implemented in MATLAB[3]. The minimum bounding rectangle is found by using the solution that finds the minimum area. The final presentation finds the largest circle fitted within the ROI (interior circle) and the smallest circle with a centre equivalent to the centroid of the ROI that can be wrapped around the ROI (exterior circle) as in Figure 3.2d.



(a) Ellipse. **(b)** Convex Hull. **(c)** Minimum bounding rectangle. **(d)** Interior circle (red) and exterior circle (blue).

Figure 3.2: Different presentations of the region of interest.

3.2.2 Shape Features

Object Area , A_{ROI},

The area of the ROI in pixels.

Determined in MATLAB by regionprops('Area'). The feature is by definition not scale invariant, but might be useful as the object area is closely related to the actual size of the plant (if all images are taken from the same height). This might be valuable when determining the size or the growth stage of a plant.

Object Perimeter , P_{ROI},

The length of the ROI contour in pixels.

Determined in MATLAB by regionprops('Perimeter').

Convex Hull Area , A_{CH},

[3] The MATLAB script is located in: `Matlab/Features/calMBR.m`

The area of the convex hull of the ROI in pixels. Determined in MATLAB by regionprops('ConvexImage') followed by regionprops('Area').

Convex Hull Perimeter , P_{CH},
The length of the convex hull contour in pixels. Determined in MATLAB by regionprops('ConvexImage') followed by regionprops('Perimeter').

Solidity Is a measure of the convexity of the objects. It is calculated as the ratio between A_{ROI} and A_{CH} also defined as the area ratio of convex hull [54], and can be calculated in MATLAB by regionprops('Solidity').

$$Solidity = \frac{A_{ROI}}{A_{CH}} \tag{3.1}$$

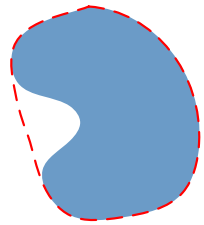


Figure 3.3: Solidity. Blue area shows the shape of ROI. Red dashed line shows the shape of the convex hull of ROI.

Convexity The ratio between P_{CH} and P_{ROI} [51].

$$Convexity = \frac{P_{CH}}{P_{ROI}} \tag{3.2}$$

Compactness Describes a ratio between the area and the major axis in the ellipse[57].:

$$Compactness = \frac{\sqrt{\left(\frac{4}{\pi}\right) \cdot A_{ROI}}}{L_{MajorAxis}} \tag{3.3}$$

Aspect Ratio The ratio between the length, $D_{\max}$, and the width, $D_{\min}$, of the minimum bounding rectangle[54].

$$AspectRatio = \frac{D_{\max}}{D_{\min}} \tag{3.4}$$

Rectangularity A measure of how rectangular-like the shape is given by the ratio between ROI area and the area of the minimum bounding rectangle [54].

$$Rectangularity = \frac{A_{ROI}}{D_{\max} \cdot D_{\min}} \tag{3.5}$$

Spherity A measure of how circle-like the shape is. It is given by the ratio between the radius of the interior circle r_i and the exterior circle r_e [54].

$$Sphericity = \frac{r_i}{r_c} \tag{3.6}$$

Eccentricity A measure of how circular a shape is. It is determined using the MATLAB build-in function regionprops('Eccentricity').

$$Eccentricity = \sqrt{1 - \frac{b^2}{a^2}} \tag{3.7}$$

where a and b respectively is the length of the major and minor inertia axis of the ROI.

Ratio of Principal Axis Defined as the ratio between the principal axis.

$$RatioOfPrincipalAxis = \frac{b}{a} \tag{3.8}$$

where a and b respectively is the length of the major and minor inertia axis of the ROI. Closely related to eccentricity.

Form Factor A third method to measures how circle-like the shape is. A perfect circle has a form factor value of 1. It is defined in [57] as:

$$FormFactor = \frac{4 \cdot \pi \cdot A_{ROI}}{{P_{ROI}}^2} \tag{3.9}$$

Hu-moment Moments in general are measures used to describe the shape of a set of points such as orientation, area or width [58, 57]. The raw moment for pixel-coordinate (x, y) in the discrete case is defined as[59]:

$$M_{pq} = \sum_x \sum_y x^p y^q I(x, y), \tag{3.10}$$

where $I(x, y)$ is the pixel intensity at coordinate (x, y). In the binary case, I can either be 1 or 0, which means that the moments are only based on the non-zero pixels. Therefore, I is removed and x, y is limited to only be defined over the region *Region* defined as the points in the binary mask that

equals 1.
Some variations of this moment are called the central moments. The central moments define the moments about the mean of x and x and is given by[59]:

$$\mu_{pq} = \sum_{x,y \in Region} \sum (x - \bar{x})^p (y - \bar{y})^q, \tag{3.11}$$

where $\bar{x}$ and $\bar{y}$ are the means of x and y, which defined from moments are given by $\bar{x} = \frac{M_{10}}{M_{00}}$ and $\bar{y} = \frac{M_{01}}{M_{00}}$. These central moments are translation invariant and can be invariant to scale by normalizing the moments to a scale given by[59]: $\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}$, where $\gamma = \frac{p+q}{2} + 1$. The seven first HU-moments are then given by[4]:

$$\phi_1 = \eta_{20} + \eta_{02} \tag{3.12a}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{3.12b}$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{3.12c}$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{3.12d}$$

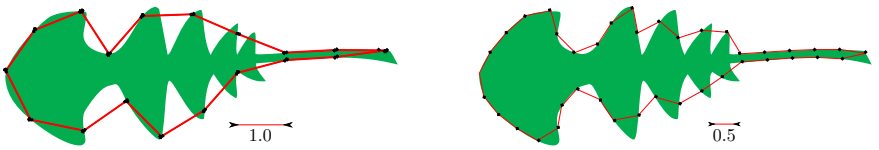
$$\begin{aligned} \phi_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + \\ & (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) \left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \end{aligned} \tag{3.12e}$$

$$\begin{aligned} \phi_6 = {} & (\eta_{20} - \eta_{02}) \left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + \\ & 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \end{aligned} \tag{3.12f}$$

$$\begin{aligned} \phi_7 = {} & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12}) \left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] - \\ & (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03}) \left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] \end{aligned} \tag{3.12g}$$

Fractal Dimension Fractal dimension is a measurement of the complexity of the boundary of an object. That is, how much does the object boundary change, when the measurement scaling of the boundary is changed? The fractal dimension value is a value describing the irregularity of an object and how much of the space it occupies. A true fractal object will still have fractal characteristics, when it is enlarged towards infinity. From this definition, it should be clear that the resolution of the object has big influence on the precision of this measurement, as details in a highly fluctuating boundary will be lost due to the sampling, which will then result in no more changes. Fractal dimension can be calculated in different ways[16, 57]. A simple

[4] The MATLAB implementation is based on [60]



(a) Ruler length = 1 unit. 18 steps around leaf contour

(b) Ruler length = 0.5 unit. 40 steps around leaf contour

Figure 3.4: The Richardson procedure for two different ruler lengths

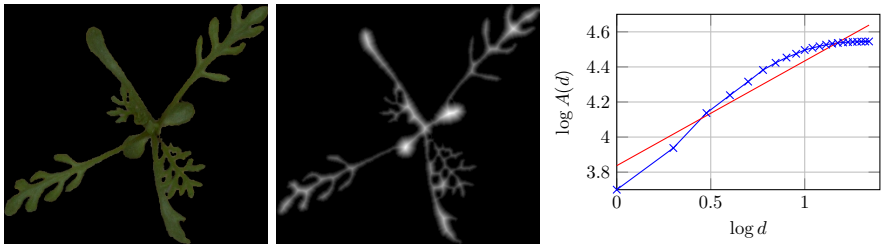
method is called the Box Counting method, in which a grid is placed on the image with a contour. The Fractal dimension is then the number of grid-cells within the contour at different grid sizes. As the grid size shrinks, the cells will be able to fill out the more fractal parts of the contour. For shapes with fractal characteristics, this means that the area covered with cells increases as the grid size is decreased.
Another method is the so-called Richardson plot. Here a ruler with a certain length is placed around the contour of the shape, starting at some point. The number of steps around the contour multiplied with the ruler length gives a perimeter measurement. By changing the length of the ruler, this measurement changes, and thereby produces the fractal dimension measurement[57]. The principle is shown in Figure 3.4.

However, according to [16, 57] a simple and precise method is the Minkovski method, which does not give identical results to the Box counting method and the Richardson plot method. Still, the results are correlated and therefore no more information will be added by using multiple methods[57].
According to this method, the distances from all pixels within the contour to the contour of the plant is found. These distances are grouped in a histogram with a bin-distance of one, indicating the distribution of pixels at certain distances from the contour. By taking the accumulative sum of this histogram, a measurement of the amount of pixels within a certain distance from the contour is found. The fractal dimension of the shape is then found by fitting a first order polynomial to a log-log plot of the measurements, from which the fractal dimension is given by Eq. 3.13 [57, 61, p. 605], which because of the fixed pixel-size can be approximated as follows:

$$FD = 2 - \lim_{d \to 0} \frac{\log A}{\log d} \approx 2 - \frac{\log A}{\log d}, \tag{3.13}$$

where $A\,(d)$ is the area of all pixels at distance $\leq d$ from the contour, which is simply the number of pixels within this distance.

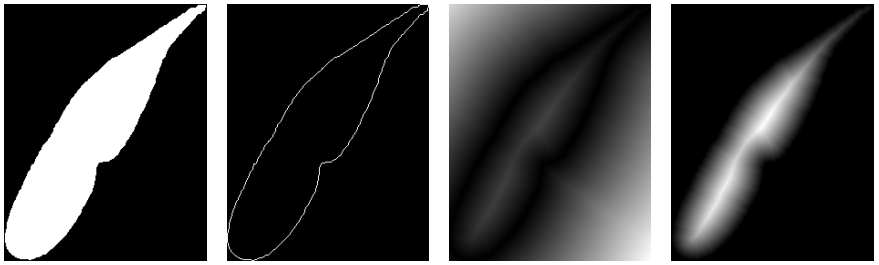
Distance Transform The distance transform $\mathbf{M}_{map}$ of a binary boundary image is a matrix or image with the same size as the binary boundary image,



(a) Input plant **(b)** Eucledean distance transform of leaf **(c)** Plot of accumulated histogram with first order line fitting

Figure 3.5: The Minkovski method for finding the fractal dimension

in which each pixel contains the distance in pixels to the closest boundary. To make the distance transform, a binary boundary image is created from the ROI as shown in Figure 3.6a and 3.6b
The distance transform is then found by assigning to all pixels within the boundary a value corresponding to the Euclidean distance to the closest boundary pixel. This is shown in Figure 3.6c. As only the area within the contour is of interest, all pixels outside the boundary is set to -1. Figure 3.6d shows the normalized distance map inside the plant mask.



(a) Plant mask. **(b)** Plant boundary. **(c)** DT normalized. **(d)** DT inside plant mask normalized.

Figure 3.6: Distance transform for the inside of a plant.

From the distance transform, $\mathbf{M}_{map}$, a distance vector $\mathbf{d}$ is created which contains all distances $d \neq -1$. This vector is dependent on both rotation and scale of the original image. To make the vector rotation invariant, $\mathbf{d}$ is sorted in ascending order to become $\mathbf{d}_{sort}$
This vector $\mathbf{d}_{sort}$ is the basic for three additional distance vectors, recommended by [17]:

- $\mathbf{d}_{scaled}$, which is $\mathbf{d}_{sort}$ normalized to the largest distance in $\mathbf{d}_{sort}$.

- $\mathbf{d}_{accu}$, which is the accumulated sum of $\mathbf{d}_{sort}$. This describes the amount of pixels with distances less than a given value.
- $\mathbf{d}_{accuScaled}$, which is $\mathbf{d}_{accu}$ normalized to the largest value in $\mathbf{d}_{accu}$.

When normalizing the vectors, the features become scale invariant. Whether this is good or not, depends on the camera set-up. If the height of the camera varies, scale will just be noise in the feature, as it does not reflect the size of the plant.

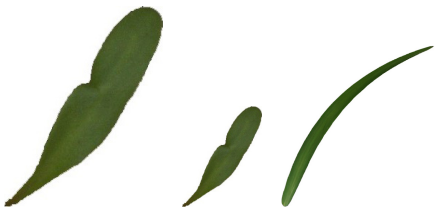
These vectors have different characteristics for different shapes, which is shown in Figure 3.7. To illustrate how the shape of the figure is reflected in the graphs, two very different leaves are used; a grass straw and a sugar beet leaf. The pixel indices are scaled to lie between -1 and 1, which will be explained later. From the two graphs it is seen that the amount of pixels with short distance to the boundary is larger for the grass straw than for the sugar beet leaf. The distance transform will therefore be able to discriminate leaves depending on their distribution of mass relative to their boundary.

In addition to these four vectors, $\mathbf{d}_{sort}$, $\mathbf{d}_{scaled}$, $\mathbf{d}_{accu}$ and $\mathbf{d}_{accuScaled}$, proposed by [17], a new scaling method is tried out. The vectors $\mathbf{d}_{sort}$ and $\mathbf{d}_{accu}$ are not scale invariant and $\mathbf{d}_{scaled}$ and $\mathbf{d}_{accuScaled}$ are scale invariant, but lots of information are removed when normalizing the distances. In the new method, $d_{scaledAreaRoot}$, the distances will be scaled according to the square root of the length of the vector, i.e. the square root of the area of the leaf. By doing this, a new vector is found, which is close to being scale invariant.
When scaling according to the square root of the area, the leaf is assumed to be a circle, for which the area is given by:

$$\begin{aligned} A &= \pi \cdot r^2 \\ &\Downarrow \\ r &= \sqrt{\frac{A}{\pi}} \end{aligned} \tag{3.14}$$

as π is a constant, we get that $r \propto \sqrt{A}$. The assumption of scale invariance is only true for circles. However, it is still close to being the case for non-round leaves. Figure 3.7h shows three input images, for which $d_{scaledAreaRoot}$ is calculated for the original input images, and the input images scaled to $1/4$ of its area. Here, it can be seen that the error is small when scaling to $1/4$ compared to the difference between the two species.
With this new scaling, it is still possible to get the information that e.g. normal leaves have pixels far from the contour compared to grass, which can not be seen from the other two scale invariant vectors, $\mathbf{d}_{scaled}$ and $\mathbf{d}_{accuScaled}$.

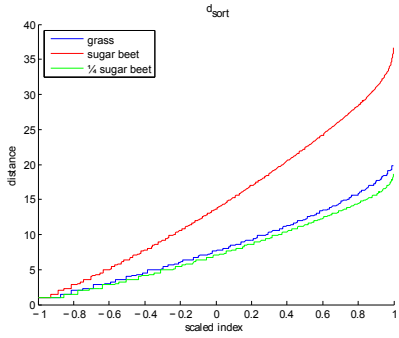
From these curves, it is necessary to extract features that are able to discriminate the different shapes. Two methods are recommended by T.



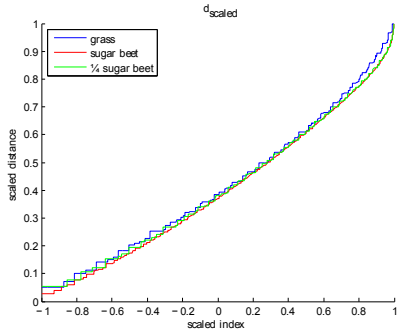
(a) Sugar beet input

(b) 1/4 Sugar beet input

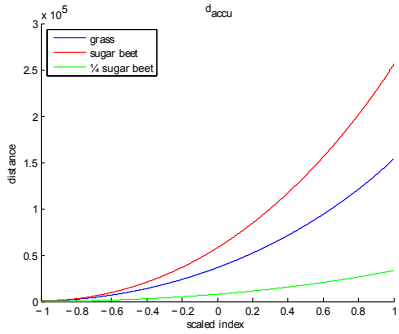
(c) Grass input



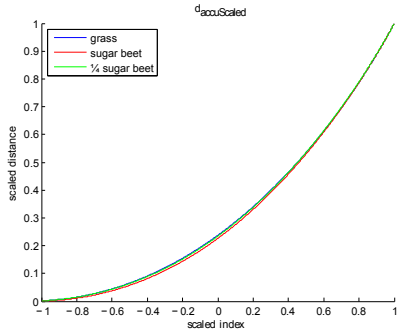
(d) Sorted distance vector, d_{sort}



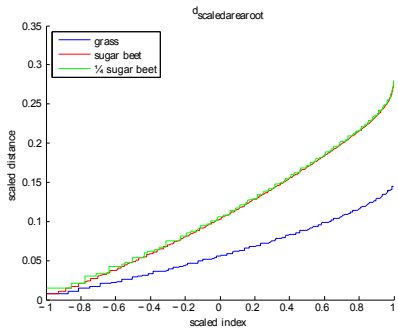
(e) Sorted and normalized distance vector, d_{scaled}



(f) Sorted and accumulated distance vector, d_{accu}



(g) Sorted, accumulated and scaled distance vector, $d_{accuScaled}$



(h) Sorted and scaled to square root of vector length, $d_{scaledAreaRoot}$

Figure 3.7: Distance transform of grass and sugar beet. The two sugar beet leaves are the same, but with different scale. Notice in (e) and (g) how d_{scaled} and $d_{accuScaled}$ are scale invariant, but the two species are almost indistinguishable. As opposed to this the new suggested method, $d_{scaledAreaRoot}$, shown in (h), is still scale invariant, but the two species are distinguishable

Giselsson [17]; re-sampling the curves and Legendre polynomial fitting. In the first method, a number of equidistant samples are taken along the index axis to represent different features. The problem with sampling is that only information about the sampling points are stored. The second method uses polynomial fitting using Legendre polynomials. The reason for using these coefficients over normal polynomial coefficients is that the coefficients from Legendre polynomials are orthogonal and thereby independent. The single features will therefore remain the same no matter the order of the polynomial.
The polynomial $P(x)$ is given as a weighted sum of the Legendre polynomials:

$$P(x) = \sum_{n=1}^{\infty} a_n p_n, \tag{3.15}$$

where a_n is the weight of the n'th Legendre polynomial, $p_n(x)$ of order $n-1$. All these polynomials are orthogonal which means that

$$\int_{-1}^{1} p_l(x)\, p_k(x) dx = 0 \ , \ \ l \neq k \tag{3.16}$$

The Legendre polynomials are normalized to $p_0(x)$, which is therefore set to 1. The polynomials are given by [62]:

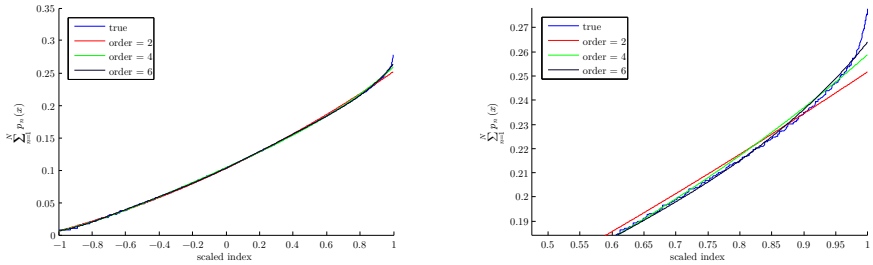
$$p_{n+1}(x) = \frac{(2n+1) \cdot x \cdot p_n(x) - n \cdot p_{n-1}(x)}{n+1} \tag{3.17}$$

The polynomial weights are calculated using MATLAB [63], where the optimum polynomial weights are found in a least square sense. For the sugar beet in Figure 3.7h, the Legendre polynomials for order $n = 2, 4$ and 6 are shown in 3.8. It is seen that the polynomials of different orders primarily differ in the area close to $x = 1$, where even the 2nd order polynomial is a good estimate for the curve in the range [-1:0.8], but not in the range [0.8:1]. For the grass sample, the second order polynomial is much closer to the real distance curve than for the sugar beet. It will therefore be interesting to find a feature that can show this difference and see if there is a tendency.

For this, the Pearson product-moment correlation coefficient, r^2, is used, which is a scale independent measurement of how well the Legendre polynomial fits the curves.

$$r^2 = \left(\frac{(y-\bar{y})^T (x-\bar{x})}{\sqrt{(y-\bar{y})^T (y-\bar{y}) \cdot (x-\bar{x})^T (x-\bar{x})}} \right)^2 \tag{3.18}$$

The error could of course be removed by increasing the polynomial order sufficiently, but by fixing the polynomial order to 2, most of the slope can be



(a) Matching Legendre polynomials to $d_{scaledarearoot}$ of sugar beet

(b) (a) focused on the area close to 1

Figure 3.8: Legendre polynomial fitting for sugar beet leaf from Figure 3.7a

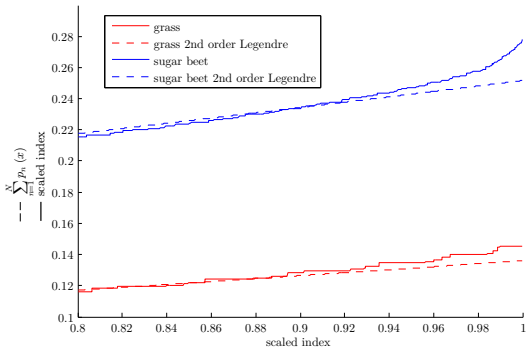


Figure 3.9: Difference from Legendre fit and distance transform for the grass and sugar beet example in 3.7

modelled except the slope close to 1, whereby r^2 only depends on this part. The example with grass and sugar beet is shown in Figure 3.9, where it should be clear that the approximation of grass by a second order Legendre polynomial is better than the approximation of a sugar beet leaf.
The distribution of r^2 for the cotyledons of the plant species, used in this project, is shown in box-plots in Figure 3.10. Even though the box-plots are overlapping, it is clear that higher order polynomials are necessary to model Scentless mayweed, which result in low correlation coefficients. However, for all species, large outliers exists.

Distance Transform Mean and Variance Two additional features related to the distance transform are described in [16]. The mean distance for the N samples inside the plant region.

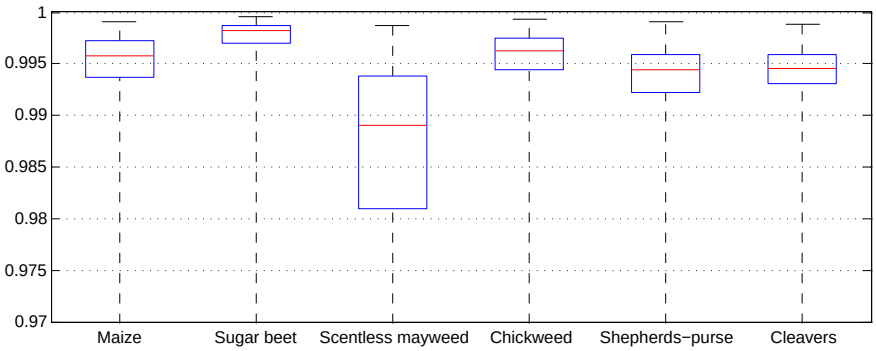


Figure 3.10: Boxplot of the correlations coefficient for the seven species, when fitting to a 2nd order Legendre polynomial

$$d_{mean} = \frac{1}{N}\sum_{n=1}^{N} d\left(n\right), \tag{3.19}$$

and the variance of the distance for the N samples inside the plant region.

$$d_{var} = \frac{1}{N}\sum_{n=1}^{N} \left(d\left(n\right) - d_{mean}\right)^2 \tag{3.20}$$

3.2.3 Contour Features

Circularity The circularity [54], also known as circular variance [51] is defined as:

$$C = \frac{\sigma_R^2}{\mu_R^2} \tag{3.21}$$

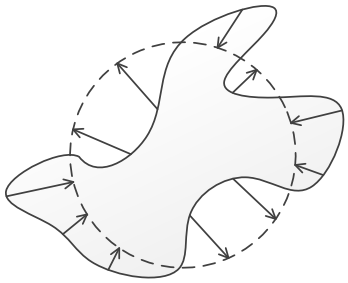
where μ_R the mean distance from the centroid of the ROI $\boldsymbol{\mu} = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}$ for all the points in the contour $\mathbf{p}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$, where $i = 0, \ldots, K-1$ and K is the number of samples around the contour.

$$\mu_R = \frac{1}{K}\sum_{i=0}^{K-1} \|\mathbf{p}_i - \boldsymbol{\mu}\| \tag{3.22}$$

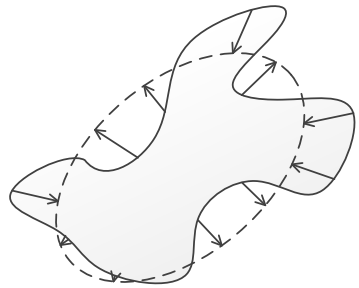
Finally σ_R^2 is the variation from the centroid of ROI.

$$\sigma_R^2 = \frac{1}{K}\sum_{i=0}^{K-1} \left(\|\mathbf{p}_i - \boldsymbol{\mu}\| - \mu_R\right)^2 \tag{3.23}$$

Measures how much the contour varies from a circle, see figure 3.11a. The function is implemented in MATLAB[5].



(a) Circularity. Measures how much the contour varies from a circle.



(b) Elliptic Variance. Measures how much the contour varies from a ellipse.

Figure 3.11: Illustration of circularity and elliptic variance.

Elliptic Variance The elliptic variance is related to circular variance or circularity, but it allows elongation [51], such that the contour can be fitted to an ellipse.

$$Evar = \frac{\sigma_{rc}^2}{\mu_{rc}^2} \tag{3.24}$$

The variable, μ_{rc} is the mean distance from the centroid of the ROI $\boldsymbol{\mu} = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}$ to all points in the boundary.

$$\mu_{rc} = \frac{1}{K} \sum_{i=0}^{K-1} \sqrt{(\mathbf{p}_i - \boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{p}_i - \boldsymbol{\mu})} \tag{3.25}$$

where $\mathbf{p}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$ for $i = 0, \ldots, K-1$ is all the points in the contour. K is the number of samples around the contour. $\mathbf{C}$ is the covariance matrix:

$$\mathbf{C} = \frac{1}{K} \sum_{i=0}^{K-1} (\mathbf{p}_i - \boldsymbol{\mu})(\mathbf{p}_i - \boldsymbol{\mu})^T \tag{3.26}$$

Finally σ_{rc}^2 is the deviation of the ellipse contour.

$$\sigma_{rc}^2 = \frac{1}{K} \sum_{i=0}^{K-1} \left(\sqrt{(\mathbf{p}_i - \boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{p}_i - \boldsymbol{\mu})} - \mu_{rc} \right)^2 \tag{3.27}$$

[5]The MATLAB script is located in:
`Matlab/Features/featureCircularVariance.m`

The elliptic variance measures how must the contour variates from an ellipse, see figure 3.11a. The function is implemented in MATLAB[6].

Elliptic Fourier Elliptic Fourier is used in [49] and [18] to approximate the contour by a number of harmonics. These harmonics are then used as features to classify the shape of an object. The contour is defined as a sorted list of point running in a clockwise direction around the boundary. The truncated Fourier approximation of a closed contour is described by the x and y coordinates.

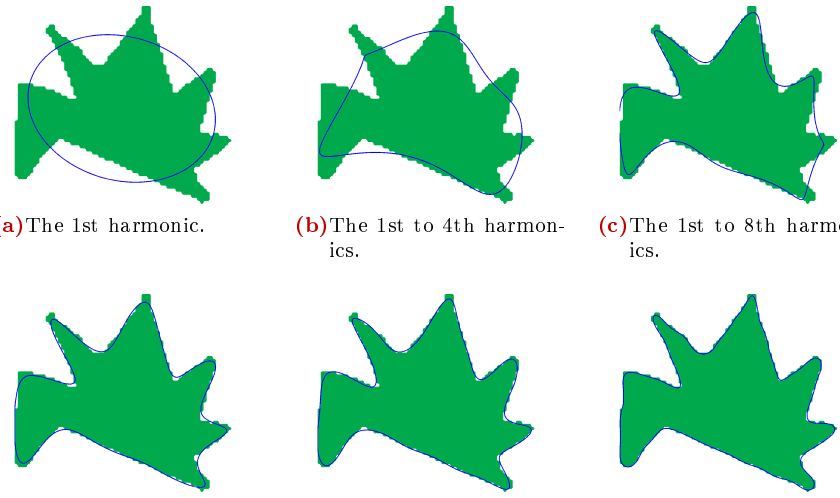
$$\begin{aligned} x_N(t) &= A_0 + \sum_{n=1}^{N} a_n \cdot \cos\left(\tfrac{2 \cdot n \cdot \pi \cdot t}{T}\right) + b_n \cdot \sin\left(\tfrac{2 \cdot n \cdot \pi \cdot t}{T}\right) \\ y_N(t) &= C_0 + \sum_{n=1}^{N} c_n \cdot \cos\left(\tfrac{2 \cdot n \cdot \pi \cdot t}{T}\right) + d_n \cdot \sin\left(\tfrac{2 \cdot n \cdot \pi \cdot t}{T}\right) \end{aligned}$$

where t defines the distance to a certain point along the contour. The value of t_p is the distance along the boundary to a pixel (in the boundary) with the index p. Traversing a full lap around the contour, will step p through integers 1 to K, where K is the total number of points in the contour. T is the distances around the whole boundary and is equivalent to $T = t_K$. The number of harmonic used in the boundary approximation is N. For each harmonic n, four coefficients are defined; a_n, b_n, c_n and d_n. The coefficients A_0 and C_0 corresponds to a frequency of 0 simply giving the x and y translation of the contour. The coefficients of a_n, b_n, c_n and d_n are defined as:

$$\begin{aligned} a_n &= \tfrac{T}{2 \cdot n^2 \cdot \pi^2} \sum_{p=1}^{K} \tfrac{\Delta x_p}{\Delta t_p} \cdot \left(\cos\left(\tfrac{2 \cdot n \cdot \pi \cdot t_p}{T}\right) - \cos\left(\tfrac{2 \cdot n \cdot \pi \cdot t_{p-1}}{T}\right)\right) \\ b_n &= \tfrac{T}{2 \cdot n^2 \cdot \pi^2} \sum_{p=1}^{K} \tfrac{\Delta x_p}{\Delta t_p} \cdot \left(\sin\left(\tfrac{2 \cdot n \cdot \pi \cdot t_p}{T}\right) - \sin\left(\tfrac{2 \cdot n \cdot \pi \cdot t_{p-1}}{T}\right)\right) \\ c_n &= \tfrac{T}{2 \cdot n^2 \cdot \pi^2} \sum_{p=1}^{K} \tfrac{\Delta y_p}{\Delta t_p} \cdot \left(\cos\left(\tfrac{2 \cdot n \cdot \pi \cdot t_p}{T}\right) - \cos\left(\tfrac{2 \cdot n \cdot \pi \cdot t_{p-1}}{T}\right)\right) \\ d_n &= \tfrac{T}{2 \cdot n^2 \cdot \pi^2} \sum_{p=1}^{K} \tfrac{\Delta y_p}{\Delta t_p} \cdot \left(\sin\left(\tfrac{2 \cdot n \cdot \pi \cdot t_p}{T}\right) - \sin\left(\tfrac{2 \cdot n \cdot \pi \cdot t_{p-1}}{T}\right)\right) \end{aligned} \tag{3.28}$$

Δx_p and Δy_p are the difference of adjacent values of respectively x and y at point p. Δt_p is the distance between two adjacent points in the contour at point p. The calculations of the coefficients above have been implemented in MATLAB[7]. The coefficients can be calculated by two for-loops running through all values of n and p. The calculations have, though, been vectorized

[6] The MATLAB script is located in:
`Matlab/Features/featureEllipticVariance.m`
[7] The MATLAB script is located in:
`Matlab/Features/ellipticFourier.m`



(a) The 1st harmonic.

(b) The 1st to 4th harmonics.

(c) The 1st to 8th harmonics.

(d) The 1st to 12th harmonics.

(e) The 1st to 16th harmonics.

(f) The 1st to 30th harmonics.

Figure 3.12: A higher number of harmonics provides a better approximation of the boundary.

and identical terms for constant value of n and p are only calculated once to improve performance. Figure 3.12 shows how a higher number of harmonics will present a better approximation of the boundary.

The elliptic Fourier coefficients will obviously depend on translation, scale and rotation of the contour, but they will also depend on the starting point of the contour. Multiple normalization steps are therefore made on the coefficients. Translation is removed by simply excluding A_0 and C_0. In Figure 3.13a and 3.13b the first 20 elliptic Fourier coefficients are shown with and without A_0 and C_0, respectively. Notice how the contour centre is placed at the origin of the plot. To make the elliptic coefficients invariant to the starting point, it is shifted until it is aligned with a semi-major axis. In [49] the starting point is shifted by doing the following transformation of the coefficients for each harmonic.

$$\begin{bmatrix} a_n^* & b_n^* \\ c_n^* & d_n^* \end{bmatrix} = \begin{bmatrix} a_n & b_n \\ c_n & d_n \end{bmatrix} \cdot \begin{bmatrix} \cos(n\theta_1) & -\sin(n\theta_1) \\ \sin(n\theta_1) & \cos(n\theta_1) \end{bmatrix} \tag{3.29}$$

where θ_1 is the angular shift between the starting point and semi-major axis of the contour.

$$\theta_1 = \frac{1}{2}\arctan\left(\frac{2(a_1 b_1 + c_1 d_1)}{{a_1}^2 + {c_1}^2 - {b_1}^2 - {d_1}^2}\right) \tag{3.30}$$

The equation might erroneously determine a shift to a semi-minor axis (instead of the semi-major axis). The atan2 function in MATLAB can be used to avoid this:

$$\theta_1 = \frac{1}{2} atan2\left(2\left(a_1 b_1 + c_1 d_1\right), {a_1}^2 + {c_1}^2 - {b_1}^2 - {d_1}^2\right) \tag{3.31}$$

To make the coefficients invariant to rotation and scaling, the contour is rotated to align the major axis of the contour to the x-axis and normalize the size of the contour in accordance to the major axis of the ellipse. In [49], rotation and normalization is made by doing a transformation and scaling of the elliptic Fourier coefficients for each harmonic.

$$\begin{bmatrix} a_n^{**} & b_n^{**} \\ c_n^{**} & d_n^{**} \end{bmatrix} = \begin{bmatrix} \cos(\psi_1) & \sin(\psi_1) \\ -\sin(\psi_1) & \cos(\psi_1) \end{bmatrix} \cdot \begin{bmatrix} a_n^{*} & b_n^{*} \\ c_n^{*} & d_n^{*} \end{bmatrix} \cdot \frac{1}{E}, \tag{3.32}$$

where ψ_1 defines the rotation of the contour and is given by

$$\psi_1 = \arctan\left(\frac{c_1^*}{a_1^*}\right) \tag{3.33}$$

the function atan2() should again be used for determining ψ_1

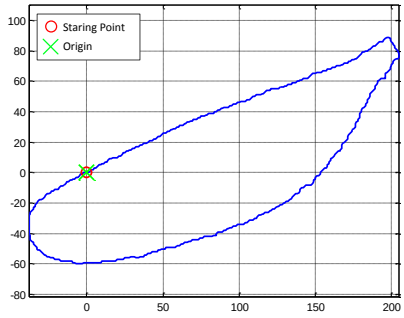
$$\psi_1 = atan2\left(c_1^*, a_1^*\right) \tag{3.34}$$

and E is the length of the major axis.

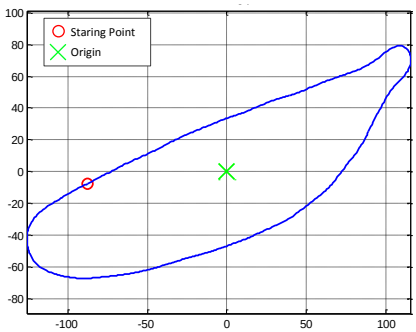
$$E = \sqrt{\left(a_1^*\right)^2 + \left(c_1^*\right)^2} \tag{3.35}$$

The result after rotation and scaling is seen in Figure 3.13d. The semi-major axis of the contour is rotated in the direction of the x-axis. Depending on the initially direction, the contour will "point" along the x-axis in either a positive or negative direction as demonstrated in Figure 3.14, where the same image e.g. has been rotated 180°. Fortunately as stated in [49] and as seen in the Table 3.1 the only difference between the coefficients is that every even set of harmonics change sign. To be fully invariant to rotation, the absolute value of every second harmonic is used. For all normalized and rotated contours the value of a_1, b_1 and c_1 is respectively 1, 0 and 0[49]. For N harmonics the number of features is therefore not $N \cdot 4$, but $N \cdot 4 - 3$, because the a_1, b_1 and c_1 coefficients are constant. The first 20 harmonics are used providing $20 \cdot 4 - 3 = 77$ features. To keep the features rotation invariant, all odd coefficients and the absolute values of the even coefficient are used.

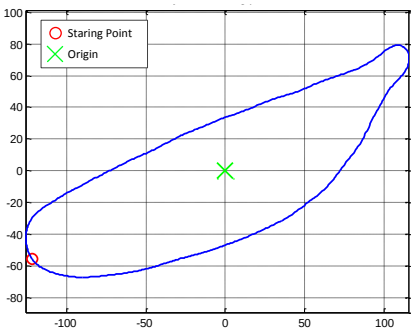
Total variation of Elliptic Fourier In [18] the approximated contours are used as a measure of the serrations of a plant contour. The measure



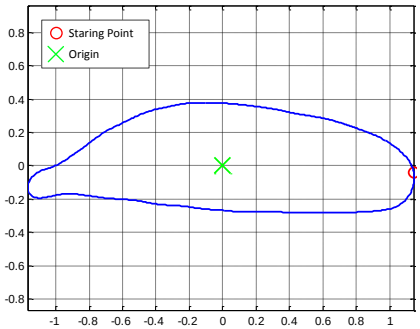
(a)Original contour.



(b)First 20 harmonics without A_0 and C_0.

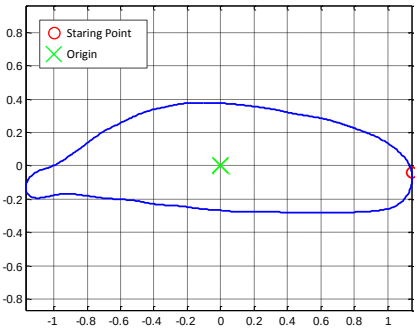


(c)Shifting of starting point.

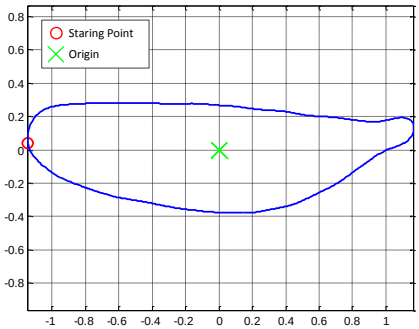


(d)Rotation and scaling of contour.

Figure 3.13: Shows the different normalization steps.



(a)Rotated and scaled contour.



(b)Rotated and scaled contour with a 180° rotation.

Figure 3.14: The elliptic coefficients are not invariant to a 180° rotation.

Harmc.	EF original image a	b	c	d	EF 180° rotated image a	b	c	d
1	1.000	0.000	0.000	-0.343	1.000	0.000	0.000	-0.343
2	0.007	0.015	-0.065	-0.033	-0.007	-0.015	0.065	0.033
3	0.096	-0.006	0.019	-0.032	0.096	-0.006	0.019	-0.032
4	-0.001	0.014	-0.011	-0.020	0.001	-0.014	0.011	0.020
5	0.031	-0.005	0.003	-0.017	0.031	-0.005	0.003	-0.017
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
19	0.000	-0.001	-0.001	0.000	0.000	-0.001	-0.001	0.000
20	0.000	0.000	0.001	0.000	0.000	0.000	-0.001	0.000

Table 3.1: The normalized coefficients of two identical images rotated by 180°. Coefficients have of same absolute size, but even harmonics change in sign. Even harmonics are marked by red.

is called the total variation of the elliptic Fourier boundary approximation, TVH.

$$TVH = \sum_{n=2}^{N} \max\left(\sqrt{(\mathbf{x}_{n-1} - \mathbf{x}_n)^2 + (\mathbf{y}_{n-1} - \mathbf{y}_n)^2}\right) \tag{3.36}$$

where N is the number of harmonics, $\mathbf{x}_n$ and $\mathbf{y}_n$ are the $\mathbf{x}$ and $\mathbf{y}$ coordinates for the approximated boundary when using n harmonics. The expression determines the maximum distance between all points in the boundary for the n and $n-1$ approximation of the contour. The maximum distance between two approximations of the contour n and $n-1$ is then summed for $n = 2, \ldots, N$. A slightly different measure has been proposed, where the mean distance of all points between the two approximations of the boundary of n and $n-1$ are calculated. The mean distance between the two approximations of n and $n-1$ is then summed for $n = 2, \ldots, N$.

$$TVH2 = \sum_{n=2}^{N} mean\left(\sqrt{(\mathbf{x}_{n-1} - \mathbf{x}_n)^2 + (\mathbf{y}_{n-1} - \mathbf{y}_n)^2}\right) \tag{3.37}$$

The second measure do not only use the largest distance between k and $k-1$ approximations, but includes all distances between two approximations by calculating the average distance.

Variance of elliptic Fourier using 5 and 30 harmonics We also propose a measure for serrations. Two approximations of the boundary are determined with elliptic Fourier by using the 5 and 30 harmonics to make respectively a coarse and a fine approximation of the contour. A leaf with a high level of serration would achieve two different approximations of the

contour, while a smooth leaf would achieve more similar contours. All the distances between the two approximated contours are determined.

$$\mathbf{d}_{EF5\_30} = \sqrt{\left(\mathbf{x}_{EF5} - \mathbf{x}_{EF30}\right)^2 + \left(\mathbf{y}_{EF5} - \mathbf{y}_{EF30}\right)^2} \tag{3.38}$$

The average distance and the variance distance are then determined.

$$\mu_{EF5\_30} = \frac{1}{K}\sum_{k=1}^{K} d_{EF5\_30,k} = mean\left(\mathbf{d}_{EF5\_30}\right) \tag{3.39}$$

$$\sigma^2_{EF5\_30} = \frac{1}{K}\sum_{k=1}^{K}\left(d_{EF5\_30,k} - \mu_{EF5\_30}\right) = var\left(\mathbf{d}_{EF5\_30}\right) \tag{3.40}$$

The average distance μ_{EF5_30} is used as a measure of scale, and the variance is normalized by dividing by μ_{EF5_30} to make it invariant to scale (as seen with the features circularity and elliptic variance).

$$\sigma^2_{ET} = \frac{\sigma^2_{EF5\_30}}{\mu^2_{EF5\_30}} \tag{3.41}$$

Distance between elliptic Fourier approximations We propose another category of features based on normalized elliptic Fourier. The distances between all points $(\mathbf{x}, \mathbf{y})$ on the boundary for the n and $n-1$ approximation of the contour are determined for $n = 2, \ldots, N$. The number of features can be reduced, by only determining the distance between the approximations for $n = 2, 3, 5, 7, 10, 14, 18, 22, 26$ harmonics.

$$\mathbf{d}_n = \sqrt{\left(\mathbf{x}_{n-1} - \mathbf{x}_n\right)^2 + \left(\mathbf{y}_{n-1} - \mathbf{y}_n\right)^2} \tag{3.42}$$

The variable $\mathbf{d}_n$ contains all distances between the boundaries for the n and $n-1$ approximation. Statistical data is then determined for $n = 2, \ldots, N$ providing $N-1$ features. The mean distance between two subsequent boundary approximations is given by:

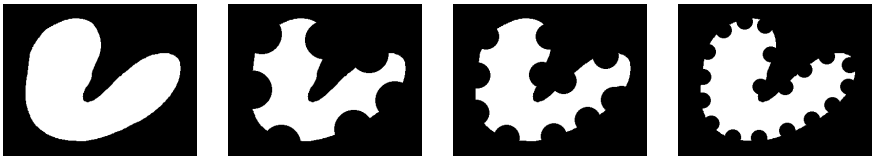
$$EFdistMean(n) = mean\left(\mathbf{d}_n\right), \tag{3.43}$$

and the accumulated mean distance between two subsequent boundary approximations.

$$EFdistAccMean(n) = \sum_{k=1}^{n} mean\left(\mathbf{d}_k\right) \tag{3.44}$$

A normalization is performed of the variance between two subsequent boundary approximations by dividing with the mean distance:

$$EFdistVarN(n) = \frac{var\left(\mathbf{d}_n\right)}{mean\left(\mathbf{d}_n\right)} \tag{3.45}$$



(a) Imaginary plant mask. **(b)** Plant mask with huge serrations. **(c)** Plant mask with medium serrations. **(d)** Plant mask with small serrations.

Figure 3.15: Shows an imaginary plant mask with different sized circular serraions.

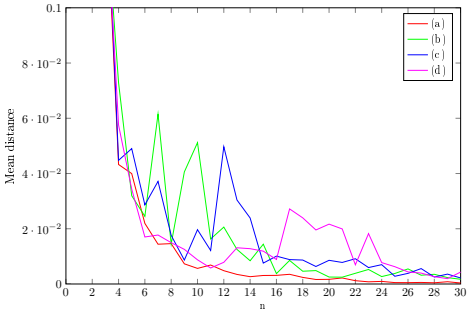
Statistics of the distance between two subsequent boundary approximations is expected to provide valuable information about the serrations or indentations of the leaf. The different statistical values do not only provide a single measure of serrations, but provides (presumably) multiple measures for serrations of different sizes. The assumption is that as the number of harmonics is increased, the boundary approximations will approach the actual boundary with a greater accuracy. Significant shape characteristics are initially fitted by a low number of harmonics, while smaller bumps and finally small serrations in the contour are fitted with a higher number of harmonics.

An illustration is provided with an imaginary plant mask with different (circular) serrations, see Figure 3.15. Figure 3.16 shows the mean distance, the scaled mean distance, the accumulated mean distance and the normalized distance variance. The plots are zoomed around the lower values to show the difference between the different plots.

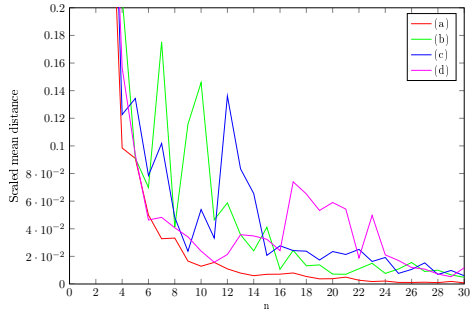
Notice how the mean distance, the scaled mean distance and the normalized variance respectively in Figure 3.16a, 3.16b and 3.16d shows no sudden peaks for the first image, (a), two peaks in the start for the second image, (b), one peak in the middle for the third image, (c) and finally a peak at the end for the fourth image, (d). The accumulated and scaled mean distance have also been included in the feature set. As to reduce the number of features included in the feature set, instead of increasing the order by one for each feature, the order is increased in steps of three. This means that the first feature will be the difference between order 1 and 4, the second will be the difference between order 4 and 7 and so on.

Skeleton Skeleton features are features extracted from the skeleton of the shapes. The skeleton can be defined as the set of centres of the largest discs that can be contained within the contour and touching the boundary of the contour at least two places[28].
The skeleton of a scentless mayweed can be seen in Figure 3.17.

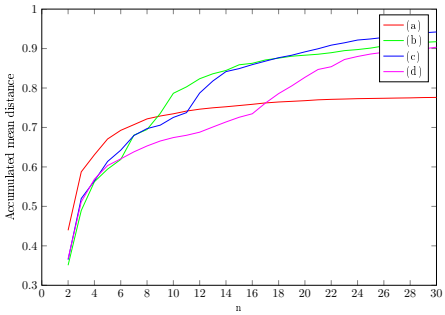
From the skeleton, a skeleton distance vector,d_{skel}, is extracted, which is a vector consisting of the minimum Euclidean distance of all skeleton



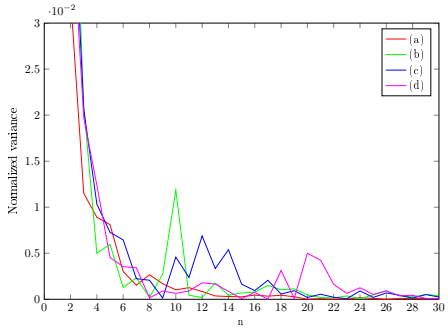
(a) Mean distance between two subsequent boundary approximations for $n = 2, \ldots, 30$.



(b) Scaled mean distance between two subsequent boundary approximations for $n = 2, \ldots, 30$.



(c) Accumulated mean distance between two subsequent boundary approximations for $n = 2, \ldots, 30$.



(d) The normalized variance between two subsequent boundary approximations for $n = 2, \ldots, 30$.

Figure 3.16: The distance statistics between two subsequent boundary approximations for $n = 2, \ldots, 30$.

Figure 3.17: Skeleton of scentless mayweed

elements to the boundary of the leaf. From this vector four features are extracted:

1. *SkeletonDistanceMax* is the maximum value of d_{skel}
2. *SkeletonDistanceMean* is the mean value of d_{skel}
3. *SkeletonDistanceLength* is the length of d_{skel}. I.e. the number of elements in d_{skel}.
4. *SkeletonDistanceVariance* is the variance of d_{skel}.

3.2.4 Colour Features

Here a selection of colour based features, using the RGB image, will be described.

Chromaticity of red, green and blue *chromaticity* is a combination of hue and saturation [28, p. 399]. The average *chromaticity* for red, green and blue is

$$\begin{aligned} avg_r &= \frac{1}{N} \sum_{Region} \frac{R}{R+G+B} \\ avg_g &= \frac{1}{N} \sum_{Region} \frac{G}{R+G+B} \\ avg_b &= \frac{1}{N} \sum_{Region} \frac{B}{R+G+B} \end{aligned} \tag{3.46}$$

The chromaticity is calculated for all pixels within the plant region *Region*. The number of pixels within the plant region is N.

Average excessive Green The average excessive green is used as a measure of green.

$$avg_{ExG} = \frac{1}{N} \sum_{Region} 2G - R - B \tag{3.47}$$

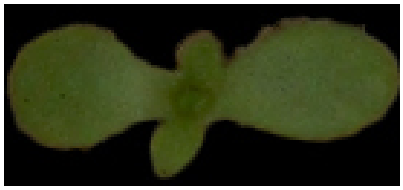
The average excessive green is calculated for all pixels within the plant *Region*, where the number of pixels within the plant region is N.

Variance of RGB The variance is a measure of how flat the colour appears. For instance will plants with dark stems like Cleavers typically have larger variation than Shepherd's-purse, where the colour of the stem is closer to the average colour of the leaves.

$$\begin{aligned} Var(R) &= E\left[(R-\mu_R)^2\right] \\ Var(G) &= E\left[(G-\mu_G)^2\right] \\ Var(B) &= E\left[(B-\mu_B)^2\right] \end{aligned} \tag{3.48}$$

3.2.5 Other Features

Stem thickness estimation After investigating images, it has been shown that shepherd's-purse and scentless mayweed have a lot of common appearances as can be seen in Figure 3.18. Therefore it will be desirable to find features that are able to find differences between these two species. One such difference, when looking at the images, is that the stem generally appears to be thinner on shepherd's-purse than on scentless mayweed. The stem thickness estimation feature, therefore, tries to estimate this thickness. The design of this feature is based on dicotyledonous plants, and the quality of the feature might change for other inputs, e.g. leaves, for which the feature is not expected to provide much information.



(a) Scentless mayweed.

(b) Shepherd's-purse

Figure 3.18: Scentless mayweed and shepherd's-purse.

To find the smallest thickness of a plant, the plant is being eroded by a single pixel, until it splits into multiple parts. The plant will split in two at the thinnest point and this point will often be found at the stem. By counting the number of iterations before this happens, i.e. the number of pixels to be removed before the plant breaks in two, we have a measurement for the size of half the stem's thickness in pixels. The thickness of the stem is then estimated as twice the number of iteration.
To bring the feature closer to being scale invariant, the feature is hereafter scaled to the square-root of the area of the plant before it was eroded.

3.3. Feature Discussion and Conclusion

The project has implemented and documented a broad range of 50 different feature descriptors providing in total 261 subfeatures. These features are selected so that they cover both shape, contour and colour features. An overview of all the features is provided in Appendix F.
The 261 features have been calculated for the 2438 plants, the 3409 cotyledons and the 1358 foliages so that they are ready for classification. Some features have originally been targeted for plants while others are targeted

for leaves. However, all features have been calculated for both plants and leaves.

The project has also proposed new features such as the stem thickness estimation and the Elliptic Fourier distance features. Variations of existing features have also been proposed, e.g. the distance transform, which has become scale invariant while keeping important information. An evaluation of the different features is provided in Chapter 5.

Of these features, no texture-based features have been implemented. this is due to the fact that a texture base is close to non-existing for the seedlings in the provided database. Vein structure and texture also make high demands on low image blurring and high resolution images complicating an in-field implementation.

An alternative to using features to characterize the plants is to use active shape models, which might provide interesting results, but such models have not been included due to the time constraint of this project.

Chapter 4

Classifier

The objective of a classifier is to take an unknown object described by a set of features and match it to a certain category/class. The features therefore have to provide enough information describing the object to make the match correct. To get information about the classes, the classifier is initially trained by a set of objects belonging to the different classes. A range of classifiers have been described in literature, but the following three have been selected in this project; k-Nearest Neighbours (kNN), Support Vector Machine (SVM) and Multivariate Gaussian (MVG).

The k-nearest neighbour classifier is a simple algorithm which is able to handle distributed feature clusters for the individual species. It is therefore useful, as the leaf and plant appearances change over time and therefore are expected to form multiple data clusters. The non-linear support vector machine is a popular classifier, which is useful in this project because of its ability to handle high dimensional feature sets and overall provides good performance for many classification problems [64]. The third classifier is the Gaussian multivariate classifier, in which the measurements are compared to Gaussian distributions, archived from training. The benefit of the Gaussian multivariate classifier is that it provides a likelihood values for the samples, which tells how much they fit in the different distribution. This likelihood value can then be used as a measure for how certain a given classification is. Likewise, the k-nearest neighbour classifier can provide a measure of the classification certainty from the distribution of samples nearby the test sample.
Based on the 261 subfeatures determined for plants and leaves in Chapter 3, the classifiers will be trained and afterwards do a classification of the plant elements.

Labelling As described in Section 2 The plants are divided into three plant elements that can be classified individually; the whole plant, single cotyledons and single foliages. To distinguish between plant elements and species a particular labelling is used. This is shown in Table 4.1. Whole plants (0) are for the seven species labelled 1-7, the single cotyledon leaves (100) are labelled 101-107 and the single foliage leaves (200) are labelled 201-207.

Species	Maize	Wheat	Sugar beet	Scentless mayweed	Chickweed	Shepherd's-purse	Cleavers
Plant	1	2	3	4	5	6	7
Cotyledon	101	102	103	104	105	106	107
Foliage	201	202	203	204	205	206	207

Table 4.1: Labeling plant, cotyledon and foliage leaves for all species. Each plant elements is also specified with a unique label 0, 100 and 200.

Confusion Matrix and classification accuracies A confusion matrix shows the performance of a classifier and is determined by comparing the result of the classification with the true label of each elements. In the following section an example is provided to show how different classification accuracies are determined based on the confusion matrix. An example of a confusion matrix and the accuracy matrix is presented for plants in Table 4.2.

		True label						
	Spe.	1	2	3	4	5	6	7
Result label	1	247	3	8	0	1	0	0
	2	1	185	15	5	4	2	1
	3	1	4	300	0	1	0	0
	4	1	2	0	551	11	10	2
	5	0	3	0	0	666	3	1
	6	0	0	0	12	18	238	1
	7	0	1	2	7	2	2	125

		True label						
	Spe.	1	2	3	4	5	6	7
Result label	1	0.988	0.015	0.025	0	0.001	0	0
	2	0.004	0.934	0.046	0.009	0.006	0.008	0.008
	3	0.004	0.020	0.923	0	0.001	0	0
	4	0.004	0.010	0	0.958	0.016	0.039	0.015
	5	0	0.015	0	0	0.947	0.012	0.008
	6	0	0	0	0.021	0.026	0.933	0.008
	7	0	0.005	0.006	0.012	0.003	0.008	0.962

Table 4.2: An example of a confusion and accuracy matrix.

The confusion matrix shows how often a given class is classified correctly and how often a class is incorrectly classified as another class. An entry in the confusion matrix is defined as $n_{ij}^{(k)}$, where j is the number of times that a true class have been classified as class i for a classifier k. An entry in the accuracy matrix is determining by the conditional probability of the classifier $e_k(\mathbf{x})$ outputting i, given that the sample $\mathbf{x}$ is of class j.

$$P\left(e_k\left(\mathbf{x}\right) = i_k | \mathbf{x} \in c_j\right) = \frac{{n_{ij}}^{(k)}}{\sum_{m=1}^{M} n_{mj}} \tag{4.1}$$

where M is the number of classes. The equation simply divides an index, $n_{ij}^{(k)}$, in the confusion matrix by the sum of all the entries in column j.

In the example provided, it is seen that class 3 in Table 4.2 is classified correctly 300 times (93.4%) and incorrectly 25 times; eight times (2.5%) as species 1, 15 times (4.6%) as species 2 and two times (0.6%) as species 7. The diagonal of the accuracy matrix shows the individual classification accuracy of each species. The classification accuracy of all species for the example are collected in Table 4.3. The classification accuracy of each plant element is

	Species							
Plant Elements	**1**	**2**	**3**	**4**	**5**	**6**	**7**	
Plant (0)	0.988	0.934	0.923	0.958	0.947	0.933	0.962	0.949
Cotyledon (100)	0.923	0	0.910	0.552	0.842	0.697	0.825	0.810
Foliage (200)	0	0	0.957	0.738	0.822	0.478	0.775	0.705
							Total	0.837

Table 4.3: The total classification accuracy and the classification accuracies of plant elements and species.

shown in the last column. The classification accuracy of a plant element is the sum of the diagonal divided by the sum of the confusion matrix. The total classification accuracy of 83.7% for all plant elements is the sum of the three diagonals divided by the sum of the three confusion matrices. The total classification accuracy is often used to provide a simple measure to describe the performance of the classifier or a set of features.

4.1. Feature scaling and feature struct

Feature Scaling The range of the feature values differs between the features, e.g. the object area of the plant samples ranges between 503 and 350980 and the solidity ranges between 0.1350 and 0.9839. Not all classifiers are invariant towards features having different ranges, and the feature values have therefore been shifted and scaled to fit in the range between 0 and 1 to optimize the performance of all classifiers. The kNN classifiers is e.g. dependent on the range of features as a plant is identified using Eucledean distance in the feature space. Features that spread samples over a large area (low density) will be less "weighted" in comparison to features that distribute samples over a small area (high density).

The MVG classifier handles features of any range as it fits a given dimension/feature with a mean value and a variance.

Feature Struct In the segmentation phase, all plants are stored in individual plant structs, but as a classifier may be trained by features of all samples this data structure is not very efficient. All plant structs are therefore rearranged into feature structs containing the feature values of all samples. One feature matrix of the size $F \times N$ is made of each of the three plant element, where F is the number of features and N is the number of samples in the given plant element. As the final identification of a plant is determined on the classification of the plant and its leaves, the struct also contains information on how the data is mapped between the two data structures.

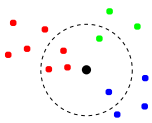
4.2. Classifiers

In this section, the three selected classifiers will be described. These are the k-nearest neighbour classifier, the non-linear support vector machine and multivariate Gaussian classifier.

4.2.1 k-Nearest Neighbours (kNN)

The k-nearest neighbours classifier is a simple classifier, in which an unknown sample is classified according to the majority of the k-nearest neighbours from the training set. The distances are calculated as Euclidean distances, for which reason it is necessary to scale the features to the same scale. In addition, the number of samples for each class in the training-set must be equal to make the classification fair for all classes.
Beside the simplicity of the classifier, another advantage is that it is independent of the distribution of the classes. This means that if e.g. cotyledons and foliages of the same species are to be classified at the same time, they could be identified even though they form two separated clusters in feature space.

Besides the classification result, the kNN-classificer is able to give a likelihood measure of how certain a given classification is. This likelihood is how many samples of class i that are present within the nearest j samples. However, for small j, this measure will be imprecise because of a sparse data basis. Likewise, for large j, the measure will be imprecise as the kNN classifier does not take the non-spherical distributions of the clusters of training samples into consideration, when using the Euclidean distance measure. Therefore, j should be chosen as a compromise of these extrema.



Red is selected for a 5 nearest neighbour classifier

The number of k neighbours have impact on the selected class, as changing k might change the majority, depending on the distribution of the training set. Therefore it is desirable to find the optimal k for this dataset.

Table 4.4 shows the classification accuracy for plants, cotyledons and foliages using all features of the given element. From it, it is seen that the optimal k depends on whether whole plants or leaves are to be classified, where plants gets the highest classification accuracy for $k = 2 - 4$, cotyledons gets the highest classification accuracy for $k = 2$ and foliages gets the highest classification accuracy for $k = 2$. These values are found by using all features, so after the features reduction presented in Chapter 5, other k's might be better. However, as it can be seen, increasing k does not change the classification accuracy dramatically, for which reason it is recommended to use a k-value of at least 3 to decrease the effect of outliers.

k	CA plant	CA cotyledon	CA foliage
1	0.933	0.875	0.849
2	0.932	0.868	0.852
3	0.933	0.864	0.846
4	0.940	0.872	0.852
5	0.935	0.868	0.842
6	0.934	0.868	0.828
7	0.928	0.866	0.832
8	0.931	0.865	0.839
9	0.925	0.856	0.815
10	0.924	0.859	0.801

Table 4.4: Classification accuracy (CA) of k-nearest-neighbours using k=1 to 10 for whole plants, cotyledons and foliages using all features. The best classification accuracies are marked with red

4.2.2 Support Vector Machine (SVM)

In this section the Support Vector Machine-classifier (SVM) will be described. For a dataset consisting of two classes, there may be several hyperplanes, which classifies the datasets evenly good. The rationale behind SVM is to find the single hyperplane $g(\mathbf{x})$ that separates the two clusters for the classes with the largest margin to the data. The hyperplane is given by[65]:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 \tag{4.2}$$

where $\mathbf{x}$ is the feature data points labelled to belong to one of the two classes ω_1 or ω_2. $\mathbf{w}$ is the direction and w_0 is the offset of the plane.

The support vectors are the points with the minimum margin to the hyperplane[65], where the margin from the data to the hyperplane is defined as twice the smallest distance from a single point in the dataset to the

hyperplane as illustrated in Figure 4.1. These distances are given by:

$$z = \frac{|g(\mathbf{x})|}{\|\mathbf{w}\|} \tag{4.3}$$

To make the plane scale-independent, the plane will be scaled, such that the margin for ω_1 is 1 and thus the margin for ω_2 is -1. With this scale, all feature-points on the ω_1-side of the plane will have a distance larger or equal to 1, where the feature-points on the ω_2-side of the plane will have a distance smaller or equal to -1.
The aim now is to maximize this margin with respect to $\mathbf{w}$ and w_0 by minimizing the norm of $\mathbf{w}$. Though to avoid calculating the squareroot, $\|\mathbf{w}\|$ is replaced by $\frac{1}{2}\|\mathbf{w}\|^2$, resulting in Eq.4.4

$$\begin{array}{ll} \text{minimize:} & J(\mathbf{w}, w_0) = \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to:} & y_i\left(\mathbf{w}^T\mathbf{x}_i + w_0\right) \geqslant 1 \quad , \quad i = 1, 2, \ldots, N \end{array} \tag{4.4}$$

where y_i is the label of x_i which is either 1 for ω_1 or -1 for ω_2.
Until now it is supposed that the two classes are separable, which often cannot be assumed. Therefore a slack variable ξ_i is related to each datapoint x_i to allow overlapping datasets. The cost function is then defined as:

$$\begin{array}{ll} \text{minimize:} & J(\mathbf{w}, w_0, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} I(\xi_i) \\ \text{subject to:} & y_i\left(\mathbf{w}^T\mathbf{x}_i + w_0\right) \geqslant 1 - \xi_i \quad , \quad i = 1, 2, \ldots, N \wedge \xi_i > 0 \end{array} \tag{4.5}$$

where

$$I(\xi_i) = \begin{cases} 1 & \xi_i > 0 \\ 0 & \xi_i = 0 \end{cases} \tag{4.6}$$

The method described until now is an ideal linear classifier. The advantage of the Support vector machine is the ability to use a non-linear kernel to map the input features $\mathbf{x}$ in space $\mathbb{R}^i$ to a higher dimensional space $\mathbb{R}^k$ in which the data can be separated using a linear classifier as illustrated in Figure 4.1.

The mapping should be done in a way such that the classes can be separated satisfactory. Several kernels exists, but a good choice of mapping is the Gaussian radial basis function (RBF) [67, 65].

$$K(\mathbf{x}, \mathbf{z}) = e^{\left(-\gamma\|\mathbf{x}-\mathbf{z}\|^2\right)} \tag{4.7}$$

Multiclass case The SVM classifier is only capable of classifying two classes, which off course is not useful when classifying multiple plant species. Therefore the classifier should be extended to handle the multi-class case. There are two methods for doing this [65, p. 217]; the one-against-one and

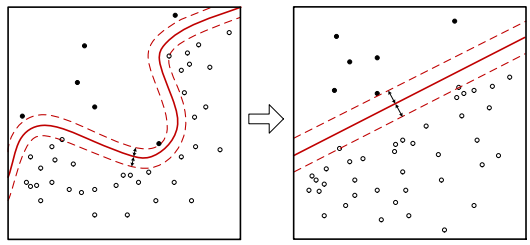


Figure 4.1: Unwrapping of data to make it separable using a linear classifier. (after [66])

the one-against-all method. In the one-against-one method a kernel is made for each pair of classes. This means that for C classes, there will be $\frac{2}{C!(2-C)!}$ SVM kernels. The input will then be classified using all these two-class classifiers for which there will be a lot of estimated classes. Among all these estimated classes the class that has been chosen the most will be chosen as the final class. In case multiple classes are chose equally many times, a new round start, but just with the remaining classes.

Another method for a multi-class SVM classifier is the one-against-all method, in which the kernels of K classifiers must be determined for C classes. For each class c a two-class problem is set up: Class c and the others. The test sample is then tested in the k classifiers which hopefully classifies the sample as "other" in all cases but one, and the sample is classified as that class.
In the case that multiple classes are selected, there are no best solution as the distance from the test sample to the separating hyperplane is only true and fair in the two-class example, because the placement of the hyperplane in the one-against-all method also depends on all the false classes.
According to [68] there are no significant deviation for any of the two methods, when used for optical character recognition. Therefore the one-against-one method is preferred, as it almost always finds a unique result.

Finding optimal parameters The optimal parameters for the RBF kernel classifier can be found using grid-search for γ and C. An evaluation of each point in the grid is, depending on the resolution of the grid, a computationally expensive procedure taking up hours in MATLAB , but should only be determined once for the given feature set[67].
Because the SVM classifier is only build for two classes, the parameters would preferably be calculated for each og the one-against-one classification, but for convenience, the parameters with the overall best performance will be found and used in all cases.
It should be noticed that the SVM classifier that comes with MATLAB have

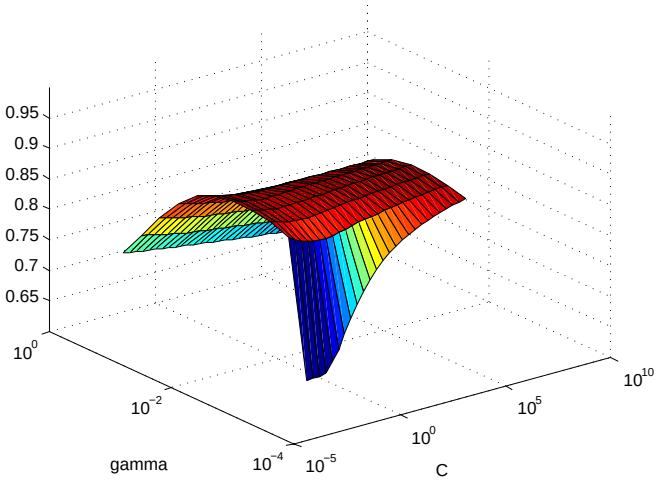


Figure 4.2: Grid search for optimal parameters for plants.

a slightly different notation for the RBF kernel than described above, as it instead of γ takes an input σ given by

$$\sigma = \sqrt{\frac{1}{2\gamma}} \tag{4.8}$$

One task of this project is to find the optimum feature subset for classification. However, this would require the parameters to be calculated for each feature combination, whereby the non-linear SVM classifier might not be the best choice. Nevertheless, for the purpose of demonstration, the parameters have been calculated for the full feature set of whole plants, which is shown in Figure 4.2. This leads to the following parameters: $\gamma = 0.0039, C = \infty$. However, these parameters should be re-calculated when the optimal feature subset has been found in Chapter 5.

4.2.3 Multivariate Gaussian (MVG)

The Multivariate Gaussian classifier (MVG) assumes that the densities of the samples are distributed in multivariate Gaussian distributions and is the generalized case of a univariate Gaussian distribution for $d > 1$[65]. In the training of MVG each class is matched to a multivariate normal distribution by determining the mean value $\boldsymbol{\mu}$ and a covariance matrix $\sum$. For class j containing N samples $\mathbf{x}$ of d dimension/features, the mean value is obtained by

$$\boldsymbol{\mu}_j = \frac{1}{N} \sum_{k=1}^{N} \mathbf{x}_k \tag{4.9}$$

The covariance matrix $\sum_j$ of class j with a size $d \times d$ is obtained by:

$$\sum\nolimits_j = \frac{1}{N} \sum_{k=1}^{N} \left(\mathbf{x}_k - \boldsymbol{\mu}_j\right) \left(\mathbf{x}_k - \boldsymbol{\mu}_j\right)^T \tag{4.10}$$

The conditional probability of being class ω_j given a test sample $\mathbf{x}_{new}$ is determined by Bayes rule.

$$P\left(\omega_j | \mathbf{x}_{new}\right) = \frac{P\left(\omega_j\right) P\left(\mathbf{x}_{new} | \omega_j\right)}{P\left(\mathbf{x}_{new}\right)}$$

where $P(\omega_j)$ is the prior probability, $P(x_{new})$ is the evidence and $P\left(\mathbf{x}_{new}|\omega_j\right)$ is the likelihood. For M number of classes the sample $\mathbf{x}_{new}$ is classified as the class with the highest probability.

$$c\left(\mathbf{x}_{new}\right) = \underset{1 \leq j \leq M}{\arg\max} \left(P\left(\omega_j | \mathbf{x}_{new}\right)\right) \tag{4.11}$$

As the evidence is independent on the class ω_j, it becomes a constant value. Assuming that the same amount of samples are presented in each class the prior also becomes a constant for all classes. As a sample is elected as the highest value and not the size, the likelihood can simply be used. The likelihood is defined as the conditional probability for a test sample $\mathbf{x}_{new}$ given the multivariate classifier j is

$$P\left(\mathbf{x}_{new} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right) = \frac{1}{\sqrt{\left(2\pi\right)^k \left|\boldsymbol{\Sigma}_j\right|}} \exp\left(-\frac{1}{2}\left(\mathbf{x}_{new} - \boldsymbol{\mu}_j\right)^T \boldsymbol{\Sigma}_j^{-1} \left(\mathbf{x}_{new} - \boldsymbol{\mu}_j\right)\right) \tag{4.12}$$

The advantage of the MVG is that the likelihood for a given class j can be used in the classifier fusion.

$$likelihood(j) = P\left(\mathbf{x}_{new} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right) \tag{4.13}$$

A drawback of the MVG classifier is that it is dependent on the dimensionality of the features space or formulated differently highly influenced by the *curse of dimensionality*. Increasing the number of dimensions/features will increase the volume of the features space exponentially and therefore decrease the density of the data - making it statistically insignificant.

4.3. Cross-Validation

The total number of samples are divided in a test and a training set to separate samples used in the training of the classifier and samples used in the

validation of the classifier. The reason for this is that a classifier might overfit the samples of the training set, providing little space for variation in the new samples and consequently fitting them into the wrong class. Meaning that an overfitted classifier might have a classification accuracy of a 100%, if the classifier is also validated with the training set, but failing to classify new samples from the test set. Dividing test and training set in portion of e.g. respectively 10% and 90% of the data set, would solve this problem, but also reduce the number of samples validating the classifier remarkably. The validation set can though be increased by using cross-validation. The cross-validation procedure divides the data in k equally sized portions. A training and a validation is then performed k individual times and in turn choosing one of the k portions as the test and the remaining portions as the training set. The k-fold validation will reduce the variance by averaging over the k different training and classifications, which thereby increases the reliability and reduces over-fitting. The result will therefore have a lower variance than a single hold-out set estimator. The size of the test set is typically set to 10% of the size of the total dataset size, as it has proven good result in many applications [69, p. 484]. To make sure that all data appears in a train-set once, k should in this case be set to 10.

To make the datasets as independent as possible, the test- and training sets are created such that leaves from the same plant only appear in either the training set or the test set, but not mixed. The reason for doing this is that leaves from the same plant have the same growth conditions, and therefore might be close to identical compared to other leaves. Therefore there is a probability of over-fitting the classifier because of this correlation, which of course should be avoided.
However, there is still a problem of providing total independence of the different plants in the dataset as the same plant is recorded in different growth stages. Selecting a plant in the test set, will not remove the same plant in different growth stages from the training set. Removing this dependency across growth stages is though not easily achieved. Even though plants are fixed in the seeding position, the trays are photographed with different translation and rotations and some plants are removed due to pruning.

4.4. Result of classification

The performance of the different classifiers are simply determined by the classification accuracy using all features as shown in Table 4.5. The MVG is useless at this stage as it performs worse than a classifier doing a random identification of species! The result is though notable as it demonstrate how the MVG classifier is affected by a feature space of high dimensions.

MVG		kNN		SVM	
Plant Element	**CA**	**Plant Element**	**CA**	**Plant Element**	**CA**
Plant	0.316	Plant	0.939	Plant	0.949
Cotyledons	0.063	Cotyledons	0.866	Cotyledons	0.745
Foliage	0.044	Foliage	0.840	Foliage	0.741
Total	**0.145**	**Total**	**0.886**	**Total**	**0.813**

Table 4.5: Classification accuracy (CA) of the three classifiers using all features. The total CA is weighted by the number of samples of plants, cotyledons and foliages

The kNN classifier performs overall the best with a 7.3 percentage point lead compared to SVM when looking at the total classification accuracy. However, SVM is slightly better at classifying plants. The classification results above are not directly related to the performance of the classifiers in the final system or in general as this depends on the number of selected features and the distribution of samples in these features. The objective of presenting these results - before feature selection and reduction - is that the classifiers should not be used naïvely.

4.5. Classifier Discussion and Conclusion

The different classifiers have been documented and tested on the whole feature set, showing the temporary results at this stage. The MVG classifier is intended for single clustered Gaussian distributions, and can be used directly on the feature set without optimizing any parameters. The MVG performs badly, but the justification of using MVG is that it returns a measure of likelihood for each classification and not just for the resulting class. The kNN classifier has the ability of handling a class consisting of multiple clusters of different distributions (i.e., not only Multivariate Gaussian distributions). On top of this, it is robust to a high dimensional features space achieving the highest classification accuracy for all features. The kNN classifier must be optimized in order to achieve the optimal k value for the three plant elements. A disadvantage of the classifier is that it does not provide a measure of likelihood. Conversely, the advantage of SVM is that is uses a non-linear kernel to map features into a higher dimensional space, providing the ability of separating classes more optimally by a linear classifier. A disadvantage of SVM is that a brute force optimization method is needed to find the SVM kernel for each plant element making the usage of SVM very time consuming. An actual evaluation of the three classifiers can not be made at the given stage. Such an evaluation would depend on the

characteristics of the final subset of features, which is determined through a feature selection method or some form of dimension reduction method. The subject of feature selection and dimension reduction is described in the following section, Chapter 5,

Chapter 5

Feature Selection and Dimension Reduction

In this chapter methods for feature selection and feature extraction are described. Feature selection is the process of reducing the number of features to an optimal subset, where feature extraction is the process of transforming existing features into a lower dimensional space. The reason for doing this is that some classifiers are sensitive to the number of features compared to the size of the dataset, for which reason it is desirable to decrease the number of features to avoid overfitting.
This phenomenon is described in literature as the *curse of dimensionality* [65, p. 55]. Expressing that an increase of dimensions/features will increase the volume of the features space lowering the density of the data or providing a sparse presentation of the data, leading some classifier to overfit the data.

Some features might be correlated or simply add noise in the features space and therefore not provide extra information to the discrimination process. Features can therefore by left out without losing information or to improve discrimination. In addition to this, decreasing the feature set will decrease the required computational power for a real-time implementation. However, timing will not be considered in this study, as the level of optimization for the individual features makes the computation times incomparable. Methods for finding the optimal subset have to be investigated as the number of possible combinations for subsets of size k for n features is given by $\frac{n!}{k!(n-k)} = \binom{n}{k}$. As the optimal feature subset may be of any size between 1 and n the total number of possible feature subset are $C(n) = \sum_{k=1}^{n} \binom{n}{k}$. For a set of 261 features the total number of possible feature

sets are $C(261) = 3.71 \cdot 10^{78}$. Running through all combinations of feature subset would be impossible within the time frame of this project[1] and another method is needed to select a good feature subset.

It should be noticed that all results are relative to the given dataset. This means that features that perform well in this study, may perform worse in others and vice versa. For instance some colour-features gives good classification results, which should not be expected for plants in the field, for which the colour variation will be bigger due to different growth conditions. Other features, like object-area, are very dependent on a static height of the camera, which also cannot be expected in the field.
Some features are also strongly dependent of a good segmentation where others are less sensitive. For instance the colour features will be almost unchanged for a mess of overlapping leaves of the same kind, where the Fourier descriptors for instance will change completely. The best features are therefore not an absolute subset, but depends of the dataset present.

5.1. Related work in the field of feature selection methods

Several feature selection algorithms exists, which overall can be divided in two groups: Those that tries to increase the classification accuracy of the object to be classified and those that tries to minimize the mutual information or correlation between features.
Features from the first group includes the forward selection algorithm (also known as the greedy-like search algorithm) which chooses the feature that benefits the most in the next iteration without looking ahead, this is a simple method, but the risk of getting trapped in a local maxima is high [70, 71]. Another method is the feature elimination algorithm which discard the features with the least importance. However, this method has the limitation that it do not re-evaluate features and also has a tendency to get trapped in local maxima [71]. A third method is to use the Genetic algorithm for feature selection [70, 72, 16]. This selection methods has been evaluated against the forward selection method by [70], where it provides slightly better results but at the expense of increasing the computational effort. Simulated annealing can also be used for feature selection, which have been done by [73], to whom it had provided good results when reducing large feature sets. Methods that try to decrease the feature correlation include the *Pearson product-moment correlation coefficient* [74]. Feature selection by decreasing the mutual information has been described by [75].

[1] Assuming that it takes 1ms to evaluate a features subset the time of testing all subset would require $1.17 \cdot 10^{68}$ years. ☺

5.2. Feature Selection and Reduction Methods

The following section describes 5 different features selection methods. Four of the feature selection methods; *individual feature performance*, *forward selection*, *recursive feature elimination*, and *recursive feature elimination using MDA* will not provide a single optimal subset, but rather a ranking of all the features. The genetic algorithm differs from the others as it optimizes for a constant number of features. The features selection methods do solely perform selection based on the kNN classifier as the processing time of just this one classifier takes weeks. The kNN classifier is used as it achieves much better accuracies than MVG and because the SVM takes hours to optimize the kernel for a single feature subset.

5.2.1 Individiual Feature Performance

A simple way to measure the performance of features, is to determine how individual features perform on the given database. The performance of a feature can be described by different measure, but as the performance is ultimately determined by how well it helps a certain classifier to discriminate classes, the classifier can be included in this process. In the following section the kNN classifier is used to determine individual feature performance solely by the classification accuracy. A MATLAB script[2] has been implemented to determine the classification accuracy of all features individually. Thereby meaning that a feature descriptor containing multiple features such as Elliptic Fourier and Distance transform are split into many individual features. The result of the MATLAB script is stored in a spreadsheet[3] to easily treat and sort the results. In Table 5.1 the top 20 ranking features have been listed based on classification accuracy. The classification accuracy is the average classification accuracy of the three plant elements.

The table shows that the highest ranking features are the five colour based features. The list also includes nine variations of the DistTransform, and in ranking order the Sphericity, HuMoments1, Compactness, RatioOfPrincipalAxes, Eccentricity and DistTransformMean. Three additional tables in Appendix G show the 20 highest ranking features for the three plant elements. The appendix provides an important fact of the project, namely that features perform differently for the three plant elements. Besides the raw features, the MATLAB script mentioned above also stores the classification accuracy of feature descriptors, which consist of multiple subfeatures. Table 5.2 shows the classification accuracies of all the feature descriptors containing multiple features without using dimensionality reduction. The

[2]The MATLAB script is located in: `Matlab/Classifiers/Decision3/Demo_ClassifyAndStoreResults.m`

[3]The spreadsheet is located in: `Database/Result/classifyResultknn25-Nov-2013`

Ranking	Feature no.	Feature Name	CA
1	17	AverageChromaticityBlue	0.521
2	19	AverageChromaticityRed	0.520
3	18	AverageChromaticityGreen	0.504
4	134	VarRGB	0.501
5	20	ExcessGreen	0.486
6	77	DistTransformRS_scaledAreaRoot9	0.482
7	120	DistTransformLP_scaledAreaRoot2	0.477
8	76	DistTransformRS_scaledAreaRoot8	0.477
9	119	DistTransformLP_scaledAreaRoot1	0.475
10	78	DistTransformRS_scaledAreaRoot10	0.472
11	75	DistTransformRS_scaledAreaRoot7	0.471
12	73	DistTransformRS_scaledAreaRoot5	0.458
13	74	DistTransformRS_scaledAreaRoot6	0.452
14	23	Sphericity	0.449
15	6	HuMoments	0.447
16	16	Compactness	0.446
17	79	DistTransformLP_Sort1	0.444
18	29	DistTransformMean1	0.443
19	26	FormFactor	0.443
20	72	DistTransformRS_scaledAreaRoot4	0.442

Table 5.1: The average classification accuracy (CA) of the plant elements of the individual features, using kNN, k=4

classification accuracy is the average classification accuracy of the three plant elements.

The table shows interesting results. First of all these more complex features shows better classification accuracies between 0.464 and 0.779, which means that the 15 best of these feature combinations provides a better performance than the best single feature, shown in Table 5.1. It also shows that variations of the distance transform, perform very well, as they account for 5 of the top 10 features. The last features within the top 10 features are features related to the chromaticity of the plants, which should not be expected to perform as well under varying illumination. Futhermore, the table shows how the different variations of a feature descriptor are ranked according to each other. The proposed DistTransformLP_scaledAreaRoot performs better than all other distance transform features by more than 5 percentage points. The EllipticFourierAbs and the variations of the Elliptic Fourier distance have performances between 0.501 and 0.668 and thus most of the features performs better than the well known HU moments. A table is provided in Appendix H showing the ranking of features for the different plant elements.

Ranking	Feature no.	Feature name	CA
1	33	DistTransformLP_scaledAreaRoot	0.779
2	30	DistTransformLP_SortScaled	0.726
3	32	DistTransformLP_AccScaled	0.712
4	31	DistTransformLP_Acc	0.693
5	28	DistTransformRS_scaledAreaRoot	0.685
6	29	DistTransformLP_Sort	0.685
7	27	DistTransformRS_AccScaled	0.673
8	43	EllipticFourierAbs	0.668
9	24	DistTransformRS_Sort	0.640
10	26	DistTransformRS_Acc	0.629
11	46	EFdist	0.615
12	25	DistTransformRS_SortScaled	0.597
13	48	EFdistAcc	0.567
14	47	EFdistScaled	0.546
15	6	HuMoments	0.511
16	49	EFdistAccScaled	0.507
17	50	EFdistVar	0.501
18	41	MinPlantThickness	0.464

Table 5.2: The average classification accuracy (CA) of feature descriptors containing multiple features using kNN, k=4

The evaluation of individual features provides an understanding and indication of how well a feature performs and how the different variations of a feature descriptor are ranked according to each other. Determining the optimal subset by purely using features with the highest classification accuracy will in most cases not provide an optimal solution as many features are correlated. An example of correlated features might be the 10 different variations of the distance transform as they all are derived from the distance transform.

5.2.1.1 Evaluation of proposed features

The following provides a small explicit evaluation of the proposed variation of the distance transform, the elliptic Fourier distance and the stem thickness estimator.

Evaluation of features derived from the distance transform In this section the features extracted from the distance transform will be presented.

In [17] four features sets are extracted from the distance transform and tested on nightshade and cornflower at growth stage BBCH 12. These four sets are:

- $\mathbf{d}_{sort}$, which is the sorted distance vector.
- $\mathbf{d}_{scaled}$, which is a normalized version of $\mathbf{d}_{sort}$.
- $\mathbf{d}_{accu}$, which is an accumulated version of $\mathbf{d}_{sort}$.
- $\mathbf{d}_{accuscaled}$, which is an accumulated and normalized version of $\mathbf{d}_{sort}$.

These four vectors are represented in two ways; either in re-sampled versions, where 10 equidistant samples are taken from each vector, or by using coefficients for a 10th order Legendre fitted polynomial. By using the last approach, [17] achieves a classification accuracy of 96.25% for $\mathbf{d}_{sort}$ and $\mathbf{d}_{accuscaled}$, and a classification accuracy of 97.5% on $\mathbf{d}_{scaled}$ using Legendre polynomial coefficients as features.
In this study, seven plant species are available, which are present at different growth stages, but which do not include nightshade and cornflower. It is therefore interesting to see how well these features perform on this, larger dataset. The project propose two variations of the distance transform as described in section 3.2.2 ,$\mathbf{d}_{scaledAreaRoot}$. Table 5.3 shows the average classification accuracy for all plant elements and the classification accuracy of plant, cotyledon and foliage using the k-Nearest Neighbours (kNN) classifier using k = 4. The features have been ranked according to the classification ac-

No.	FeatureName	Plant		Cotyledon		Foliage		Total	
		Rank	CA	Rank	CA	Rank	CA	Rank	CA
33	DistTransformLP_scaledAreaRoot	1	0.803	1	0.762	1	0.777	1	0.779
30	DistTransformLP_SortScaled	2	0.740	2	0.709	2	0.743	2	0.726
32	DistTransformLP_AccScaled	3	0.729	3	0.697	3	0.719	3	0.712
31	DistTransformLP_Acc	6	0.711	4	0.677	4	0.703	4	0.693
28	DistTransformRS_scaledAreaRoot	5	0.718	5	0.659	5	0.692	5	0.685
29	DistTransformLP_Sort	4	0.722	6	0.656	6	0.691	6	0.685
27	DistTransformRS_AccScaled	7	0.699	7	0.649	7	0.687	7	0.673
24	DistTransformRS_Sort	8	0.663	8	0.621	9	0.647	8	0.640
26	DistTransformRS_Acc	9	0.633	9	0.615	8	0.654	9	0.629
25	DistTransformRS_SortScaled	10	0.594	10	0.583	10	0.638	10	0.597

Table 5.3: Classification accuracies (CA) for a 4-nearest neighbour classifier using the distance transform features.

curacy that they provide, showing that DistTransformLP_scaledAreaRoot performs best for all plant elements. The table shows that the features are ranked relatively similar for the different plant elements. The Legendre polynomial variation (LP) of the distance transform generally performs better than the resampled (RS) variation. Classification accuracies of the different plant species are also presented in Appendix I using the kNN with a k value

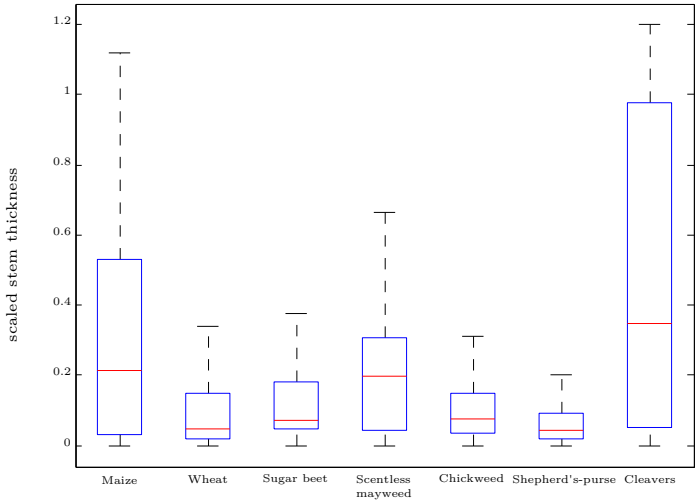


Figure 5.1: Boxplot of smallest thickness of all seven species, evaluated on whole plants

of respectively 4, 1 and 1 for plant, cotyledon and foliage leaves. The classification accuracy is much lower than the results presented in [17] as five additional plants have been included, which at the same time are present at different growth stages.

Evaluation of the stem Thickness estimator Here the *minimum stem thickness*-feature, will be evaluated. The feature was made with the purpose of discriminating scentless mayweed from shepherd's-purse, as the appearences of them are very alike, except that scentless mayweed tends to have a slightly thicker stem than shepherd's-purse. To show the performance of this feature alone, the classification accuracy for each of the seven species is shown in Table 5.4. From this it is clear that this feature is not very valuable

Species							
Maize	Wheat	Sugar beet	Scentless mayweed	Chickweed	Shepherd's-purse	Cleavers	Total
0.360	0.071	0.206	0.414	0.464	0.126	0.146	0.323

Table 5.4: Classification accuracies (CA) for a 4-nearest neighbour classifier using the minimum stem thickness

on its own.
To demonstrate the ability of the feature to discriminate each of the species from each other, a boxplot have been made in Figure 5.1. From this it can be seen that this feature is able to distinguish most scentless mayweed from shepherd's-purse plants. However, there are still an overlap, which means

that this feature alone is not enough to discriminate the two species. When looking at all species, maize, scentless mayweed and cleavers do in general have thick stems compared to the four other species, which seems reasonable when looking at the samples in shown in Appendix O.
A thing that should be noticed is that for all species, there are samples with a scaled thickness close to zero. This is a consequence of rough leaf edges from the segmentation, which means that the plants are divided in two parts after only few erosions. Therefore, there are room for optimization, so that such cases can be avoided.

Elliptic Fourier Distance Based on the Elliptic Fourier, different Elliptic Fourier distance features have been proposed in Chapter 3. The Elliptic Fourier distance features are distance statistics between subsequent approximations of Elliptic Fourier for an increasing number of harmonics. The classification accuracies are presented in Table 5.5, showing that EFdist

No.	FeatureName	Plant		Cotyledon		Foliage		Total	
		Rank	CA	Rank	CA	Rank	CA	Rank	CA
47	EFdist	1	0.698	3	0.519	1	0.708	1	0.615
49	EFdistAcc	2	0.585	2	0.526	3	0.637	2	0.567
48	EFdistScaled	3	0.579	1	0.527	4	0.535	3	0.546
50	EFdistAccScaled	4	0.533	4	0.486	5	0.510	4	0.507
51	EFdistVar	5	0.509	5	0.425	2	0.679	5	0.501

Table 5.5: Result of Elliptic Fourier distance features

generally performs the best with a classification accuracy of 0.615 followed by EFdistAcc, EFdistScaled, EFdistAccScaled and finally EFdistVar. The EFdist performs worse than most distance transform features and the elliptic Fourier features, but will still provide a good discrimination between classes. As the EFdist is based on Elliptic Fourier, they are likely to be highly correlated. Assuming that they are not highly correlated, they will combined improve discrimination. The classification accuracy of the features Elliptic Fourier, EFdist and the features of Elliptic Fourier+EFdist combined is presented in Table 5.6. The classification accuracy is improved by 2.2

No.	FeatureName	Plant	Cotyledon	Foliage	Total
47	EFdist	0.698	0.519	0.708	0.615
43	EllipticFourierAbs	0.668	0.725	0.612	0.711
43+47	EFdist+EllipticFourierAbs	0.747	0.723	0.731	0.733

Table 5.6: The classification accuracy of EFdist and Elliptic Fourier, individually and combined to investigate correlation.

percent points, indicating that EFdist provides additional discriminating information especially for Plants and foliage leaves.

5.2.2 Multiple Discriminant Analysis (MDA)

Multiple Discriminant Analysis (MDA) is a dimension reduction method that takes c classes of d dimensions/features and reduces the number of dimensions to a $(c-1)$ dimensional space. MDA assumes that the feature dimension d is larger or equal to the number of plant species c that are being classified[69].
In this specific case it will reduce the number of features from 261 to only 6 features as the data set contains 7 classes/species. The method is highly related to PCA described in Appendix B as it also reduces the number of dimensions by taking a linear combinations of all the features. The difference is that PCA finds the projection to a lower dimensions space that keeps the highest variance of the data, while MDA finds the projection that keeps the highest discrimination between classes. MDA keeps discrimination between classes by maximizing the distance between classes, while minimizing the variance within each class. An example is illustrated in Figure 5.2. The objective is to find a projection $\mathbf{w}$ (for two dimensions onto a line) providing the highest discrimination. The projected mean for first and second class is denoted $\tilde{\mu}_1$ and $\tilde{\mu}_2$, while the projected variation for the first and second class is denoted $\tilde{\sigma}_1^2$ and $\tilde{\sigma}_2^2$. MDA will optimize discrimination by making an projection that will maximize the distance between projected means $|\tilde{\mu}_1 - \tilde{\mu}_2|$, and minimizing the sum of the projected variance of the two classes $\tilde{\sigma}_1^2 + \tilde{\sigma}_2^2$.
The discrimination criteria makes the MDA linear reduction method more

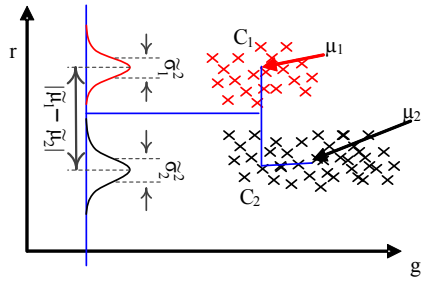


Figure 5.2: An example of MDA discriminating two classes in a two dimensional space. Discrimination is maximized by increasing distance between projected means $|\tilde{\mu}_1 - \tilde{\mu}_2|$, and minimizing the projected variance of the two classes $\tilde{\sigma}_1^2$ and $\tilde{\sigma}_2^2$.

optimal than PCA as the goal is to classify at a lower dimensional feature

space. The actual derivation of MDA is described in Appendix C. An implemented MATLAB function[4] calculates the linear projection to transform the features into a $(c-1)$ dimensional space. The linear projection $\mathbf{w}$ determined by MDA is then used on each plant sample $\mathbf{x}$

$$\mathbf{y} = \mathbf{w}^T \mathbf{x}, \tag{5.1}$$

to make a plant sample $\mathbf{y}$ of lower dimensionality. One advantage of MDA is that it provides great visualization of data of high dimensionality. Figure 5.3 shows the three dimensions of MDA that provides the highest discrimination between classes. From this it is seen how the different classes are more or less divided into separated clusters. Figure 5.4 shows the histogram

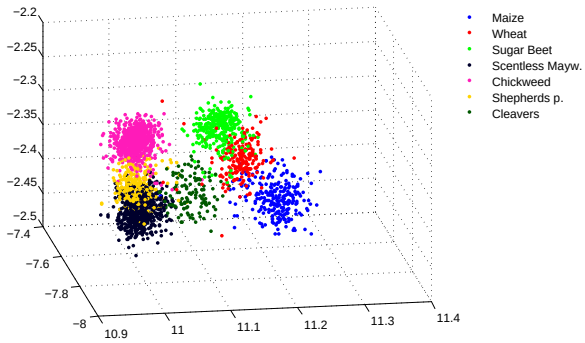


Figure 5.3: 3D plot of the 3 most discriminating dimensions of the MDA of plants.

and Gaussian distributions of the data for each dimension of the MDA. Starting with the dimension of highest discriminant power in the top left corner, the next highest in top right corner and so forth down to the bottom right corner. It is seen how the distributions overlap each other more as the dimension becomes of minor discriminant power. The classification accuracies[5] are determined for MVG and the kNN classifiers (k=4), when using all features without any dimension reduction and when using MDA and PCA (6 dimensions)[6]. The results show how the feature reduction improves the result of the classification. The MVG is highly affected by the number of features, resulting in a very low classification accuracy of 0.108 when no dimensionality reduction is made. In the given case the feature space contains 261 dimensions requiring a must greater data set. Reducing

[4] The MATLAB script is located in: `Matlab/Classifiers/myMDA_Features.m`

[5] The Classification accuracy is a weighted average of the three plant elements, depending on the number of samples of the elements

[6] SVM has been left out of the table as the optimal sigma and a box constraint must be found for each plant element and when using no feature reduction and when using MDA and PCA for 6 dimensions.

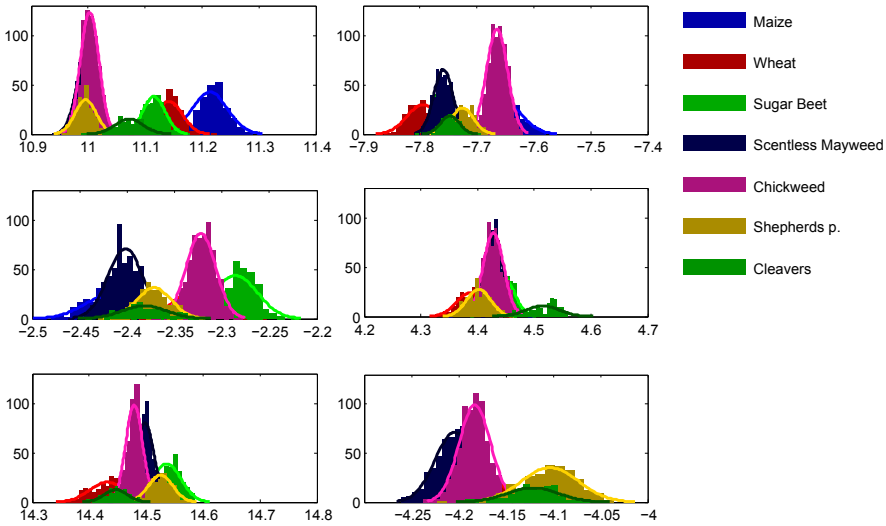


Figure 5.4: Showing how the classes are discriminated in the 6 dimensions of MDA. Starting with the highest discriminant power in the top left corner, the second highest in the top right position and so forth.

Classifier	Dimension Reduction		
	None	PCA	MDA
MVG	0.108	0.566	0.833
kNN	0.885	0.825	0.869

Table 5.7: Classification accuracy of classifieres for all features when using no dimension reduction and when using MDA and PCA (6 dimensions).

the features space to 6 dimensions with PCA improves the result to 0.566. Reducing the features space with MDA to maintain discrimination between classes improves the classification accuracy to 0.833. The kNN classifier performs well for all settings showing high robustness with samples of high dimensionality. MDA will though provide a remarkable improvement to kNN as the affect of noisy or bad features have been minimized in the MDA reduction of the data. MDA is though not an optimal feature reduction method as it will have the chance of removing important information and worsen the classification accuracy.

5.2.3 Forward Selection

In the forward selection algorithm, a list of features is selected in an iterative process in which the feature that adds most to the classification

accuracy of the existing feature list is added to the feature list. To rank all features, the algorithm will have as many iterations as there are features. In each of the iterations a classification is made for each of the features that have not been selected yet, in combination with the ones that have been selected in earlier iterations. The principle is shown in Algorithm 2.

Algorithm 2: Forward feature selection algorithm

var $featurelist_0$ is empty;
var $remainFeatures = Allfeatures$;
for $i \leftarrow 1$ **to** $N_{Allfeatures}$ **do**
  for $n \leftarrow 1$ **to** $N_{remainFeatures}$ **do**
    $tmpfeaturelist_n = [featurelist_i \oplus remainFeatures_n]$;
    Train using $tmpfeaturelist_n$;
    $classAcc_n$ = classification accuracy using $tmpfeaturelist_n$;
  end
  $featurelist_{i+1} = tmpfeaturelist_n$ with max $classAcc_n$.
  $remainFeatures = [Allfeatures \ominus featurelist_{i+1}]$
end

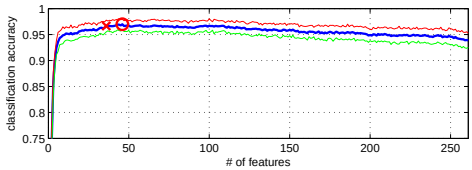
The problem with the forward selection algorithm is the dependency of one feature to the former selected features. That is a problem, as some features might have poor discrimination power alone, but in combination provide stronger discrimination power than the features selected using the forward feature selection algorithm. There are therefore a risk of getting trapped in a local maxima.

The graphs in Figure 5.5 show the most important features for respectively whole plants, cotyledons and foliage. The figure is made by the average of 5 runs using a 10-fold cross validation, providing 50 different validations[7]. As it is seen, the variance of the classifications is largest for the foliage leaves and smallest for the plants, which might indicate that the leaf samples are not as representative for the dataset as the plants.

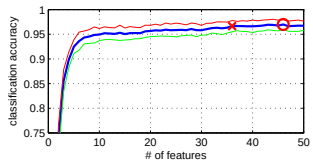
The indices on the x-axis can be looked up in Appendix J, which shows a full list of the features and the classification accuracies using all features up to the selected one.

By assuming that the classification accuracies from the 50 validations are Gaussian distributed (which in not totally correct due to the bounded range), the mean and standard deviation of the classifications are found. The blue line in Figure 5.5 indicates the mean classification accuracy from 50 cross-validations, the red line indicates the mean value plus the standard

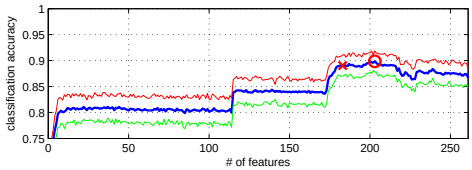
[7] The forward selection method selects features by only a single 10-fold cross validation for each iteration



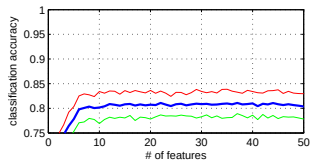
(a) Forward selection on plants. The mean classification accuracy is 96.7% for a subset of 36 features, but a classification accuracy of 95.0% can still be achieved with only 10 features.



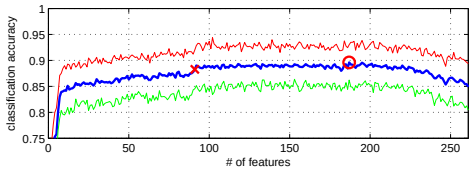
(b) Same plot as (a), but zoomed on the first 50 features.



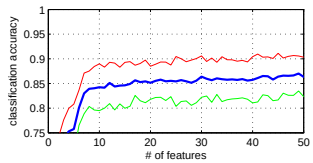
(c) Forward selection on cotyledons. The mean classification accuracy is 89.0% for a subset of 183 features.



(d) Same plot as (c), but zoomed on the first 50 features.



(e) Forward selection on foliage. The mean classification accuracy is 88.3% for a subset of 91 features, but a classification accuracy of 85.2% can be achieved with only 12 features.



(f) Same plot as (e), but zoomed on the first 50 features.

Figure 5.5: Forward selection for (a) Whole plants, (c) Cotyledons and (e) Foliages. The blue line indicates the mean value from the 50-fold cross-validation, the red line indicates the mean value plus the standard deviation and the green line indicates the mean minus the standard deviation from the cross validation. The red ∘ indicates the maximum classification accuracy and the red × indicates that selected subset.

deviation of the validations, and the green line indicates the mean value minus the standard deviation of the validations. The maximum classification rate is marked by a red ∘, but by looking at the graphs, it is seen that they are almost flat after only few features, for which reason, the feature subset at the maximum classification rate might not be the best choice.
It should be noticed that for cotyledons, there are two steep steps in the classification accuracy around 115 and 175 features. This is probably due to the algorithm getting trapped in a local minima, which indicates that this might not be the best feature ranking method. The features that causes these two steps are both related to distance transform.

T-test as stopping criteria As there are some variation in the different classification rates from the cross-validations, a t-test is made to find a good choice of subset with as few features as possible. For the t-test a null-hypothesis is made for each number of feature, saying that the results from the cross-validations at this point and the results from the cross validations at the point of the maximum classification rate are from the same distribution with the same mean. The t-test for a single hypothesis is given by

$$t = \frac{\bar{x}_k - \bar{x}_{\max}}{\sqrt{\frac{s_k^2}{n} + \frac{s_{\max}^2}{n}}}, \tag{5.2}$$

where $\bar{x}_k$ and s_k^2 are the sample mean and variance, respectively, at the distribution using k features and $\bar{x}_{\max}$ and $s_{\max}^2$ are similar measures at the distribution with the maximum mean classification accuracy. By using a 5% significance level, some of these null-hypothesises are rejected which means that they are significant lower than the classification accuracy at the maximum. But for others, this hypothesis can not be rejected, which means that we cannot conclude that one is better than the other. Therefore the feature subset with the smallest number of features, which are not significant lower than the feature subset with the highest mean classification accuracy, is chosen.
The subset chosen from the t-test is marked with a red × on the graphs.

By looking at the top 10 selected features for plants, cotyledons and foliages in Table 5.8, it is seen the plant is best described by a variety of features, whereas cotyledons in general are best described by using the variations of the distance transform, which accounts for four of the top 10 feature, but also caused big increases in the classification accuracy as earlier described.
For foliages, seven of the top 10 features are variations of the distance transform.

#	Plants	Cotyledons	Foliages
1	AverageChromaticityBlue1	DistTransformRS_Acc1	DistTransformRS_scaledAreaRoot4
2	DistTransformRS_Sort4	DistTransformRS_Sort8	DistTransformRS_AccScaled7
3	DistTransformRS_scaledAreaRoot6	HuMoments8	Convexity1
4	ExcessGreen1	DistTransformLP_scaledAreaRoot5	DistTransformLP_SortScaled8
5	Sphericity1	Convexity1	Solidity1
6	EllipticVariance1	VarRGB1	DistTransformRS_AccScaled3
7	EFdistScaled5	EllipticFourierAbs58	DistTransformRS_AccScaled5
8	VarRGB1	SkeletonDistanceMax1	DistTransformRS_scaledAreaRoot1
9	DistTransformRS_Sort9	DistTransformLP_AccScaled5	EllipticFourierAbs64
10	EFdistAcc1	EFdistAcc8	DistTransformLP_Sort1

Table 5.8: The ten first features for plants, cotyledons and foliages, achieved by using the forward selecting algorithm

5.2.4 Recursive Feature Elimination

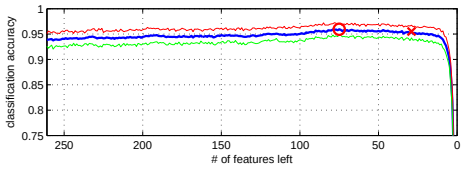
The Recursive feature elimination algorithm is quite similar to the forward selection algorithm, but instead of adding a new feature in each iteration, the least important feature is removed from a list containing all features. This algorithm is therefore not influenced on the order of the remaining features such as in the forward selection algorithm. Still, the recursive feature elimination algorithm can get trapped in local minima. The Recursive feature elimination algorithm is described in Algorithm 3.

Algorithm 3: Recursive feature elimination algorithm

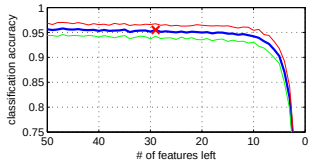
var $featurelist_1 = Allfeatures$;
var $remainFeatures = Allfeatures$;
for $i \leftarrow 1$ **to** $N_{featurelist_1} - 1$ **do**
  for $n \leftarrow 1$ **to** $N_{remainFeatures}$ **do**
    $tmpfeaturelist_n = [featurelist_i \ominus remainFeatures_n]$;
    Train using $tmpfeaturelist_n$;
    $classAcc_n$ = classification accuracy using $tmpfeaturelist_n$;
  end
  $featurelist_{i+1} = tmpfeaturelist_n$ with max $classAcc_n$;
  $remainFeatures = [Allfeatures \ominus featurelist_{i+1}]$;
end

The calculations for the recursive feature elimination algorithm are quite computationally intensive, as the required number of classification is $\sum_{x=1}^{N} x = 34191$, for $N = 261$ features, where each classification might consist of multiple cross-validations
By using the feature elimination algorithm the three graphs in Figure 5.6 are created, showing the order in which the least important features are removed for whole plants, cotyledons and foliage respectively. The figure is made by the average of 5 runs using a 10-fold cross validation, providing 50 different

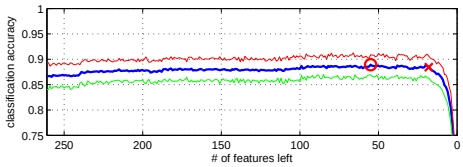
validations[8]. The indices on the x-axis can be looked up in Appendix K,



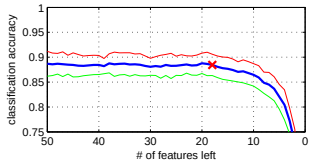
(a) Recursive feature elimination on plants. The mean classification accuracy is 95.5% for a subset of 29 features.



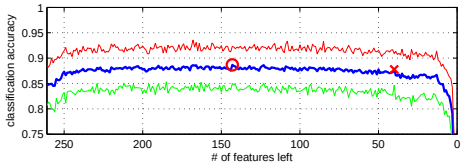
(b) Same plot as (a), but zoomed on last 50 features.



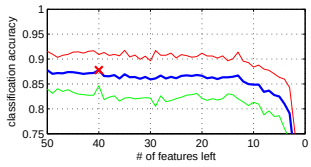
(c) Recursive feature elimination on cotyledons. The mean classification accuracy is 88.5% for a subset of 18 features.



(d) Same plot as (c), but zoomed on last 50 features.



(e) Recursive feature elimination on foliages. The mean classification accuracy is 87.8% for a subset of 40 features.



(f) Same plot as (e), but zoomed on last 50 features.

Figure 5.6: Recursive feature elimination on (a) Whole plants, (c) Cotyledons and (e) Foliages. The blue line indicates the mean value from the 50-fold cross-validation, the red line indicates the mean value plus the standard deviation and the green line indicates the mean minus the standard deviation from the cross validation. The red ○ indicates the maximum classification accuracy and the red × indicates that selected subset.

which shows a full list of the features and the classification accuracies after removing all features up to the selected one.

As with the graphs from the forward selection, the blue line indicates the mean classification accuracy from a 50 validations, the red line indicates the mean value plus the standard deviation of the validations, and the green line

[8] The Recursive feature elimination algorithm selects features by only a single 10-fold cross validation for each iteration

indicates the mean value minus the standard deviation of the validations. To find the optimal subset, the same procedure as with the forward selection is used, where the chosen feature subset is the one with the smallest number of features and which with a 5% significance level does not perform significant worse that the best subset.
The classification accuracy for the chosen subset is marked with a red ○ and the classification accuracy for the subset at the maximum classification accuracy is marked with a red×.

The top 10 ranking features are shown in Table 5.9.

#	Plants	Cotyledons	Foliages
1	ExcessGreen1	DistTransformRS_SortScaled9	DistTransformRS_SortScaled5
2	DistTransformLP_Sort2	ObjectArea1	ObjectArea1
3	Compactness1	DistTransformRS_SortScaled2	DistTransformRS_AccScaled2
4	AverageChromaticityRed1	DistTransformRS_AccScaled4	ConvexHullArea1
5	DistTransformRS_AccScaled4	DistTransformRS_AccScaled2	DistTransformRS_SortScaled7
6	EFdistAccScaled4	VarRGB1	HuMoments7
7	AverageChromaticityGreen1	DistTransformLP_SortScaled6	EllipticVariance1
8	Rectangularity1	EllipticFourierAbs58	HuMoments1
9	DistTransformLP_SortScaled5	DistTransformLP_Sort8	DistTransformRS_AccScaled4
10	EFdistAccScaled7	DistTransformRS_scaledAreaRoot5	DistTransformRS_AccScaled6

Table 5.9: The 10 best features for plants, cotyledons and foliages respectively found by recursive feature elimination

When looking at the features, describing whole plants, cotyledons and foliages, it is seen that DistTransformRS_AccScaled4, DistTransformRS_AccScaled2 and ObjectArea1 are the only features that are present more than once in the ten best features for the three elements.

When we extend to the top 20 features it is seen that whole plants use more of the "standard" features such as *Solidity*, *Rectangularity* and *Compactness*, where the classifications of leaves primarily depend on the elliptic Fourier descriptors or the distance transform features. Especially the different variations of the distance transform feature are good at describing leaves, as they account for 15 of the top 20 features for cotyledons and 9 of the top 20 features for foliages.

For the distance transform features, it is seen that among the top 20 features for the three tests, the re-sampled and the Legendre polynomial variations are almost equally represented.
The feature, *MinPlantThichness*, which is the feature that was suggested in Section 3.2.5 as a way to discriminate scentless mayweed and shepherd's-purse, is not present at all in the top 20 features found from the recursive feature extraction.

5.2.5 Recursive Feature Elimination using Multiple Discriminant Analysis

The multidiscriminan analysis (MDA) makes a weighting of each feature as they are reduced to a lower dimensional feature space. Features providing a low discrimination are weighted low, while features providing high discrimination are high weighted. To make sure that the weighting is independent on the magnitude or the absolute value of the features, a feature $\mathbf{f}$ of N samples is subtracted by the smallest value to shift the features values above zero.

$$\tilde{\mathbf{f}} = \mathbf{f} - \min(\mathbf{f}) \tag{5.3}$$

The length of all features are then normalized by the norm of all the samples.

$$\tilde{\tilde{f}}_i = \frac{\tilde{f}_i}{\left\|\tilde{\mathbf{f}}\right\|} \tag{5.4}$$

For uncorrelated features the weight of a feature is an indication of the discriminate power for that feature. For correlated features the weight is distributed between correlated features, consequently meaning that many correlated features are weighted weaker. Selecting feature only based on the weights of the MDA is therefore not an optimal selection method as proposed in [76]. Instead this study propose a feature elimination method by only removing the feature with the lowest weight for each iteration. This means that only one of the correlated features will be removed and thereby increase the weights of the rest of the correlated features. MDA works optimal when data is spread in single clusters with a Gaussian distribution, and the feature elimination procedure using MDA is therefore not always optimal. In comparison to the procedure described in "Recursive Feature Elimination" allowing multiple clusters with non-Gaussian distributions the "Recursive Feature Elimination using Multiple Discriminant Analysis" is seemingly not preferred. However, the advantage of "Recursive Feature Elimination using Multiple Discriminant Analysis" is that the computations do not explode exponentially, but close to proportional for an increasing number of features. In MATLAB the execution of "Recursive Feature Elimination" takes days (around 120 hours for cotyledons), while the "Recursive Feature Elimination using Multiple Discriminant Analysis" is executed in seconds (around 15 seconds for cotyledons). In Table 5.10 the classification accuracies using kNN with the optimal subset in accordance with "Recursive Feature Elimination using Multiple Discriminant Analysis" is shown.

In Appendix L the classification accuracies for a decreasing number of features are determined using the Recursive Feature Elimination using Multiple Discriminant Analysis procedure for kNN, kNN+MDA, MVG and MVG+MDA. The procedure using MDA for recursive feature elimination provides good

	Number of features	Classification accuracy
Plant	127	0.943
Cotyledon	140	0.865
Foliage	230	0.867

Table 5.10: Classification accuracy of kNN for the optimal subset in accordance with Recursive Feature Elimination using Multiple Discriminant Analysis.

results for plants, but provides worse results for cotyledons and foliage leaves. The reason for this is presumably that classes are not distributed in single Gaussian distributions.

5.2.6 Genetic algorithm for feature selection

The genetic algorithm, described in Section 2.3.1, is another method that can be used to find good feature combinations[70].
In the genetic algorithm, the chromosomes p_n that make up the feature subsets, consists of n entries, where each entry refers to the indices of the features in the list of all features. The aim is to let the chromosomes evolve to better subsets based on the classification accuracy og them. The procedure of the algorithm is the same as described in Algorithm 1, but instead, the fitness of each chromosome, i.e. how good it performs, is found using the normalized classification accuracy of a k-nearest-neighbour-classifier.

The feature subsets shown in Table 5.11, 5.12 and 5.13 are found for plants, cotyledons and foliages respectively by using 45 features for each chromosome, 800 iterations, a cross-over probability of 0.7 and a mutation probability of 0.1. For each iteration, the probabilities is decreased by $1/800$ of the current values to stabilize the final result. With these 45 features the maximum classification accuracy is 93.7%, 84.3% and 85.3% for plants, cotyledons and foliages, respectively. This is a bit poor compared to the classification accuracies found by the forward selection and recursive feature elimination.
By looking at the selected features, it is also seen that the Fourier features and the distance transform features accounts for most of the selected features. However, it should be remembered that these also account for most of the 261 features.

EllipticFourier77	EllipticFourier25	EFdistVar7
DistTransformLP_AccScaled8	EllipticFourier4	EllipticFourierAbs67
FormFactor1	DistTransformRS_Sort8	DistTransformLP_AccScaled2
DistTransformRS_SortScaled7	EllipticFourierAbs20	AverageChromaticityRed1
DistTransformLP_Sort2	AverageChromaticityGreen1	EllipticFourierAbs32
SkeletonDistanceMax1	DistTransformRS_AccScaled8	EllipticFourierAbs1
EllipticFourier19	EllipticFourier52	EllipticFourier74
EllipticFourierAbs10	DistTransformLP_Acc6	Sphericity1
ExcessGreen1	DistTransformRS_SortScaled9	EllipticFourierAbs2
DistTransformLP_scaledAreaRoot4	DistTransformRS_scaledAreaRoot3	EllipticFourier5
EFdist3	EFdistScaled6	DistTransformLP_Acc7
EllipticFourier23	DistTransformLP_scaledAreaRoot8	EFdistAcc2
DistTransformLP_Sort10	DistTransformRS_Acc1	DistTransformRS_AccScaled7
DistTransformLP_Sort9	EllipticFourierAbs36	EllipticFourierAbs57
EllipticFourierAbs7	EFdistScaled8	DistTransformLP_Sort3

Table 5.11: Best 45 features for classification of plants found by using the genetic algorithm. The features are not ordered by the performance of the individual features. The classification accuracy by using these features is 93.7%

DistTransformLP_SortScaled5	DistTransformRS_AccScaled2	EllipticFourier54
EllipticFourier21	EFdistScaled6	EFdist5
EllipticFourier7	EllipticFourierAbs74	DistTransformRS_SortScaled8
ObjectArea1	EllipticFourier42	EFdistAccScaled6
EllipticFourier67	DistTransformRS_SortScaled2	EllipticFourierAbs37
DistTransformLP_AccScaled10	DistTransformRS_Sort6	VarRGB1
DistTransformLP_scaledAreaRoot3	EllipticFourierAbs22	EllipticFourierAbs77
EFdistVar6	DistTransformRS_AccScaled1	ObjectArea1
DistTransformLP_scaledAreaRoot3	EllipticFourierAbs77	DistTransformRS_SortScaled6
EllipticFourier19	EllipticFourier22	DistTransformRS_scaledAreaRoot10
EllipticFourierAbs17	EllipticFourier43	DistTransformLP_AccScaled7
EllipticFourier37	DistTransformRS_AccScaled1	DistTransformLP_AccScaled7
SkeletonDistanceMean1	DistTransformRS_AccScaled5	EllipticFourier9
HuMoments6	EF_TVH_Max1	EllipticFourier44
DistTransformRS_SortScaled9	ConvexHullPerimeter1	DistTransformLP_AccScaled7

Table 5.12: Best 45 features for classification of cotyledons found by using the genetic algorithm. The features are not ordered by the performance of the individual features. The classification accuracy by using these features is 84.3%

EllipticFourier77	EllipticFourier5	EllipticFourierAbs14
EllipticFourier31	DistTransformRS_SortScaled8	DistTransformRS_SortScaled5
AspectRatio	CircularVariance1	DistTransformRS_AccScaled4
HuMoments8	EFdistAccScaled7	EllipticFourierAbs34
EllipticFourier31	ExcessGreen1	EllipticFourierAbs70
EllipticFourier26	EllipticFourier60	DistTransformLP_scaledAreaRoot4
EllipticFourier13	EllipticFourierAbs76	ObjectPerimeter1
Solidity	ObjectArea	EllipticFourier66
EllipticFourierAbs61	EllipticFourier9	DistTransformRS_SortScaled7
DistTransformRS_AccScaled2	DistTransformLP_AccScaled5	EFdist9
EllipticFourierAbs30	SkeletonDistanceLength1	EllipticFourier72
DistTransformRS_Sort4	SkeletonDistanceMax1	EFdistAccScaled6
EllipticFourier68	DistTransformLP_Acc6	EFdistAcc5
DistTransformLP_Acc7	DistTransformLP_Sort8	EllipticFourierAbs55
EFdistVar6	EllipticVariance1	EllipticFourierAbs17

Table 5.13: Best 45 features for classification of foliages found by using the genetic algorithm. The features are not ordered by the performance of the individual features. The classification accuracy by using these features is 85.3%

						Ranking methods									
		All features		Genetic alg.		Indi. Feat.		Forward sel.		RFE		RFE MDA		MDA	
Classifier	Plant Ele.	N	AC	N	AC	N	AC	N	AC	N	AC	N	AC	N	AC
MVG	Plant	261	0.444	45	0.840	34	0.925	31	0.928	12	0.924	65	0.872	261	0.948
	Cotyledon	261	0.065	45	0.582	86	0.734	43	0.369	27	0.796	82	0.696	261	0.806
	Foliage	261	0.127	45	0.429	71	0.693	96	0.486	4	0.598	60	0.599	261	0.694
	Total		**0.205**		**0.640**		**0.791**		**0.580**		**0.802**		**0.737**		**0.833**
kNN	Plant	261	0.941	45	0.937	48	0.949	36	0.967	29	0.955	84	0.937	261	0.953
	Cotyledon	261	0.871	45	0.843	113	0.871	183	0.890	18	0.885	135	0.860	261	0.832
	Foliage	261	0.841	45	0.853	27	0.851	12	0.883	40	0.878	144	0.848	261	0.772
	Total		**0.889**		**0.877**		**0.894**		**0.915**		**0.907**		**0.884**		**0.862**
SVM	Plant							36	0.557	29	0.951				
	Cotyledon							183	0.507	18	0.889				
	Foliage							12	0.471	40	0.832				
	Total								**0.517**		**0.899**				

Table 5.14: The number of features used and the classification accuracy for the kNN and MVG classifier for the different features subsets and the MDA dimension reduction method. RFE stands for recursive feature selection.

5.3. Result of the Feature Selection Methods

A fast and good overview of all the selection methods is hardly achievable, due to the many potential options. First the classification accuracy is determined for two classifiers; MVG and kNN. Secondly the classifiers are used on plants, cotyledons and foliage leaves and thirdly, five different features selection methods have been used. The result of the different features selection methods are therefore presented in two sections.

5.3.1 Accuracy of Feature Selection Methods

The first section presents the classification accuracy for the different, optimal feature subsets. The genetic algorithm differs from the others as it optimizes for a constant number of features. The four other feature selection methods; *individual feature evaluation*, *forward selection*, *Recursive feature elimination*, and *Recursive feature elimination using MDA* will not provide a single optimal subset, but rather a ranking of all the features. E.g. the *individual feature evaluation* method ranks the features based on how well they perform individually or e.g. the *forward selection* method ranks features based on when they are added in the forward selection procedure. After the features have been ranked by one of the four features selection methods, the classification accuracy is calculated for an increasing number of the ranked features. The highest classification accuracy is then determined, and by the t-test a subset is selected as described in Section 5.2.3. The classification accuracy of the subset of each methods is presented in Table 5.14.

Generally the MVG classifier performs worse than kNN in all settings, which partly is because the selection methods are optimized with kNN.

The MVG classifier has a bad performance when using all features, which can be explained by non-Gaussian distributed data and due to the *curse of*

dimensionality. A higher classification accuracy is achieved by using any of the different features selection strategies. The best performing feature selection method is forward selection, which achieves a total classification accuracy of 91.5%. By using all features together with the dimension reduction method MDA, a total classification accuracy of 86.2% is achieved. The disadvantage of using MDA is that no features are removed.

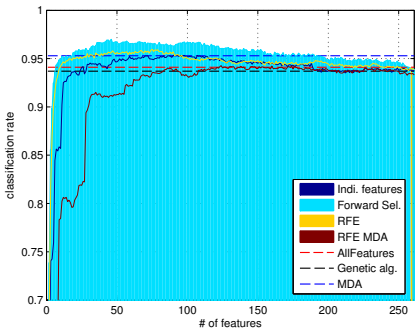
The SVM classifier has only been used to test the feature subset achieved by the *forward selection* and *recursive feature elimination* with a unoptimized kernel, as the kernel must be trained for each subset. The best performances of the method is better than MVG, and almost similar to kNN for the subsets achieved using *recursive feature elimination*.

The kNN classifier performs the best, but also very similar for all the different feature subsets. However, the forward selection method improves by just around 2 to 3 percentage points compared to the subset found by the genetic algorithm and when using all features. The real advantage of using a feature selection method for the kNN is to reduce the number of features. The kNN classifier achieves a classification accuracy of 0.894 by using the 48, 113, and 27 top ranking features for plants, cotyledons and foliages, respectively.

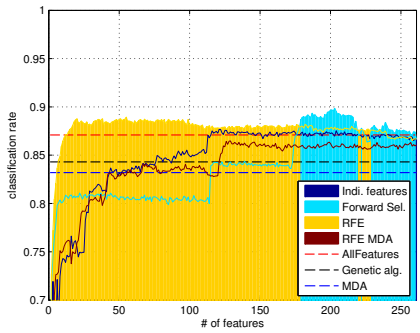
5.3.2 Feature Selection Method Evaluation

Another interesting characteristic of the feature selection methods is how few features that are necessary to achieve a given classification accuracy. To present this in a manageable way, the feature selection methods are presented in the same plot for an increasing number of features. The colours of the area below the graphs are determined by the feature selection method that performs best for the given number of features. Feature subsets using a constant number of features, and therefore a constant classification accuracy, is inserted in the plot as straight lines. The subset having a constant accuracy is; *all features* (red dashed line), the genetic algorithm (black dashed line) and the dimension reduction method MDA used on all features(blue dashed line).
In Figure 5.7a, 5.7b and 5.7c the plots are shown for plants, cotyledons leaves and foliage leaves, respectively.

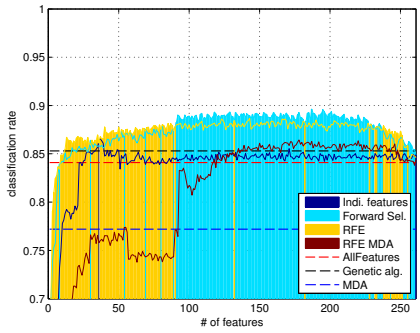
In all three plots the highest classification accuracy is achieved using the *forwards selection* method. For plants (Figure 5.7a) the *forward selection* method is superior to the other ranking methods for any number of features apart from only a small section. For cotyledon (Figure 5.7b) and foliage (Figure 5.7c) the *forwards selection* method performs bad for a low number



(a) The accuracy of multiple feature selection methods for an increasing number of features using **plant**.



(b) The accuracy of multiple feature selection methods for an increasing number of features using **cotyledon**



(c) The accuracy of multiple feature selection methods for an increasing number of features using **foliage**

Figure 5.7: The accuracy of different feature selection methods: The *individual feature evaluation*, *forward selection*, *recursive feature elimination (RFE)*, *recursive feature elimination+MDA (RFE MDA)*. The constant dashed lines show the accuracy of *all features*, the *genetic algorithm* and all features with MDA (MDA).The colours of the area under the graphs are determined by the highest performing feature selection algorithm for a given number of features.

of features, and the *RFE* selection method is preferred for achieve a high performance using a small set of features.

5.4. Feature Selection Discussion and Conclusion

Five different features selection methods and two dimension reduction methods have been investigated. For MVG, the feature selection methods improve its accuracy greatly as it handles many dimensions poorly. The results of using feature selection methods can though be improved if the MVG classifier is used in the feature selection. The optimal classification accuracy for MVG is achieved with all features using MDA for dimension reduction. For kNN, the accuracy for different feature selection methods is close to similar, but the highest accuracy is achieved using the forward selection method which improves results by about 2 percentage points at its maximum, compared to the second highest methods.

Two classifiers using different subsets are selected for the classifier fusion which will be described in the following section, Chapter 6.

1. The kNN classifier using the 46, 183 and 12 highest ranking features for plants, cotyledons and foliages, respectively, determined by the forward selection method.
2. The MVG classifier using all features and the MDA dimension reduction procedure.

Chapter 6

Classifier Fusion

Until now the plant elements (cotyledon leaves, foliage leaves and whole plants) have been classifies individually. The aim now is to combine these classifications to determine the species of the plant when the elements of the plant have been classified as multiple species.

Combining the results of multiple classifiers to achieve an improved identification is in general terms described as classifier fusion defining a broad range of different methods. Creating a fusion between systems may be performed at different levels of abstraction namely; data level fusion, feature level fusion and classifier fusion[77]. This project will only treat classifier fusion as this method solely looks on the output of the classifiers without optimizing neither data segmentation nor the provided feature set. Classifier fusion also known as decision fusion or mixture of experts are generally divided in two groups [77]. One group do not treat the output of the classifier but seeks to select a single or a subset of classifiers to achieve the optimal identification. The classifier fusion will only use one classifier per plant element, allowing little space for improvement. The second group of classifier fusion, most relevant to the current problem, seeks to identify the true class only based on the result of the classifiers. This study investigate three types of classifier fusion techniques. First Bayesian Belief Networks is investigated, then Bayes Belief Integration (BBI) and finally a voting procedure that includes BBI for combinations of equal votes.

To handle the plants, the term *plant struct* is introduced. A plant struct may contain three different plant elements; the whole plant, single cotyledon leaves and single foliage leaves. After classification each plant struct contains a vector with the result for each plant element, defined as a combination.

6.1. Bayesian Belief Network (BBN) for species decision

In this section, a decision procedure based on Bayesian Belief Networks (BBN) is described. BBN are networks which describes probabilities of events given all cases from lower levels in the network[78]. The aim is to use this to estimate the probability of a species, given an observation of the whole plant and a series of its leaves.

As the size of the network will be quite large, a two species classification problem of sugar beet and cleavers will be walked through in this example, where cotyledons and foliages are grouped together as 'leaves'. The Bayesian network seen in Figure 6.1 illustrates the network for sugar beet, labelled S, where a similar network should be created for the cleavers, labelled C.
The top node with $S^\star$ indicates the true label for the plant. In this case, the true species is sugar beet, likewise, the true label for the other network would be $C^\star$.
The *a priori* probability of $S^\star$ is written as $P(S^\star)$ and is the naïve guess, which would be the fraction of sugar beet in the data set.
The second level of the network is the classification of the whole plant as either S, C or $\emptyset$. The class $\emptyset$ defines the case where leaves are detected but not connected to any plant, which could happen if for instance parts of the plant are overlapped by other plants, whereby the plant element have been discarded. Though this case will be ignored in this walk through.
An example of the probability for the whole plant classified as S given that the true species is $S^\star$ is written as $P(S|S^\star)$.
The third level of the network is the classification of a number of leaves as S. This will typically be a number between 0 and 4 in this data set. E.g. is the probability of zero leaves identified as S given that the true species is $S^\star$ and the whole plant has been classified as C written as $P(0S|S^\star, C)$.
The fourth level of the network is the classification of a number of leaves as C. This will also typically be a number between 0 and 4. The probability of two leaves identified as C given that the true species is $S^\star$, the whole plant has been classified as S and zero leaves has been classified as S is written as $P(4C|S^\star, S, 0S)$.

The aim is to find the posteriori probability for $S^\star$ and $C^\star$ given a detection of a plant with x leaves classified as S and y leaves classified as C.
If the whole plant has been classifies as S, the probability for $S^\star$ and $C^\star$ are respectively given by:

$$P(S^\star|S, xS, yC) \tag{6.1}$$

$$P(C^\star|S, xS, yC) \tag{6.2}$$

The rest of the derivation will be based on Eq. 6.1, but the procedure for Eq. 6.2 is the same. From Bayes rule we have that the conditional probability

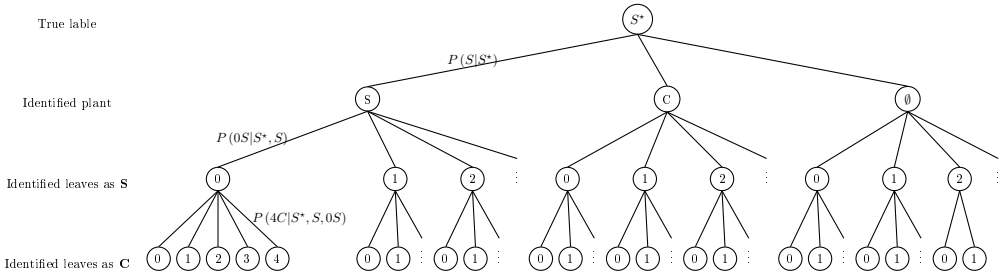


Figure 6.1: Bayesian Network

of A given B can be rewritten as the joint probability of A and B divided by the *a priori* probability of B:

$$P(A|B) = \frac{P(A,B)}{P(B)} \tag{6.3}$$

Hereby Eq. 6.1 can be written as:

$$P(S^\star|S, xS, yC) = \frac{P(S^\star, S, xS, xC)}{P(S, xS, yC)} \tag{6.4a}$$

$$= \frac{P(yC|S^\star, S, xS) \cdot P(S^\star, S, xS)}{\sum_{P^\star} P(P^\star, S, xS, yC)} \tag{6.4b}$$

$$= \frac{P(yC|S^\star, S, xS) \cdot P(xS|S^\star, S) \cdot P(S^\star, S)}{\sum_{P^\star} P(P^\star, S, xS, yC)} \tag{6.4c}$$

$$= \frac{P(yC|S^\star, S, xS) \cdot P(xS|S^\star, S) \cdot P(S|S^\star) \cdot P(S^\star)}{\sum_{P^\star} P(P^\star, S, xS, yC)} \tag{6.4d}$$

$$= \frac{P(yC|S^\star, S, xS) \cdot P(xS|S^\star, S) \cdot P(S|S^\star) \cdot P(S^\star)}{\sum_{P^\star} P(yC|P^\star, S, xS) \cdot P(xS|P^\star, S) \cdot P(S|P^\star) \cdot P(P^\star)} \tag{6.4e}$$

, where $P^\star$ is the different true labels, which in this example is sugar beet, $S^\star$, or cleavers, $C^\star$. When looking at Eq. 6.4e, it is seen that the denominator is constant for all species, and therefore can be left out. It is also seen that all factors can be read from nodes in the tree.
The problem now is to find the probabilities for the different nodes in the tree.

Finding the probabilities A problem with the Bayesian network is that the probabilities of the nodes should be learned from the data, which means that a very large data set is required for a large network[79]. For a general case with N species, where each plant can have between 0 and L leaves, the number of nodes of each of the N trees will be:

$$1 + N + N \sum_{n=1}^{N} (L+1)^n \tag{6.5}$$

For seven species this makes 683,593 nodes in each of the seven networks when assuming that each plant can have up to four leaves. By increasing the number of species by just one more, the number becomes 3,906,249 nodes, which means that the required dataset would be enormous and methods for estimating the probabilities must be found.

When finding the probabilities for each node, the "identified plant" level of the tree is the easiest to find, as this is the classification accuracy of each species on plant level. The probabilities of each of the sub-nodes is then the fraction of that outcome, divided by the total outcomes in that branch on that level.

Test of Bayesian belief network As stated earlier, the Bayesian network require a very large amount of samples to cover the highest levels of the tree. For that reason, a two-class network has been made and implemented in MATLAB to show the possibilities. The example has been made on Sugar beet and Scentless mayweed, where foliages have been excluded such that the plants only can have one kind of leaves.
For classification the kNN classifier has been used, with the optimal values for k, as described in section 4.2.1.

The *a priori* probability for the true species is set to 0.5 for both classes to thereby ignore prior knowledge of the distribution of plants.
Since this example uses only two classes, the initial classification accuracy is already high, where 15 of 900 plant samples were misclassified. After using this network, this number is reduced 12 of 900, which is a reduction of 20% of the errors[1].

6.2. Variations of Bayes Belief Integration (BBI)

The following section describes Bayes Belief Integration (BBI) using different settings.

[1] The MATLAB script is located in `Matlab/TwoStepClassifier/Demo_BayesianNetwork.m`

6.2.1 Bayes Belief Integration

The BBI is defined as a soft-output method as it uses fuzzy measures in the range [0,1] to define an evidence; probability, possibility, necessity, belief and plausibility[77]. In BBI the confusion matrix of each classifier is used to determine the belief for a given combination. A combination is defined as the classification result for each plant element in a struct. The confusion matrix PT_k of the classifier labelled k is defined as[2]:

$$PT_k = \begin{pmatrix} n_{11}^{(k)} & \cdots & n_{1j}^{(k)} & \cdots & n_{1M}^{(k)} \\ \vdots & \ddots & & & \vdots \\ n_{i1}^{(k)} & & n_{ij}^{(k)} & & n_{iM}^{(k)} \\ \vdots & & & \ddots & \vdots \\ n_{M1}^{(k)} & \cdots & n_{Mj}^{(k)} & \cdots & n_{MM}^{(k)} \end{pmatrix} \tag{6.6}$$

, where M is the number of classes and $n_{ij}^{(k)}$ is the number of times that samples with true class j have been classified as class i. In [80] the confusion matrix is used to create what is defined as a belief measure of correct assignments. The belief is found by determining the conditional probability of sample $\mathbf{x}$ being class j, when the classifier $e_k(\mathbf{x})$ outputs i.

$$Bel\left(\mathbf{x} \in c_j | e_k\left(\mathbf{x}\right)\right) = P\left(\mathbf{x} \in c_j | e_k\left(\mathbf{x}\right) = i_k\right) \tag{6.7}$$

The belief is defined as:

$$P\left(\mathbf{x} \in c_j | e_k\left(\mathbf{x}\right) = i_k\right) = \frac{{n_{ij}}^{(k)}}{\sum_{m=1}^{M} n_{im}} \tag{6.8}$$

In words Eq. 6.8 simply transforms the confusion matrix by dividing an entry $n_{ij}^{(k)}$ by the sum of all the entries in row (i) [3]. In Table 6.1 an example of a confusion matrix and a matrix of the belief measures are shown for a kNN classifer for plants. The belief matrix should be determined for both whole plants, cotyledon leaves and foliage leaves. The full example of using the kNN classifier is provided in Appendix M.

After the belief matrix have been determined for each plant elements, it is possible to combine the result of each classifier into a new belief measure

[2] The matrix looks slightly different from the original paper as the matrix has been transposed. The matrix is only of size $M \times M$ and not $M \times M+1$ as an extra class have been included in the paper allowing the classifier to identify a sample to no class.

[3] The belief matrix should NOT be confused with the accuracy matrix as an entry $n_{ij}^{(k)}$ is divided by the sum of all the entries in column (j).

		True label							
	Spe.	1	2	3	4	5	6	7	Tot.
Result label	1	242	0	2	0	0	0	0	244
	2	2	185	0	0	1	0	0	188
	3	3	8	316	1	0	1	4	333
	4	1	2	2	567	8	12	3	595
	5	1	3	2	2	691	9	2	710
	6	0	0	0	4	3	232	1	240
	7	1	0	3	1	0	1	120	126

		True label						
	Spe.	1	2	3	4	5	6	7
Result label	1	0.992	0	0.008	0	0	0	0
	2	0.011	0.984	0	0	0.005	0	0
	3	0.009	0.024	0.949	0.003	0	0.003	0.012
	4	0.002	0.003	0.003	0.953	0.013	0.020	0.005
	5	0.001	0.004	0.003	0.003	0.973	0.013	0.003
	6	0	0	0	0.017	0.013	0.967	0.004
	7	0.008	0	0.024	0.008	0	0.008	0.952

Table 6.1: Confusion matrix and the belief matrix for a kNN classifier used on plants ($k = 1$).

Bel_{class} for each class j.

$$Bel_{class}(j) = \frac{P(\mathbf{x} \in c_j|EN)}{\prod_{\forall k \in classifiers} P(\mathbf{x} \in c_j|EN)} \prod_{\forall k \in classifiers} P(\mathbf{x} \in c_j|e_k(\mathbf{x}) = i_k) \tag{6.9}$$

, where $classifiers$ is a list of all classifiers to be used; one classifier for each plant element in the struct.
The sample $\mathbf{x}$ is identified as the class j achieving the highest belief $Bel_{class}(j)$ for classes $j = 1, ..., M$.

$$c(\mathbf{x}) = \arg\max_{1 \leq j \leq M} (Bel_{class}(j)) \tag{6.10}$$

EN describes the environment from which the samples are drawn. By using the belief matrix defined in Table 6.1 changes towards classes with a high number of samples are more likely to happen than changes to classes with a low number of samples if the classification errors for the classes is identical. If the sample distribution used to make the belief matrix does not represent the real distribution of the environment from which the test samples are drawn, this bias will certainly change the outcome to the worse. Therefore a weighting of the belief can be used as described in [81]. However, as the belief matrix in this project is based on the same distribution as the test samples, this weighting is not used. The first term in Eq. 6.9, is not achievable from hard assignment classifiers like kNN or SVM. Thereby 6.9 can be changed to a simplified belief term:

$$Bel_{class}(j) = \prod_{\forall k \in classifiers} P(\mathbf{x} \in c_j|e_k(\mathbf{x}) = i_k) \tag{6.11}$$

In this project three different types of classifier are used; one for each type of plant element. To easily distinguish between the three classifiers, they are defined as $k = 0, 100, 200$ for the plant, cotyledon and the foliage classifier, respectively.

In Figure 6.2 the output e_k of different classifiers are handled by the BBI. The first struct contains only one plant element/one classifier, whereby

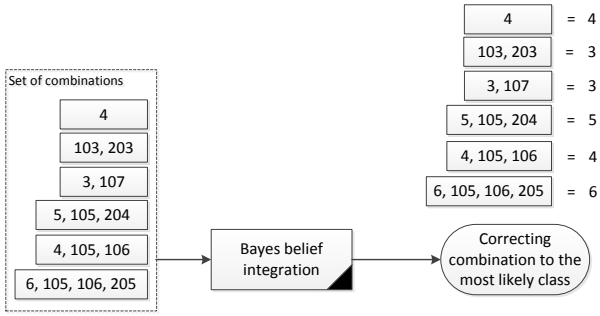


Figure 6.2: Decision procedure based on likelihood.

there are no reason to use classifier fusion. The result is simply the output of e_1 (the plant classifier) $c(\mathbf{x}) = e(\mathbf{x})_1 = 4$. The second struct contains non-conflicting classifications as $c(\mathbf{x}) = e(\mathbf{x})_{100} = e(\mathbf{x})_{200} = 3$ needing no classifier fusion. The third, fourth, fifth and sixth example shows conflicting output justifying the use of BBI. An example is provided for the fifth struct with the output $e_0 = 4, e_{100} = 105, e_{100} = 106$ using three classifiers $k = 0, 100, 100$. The belief has to be calculated for all classes $j = 1, \ldots, 7$, but the belief measure of class $j = 4, 5, 6$ is only presented assuming that they are the most likely classes. The calculations are based on the numbers in the belief matrices provided in Appendix M.

$$Bel_{class}(4) = \prod_{\forall k \in classifiers} P(\mathbf{x} \in c_4 | e_k(\mathbf{x}) = i_k) \tag{6.12a}$$
$$= P(\mathbf{x} \in c_4 | e_0(\mathbf{x}) = 4) \cdot P(\mathbf{x} \in c_4 | e_{100}(\mathbf{x}) = 105) \cdot P(\mathbf{x} \in c_4 | e_{100}(\mathbf{x}) = 106)$$
$$= 0.953 \cdot 0.020 \cdot 0.134 = \underline{\underline{2.51 \cdot 10^{-3}}}$$
$$Bel_{class}(5) = \prod_{\forall k \in classifiers} P(\mathbf{x} \in c_5 | e_k(\mathbf{x}) = i_k) \tag{6.12b}$$
$$= P(\mathbf{x} \in c_5 | e_0(\mathbf{x}) = 4) \cdot P(\mathbf{x} \in c_5 | e_{100}(\mathbf{x}) = 105) \cdot P(\mathbf{x} \in c_5 | e_{100}(\mathbf{x}) = 106)$$
$$= 0.013 \cdot 0.931 \cdot 0.067 = \underline{\underline{8.42 \cdot 10^{-4}}}$$
$$Bel_{class}(6) = \prod_{\forall k \in classifiers} P(\mathbf{x} \in c_6 | e_k(\mathbf{x}) = i_k) \tag{6.12c}$$
$$= P(\mathbf{x} \in c_6 | e_0(\mathbf{x}) = 4) \cdot P(\mathbf{x} \in c_6 | e_{100}(\mathbf{x}) = 105) \cdot P(\mathbf{x} \in c_6 | e_{100}(\mathbf{x}) = 106)$$
$$= 0.020 \cdot 0.033 \cdot 0.745 = \underline{\underline{5.03 \cdot 10^{-4}}}$$
$$\tag{6.12d}$$

The fourth class achieves the highest belief and the struct is therefore identified as this class.

6.2.2 Combing voting and BBI

Another approach have been provided for classifier fusion based on voting. The principle of voting is that each classifier has a single vote. The class with the highest number of votes is simply selected as the winning class. In a case where multiple classes are ranked evenly high, the BBI model steps in to determine the winning class. In Figure 6.3 a small decision tree shows how six plant structs are processed. The combination of struct 1, 2 and 4 can all be determined by voting as a single class gets the most votes. The combination of struct 3, 5 and 6 ends up having multiple classes with the same number of votes and are handled by BBI.

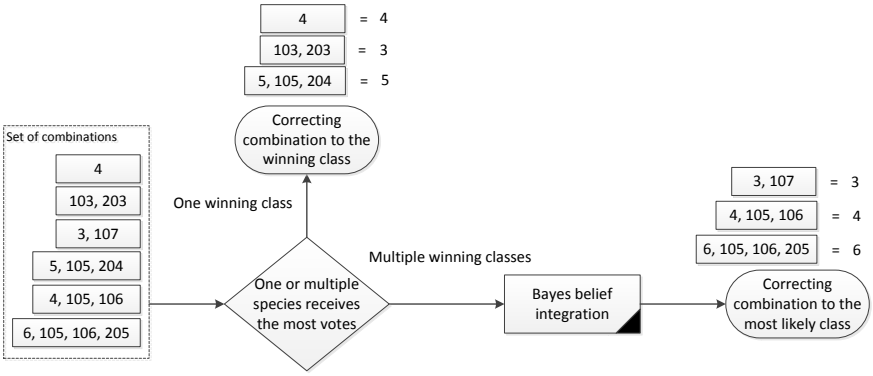


Figure 6.3: Decision procedure based on voting.

6.2.3 Settings for BBI

Multiple "Settings" have been provided to investigate different variations of classifier fusion .

Voting The *voting* settings determines if voting should be used together with BBI, basically selecting one of the two classifier fusion method described in section 6.2.1 and 6.2.2.

UseBelief The *useBelief* setting changes the classifier fusion method in a more fundamental way as it sets the classifier to either use the belief matrix or the likelihood of each class provided by the classifier. Disabling the *useBelief* setting is only possible for the MVG classifier as kNN and SVM

	Class						
Combination	**1**	**2**	**3**	**4**	**5**	**6**	**7**
4	0	0	0	**0.993**	0.001	0.006	0
105	0.173	0	0.048	0.180	**0.362**	0.195	0.041
106	0	0	0	0.104	0.001	**0.895**	0
Likelihood	0	0	0	**0.0186**	0	0.0011	0

	Class						
Combination	**1**	**2**	**3**	**4**	**5**	**6**	**7**
4	0.007	0.007	0.007	**0.993**	0.007	0.007	0.007
105	0.638	0.638	0.638	0.638	**0.362**	0.638	0.638
106	0.105	0.105	0.105	0.105	0.105	**0.895**	0.105
Likelihood	0.000	0.000	0.000	**0.067**	0.000	0.004	0.000

Table 6.2: Shows the likelihood of the MVG classifier for samples of 4, 105 and 106. Two settings are provided using likelihood; top one uses the likelihood for each class and the bottom one uses only the value of the most likely class, the likelihood of the other classes is the value 1 subtracted by the most likely value

do not provide such measures. As described in Section 4.2.3 the classifier returns a likelihood for a given class, that can be used as a fuzzy measure in the same way as the belief in Eq. 6.11. One fundamental difference of using the likelihood of a classifier is that it changes for every new sample, while the belief matrix remains the same (after training) for all samples. The advantage of this is that a struct is identified based on the likelihood of each classifier for a given sample and not the general belief for such a sample. An example using struct 5, where $e_0 = 4, e_{100} = 105, e_{100} = 106$ is presented[4] in Table 6.2. By Eq. 6.11 the belief of class j is determined by the product of all the values in each column j in Table 6.2. Two tables are provided, the first shows a setting, where the likelihood of all classes are used, the second setting uses only the value of the most likely class, the likelihood of the other classes are the value of 1 subtracted by the most likely value (likelihoodSimple).

Leaves Only Count Once The final setting removes identical classification of leaves, making identical leaves only count as one. E.g. meaning that [3 101 101 103 207 207] = [3 101 103 207].

[4] Actually there is no combination within the dataset having this combination, but the values of actual individual classifications of 4, 105 and 106 have been used

6.2.4 Result of BBI

To evaluate the different fusion methods, two subset found in Chapter 5 are used. The kNN classifier is used without MDA selecting the 36, 183 and 12 highest ranking features of Forward Selection for respectively plants, cotyledons and foliage leaves. As also likelihood must be evaluated, all features using the MDA and the MVG classifier, is used.

Result of Classifier Fusion with MVG The different fusion methods and settings using the MVG classifier with MDA is seen in Table 6.3[5].

	Settings				
#	CountLeavesOnces	Voting	Belief (Accuracy)	Likelihood (Accuracy)	LikelihoodSimple (Accuracy)
1	0	0	0.917	0.897	0.900
2	0	1	0.882	0.851	0.854
3	1	0	0.937	0.901	0.905
4	1	1	0.929	0.897	0.898

Table 6.3: The accuracy of different classifier fusion settings using the MVG classifier.

The settings from Table 6.3 is prioritized as the fusion works better, when the setting *CountLeavesOnce* is enabled and the setting *Voting* is disabled. The *Belief* settings performs for the same setting better than the two *Likelihood* based settings. The *Likelihood* setting should therefore not be prioritizing. The two likelihood based settings generally performs very similar achieving a maximum identification accuracy of 0.897 and 0.900 for setting 1.

After a struct has been determined as a certain class, the conflicting elements are corrected to the class identified by the classifier fusion. To show the improvement of classifier fusion the results of the MVG classifier with the optimal setting (setting 3) using belief is presented in Table 6.4 showing the accuracy of a struct and each plant element before and after correction.
The fusion will not improve plants significantly as the leaves performs worse in general. However, the classification accuracy of cotyledons and foliages are improved remarkably by, respectively, 15.47 and 20.2 percentage points, thereby improving the total classification accuracy by 11.16 percentage points.

The confusion and the accuracy matrix is presented in Table 6.5.
This table shows that Maize (1) is classified with a high accuracy of 0.976,

[5] A test and training set is picked at random when training a classifier, meaning that the result changes slightly from run to run. The result of each setting is therefore determined as the average value of 20 runs each run doing 10 cross validations.

	nElements	Accuracy Before	Accuracy After
Plant	**2436**	0.949	0.949
Cotyledon	**3409**	0.802	0.957
Foliage	**1358**	0.712	0.914
All	**7203**	0.835	0.946
Struct	**2813**	N/D	0.937

Table 6.4: Accuracy before and after fusion MVG.

		True label						
	Spe.	1	2	3	4	5	6	7
Result label	1	245	3	7	0	1	0	5
	2	0	186	9	1	3	1	2
	3	4	3	432	1	1	2	4
	4	1	2	2	553	9	20	13
	5	0	2	5	2	682	17	5
	6	0	1	0	14	8	228	12
	7	0	1	4	6	2	6	299

		True label						
	Spe.	1	2	3	4	5	6	7
Result label	1	0.976	0.015	0.015	0	0.001	0	0.014
	2	0	0.930	0.019	0.002	0.004	0.004	0.006
	3	0.016	0.015	0.935	0.002	0.001	0.007	0.012
	4	0.004	0.010	0.004	0.952	0.013	0.071	0.037
	5	0	0.010	0.011	0.003	0.959	0.061	0.014
	6	0	0.005	0	0.024	0.011	0.814	0.035
	7	0	0.005	0.009	0.010	0.003	0.021	0.862

Table 6.5: The confusion and accuracy matrix.

while Shepherd's-purse (6) and Cleavers(7) performs the worst with an accuracy of 0.814 and 0.862, respectively.

Result of Classifier Fusion with kNN The following section presents the final results of the system using, what has been found, to be the optimal classifier and feature subset. First the different classifier fusion settings of kNN is shown in Table 6.6[6].

#	Settings CountLeavesOnce	Voting	Belief (Accuracy)
1	0	0	0.9406
2	0	1	0.9312
3	1	0	0.9575
4	1	1	0.9560

Table 6.6: The accuracy of different classifier fusion settings using the kNN classifier.

[6] A test and training set is picked at random when training a classifier, meaning that the result changes slightly from run to run. The result of each setting is therefore determined as the average value of 20 runs each run doing 10 cross validations.

Using the optimal setting (setting = 5) the improvements of classifier fusion is presented in Table 6.7 showing the accuracy of a struct and each plant element before and after correction.

	nElements	Accuracy Before	Accuracy After
Plant	**2436**	0.966	0.964
Cotyledon	**3409**	0.891	0.968
Foliage	**1358**	0.847	0.944
All	**7203**	0.908	0.962
Struct	**2813**	**N/D**	0.958

Table 6.7: Accuracy before and after fusion using kNN.

The classification accuracy after fusion have degraded slightly for plants, but improves cotyledons and foliage leaves of 7.7 and 9.7 percentage points. The total classification accuracy of all plant elements achieves an impressive weighted rate of 0.962. The structs have an identification accuracy of 0.958 showing the actually and final system performance. The confusion and the accuracy matrix is presented in Table 6.8

This table shows that Sugar beet (3), Scentless mayweed (4) and Chickweed

		True label						
	Spe.	1	2	3	4	5	6	7
Result label	1	242	1	3	0	0	0	3
	2	2	184	0	0	0	0	0
	3	3	7	450	1	0	1	6
	4	1	2	4	568	7	19	6
	5	1	3	2	1	691	13	4
	6	0	0	0	6	8	241	13
	7	1	1	0	1	0	0	308

		True label						
	Spe.	1	2	3	4	5	6	7
Result label	1	0.964	0.005	0.006	0	0	0	0.009
	2	0.008	0.920	0	0	0	0	0
	3	0.012	0.035	0.974	0.002	0	0.004	0.017
	4	0.004	0.010	0.009	0.978	0.010	0.068	0.017
	5	0.004	0.015	0.004	0.002	0.972	0.046	0.012
	6	0	0	0	0.010	0.011	0.861	0.037
	7	0.004	0.005	0	0.002	0	0	0.888

Table 6.8: The confusion and accuracy matrix.

(5) are classified with a high accuracy of above 0.97, while Shepherd's-purse (6) and Cleavers(7) performs the worst with an accuracy of 0.861 and 0.888, respectively.

Five examples of wrong classifications are shown for each of the seven species in Appendix N[7] to somehow present when the classifier fails. The appendix shows how most of the wrong classification are either plants that have been labelled incorrectly, non-ideal in shape or have rough edges due

[7] Images of all wrong classifications are provided with the disc that came with the report

to bad segmentation.

The classifier fusion have improved the total classification accuracy from 0.908 to 0.962 for the plant struct, but another essential improvement of using classifier fusion for plants is that more structs are enabled for identification. A procedure only classifying whole plants will identify poorly the 377 (13.4%) structs that only contains leaves and a procedure only classifying leaves will fail to identify the 705 (25%) structs, where leaves could not be extracted from the plant.

The classification accuracies have already been optimized leaving little space for improvement. Using the proposed resampling distance transform (DistTransformRS_Sort) and the kNN classifier. The classification accuracies before and after fusion is shown in Table 6.9.

The example shows that the classifier fusion can provide larger improve-

		Accuracy	
	nElements	**Before**	**After**
Plant	**2436**	0.663	0.704
Cotyledon	**3409**	0.754	0.812
Foliage	**1358**	0.609	0.725
All	**7203**	0.696	0.759
Struct	**2813**	**N/D**	0.717

Table 6.9: Classification accuracy before and after classifier fusion using the kNN classifier with the 10 subfeatures of the feature descriptor DistTransformRS_Sort .

ments in a case where the classification accuracies is not as high. The classification accuracies improve with 4.16, 5.71 and 11.6 percentage points for plants, cotyledons and foliage, respectively. An improvement of 5.48 percentage points is also seen from plant without leaves to plants with leaves (struct).

6.3. Classifier Fusion Discussion and Conclusion

The BBN classifier fusion method provides a more complex model-based procedure achieving promising results for two species. However, the network has a great disadvantage in that the tree structure growths exponential for an increasing number of species. Treating just seven species would require an unrealistic amount of data. This ultimately makes the BBN classifier undesirable for plant identification, as soon as one has a medium to large number of species to deal with.

The Bayes Belief Integration classifier fusion method has been described and applied to the data of this project using a range of different settings. The Bayes Belief Integration method enables not just a fusion of the classifiers used for leaves and plants, it also succeeds in correcting conflicting classifiers and achieving a higher classification accuracy. After fusion the whole plant, single cotyledon and single foliage achieve a classification accuracy of, respectively, 0.964, 0.968 and 0.944 and improving with -0.02, 7.66 and 9.66 percentage points. Ultimately the project achieves a plant (struct) identification accuracy of 0.958.

Chapter 7

Results and discussion

The following chapter will be a summary of the main results achieved in the different parts of the project. This chapter will also discuss results in relation to the use of plant recognition for weed control.

During the segmentation phase, plants have been extracted from the background gravel. This have been done using the difference of excess green and excess red, which has proven to cast off good results. However, under different lighting conditions or with a different background, the background may not be removed as efficiently, for which reason optimal colour weights have been investigated and a cost function has been made to optimize for various colour segmenting cases.

All plants have been inspected manually in order to avoid overlapping plants by manually segmenting these into single plants. This manual action has been valuable to ensure a large data set within a short period of time. However, the manual input has to be replaced by an automated process in the future, so as to make the procedure valuable for weed control. All plants have been sown under controlled conditions, whereby the quality of samples is generally high. In the field, however, there will be a larger tendency for plants to be misshapen, either because of difference in lighting and nutrients or simply because plants may have lost leaves. Therefore, the results must be seen in the light of these circumstances. The plant segmentations has provided 2813 plant samples distributed on seven species, each covering multiple growth stages.

One of the goals of this project has been to improve identification by combining the classification of leaves and whole plants. It has therefore been necessary to find ways to automatically detect leaves on plants and cut them off. Two methods have been analysed and implemented: The first

method is based on the leaf's colour and the distribution of pixels, whereas the second method is based on the contour of the plant. The first method has the advantage that it can operate on overlapping leaves, but it is slow and only able to deal with convex leaves; a feature which excludes many foliages. The latter method is fast and because of modifications of the *Plant Stem Emerging Point* algorithm, this method has been extended to handle many different leaf shapes, though it cannot handle overlapping leaves. Furthermore, this method has been used to automatically cut off leaves, so that these can be classified. Nevertheless, it has been necessary to conduct a manual inspection so as to ensure that the leaves that have been found, in fact, really are leaves. Furthermore, all cut-off leaves have been ranked according to how ideal they appear in the images, and the growth stages of the leaves have been estimated. This information has been stored, but it has not been put to use in this project. Moreover, the leaves are manually labelled as either foliages or cotyledons, which has helped to eliminate some variance in the dataset to ease the classification. In the future, however, such labelling will not be possible and all leaves should be gathered in the same pool.
After the leaf segmentation, a total of 3409 cotyledons and 1358 foliages has been extracted from the plants.

Based on the plant and leaf samples, 50 features descriptors have been developed and implemented providing in total 261 feature values. Of these features, the new variations of the Distance Transform [17] has been developed and proven to be the best single feature, apart from the colour features.
Similarly, a feature has been derived from the elliptic Fourier transform, which is able to describe the indentations of leaves. A third new feature is the stem thickness estimating feature, which has been constructed so that it discriminates plants with different stem thickness. By looking at the forward feature selection and the recursive feature elimination, it is seen that this feature does not help discriminating the plants significantly when combining it with the other features in the feature set.

Three classifiers have been used to identify plants and leaves; the MVG, the kNN and the SVM classifier. The MVG works poorly for high dimensional spaces and handles best a single clustered, Gaussian distributed class. The advantage of the MVG is that it provides a likelihood measure for each class improving the classifier fusion. The kNN classifier generally performs the best as it handles many features with multiple clusters of arbitrary distribution. A minor drawback of the kNN classifier is that it favours classes with a high number of samples. The SVM classifier is not preferred as is requires a time consuming process of optimization the kernel for a given dataset. However, when optimized, it perform very well.

In order to achieve the highest accuracy possible, five different features

selection methods and two dimension reduction methods have been applied. The five feature selection methods are; *genetic algorithm*, *individual feature performance*, *forward selection*, *recursive feature elimination* and *recursive feature elimination using MDA*. The forward selection method performs the best for the k-nearest neighbour classifier by using respectively 36, 183 and 91 features for plants, cotyledon and foliage leaves. The k-nearest neighbour classifier achieves the highest classification accuracies with 96.7%, 89.0% and 88.3% for whole plants, cotyledons and foliages, respectively. The two feature dimension reduction methods, PCA and MDA, have also been applied. Both methods seek to find a linear combination of all features to a lower the dimensional space, but as MDA keeps discrimination between classes, it achieves better classification accuracies. The drawback of MDA is that it shrinks the information to e.g. a six dimensions feature space for seven species potentially removing some discriminating information. MDA provides the best reduction of features for the kNN classifier achieving an accuracy of 95.3%, 83.2% and 77.2% for whole plants, cotyledons and foliages, respectively.

As previously mentioned, one of the goals of the project has been to test whether classification of plants could be improved by handling plants as both whole plants as well as individual leaves. A method to combine the classifications of plants and leaves using Bayesian networks has been described and a two-species example has been implemented. Even though this method has provided an increased overall classification accuracy, it is not suitable for the multi-species problem, as the training data required increase exponentially with the number of species.
Another method for combining the classification of leaves with the classification of whole plants is Bayes Belief Integrator. This method have shown promising results. The classification accuracy for plants is almost unchanged, whereas the classification rate of the leaves increases by 7.7 and 9.7 percentage points for cotyledons and foliages by combining the classification of leaves and plants. An example on classifier fusion using only the new proposed variation of the Distance Transform improves the classification accuracies from 66.3% to 70.4% for plants.

This result in the overall classification of leaves and whole plants combined, should be compared to others in the field of plant recognition.

T. M. Giselsson, in his study, achieves an accuracy of 94.8% for 1700 plant samples of eight species, where each species is only represented at one growth stage. [16].

B. Åstrand et al. have classified sugar beet in a field. They are able to distinguish sugar beet from weeds by using only colour features whereby they achieve an accuracy of 91%[27].

For leaves, D. J. Hearn has achieved an accuracy of 72% for 2420 leaves

Maize	Wheat	Sugar beet	Scentless mayweed	Chickweed	Shepherd's-purse	Cleavers
96.4%	92.0%	97.4%	97.8%	97.2%	86.1%	88.8%

Table 7.1: Final Result by combining plant and leaf classification

from 151 different species, by using only Fourier descriptors [19]. Hearn uses images acquired using a digital scanner.

JX. Du et al. achieve an accuracy of 93.6% for 1800 leaves divided over 25 species[20].

When viewing this report in the light of these results, the overall classification accuracy of 95.8% for 2813 plant samples, found in this project, distributed on seven species and different grown stages is a result that can compete with existing research in the field of plant recognition.
However, as each of these studies are based on different databases, the results can hardly be compared. It is therefore worth considering, whether this project's database should be made available for the public to access and contribute to, so that future studies can be based on comparable data.

The final result for the seven species is presented in Table 7.1. Some of the examples from the wrong classifications have been extracted and placed in Appendix N[1]. From these examples, it is seen that some misclassifications are simply the result of wrong labelling and non-ideal plants. Other misclassifications are due to an un-sharp segmentation, resulting in a noisy edge or plant mask.

[1] images of all wrong classifications is provided on the disc that comes with the report

Chapter 8

Future work

The image database used in this project has been very valuable as it has provided a good foundation for the work undertaken here. However, the database contains five more species, which have not been included in the tests. These species are *Charlock*, *Fat Hen*, *Cranesbill*, *Black-grass* and *Loose Silky-bent*.
Including these plants would be a manageable task, due to the segmentation tools developed in this project, although it will probably be necessary to adjust the features descriptors or make new ones to handle these species.

For instance *Black grass* and *Loose Silky-bent* have many of the same characteristics as *Winter Wheat*, wherefore they are likely to be confused in the current system.

Except from the plants, that have now been segmented, the image database also contains plants at higher growth stages. These higher-stage plants have the side-effect of overlapping each other. Such behaviour should also be expected in the field, for which reason the system should be able to handle these cases. Still, as the classification procedure is based on single plants and single leaves, classification methods for overlapping plants are yet to be found.
A proposal for a fully automated system as e.g. a weeding machine is ready for further development. The proposal is shown in Figure 8.1, where parts implemented in the current system have been marked green. The first step of plant recognition is to extract connected, green objects from the images similar to the colour segmentation use in this project. Based on the experience from the database, these green connected objects could both be a single leaves, single plants and overlapping plants. The next step introduces an extra classifier step, *Leaf/Growth stage estimation*, that determines if the

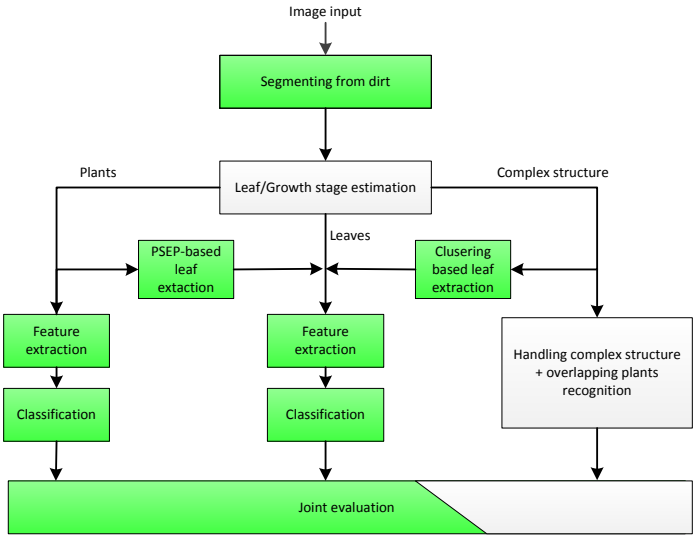


Figure 8.1: Overview of further development

element is a *leaf*, a *simple plant* (plants having up to four leaves), or *overlapping/complex plants* with more than four leaves. Simple plants, leaves and overlapping/complex plants are then classified individually, and each object is thereby determined in a two-step classification. The result of each classification is then joint by using classifier fusion in the final step. In general, a two-step classifier is not preferable, unless, as in this case, each object is treated differently. If the elements were to be treated equally, another single classifier would have been preferable. A procedure for handling small, non-overlapping plants, where the plants species decision is based on both whole plants and its leaves, has been described in this project. However, the system is not able to handle more complex structures such as plants at higher growth stages or overlapping plants. The reason for this is that the complexity of these objects are not preferably handled by shape and contour features. Because of the complexity of segmenting overlapping plants, the species classification of those plants should be based primarily on the appearance of the mess of plants rather than on individual plants and leaves. A clustering based method for finding whole leaves from overlapping plants has been described in this project, which, in combination with the leaf features, could help when classifying the mess. When it is not possible to extract leaves by using this leaf extracting method, other recognition methods must be used. One such method that could be investigated is the bag-of-words model, where the images are divided into small patches, which are described using a set of Scale-invariant feature transform (SIFT) descriptors. These

descriptors are then held up against a codebook of known classes, where the distribution of matches should reflect the distribution of species in the image.

To shortly demonstrate the possibility of a fully automated system, the elements of the seven species have been labelled as either a simple plant, a leaf and overlapping/complex plants. A test determines classification accuracies in Table 8.1 by using all features and the k nearest neighbour classifier.

Plant objects	Classification Accuracy
Leaf	0.91
Simple Plant	0.93
Overlapping/Complex Plant	0.67

Table 8.1: The classification accuracies for leaves, simple plants and overlapping/complex plants.

These results are not optimized, but albeit this, they still provide good classification accuracies of 0.91, 0.93 and 0.67. Furthermore these classification are not strict, meaning that a simple plant can be handled as an overlapping/Complex plants and visa versa.

To decrease some of the variation of leaves within a plant species, a method to straighten bending leaves is proposed. Even so, the method needs to be optimized in order to handle more irregular leaf shapes before it is suitable for an automated process.

When looking at the classification results, it is remarkable that the classification accuracy of whole plants is as high as it actually is. This is partly due to the data set, which is made under controlled conditions at Flakkebjerg. This means that defects, such as loss of leaves, are not present in the data set. Thereby, this fact makes the plant samples more similar and thus increases the accuracy of their classification. The natural next step would therefore be to include non-ideal samples into the data in order to see how this affect the method's ability to classify the species. This development would likely decrease the classification accuracy of whole plants, but as the final decision of the system on which species, a plant belongs to, is based on the classification of both whole plants and their leaves, it would be interesting to investigate if the classifier fusion would be able to handle this. In the first place, this could simply be done by manipulating the images, so that an expansion of dataset could be avoided. Later on, however, images from the field should be included to investigate the robustness of the system against the variations given by light and different nutrients, which might decrease the good discrimination power, that the colour features provid.

None of the features, that have been tested in this project, are based on texture. This is primarily due to very flat appearance at the lower growth-stages of the selected species. Still, texture based features should be reconsidered when adding new species to the dataset.

A problem with the results from this project is that they are hardly comparable with other studies, as the database is different from the databases used by others. A way to move the field of plant recognition forward, would be to make a common database, which preferably could be open for others to use and to add additional samples. By doing this, methods for plant recognition could be tested under the same conditions and provide comparable results for different plant recognition features, classifiers and other methods.

Chapter 9

Conclusion

The objective of this project has been to investigate methods and procedures for implementing a computer vision based system for plant identification. A challenge when using pattern recognition on plants is the variation within the same species at different growth stages. The project has successfully applied and investigated different measures in order to minimize the variation between plants and thereby ultimately achieve higher identification accuracy.

Segmentation The plant identification is based on a database of 12 species providing images of the plants in three to eleven different growth stages according to the BBCH index. A segmentation tool has been provided for extracting plants and leaves automatically from images supported by a manual labelling of the species. Plants are extracted from images by a colour segmentation method. Three approaches using different colour transformation have been applied; HSI, ExG and ExG-ExR. The ExG-ExR method is chosen as the best procedure as it keeps related plant pieces together. Furthermore, three general colour segmentation methods have been proposed.

Two methods have been investigated to extract leaves from the plant. The first method is a modification of the *plant stem emerging point*-algorithm using the contour of plants to extract leaves. The modification of the algorithm even makes it possible to handle leaves with complicated contours such as foliage leaves. The other method finds leaves by creating and connecting clusters using the Gustafson-Kessel C-means algorithm in combination with the genetic algorithm. This method is able to extract leaves from overlapping plants, but as the algorithm prefer convex leaves, the plant stem estimation algorithm algorithm is used for the further processing.
The shape characteristics of a leaf within the same species have some variations due to bending leaves; either for natural reason or because of a skewed camera angle. A leaf straightening algorithm is therefore proposed having

the purpose of reducing the variance of bending. Though the procedure seems promising, some modifications are needed for it to handle more leaf shapes.

Features A broad range of 50 features and 261 subfeatures have been implemented covering both shape, contour and colour features. These features include both common features like object area, solidity and eccentricity, but also more complex features such as e.g. fractal dimension, distance transform based features and a rotational invariant elliptic Fourier descriptor. The project proposes multiple features including a partly scale invariant variation of the distance transform that performs better than other distance transform features. The elliptic Fourier distance features provide different statistics of distances between adjacent elliptic Fourier approximations thereby increasing discrimination between classes.

Classifier Three classifiers have been used to identify plants and leaves. These are the *k-nearest neighbour* classifier, the *multivariate Gaussian classifier* and the *support vector machine* classifier. The *k-nearest neighbour* classifier generally performs the best as it handles high dimensional feature spaces with multiple clusters of arbitrary distribution. The multivariate Gaussian classifier works poorly for high dimension spaces and cannot handle distributed classes. The advantage of the multivariate Gaussian classifier is that it gives a likelihood estimation for each class that describes the certainty of its choices.

Feature selection and dimension reduction In order to achieve the highest accuracy possible, five different features selection methods and two dimension reduction methods have been applied. By using these, the number of features necessary to achieve the maximum classification is 34, 49 and 15 for whole plants, cotyledons and foliages, respectively. Two feature dimension reduction methods, *principal component analysis* and *multidiscriminant analysis*, have also been applied. The *multidiscriminant analysis* performs the best as it keeps discrimination between classes, and achieves the best result for the multivariate Gaussian classifier.

Classifier fusion The final step combines the classification of whole plants, cotyledon and foliage leaves in a joint identification of the species by using Bayes belief integration, but the Bayesian belief network has also been investigated.

After classifier fusion the system finally achieves a plant identification accuracy of 95.8% for seven species.

The work of this project has investigated and implemented a range of different procedures for plant identification using RGB images in all steps of

a recognition procedure. Apart from joining the work of others in the field of plant recognition and pattern recognition in general, new methods and improvements to existing methods have been proposed. The overall system proposes a procedure for extracting plants and leaves in order to exploit the discriminating power of both. This procedure aims at finally combining the two for an improved identification, proving that a computer vision based system can be used to identify plant species with high identification accuracy.

Appendices

Appendix A

Documentation of two Colour segmenation methods

Two of the three proposed colour segmentation procedures including limitations of the procedures are provided in this appendix.

Limitations of the colour segmentation method The general procedure of projecting data unto a line have some limitations as the RGB colour space is projected with a linear projection onto a line and thresholded in only two classes; foreground (wanted) and background (unwanted) elements. The limitation of only discriminating colours in only two classes involves a problem. Imagine that a low saturation of green is selected as foreground while a high saturation of green and all other colours are selected as background. The RGB colour space can - in this case - impossibly be separated in fore- and background with only a single threshold as this would require three classes. These methods can therefore only segment the highest saturation of some colour to a desired lower saturation level and not opposite. Another limitation of the linear projection is that the methods only works well in the corners of the RGB colour space; red, green, blue, cyan, magenta, yellow, white and black. These limitations are though also the major strength of these methods as - with the ExG-ExR - they segments the different shades of a specified colour providing a more robust segmentation in different lighting. The advantage of the three methods - in comparison to the ExG-ExR method - is though that adjustments can be performed, allowing a different colour than green to be segmented and allowing a more strict or less strict colour segmentation.

The second method uses a cost function related to Eq. 2.8 used in the

first method, but includes (> 0) in the equation.

$$J(r,g,b) = \left\| \begin{bmatrix} \mathbf{R} \\ \mathbf{G} \\ \mathbf{B} \end{bmatrix}^T \begin{bmatrix} r \\ g \\ b \end{bmatrix} > 0 - \mathbf{OSM} \right\|^2 \tag{A.1}$$

The added part of the cost function provides a more correct solution as the used segmentation is performed by thresholding positive values. The cost function must though be optimized in a fundamentally different way as it becomes non-linear. To solve a non-linear problem like this analytically is very hard (if not impossible), and the problem is solved using a numerical optimization method.
The equation below will be true for all positive values of a. SM is the segmentation mask.

$$\begin{bmatrix} \mathbf{R} \\ \mathbf{G} \\ \mathbf{B} \end{bmatrix}^T \begin{bmatrix} a \cdot r \\ a \cdot g \\ a \cdot b \end{bmatrix} > 0 = SM \tag{A.2}$$

The component g is therefore set to 1 to avoid this.

$$J(r,g,b) = \left\| \begin{bmatrix} \mathbf{R} \\ \mathbf{G} \\ \mathbf{B} \end{bmatrix}^T \begin{bmatrix} r \\ 1 \\ b \end{bmatrix} > 0 - \mathbf{OSM} \right\|^2 \tag{A.3}$$

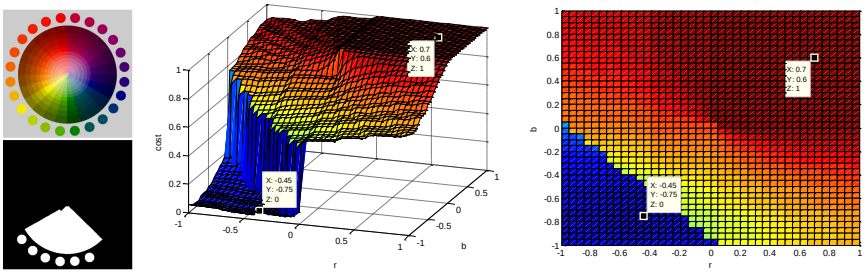
The cost function is now only dependent on r and b, and the feasible set in the optimization problem is reduced from a 3d space to a 2d space. In Figure A.1b the cost of different r and b values have been plotted. The image is seen in Figure A.1a.

Further research can be performed in finding a good optimization method, but to focus on more relevant parts of the project a simple brute force optimization is used by calculating the cost function for a set of r and b values. The method has been implemented in MATLAB[1].

A selection of possible methods is though provided. A gradient method[82, ch. 8] (either a fixed-step-size gradient or a steepest descent including a 1D search) would be fast, but requires the function to be convex - only a single minimum - and to contain no flat surfaces . A gradient function therefore falls on both criteria as flat areas are presented in the graph and multiple local minima are obtain when red/blue are mistakenly segmented in the OSM[2]. An optimization method called simulated annealing [82, ch. 14.3]

[1] The MATLAB script is located in:
`Matlab/Segmenting/SegmentingTool/NewColorSegment/ColorSegmentationNonLinear.m`

[2] The gradient method might though be relevant as the max and min in the not thresholded image can be used in flat areas and because large section of other than green must be selected to add addition local minima.



(a) RGB image and OSM mask.

(b) The cost for multiple values of r and b, $g = 1$ in a 3d plot. Min and max are marked.

(c) The cost for multiple values of r and b, $g = 1$ in a surface plot. Min and max are marked.

Figure A.1: The cost for different values of r and b.

won't perform well on large flat areas and should not be used. Two optimization procedures; the stochastic region contraction[83] and the particle swarm algorithm[82, ch. 14.4] are more computationally intensive, but might provide a better solutions than brute force.

Fishers Linear Discriminant The final colour segmentation implementation uses the Fishers Linear Discriminant method to likewise determine the r, g and b component, but adds also a threshold value adding an extra degree of freedom. The Fishers Linear Discriminant method projects d dimensional sample, $\mathbf{x}$, onto a line with a linear combination. This is equivalent to Eq. 2.6.

$$y = \mathbf{w}^T\mathbf{x} = \mathbf{x}^T\mathbf{w} = \begin{bmatrix} R \\ G \\ B \end{bmatrix}^T \begin{bmatrix} r \\ g \\ b \end{bmatrix} \tag{A.4}$$

The optimization in Fisher finds the $\mathbf{w}$ by maximizing the distance between the projected mean $\tilde{m}_i$ of each class i and minimizing the projected variance $\tilde{s}_i^2$ of each class i.

$$\max |\tilde{m}_1 - \tilde{m}_2| \quad \text{and} \quad \min \left(\tilde{s}_1^2 + \tilde{s}_2^2\right) \tag{A.5}$$

The cost is defined as

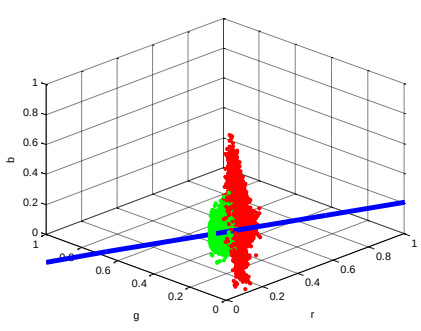
$$J(\mathbf{w}) = \frac{|\tilde{m}_1 - \tilde{m}_2|}{(\tilde{s}_1^2 + \tilde{s}_2^2)} \tag{A.6}$$

Increasing $|\tilde{m}_1 - \tilde{m}_2|$ and decreasing $\left(\tilde{s}_1^2 + \tilde{s}_2^2\right)$ will maximize $J(\mathbf{w})$. The maximum is defined as $\mathbf{w}_{fisher}$ and determined by the following optimization

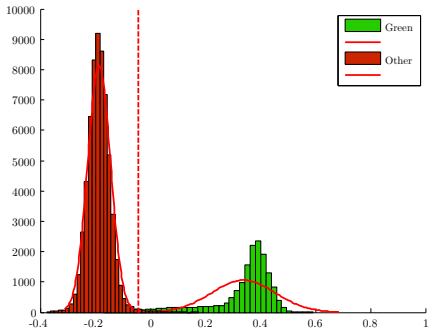
equation.

$$\mathbf{w}_{fisher} = \arg\max\left(J\left(\mathbf{w}\right)\right) = S_w^{-1}\left(m_1 - m_1\right) \tag{A.7}$$

The last term is a solution to the optimization problem derived in [69, pp. 117]. In Figure A.2a all pixels are plotted in an RGB space. The w_{fisher} line has been added in the plot with the values of r = -2.4, g = 3.7232 and b = -1.7936. In Figure A.2b all pixels have been projected on to the w_{fisher} line and matched with a Gaussian distribution. The threshold is defined as the intersection of the two distributions and found to be -0.042. The OSM pixels shows a rather screwed Gaussian distribution and the threshold might therefore be modified. The RGB image and the OSM is shown in Figure A.2c. In Figure A.2d the pixels are projected onto a 1D or greyscale image and thresholded at the intersection of the two distributions.

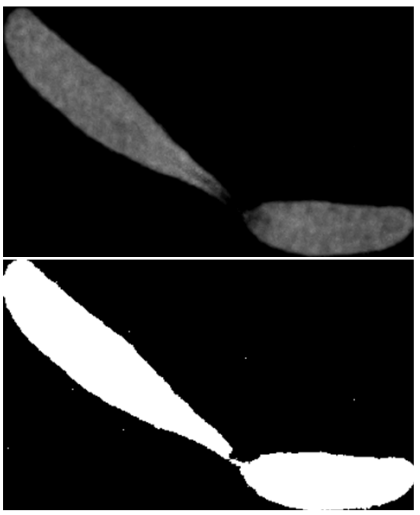


(a) All pixels are plotted in the RGB space. The w_{fisher} line is showing.



(b) Data after being projected to the optimal line.

(c) The RGB image and OSM mask.



(d) The projected greyscale image and the thresholded image.

Figure A.2: Segmenting colour with Fisher.

Appendix B

Principal Components Analysis (PCA)

From [69, p. 115] the process is described. We start with the cost function

$$J_0(\mathbf{x}_0) = \sum_{k=1}^{n} \|\mathbf{x}_0 - \mathbf{x}_k\|^2 \tag{B.1}$$

Where $\mathbf{x}_0$ is a general representation of the d dimensional feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$, where n is the number of samples. Here our aim is to minimize J_0 in a least square manner. The solution to this problem is $\mathbf{x}_0 = \boldsymbol{\mu}$, that would say; the mean of the dataset. This mean $\boldsymbol{\mu}$ is the best one-component expression of our dataset, and can be expressed as

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{k=1}^{N} \mathbf{x}_k \tag{B.2}$$

To express the variation, we have to add a term, as μ does not express any variation.

$$\mathbf{x} = \boldsymbol{\mu} + a\mathbf{e} \tag{B.3}$$

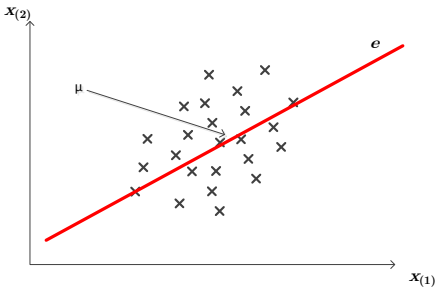


Figure B.1: 2D example of Principal Components Analysis

where $\mathbf{e}$ is a unitvector pointing in the direction of our high-dimension feature-space with the largest variation and a is a scalar, specifying the length of the vector. By chaging the cost function with this representation we get

$$J\left(a_1,\ldots,a_n,\mathbf{e}\right)=\sum_{k=1}^{N}\left\|\left(\boldsymbol{\mu}+a_k\mathbf{e}\right)-\mathbf{x}_k\right\| \tag{B.4}$$

By minimizing this cost function, we get a set of weights expressing how the k samples best can be expressed in an one-dimensional subspace spanned by $\mathbf{e}$. That is, how far from μ along $\mathbf{e}$ the k samples lay. To minimize the cost function, we take the partial derivative J with respect to a_k, and setting it to zero, we get

$$\underbrace{\min}_{a_k} J=\frac{\partial}{\partial a_k}=0\Rightarrow 2a_k-2\mathbf{e}^T\left(x_k-\boldsymbol{\mu}\right)\Rightarrow a_k=\mathbf{e}^T\left(\mathbf{x}_k-\boldsymbol{\mu}\right) \tag{B.5}$$

Which, according to [69] means that geometrically the least-squares solution is the projection of $\mathbf{x}_k$ onto the vector $\mathbf{e}$ through μ. The last thing is to find the direction of $\mathbf{e}$. To find the solution we insert a_k into Eq. B.5 which gives us

$$J=-\mathbf{e}^T S\mathbf{e}+\underbrace{\sum_{k}\left\|\mathbf{x}_k-\boldsymbol{\mu}\right\|^2}_{\text{not dependent on }\mathbf{e}} \tag{B.6}$$

Here $\mathbf{S}$ is the sample covariance matrix, which is generated in the following way:

$$\mathbf{S}=\sum_{k=1}^{N}\left(\mathbf{x}_k-\boldsymbol{\mu}\right)\left(\mathbf{x}_k-\boldsymbol{\mu}\right)^t \tag{B.7}$$

because the last term of Eq. B.6 is independent of $\mathbf{e}$ we just skip it when minimizing J with respect to $\mathbf{e}$. This minimization problem is solved using Lagrange, as we have the constrain, that $\|\mathbf{e}\|=1$

$$\tilde{J}=\mathbf{e}^T S\mathbf{e}-\lambda\left(\mathbf{e}^T\mathbf{e}-1\right) \tag{B.8}$$

where λ is the Lagrange multiplier. We differentiate $\tilde{J}$ with respect to $\mathbf{e}$ and set it to zero, from which we get

$$\frac{\partial\tilde{J}}{\partial\mathbf{e}}=2\mathbf{S}\mathbf{e}-2\lambda\mathbf{e}=0\quad\Rightarrow\quad\mathbf{S}\mathbf{e}=\lambda\mathbf{e} \tag{B.9}$$

which is a normal eigenvalue problem. We want to find $\mathbf{e}$, that satisfies this

$$\mathbf{e}^T\mathbf{S}\mathbf{e} = \mathbf{e}^T\lambda\mathbf{e} = \lambda\underbrace{\mathbf{e}^T\mathbf{e}}_{=1} = \lambda \tag{B.10}$$

The largest eigenvalue of the sample covariance matrix $\mathbf{S}$ will satisfy this. Let us call the largest eigenvalue λ_1 and the FIRST PRINCIPAL COMPONENT $\mathbf{e}_1$. We now have the direction of the line.

Appendix C

Multiple Discriminant Analysis (MDA)

Multiple Discriminant Analysis is a dimension reduction method that takes c classes of d dimensions/features and reduces the number of dimensions to a $(c-1)$ dimensional space. MDA assumes, that the feature dimension d is larger or equal to the number of plant species c, that are being classified[69]. Derivation starts by determining the within-class scatter $\mathbf{S}_w$ as in Fishers's linear discriminant described in Appendix A, but extended to c classes.

$$\mathbf{S}_w = \sum_{i=1}^{c} \mathbf{S}_i \tag{C.1}$$

where the scatter matrix $\mathbf{S}_i$ again is given by

$$\mathbf{S}_i = \sum_{\mathbf{x} \in D_i} (\mathbf{x} - \boldsymbol{\mu}_i) (\mathbf{x} - \boldsymbol{\mu}_i)^T \tag{C.2}$$

and the between class scatter is given by:

$$\mathbf{S}_b = \sum_{i=1}^{c} n_i (\boldsymbol{\mu}_i - \mathbf{M}) (\boldsymbol{\mu}_i - \mathbf{M})^T \tag{C.3}$$

where n_i is the number of samples in class i and $\mathbf{M}$ is total mean for all classes defined by

$$\mathbf{M} = \frac{1}{n} \sum_{i=1}^{c} n_i \boldsymbol{\mu}_i, \tag{C.4}$$

where n is the total number of samples. With this total mean, we can define a *total scatter matrix*, which turn out to be the sum of the within- and between-class scatter matrices.

$$\mathbf{S}_t = \mathbf{S}_b + \mathbf{S}_w \tag{C.5}$$

The projection from the d-dimentions of the features to $(c-1)$ dimension is

$$\mathbf{y} = \mathbf{W}^T \mathbf{x} \tag{C.6}$$

whereby the projected scatter-matrices $\tilde{\mathbf{S}}_b$ and $\tilde{\mathbf{S}}_w$ can be described by

$$\tilde{\mathbf{S}}_w = \mathbf{W}^T \mathbf{S}_w \mathbf{W} \tag{C.7}$$

$$\tilde{\mathbf{S}}_b = \mathbf{W}^T \mathbf{S}_b \mathbf{W} \tag{C.8}$$

The aim now is to find the subspace, described by $\mathbf{W}$, where the projection of the features for the different classes have as large separation as possible while keeping the in-class separation as small as possible. To do this, the following cost function $J(\mathbf{W})$ is created, which we want to maximize. I.e. we want to maximize the within-class scattering while minimizing the between-class scattering.

$$J(\mathbf{w}) = \frac{\left|\tilde{\mathbf{S}}_b\right|}{\left|\tilde{\mathbf{S}}_w\right|} = \frac{\left|\mathbf{W}^T \mathbf{S}_b \mathbf{W}\right|}{\left|\mathbf{W}^T \mathbf{S}_w \mathbf{W}\right|} \tag{C.9}$$

where the columns in $\mathbf{W}$ are the eigenvectors for the $(c-1)$ solutions to the eigenvalue problem:

$$\mathbf{S}_b \mathbf{w}_i = \lambda_i \mathbf{S}_w \mathbf{w}_i \tag{C.10}$$

This matrix $\mathbf{W}$ maps the features from the high dimension space to a $(c-1)$-dimensional subspace, which maximises $J(\mathbf{w})$

By sorting the eigenvectors according to the corresponding eigenvalues, the six $(c-1)$ first eigenvectors contains the contribution of all features to this lower-dimension feature-set. The weight of each feature is then found by taking the sum of the absolute value of each row in $\mathbf{W}$.

Appendix D

Extracting Boundary

The input of the algorithm is a binary mask of the plant as shown in Figure D.1b. The contour must be determined as a list of points running clockwise around the boundary. The boundary is shown in Figure D.1c. Initially different morphological operations are performed on the plant mask.

- Holes within the plant mask is filled with the MATLABfunction imfill(bw,'holes') to avoid one/more boundaries inside the object.
- The morphological operation open with a 3x3 square mask is performed to remove noise in the boundary of the object.
- The MATLAB function bwmorph(bw,'remove') with the argument 'remove' is used to extract only the boundary.
- The MATLAB function bwmorph(bw,'shrink',Inf) with the argument 'shrink' makes the thinnest possible boundary.

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

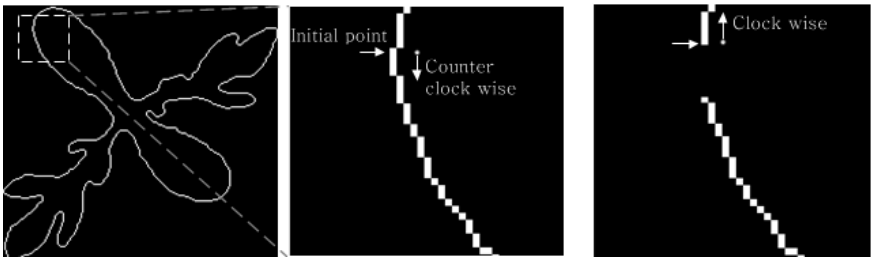
Operation 1,2 and 4 are performed to avoid rare case that cause errors when the sorted coordinate list is made. In figure D.1 the RGB image, the binary mask and the boundary of a plant is shown.

Sorting the boundary is performed by first getting a random pixels from the boundary. This pixel is stored, marked as the current position and moved from the boundary image. The next pixel in the boundary is simply the pixels from the boundary with the shortest distance to the current position.

(a) RGB image **(b)** Plant binary mask **(c)** Plant boundary

Figure D.1: Shows the RGB image, the binary mask and the extracted boundary of the plant.

The closest pixels is stored (unless the distance is to big), marked as the current position and removed from the boundary image. The procedure is processed until all pixels in the boundary have been removed.
This simple procedure will initially run in a random direction (either clockwise or counter clockwise) as the starting point will have two neighbor pixels. In Figure D.2a the problem is illustrated, showing that the procedure will run in a counter clockwise direction as the button pixels is closest.

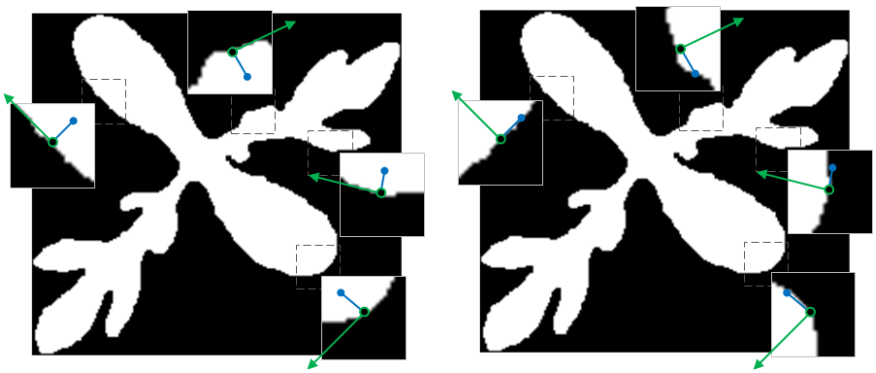


(a) The starting point has a possibility of running in both directions. **(b)** Corrected to run in the correct direction

Figure D.2: Illustration of how the starting point has a possibility of running in a counter clockwise direction.

To avoid this problem pixels in the wrong direction must be temporally removed, so the procedure will run in the right direction as in D.2b.
To choose the right direction the second point will not always be in an up direction as in Figure D.2 as the starting point e.g. might be located on the other side of the plant. A small mask is therefore selected from the plant mask around the starting point. The small mask contains information about the direction of the interior (marked with a blue dot arrow) in Figure D.3a. The correct direction will now always point in the direction of the

green arrow. The mask is rotated -90° as in Figure D.3b to remove (only temporally) neighbor pixels placed in the wrong direction by multiplying (Hadamard product) on the boundary. A mask of only size 3x3 around the starting point is sufficient.



(a) The starting point has a possibility of running in both directions.

(b) Corrected to run in the correct direction

Figure D.3: Illustration of how the starting point has a possibility of running in a counter clockwise direction.

The boundary is sorted in a function implemented in MATLAB[1] which includes a procedure for finding the first two points [2]

[1] The MATLAB script is located in: `Matlab/Segmenting/PSEP/PSEP_SortedEdgeList.m`
[2] The MATLAB script is located in: `Matlab/Segmenting/PSEP/PSEP_findTheFirst2Points.m`

Appendix E

Total variation denoiseing

This is a denoising technique, that tries to remove noise while keeping edges. The noisy image $\mathbf{b}$ can be decomposed into a clean image $\mathbf{u}$ and noise $\mathbf{n}$.

$$\mathbf{b}(x,y) = \mathbf{u}(x,y) + \mathbf{n}(x,y) \tag{E.1}$$

The aim is to minimize the difference between $\mathbf{u}$ and $\mathbf{b}$ by using a regularization term, which described the total variation across a boundary[84].

$$\arg\min_{\mathbf{u}} \left\{ \frac{1}{2}\lambda \|\mathbf{u}-\mathbf{b}\|_2^2 + \frac{1}{2}\|\mathbf{u}\|_{TV} \right\}, \tag{E.2}$$

where the latter term $\|\mathbf{u}\|_{TV}$ does not use the L2-norm but instead uses the total variation norm, which is the L1 norm of the derivative of $\mathbf{u}$ [84]. Said in a graphical way, the total variation norm is the total distance along the intensity-axis as illustrated in Figure E.1. The minimization term can be rewritten as

$$\arg\min_{\mathbf{u}} \left\{ \frac{1}{2}\|\mathbf{u}\|_{TV} \right\} \text{ subject to } \lambda \|\mathbf{u}-\mathbf{b}\|_2^2 \tag{E.3}$$

In words this means, that we want to minimize fluctuation but still want the output $\mathbf{u}$ to look like the original input $\mathbf{b}$ as much as possible. Without

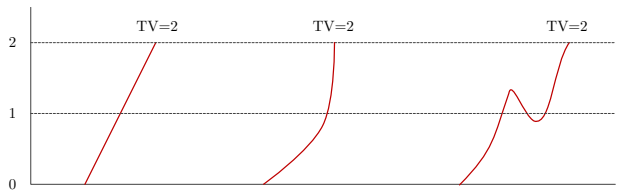


Figure E.1: Total variation for three different, one dimentional surfaces

this constrain, the output would just be flat, but in combination the result will retain sharp edges. λ is a weight factor, which determine how much we want the original image over the flat image.
As we assume, that the centre vein is bending, we treat the image isotropic.

$$\|\mathbf{u}\|_{TV_{ISO}} = \int_{\mathbf{x}} |\nabla \mathbf{u}|_2 d\mathbf{x} = \sqrt{\mathbf{u}_x + \mathbf{u}_y} \tag{E.4}$$

The function, that we want to minimize can thereby be described as:

$$f(x, y, u, u_x, u_y) = \frac{1}{2}\lambda(u-b)^2 + \sqrt{u_x^2 + u_y^2} \tag{E.5}$$

To solve this constrained problem we use Euler Lagrange [85]. The $\mathbf{u}$ that satisfies this, is the solution to the total variation denoising problem:

$$\frac{\partial f}{\partial u} - \frac{d}{dx}\frac{\partial f}{\partial u'} = \frac{\partial f}{\partial u} - \frac{d}{dx}\frac{\partial f}{\partial u_x} - \frac{d}{dy}\frac{\partial f}{\partial u_y} = 0 \tag{E.6}$$

To ease the calculation, we split Eq. E.6 in small parts:

$$\frac{\partial f}{\partial u} = \frac{\partial}{\partial u}\left(\frac{1}{2}\lambda(u-b)^2 + \underbrace{\sqrt{u_x^2 + u_y^2}}_{=0}\right) = \frac{1}{2}\lambda \cdot \frac{\partial}{\partial u}\left(u^2 + b^2 - 2ub\right) = \lambda(u-b)$$

$$\frac{\partial f}{\partial u_x} = \frac{\partial}{\partial u_x}\left(\frac{1}{2}\lambda(u-b)^2 + \sqrt{u_x^2 + u_y^2}\right) = \frac{\partial}{\partial u_x}\left(\sqrt{u_x^2 + u_y^2}\right) = \frac{u_x}{\sqrt{u_x^2 + u_y^2}}$$

$$\frac{\partial f}{\partial u_y} = \frac{\partial}{\partial u_y}\left(\frac{1}{2}\lambda(u-b)^2 + \sqrt{u_x^2 + u_y^2}\right) = \frac{\partial}{\partial u_x}\left(\sqrt{u_x^2 + u_y^2}\right) = \frac{u_y}{\sqrt{u_x^2 + u_y^2}}$$

Hereby Eq. E.6 becomes:

$$\begin{array}{lllllll} \frac{\partial f}{\partial u} & - & \frac{d}{dx}\frac{\partial f}{\partial u_x} & - & \frac{d}{dx}\frac{\partial f}{\partial u_y} & = & 0 \\ \lambda(u-b) & - & \frac{d}{dx}\left(\frac{u_x}{\sqrt{u_x^2+u_y^2}}\right) & - & \frac{d}{dx}\left(\frac{u_y}{\sqrt{u_x^2+u_y^2}}\right) & = & 0 \end{array} \tag{E.7}$$

The only thing left now is to find the $\mathbf{u}$ satisfying this. This $\mathbf{u}$ is the solution to the total variation problem and can be found by using eg. steepest descent. By using steepest descent the update process will be:

$$u^{t+1} = u^t - \varepsilon\left(\lambda(u-b) - \frac{d}{dx}\left(\frac{u_x}{\sqrt{u_x^2+u_y^2}}\right) - \frac{d}{dy}\left(\frac{u_y}{\sqrt{u_x^2+u_y^2}}\right)\right) \tag{E.8}$$

Where ε is the step-size and t indicates the iteration[1].

[1] The MATLAB script is based on [86] and located in: \Matlab\Plant normalization\tv.m

Appendix F

Feature Overview

Table F.1 shows a complete list of the 50 different feature descriptors.

Number	Feature	Number of Features
1	ObjectArea	1
2	ObjectPerimeter	1
3	ConvexHullArea	1
4	ConvexHullPerimeter	1
5	Solidity	1
6	HuMoments	8
7	FractalDimension	1
8	Convexity	1
9	Compactness	1
10	ChromaticityBlue	1
11	ChromaticityGreen	1
12	ChromaticityRed	1
13	ExcessGreen	1
14	AspectRatio	1
15	Rectangularity	1
16	Sphericity	1
17	CircularVariance	1
18	Eccentricity	1
19	FormFactor	1
20	EllipticVariance	1
21	RatioOfPrincipalAxes	1
22	DistTransformMean	1
23	DistTransformVariance	1
24	DistTransformRS_Sort	10
25	DistTransformRS_SortScaled	9
26	DistTransformRS_Acc	10
27	DistTransformRS_AccScaled	9
28	DistTransformRS_scaledAreaRoot	10
29	DistTransformLP_Sort	10
30	DistTransformLP_SortScaled	10
31	DistTransformLP_Acc	10
32	DistTransformLP_AccScaled	10
33	DistTransformLP_scaledAreaRoot	10
34	DistLPCorrelation	1
35	SkeletonDistanceLength	1
36	SkeletonDistanceMax	1
37	SkeletonDistanceMean	1
38	SkeletonDistanceVariance	1
39	VarRGB	1
40	MeanDistBetweenHulls	1
41	MinPlantThickness	2
42	EllipticFourierAbs	77
43	EF_TVH_Max	1
44	EF_TVH_Mean	1
45	EFvar	1
46	EFdist	9
47	EFdistScaled	9
48	EFdistAcc	9
49	EFdistAccScaled	8
50	EFdistVar	9
	Total	**261**

Table F.1: Feature list of all feature descriptors.

Appendix G

Classification Accuracy of Individual Features

The Tables G.1, G.2 and G.3 shows the top 20 ranking features based on the classification accuracy of individual features for each plant element respectively plants, cotyledon leaves and foliage leaves. The tables supports an important fact of the project, namely that features perform different for the three plant elements. For plants the highest ranking features are the 5 color based features. The list also includes 9 variations of the DistTransform, and in ranking order the SkeletonDistanceMean (6th place), SkeletonDistanceVariance (10th place), Sphericity (12th place), HuMoments (15th place) and Compactness (20th place). For cotyledons leaves the highest 3 ranking features are still color features, but includes now the features such as Eccentricity (4th place), RatioOfPrincipalAxes (7th place), AspectRatio(14th place) and Elliptic Fourier (16th place). For foliage leaves the highest raking features includes only one color feature (9th place), the top seven ranking features are variations of the DistTransform and includes new features such as MeanDistBetweenHulls (8th place) , DistLPCorrelation1 (10th place), EllipticVariance (15th place), Solidity (18th place) and EFDistAcc (4th place).

Ranking	Feature number	Feature name	CA
1	19	AverageChromaticityRed	0.502
2	134	VarRGB	0.500
3	17	AverageChromaticityBlue	0.500
4	18	AverageChromaticityGreen	0.491
5	20	ExcessGreen	0.474
6	132	SkeletonDistanceMean	0.451
7	79	DistTransformLP_Sort1	0.434
8	133	SkeletonDistanceVariance	0.422
9	29	DistTransformMean	0.418
10	80	DistTransformLP_Sort2	0.417
11	77	DistTransformRS_scaledAreaRoot9	0.416
12	23	Sphericity	0.413
13	119	DistTransformLP_scaledAreaRoot1	0.412
14	76	DistTransformRS_scaledAreaRoot8	0.406
15	78	DistTransformRS_scaledAreaRoot10	0.393
16	74	DistTransformRS_scaledAreaRoot6	0.389
17	73	DistTransformRS_scaledAreaRoot5	0.385

Table G.1: Classification accuracy (CA) of top 20 ranking features for **plants** based on the classification accuracy of individual features

Ranking	Feature number	Feature name	CA
1	55	DistTransformRS_Acc6	0.627
2	38	DistTransformRS_Sort8	0.626
3	50	DistTransformRS_Acc1	0.618
4	37	DistTransformRS_Sort7	0.612
5	56	DistTransformRS_Acc7	0.603
6	49	DistTransformRS_SortScaled9	0.590
7	41	DistTransformRS_SortScaled1	0.569
8	33	DistTransformRS_Sort3	0.566
9	31	DistTransformRS_Sort1	0.561
10	32	DistTransformRS_Sort2	0.561
11	36	DistTransformRS_Sort6	0.560
12	40	DistTransformRS_Sort10	0.554
13	39	DistTransformRS_Sort9	0.552
14	51	DistTransformRS_Acc2	0.551
15	58	DistTransformRS_Acc9	0.550
16	29	DistTransformMean1	0.549
17	42	DistTransformRS_SortScaled2	0.546

Table G.2: Classification accuracy of top 20 ranking features for **cotyledon** based on the classification accuracy of individual features

Ranking	Feature number	Feature name	CA
1	72	DistTransformRS_scaledAreaRoot4	0.581
2	59	DistTransformRS_Acc10	0.573
3	58	DistTransformRS_Acc9	0.562
4	52	DistTransformRS_Acc3	0.556
5	55	DistTransformRS_Acc6	0.552
6	66	DistTransformRS_AccScaled7	0.549
7	69	DistTransformRS_scaledAreaRoot1	0.546
8	1	ObjectArea1	0.541
9	71	DistTransformRS_scaledAreaRoot3	0.534
10	37	DistTransformRS_Sort7	0.532
11	65	DistTransformRS_AccScaled6	0.531
12	67	DistTransformRS_AccScaled8	0.529
13	70	DistTransformRS_scaledAreaRoot2	0.528
14	50	DistTransformRS_Acc1	0.525
15	57	DistTransformRS_Acc8	0.525
16	45	DistTransformRS_SortScaled5	0.524
17	62	DistTransformRS_AccScaled3	0.521

Table G.3: Classification accuracy of top 20 ranking features for **foliage** based on the classification accuracy of individual features

Appendix H

Classification Accuracy Of Feature Descriptors Containing Multiple Features

The feature descriptors have been ranked for average classification accuracy of all plant elements, showing the ranking of features for plants, cotyledons leaves and foliage leaves. Generally variations of the DistTransform perform well for all elements. Especially the proposed DistTransformLP_scaledAreaRoot is the highest ranked feature for all plant elements. All the LP (Legendre) variation of the distTransform rank higher than the RS methods, and the DistTransformLP_SortScaled and DistTransformLP_AccScaled are ranked on either 2nd or 3rd place. No feature is constantly ranked as the worst feature, for plants the MinPlantThickness is the lowest, for cotyledon leaves the EFdistVar is ranked the lowest and for foliage leaves the EFdistAcc-Scaled ranked the lowest. The feature EllipticFourierAbs is generally the best feature for all plant elements when not including any variations of the DistTransform.

Appendix H. Classification Accuracy Of Feature Descriptors Containing Multiple Features

No.	Feature name	All		Plant		Cotyledon		Foliage	
		Rank	CA	Rank	CA	Rank	CA	Rank	CA
33	DistTransformLP_scaledAreaRoot	1	0.779	1	0.803	1	0.762	1	0.777
30	DistTransformLP_SortScaled	2	0.726	2	0.740	2	0.709	2	0.743
32	DistTransformLP_AccScaled	3	0.712	3	0.729	3	0.697	3	0.719
31	DistTransformLP_Acc	4	0.693	7	0.711	4	0.677	6	0.703
28	DistTransformRS_scaledAreaRoot	5	0.685	6	0.718	5	0.659	7	0.692
29	DistTransformLP_Sort	6	0.685	5	0.722	6	0.656	8	0.691
27	DistTransformRS_AccScaled	7	0.673	8	0.699	7	0.649	9	0.687
43	EllipticFourierAbs	8	0.668	4	0.725	10	0.612	4	0.711
24	DistTransformRS_Sort	9	0.640	11	0.663	8	0.621	12	0.647
26	DistTransformRS_Acc	10	0.629	12	0.633	9	0.615	11	0.654
42	EllipticFourier	11	0.624	10	0.681	11	0.588	16	0.612
47	EFdist	12	0.615	9	0.698	15	0.519	5	0.708
25	DistTransformRS_SortScaled	13	0.597	13	0.594	12	0.583	13	0.638
49	EFdistAcc	14	0.567	14	0.585	14	0.526	14	0.637
48	EFdistScaled	15	0.546	15	0.579	13	0.527	18	0.535
6	HuMoments	16	0.511	18	0.450	16	0.514	15	0.613
50	EFdistAccScaled	17	0.507	16	0.533	17	0.486	19	0.510
51	EFdistVar	18	0.501	17	0.509	19	0.425	10	0.679
41	MinPlantThickness	19	0.464	19	0.432	18	0.441	17	0.580

Table H.1: Classification accuracy (CA) of feature descriptors containing multiple features.

Appendix I

Evaluation of Distance Transform

	Maize 1	Winter wheat 2	Sugar beet 3	Scentless mayweed 4	Chickweed 5	Shepherd's-purse 6	Cleavers 7	Total
$\mathbf{d}_{sort_{RS}}$	0.576	0.217	0.692	0.790	0.784	0.459	0.492	0.656
$\mathbf{d}_{sort_{LP}}$	0.624	0.242	0.717	0.830	0.888	0.580	0.538	0.599
$\mathbf{d}_{scaled_{LP}}$	0.672	0.227	0.757	0.845	0.882	0.588	0.647	0.642
$\mathbf{d}_{scaled_{RS}}$	0.552	0.181	0.609	0.723	0.727	0.474	0.300	0.699
$\mathbf{d}_{Accu_{RS}}$	0.508	0.394	0.643	0.786	0.824	0.267	0.400	0.706
$\mathbf{d}_{Accu_{LP}}$	0.600	0.232	0.760	0.869	0.879	0.514	0.515	0.721
$\mathbf{d}_{AccuScaled_{LP}}$	0.648	0.258	0.748	0.849	0.861	0.592	0.600	0.739
$\mathbf{d}_{AccuScaled_{RS}}$	0.624	0.273	0.732	0.838	0.826	0.463	0.554	0.722
$\mathbf{d}_{scaledAreaRoot_{RS}}$	0.72	0.303	0.785	0.753	0.821	0.506	0.662	0.730
$\mathbf{d}_{scaledAreaRoot_{LP}}$	0.752	0.434	0.852	0.883	0.908	0.675	0.723	0.806
$\mathbf{r}^2$	0.12	0.116	0.382	0.463	0.398	0.235	0.069	0.325

Table I.1: Classification accuracies for a 4-nearest neighbour classifier using the distance transform features on whole plants.

	Maize 1	Sugar beet 3	Scentless mayweed 4	Chickweed 5	Shepherd's-purse 6	Cleavers 7	Total
$\mathbf{d}_{sort_{RS}}$	0.758	0.904	0.530	0.836	0.386	0.789	0.755
$\mathbf{d}_{sort_{LP}}$	0.799	0.728	0.290	0.636	0.218	0.706	0.737
$\mathbf{d}_{scaled_{LP}}$	0.851	0.748	0.225	0.477	0.127	0.640	0.775
$\mathbf{d}_{scaled_{RS}}$	0.737	0.897	0.557	0.793	0.386	0.759	0.678
$\mathbf{d}_{Accu_{RS}}$	0.784	0.901	0.639	0.847	0.404	0.789	0.714
$\mathbf{d}_{Accu_{LP}}$	0.799	0.704	0.238	0.527	0.078	0.477	0.587
$\mathbf{d}_{AccuScaled_{LP}}$	0.907	0.031	0.114	0.033	0.023	0.041	0.514
$\mathbf{d}_{AccuScaled_{RS}}$	0.799	0.672	0.495	0.796	0.500	0.640	0.495
$\mathbf{d}_{scaledAreaRoot_{RS}}$	0.835	0.737	0.644	0.788	0.474	0.690	0.092
$\mathbf{d}_{scaledAreaRoot_{LP}}$	0.923	0.038	0.119	0.041	0.018	0.043	0.097
$\mathbf{r}^2$	0.603	0.018	0.025	0.013	0.000	0.003	0.047

Table I.2: Classification accuracies for a 1-nearest neighbour classifier using the distance transform features on cotyledons.

	Sugar beet 3	Scentless mayweed 4	Chickweed 5	Shepherd's-purse 6	Cleavers 7	Total
$\mathbf{d}_{sort_{RS}}$	0.087	0.668	0.521	0.435	0.762	0.585
$\mathbf{d}_{sort_{LP}}$	0.696	0.433	0.398	0.230	0.654	0.669
$\mathbf{d}_{scaled_{LP}}$	0.913	0.228	0.071	0.062	0.390	0.662
$\mathbf{d}_{scaled_{RS}}$	0.435	0.763	0.638	0.438	0.861	0.736
$\mathbf{d}_{Accu_{RS}}$	0.304	0.715	0.712	0.441	0.831	0.742
$\mathbf{d}_{Accu_{LP}}$	0.913	0.199	0.023	0.050	0.156	0.419
$\mathbf{d}_{AccuScaled_{LP}}$	1,000	0.040	0.000	0.022	0.048	0.192
$\mathbf{d}_{AccuScaled_{RS}}$	0.348	0.742	0.790	0.627	0.844	0.128
$\mathbf{d}_{scaledAreaRoot_{RS}}$	0.217	0.841	0.783	0.553	0.797	0.044
$\mathbf{d}_{scaledAreaRoot_{LP}}$	1,000	0.042	0.000	0.019	0.048	0.044
$\mathbf{r}^2$	1.000	0.044	0.000	0.012	0.035	0.041

Table I.3: Classification accuracies for a 1-nearest neighbour classifier using the distance transform features on Foliages.

Appendix J

Forward selection Result

Table J.1 shows the features ranged from the least to the maximum importance for the classification accuracy tested on whole plants.

#	Feature	CA	#	Feature	CA	#	Feature	CA
1	AverageChromaticityBlue1	0.501		Convexity1	0.964	175	EllipticFourierAbs75	0.956
	DistTransformRS_Sort4	0.739		EllipticFourierAbs67	0.963		EllipticFourierAbs54	0.954
	DistTransformRS_scaledAreaRoot6	0.856	90	DistTransformLP_Acc1	0.963		EllipticFourierAbs44	0.952
	ExcessGreen1	0.898		DistTransformLP_Acc6	0.963		EllipticFourierAbs25	0.954
5	Sphericity1	0.925		Compactness1	0.965		EllipticFourierAbs17	0.956
	EllipticVariance1	0.936		EllipticFourierAbs74	0.965	180	EFdistAcc6	0.954
	EFdistScaled5	0.943		DistTransformLP_AccScaled10	0.966		Eccentricity1	0.953
	VarRGB1	0.945	95	DistTransformLP_Acc9	0.966		EllipticFourierAbs71	0.954
	DistTransformRS_Sort9	0.949		DistTransformRS_Sort8	0.966		DistTransformMean1	0.954
10	EFdistAcc1	0.949		EllipticFourierAbs12	0.967		DistTransformLP_scaledAreaRoot6	0.953
	DistTransformLP_AccScaled6	0.952		DistTransformRS_scaledAreaRoot8	0.967	185	EllipticFourierAbs55	0.953
	DistTransformLP_Acc5	0.952		EllipticFourierAbs18	0.965		EllipticFourierAbs52	0.955
	DistTransformLP_Sort4	0.951	100	EllipticFourierAbs70	0.967		ConvexHullPerimeter1	0.954
	DistTransformRS_Sort3	0.953		DistTransformRS_AccScaled6	0.966		EFdistScaled8	0.954
15	EllipticFourierAbs30	0.950		EF_TVH_Max1	0.967		DistTransformRS_SortScaled3	0.955
	MinPlantThickness2	0.952		DistTransformRS_Acc4	0.966	190	EllipticFourierAbs35	0.954
	EFdist9	0.953		DistTransformLP_Acc2	0.967		DistTransformLP_AccScaled1	0.952
	EllipticFourierAbs47	0.952	105	EllipticFourierAbs39	0.966		RatioOfPrincipalAxes1	0.953
	Solidity1	0.956		SkeletonDistanceMean1	0.966		EllipticFourierAbs62	0.952
20	DistTransformLP_Acc10	0.957		EFdistAcc8	0.966		EFdist1	0.953
	DistTransformRS_Acc5	0.958		DistTransformLP_Acc8	0.967	195	EllipticFourierAbs50	0.952
	DistTransformLP_Sort3	0.957		EFdistScaled7	0.967		DistTransformLP_AccScaled2	0.952
	EFdistVar1	0.959	110	DistTransformLP_AccScaled7	0.965		EllipticFourierAbs28	0.951
	EFdistVar9	0.958		EllipticFourierAbs11	0.963		DistTransformRS_SortScaled5	0.950
25	HuMoments6	0.959		ObjectPerimeter1	0.966		EllipticFourierAbs14	0.948
	HuMoments4	0.959		EllipticFourierAbs27	0.963	200	EFdistAcc5	0.949
	DistTransformRS_Acc10	0.958		DistTransformRS_SortScaled8	0.964		EllipticFourierAbs2	0.948
	AverageChromaticityRed1	0.961	115	EllipticFourierAbs64	0.963		EFdistAccScaled8	0.951
	FormFactor1	0.958		EllipticFourierAbs66	0.962		EllipticFourierAbs5	0.950
30	EllipticFourierAbs59	0.958		AverageChromaticityGreen1	0.965		DistTransformRS_SortScaled7	0.948
	EFdist3	0.960		ObjectArea1	0.963	205	DistTransformRS_AccScaled2	0.951
	DistLPCorrelation1	0.962		EFdistVar7	0.963		EllipticFourierAbs58	0.949
	DistTransformRS_Sort5	0.963	120	EllipticFourierAbs38	0.962		DistTransformRS_scaledAreaRoot2	0.951
	CircularVariance1	0.962		EllipticFourierAbs72	0.963		DistTransformLP_scaledAreaRoot2	0.949
35	EFdistScaled9	0.963		EFdistVar5	0.963		EllipticFourierAbs21	0.951
	DistTransformRS_AccScaled4	0.967		DistTransformLP_scaledAreaRoot5	0.960	210	DistTransformLP_SortScaled5	0.950
	DistTransformRS_Acc7	0.967		EFdist8	0.961		EFdistVar3	0.949
	EllipticFourierAbs9	0.967	125	DistTransformRS_scaledAreaRoot7	0.961		EllipticFourierAbs53	0.950
	DistTransformLP_SortScaled2	0.966		DistTransformRS_scaledAreaRoot10	0.960		DistTransformLP_SortScaled6	0.947
40	DistTransformLP_SortScaled4	0.966		DistTransformRS_Acc6	0.961		EFdistAccScaled4	0.948
	EFdistAccScaled2	0.967		EllipticFourierAbs32	0.961	215	EFdist5	0.948
	EllipticFourierAbs20	0.968		EllipticFourierAbs31	0.961		DistTransformRS_SortScaled4	0.948
	EFdistScaled6	0.969	130	DistTransformRS_Sort1	0.960		DistTransformRS_AccScaled9	0.947
	DistTransformRS_Acc8	0.969		DistTransformRS_AccScaled8	0.961		EllipticFourierAbs42	0.949
45	DistTransformLP_Acc3	0.968		DistTransformLP_scaledAreaRoot1	0.960		EllipticFourierAbs19	0.949
	HuMoments3	0.970		EllipticFourierAbs51	0.958	220	EllipticFourierAbs68	0.948
	SkeletonDistanceVariance1	0.966		EFdistAccScaled5	0.960		EllipticFourierAbs29	0.949
	DistTransformRS_Sort10	0.967	135	FractalDimension1	0.960		DistTransformRS_scaledAreaRoot3	0.949
	HuMoments2	0.967		EllipticFourierAbs41	0.957		EllipticFourierAbs16	0.950
50	DistTransformRS_Sort2	0.967		DistTransformLP_Sort9	0.959		ETvar1	0.948
	DistTransformRS_SortScaled1	0.965		EFdistScaled3	0.959	225	EllipticFourierAbs1	0.947
	DistTransformRS_AccScaled1	0.965		DistTransformLP_SortScaled9	0.958		EFdist4	0.948
	EFdistAcc4	0.966	140	EFdistAcc2	0.960		EllipticFourierAbs77	0.950
	EllipticFourierAbs60	0.967		EllipticFourierAbs73	0.959		EllipticFourierAbs61	0.948
55	DistTransformRS_Acc3	0.967		EllipticFourierAbs63	0.959		EllipticFourierAbs26	0.949
	AspectRatio1	0.967		EllipticFourierAbs23	0.957	230	DistTransformLP_AccScaled9	0.949
	DistTransformLP_Acc4	0.968		EFdist6	0.957		DistTransformRS_AccScaled7	0.947
	EllipticFourierAbs10	0.967	145	DistTransformLP_Sort10	0.957		DistTransformRS_AccScaled5	0.948
	DistTransformLP_scaledAreaRoot8	0.965		EFdistScaled4	0.960		EllipticFourierAbs33	0.948
60	DistTransformRS_Acc1	0.965		DistTransformLP_Sort7	0.959		SkeletonDistanceLength1	0.949
	ConvexHullArea1	0.966		EFdistVar4	0.957	235	MinPlantThickness1	0.948
	EllipticFourierAbs56	0.965		EllipticFourierAbs76	0.958		EFdistAcc3	0.947
	DistTransformRS_Acc2	0.966	150	EFdistAccScaled3	0.959		DistTransformLP_scaledAreaRoot3	0.946
	DistTransformLP_AccScaled3	0.966		DistTransformLP_SortScaled1	0.958		DistTransformRS_scaledAreaRoot4	0.946
65	HuMoments1	0.967		DistTransformRS_Sort7	0.959		DistTransformLP_AccScaled8	0.945
	DistTransformLP_Sort6	0.967		DistTransformRS_scaledAreaRoot9	0.958	240	EF_TVH_Mean1	0.947
	EFdistAccScaled7	0.967		DistTransformRS_SortScaled6	0.956		MeanDistBetweenHulls1	0.948
	DistTransformLP_Sort1	0.967	155	DistTransformLP_scaledAreaRoot4	0.957		DistTransformLP_AccScaled5	0.949
	DistTransformLP_Sort5	0.966		EllipticFourierAbs13	0.954		DistTransformLP_SortScaled3	0.947
70	DistTransformRS_Sort6	0.967		DistTransformRS_SortScaled2	0.955		EllipticFourierAbs24	0.948
	EllipticFourierAbs36	0.965		EFdistAcc9	0.955	245	DistTransformLP_SortScaled8	0.945
	DistTransformLP_Sort8	0.966		EllipticFourierAbs7	0.955		EllipticFourierAbs4	0.946
	EFdist7	0.966	160	EFdistVar2	0.954		EllipticFourierAbs57	0.946
	EFdistVar8	0.966		DistTransformRS_SortScaled9	0.953		DistTransformLP_SortScaled10	0.945
75	DistTransformRS_AccScaled3	0.965		EFdistAccScaled1	0.957		EFdistScaled1	0.947
	EllipticFourierAbs15	0.965		EllipticFourierAbs48	0.955	250	DistTransformLP_scaledAreaRoot7	0.946
	DistTransformVariance1	0.966		EllipticFourierAbs65	0.956		EllipticFourierAbs49	0.946
	EllipticFourierAbs43	0.964	165	EllipticFourierAbs6	0.955		EllipticFourierAbs22	0.943
	HuMoments7	0.964		EllipticFourierAbs34	0.953		DistTransformLP_Acc7	0.945
80	EFdist2	0.967		EllipticFourierAbs46	0.954		EllipticFourierAbs3	0.943
	SkeletonDistanceMax1	0.964		HuMoments5	0.956	255	EFdistScaled2	0.942
	Rectangularity1	0.964		EllipticFourierAbs69	0.953		EllipticFourierAbs45	0.941
	HuMoments8	0.964	170	EllipticFourierAbs8	0.954		DistTransformRS_scaledAreaRoot1	0.941
	DistTransformLP_Sort2	0.964		EFdistVar6	0.952		EFdistAcc7	0.941
85	EFdistAccScaled6	0.965		EllipticFourierAbs37	0.954		DistTransformLP_scaledAreaRoot10	0.939
	DistTransformLP_AccScaled4	0.964		DistTransformRS_Acc9	0.952	260	DistTransformLP_scaledAreaRoot9	0.939
	DistTransformRS_scaledAreaRoot5	0.963		EllipticFourierAbs40	0.954		DistTransformLP_SortScaled7	0.939

Table J.1: Order of feature importance from forward selection tested on **Plants**. The classification accuracy (CA) indicates the classification rate after the feature has been added to the subset

#	Feature	CA	#	Feature	CA	#	Feature	CA
1	DistTransformRS_scaledAreaRoot4	0.594		EllipticFourierAbs38	0.875	175	EllipticFourierAbs6	0.886
	DistTransformRS_AccScaled7	0.700		DistTransformRS_SortScaled9	0.875		EllipticFourierAbs4	0.884
	Convexity1	0.739	90	EllipticFourierAbs35	0.878		DistTransformLP_AccScaled2	0.890
	DistTransformLP_SortScaled8	0.753		EllipticVariance1	0.883		EFdistScaled1	0.893
5	Solidity1	0.758		DistTransformRS_SortScaled4	0.890		EllipticFourierAbs43	0.887
	DistTransformRS_AccScaled3	0.801		DistTransformLP_SortScaled1	0.889	180	DistTransformRS_scaledAreaRoot6	0.887
	DistTransformRS_AccScaled5	0.830		EllipticFourierAbs30	0.886		DistTransformLP_scaledAreaRoot4	0.886
	DistTransformRS_scaledAreaRoot1	0.839	95	EllipticFourierAbs60	0.885		DistTransformLP_Acc2	0.880
	EllipticFourierAbs64	0.840		Sphericity1	0.885		EllipticFourierAbs1	0.884
10	DistTransformLP_Sort1	0.842		MinPlantThickness2	0.890		EFdistAccScaled7	0.892
	EllipticFourierAbs23	0.842		EllipticFourierAbs49	0.886	185	DistTransformLP_Sort4	0.892
	SkeletonDistanceMax1	0.851		DistTransformLP_Sort3	0.889		DistTransformLP_Sort2	0.884
	DistTransformRS_Acc6	0.844	100	EllipticFourierAbs14	0.889		EF_TVH_Max1	0.896
	EllipticFourierAbs53	0.846		DistTransformLP_scaledAreaRoot6	0.886		FormFactor1	0.892
15	DistTransformRS_Acc1	0.846		AspectRatio1	0.889		EllipticFourierAbs55	0.892
	DistTransformLP_AccScaled4	0.849		DistTransformLP_Acc1	0.885	190	DistTransformRS_scaledAreaRoot8	0.889
	DistTransformLP_Sort8	0.856		AverageChromaticityGreen1	0.889		DistTransformLP_Sort6	0.886
	EllipticFourierAbs31	0.853	105	EFdistVar3	0.890		DistTransformRS_scaledAreaRoot10	0.890
	DistTransformRS_Acc4	0.854		EFdist5	0.886		EFdistScaled5	0.890
20	DistTransformLP_SortScaled6	0.852		EllipticFourierAbs12	0.883		EllipticFourierAbs7	0.892
	DistTransformLP_AccScaled3	0.855		EFdistScaled2	0.891	195	EllipticFourierAbs15	0.896
	DistTransformRS_Acc7	0.858		SkeletonDistanceLength1	0.892		EllipticFourierAbs48	0.894
	DistTransformLP_Acc6	0.854	110	DistTransformLP_Acc4	0.886		EFdist2	0.891
	EllipticFourierAbs41	0.855		MinPlantThickness1	0.888		EllipticFourierAbs9	0.888
25	EllipticFourierAbs36	0.854		DistTransformLP_Acc9	0.886		FractalDimension1	0.891
	EllipticFourierAbs28	0.857		EFdistVar8	0.891	200	EllipticFourierAbs18	0.892
	DistTransformRS_scaledAreaRoot5	0.854		DistTransformRS_AccScaled9	0.890		EllipticFourierAbs29	0.894
	DistTransformRS_Sort9	0.852	115	SkeletonDistanceMean1	0.888		DistTransformRS_Acc2	0.887
	DistTransformRS_scaledAreaRoot9	0.856		EllipticFourierAbs2	0.886		DistTransformRS_scaledAreaRoot3	0.891
30	EllipticFourierAbs46	0.864		EllipticFourierAbs27	0.889		DistTransformLP_SortScaled4	0.890
	EllipticFourierAbs63	0.860		EFdistAccScaled2	0.893	205	EllipticFourierAbs45	0.889
	EllipticFourierAbs11	0.857		EllipticFourierAbs26	0.891		DistTransformLP_scaledAreaRoot2	0.889
	EllipticFourierAbs37	0.860	120	EllipticFourierAbs42	0.888		EllipticFourierAbs62	0.886
	EllipticFourierAbs66	0.859		DistTransformLP_Sort9	0.889		RatioOfPrincipalAxes1	0.888
35	EFdistAccScaled5	0.858		ExcessGreen1	0.887		EllipticFourierAbs54	0.890
	DistTransformLP_AccScaled8	0.858		EllipticFourierAbs22	0.890	210	EFdistAcc9	0.891
	DistTransformLP_SortScaled5	0.857		EllipticFourierAbs34	0.890		EFdistAcc8	0.884
	EFdistAcc1	0.859	125	EFdist4	0.890		DistTransformRS_AccScaled6	0.887
	EFdistAcc2	0.856		EllipticFourierAbs73	0.888		DistTransformVariance1	0.885
40	EFdist9	0.857		EllipticFourierAbs69	0.890		EFdistScaled3	0.882
	EFdistAcc4	0.861		EFdistVar5	0.892	215	EllipticFourierAbs61	0.888
	DistTransformLP_SortScaled7	0.865		EllipticFourierAbs47	0.893		EFdistAccScaled8	0.888
	EllipticFourierAbs20	0.865	130	DistTransformRS_Sort10	0.889		EFdistScaled8	0.884
	DistTransformLP_SortScaled9	0.858		EllipticFourierAbs77	0.889		EFdistAcc5	0.886
45	DistTransformLP_Sort5	0.864		EllipticFourierAbs51	0.884		EFdistScaled7	0.885
	DistTransformLP_Sort7	0.866		EFdistVar1	0.893	220	DistTransformMean1	0.890
	EllipticFourierAbs67	0.866		EllipticFourierAbs68	0.891		EFdistAcc7	0.887
	EFdistAccScaled1	0.866	135	DistTransformRS_Acc9	0.889		DistTransformLP_Acc3	0.884
	DistTransformLP_AccScaled7	0.870		ETvar1	0.888		DistTransformLP_scaledAreaRoot8	0.882
50	EF_TVH_Mean1	0.863		DistTransformLP_SortScaled10	0.891		DistTransformRS_Sort1	0.888
	VarRGB1	0.867		EFdistAccScaled3	0.891	225	EllipticFourierAbs70	0.886
	DistTransformLP_Acc5	0.873		SkeletonDistanceVariance1	0.891		DistTransformRS_Sort7	0.885
	EFdistVar7	0.867	140	EllipticFourierAbs71	0.888		AverageChromaticityRed1	0.884
	DistTransformLP_SortScaled3	0.871		AverageChromaticityBlue1	0.891		DistTransformRS_Sort6	0.877
55	CircularVariance1	0.871		DistTransformRS_SortScaled3	0.890		Eccentricity1	0.881
	EllipticFourierAbs75	0.871		EllipticFourierAbs8	0.889	230	DistTransformLP_AccScaled6	0.878
	DistTransformLP_scaledAreaRoot9	0.869		EllipticFourierAbs5	0.886		EFdistVar4	0.880
	EllipticFourierAbs33	0.872	145	EllipticFourierAbs13	0.891		EllipticFourierAbs76	0.876
	EllipticFourierAbs3	0.870		MeanDistBetweenHulls1	0.887		DistTransformRS_Sort2	0.876
60	DistTransformRS_scaledAreaRoot2	0.865		Rectangularity1	0.892		DistTransformRS_scaledAreaRoot7	0.878
	DistTransformLP_scaledAreaRoot10	0.869		HuMoments3	0.892	235	EFdistAcc6	0.881
	HuMoments1	0.864		DistTransformLP_AccScaled1	0.890		EFdistVar2	0.879
	EFdist7	0.861	150	EFdistAcc3	0.889		DistTransformRS_SortScaled2	0.875
	DistTransformRS_SortScaled1	0.864		DistTransformRS_Acc8	0.888		DistTransformLP_SortScaled2	0.874
65	EllipticFourierAbs32	0.870		EllipticFourierAbs44	0.889		DistTransformRS_Sort4	0.870
	DistTransformLP_Acc7	0.869		EFdistScaled4	0.893	240	DistTransformRS_Sort5	0.866
	DistTransformLP_scaledAreaRoot1	0.868		DistTransformRS_Acc3	0.891		EFdist6	0.866
	EFdistScaled9	0.866	155	DistTransformLP_scaledAreaRoot3	0.891		EllipticFourierAbs16	0.869
	DistTransformRS_AccScaled8	0.875		EllipticFourierAbs50	0.888		DistTransformRS_Sort3	0.871
70	EllipticFourierAbs65	0.873		DistTransformRS_Acc10	0.887		HuMoments2	0.866
	EFdistAccScaled4	0.871		EllipticFourierAbs24	0.887	245	DistTransformRS_AccScaled2	0.866
	EllipticFourierAbs19	0.872		EllipticFourierAbs17	0.890		HuMoments5	0.859
	DistTransformLP_AccScaled10	0.870	160	EFdistVar9	0.893		ObjectPerimeter1	0.853
	EllipticFourierAbs39	0.869		DistTransformRS_AccScaled1	0.891		DistTransformRS_AccScaled4	0.859
75	EllipticFourierAbs52	0.875		EFdist8	0.889		DistTransformRS_SortScaled8	0.865
	DistTransformLP_AccScaled5	0.871		EFdist1	0.888	250	DistTransformRS_SortScaled5	0.867
	DistTransformLP_Acc10	0.875		EFdistVar6	0.887		ConvexHullArea1	0.867
	DistTransformLP_AccScaled9	0.872	165	EllipticFourierAbs74	0.892		ConvexHullPerimeter1	0.864
	EllipticFourierAbs72	0.867		EllipticFourierAbs25	0.887		HuMoments6	0.860
80	DistTransformLP_scaledAreaRoot7	0.876		EllipticFourierAbs10	0.891		Compactness1	0.858
	EFdist3	0.878		EFdistScaled6	0.889	255	HuMoments8	0.858
	DistTransformLP_Sort10	0.870		EllipticFourierAbs57	0.892		HuMoments7	0.863
	EllipticFourierAbs40	0.870	170	EFdistAccScaled6	0.890		ObjectArea1	0.861
	DistTransformLP_scaledAreaRoot5	0.868		DistTransformLP_Acc8	0.888		HuMoments4	0.858
85	DistLPCorrelation1	0.872		EllipticFourierAbs21	0.884		DistTransformRS_Sort8	0.854
	EllipticFourierAbs59	0.872		EllipticFourierAbs56	0.889	260	DistTransformRS_SortScaled7	0.854
	DistTransformRS_Acc5	0.870		EllipticFourierAbs58	0.889		DistTransformRS_SortScaled6	0.849

Table J.2: Order of feature importance from forward selection tested on **cotyledons**. The classification accuracy (CA) indicates the classification rate after the feature has been added to the subset

#	Feature	CA	#	Feature	CA	#	Feature	CA
1	DistTransformRS_scaledAreaRoot4	0.594		EllipticFourierAbs38	0.874816	175	EllipticFourierAbs6	0.886009
	DistTransformRS_AccScaled7	0.700		DistTransformRS_SortScaled9	0.874963		EllipticFourierAbs4	0.884389
	Convexity1	0.739	90	EllipticFourierAbs35	0.877614		DistTransformLP_AccScaled2	0.89028
	DistTransformLP_SortScaled8	0.753		EllipticVariance1	0.882769		EFdistScaled1	0.892636
5	Solidity1	0.758		DistTransformRS_SortScaled4	0.890133		EllipticFourierAbs43	0.88704
	DistTransformRS_AccScaled3	0.801		DistTransformLP_SortScaled1	0.888954	180	DistTransformRS_scaledAreaRoot6	0.887187
	DistTransformRS_AccScaled5	0.830		EllipticFourierAbs30	0.886303		DistTransformLP_scaledAreaRoot4	0.885567
	DistTransformRS_scaledAreaRoot1	0.839	95	EllipticFourierAbs60	0.885272		DistTransformLP_Acc2	0.879823
	EllipticFourierAbs64	0.840		Sphericity1	0.885125		EllipticFourierAbs1	0.883947
10	DistTransformLP_Sort1	0.842		MinPlantThickness2	0.890133		EFdistAccScaled7	0.892194
	EllipticFourierAbs23	0.842		EllipticFourierAbs49	0.886451	185	DistTransformLP_Sort4	0.892489
	SkeletonDistanceMax1	0.851		DistTransformLP_Sort3	0.888513		DistTransformLP_Sort2	0.884389
	DistTransformRS_Acc6	0.844	100	EllipticFourierAbs14	0.889102		EF_TVH_Max1	0.896171
	EllipticFourierAbs53	0.846		DistTransformLP_scaledAreaRoot6	0.886303		FormFactor1	0.892047
15	DistTransformRS_Acc1	0.846		AspectRatio1	0.889102		EllipticFourierAbs55	0.891605
	DistTransformLP_AccScaled4	0.849		DistTransformLP_Acc1	0.885125	190	DistTransformRS_scaledAreaRoot8	0.889396
	DistTransformLP_Sort8	0.856		AverageChromaticityGreen1	0.888954		DistTransformLP_Sort6	0.885567
	EllipticFourierAbs31	0.853	105	EFdistVar3	0.889838		DistTransformRS_scaledAreaRoot10	0.890133
	DistTransformRS_Acc4	0.854		EFdist5	0.885567		EFdistScaled5	0.890133
20	DistTransformLP_SortScaled6	0.852		EllipticFourierAbs12	0.883211		EllipticFourierAbs7	0.892194
	DistTransformLP_AccScaled3	0.855		EFdistScaled2	0.891016	195	EllipticFourierAbs15	0.895582
	DistTransformRS_Acc7	0.858		SkeletonDistanceLength1	0.8919		EllipticFourierAbs48	0.89352
	DistTransformLP_Acc6	0.854	110	DistTransformLP_Acc4	0.885567		EFdist2	0.890722
	EllipticFourierAbs41	0.855		MinPlantThickness1	0.887629		EllipticFourierAbs9	0.887629
25	EllipticFourierAbs36	0.854		DistTransformLP_Acc9	0.886009		FractalDimension1	0.891163
	EllipticFourierAbs28	0.857		EFdistVar8	0.891016	200	EllipticFourierAbs18	0.8919
	DistTransformRS_scaledAreaRoot5	0.854		DistTransformRS_AccScaled9	0.889985		EllipticFourierAbs29	0.893667
	DistTransformRS_Sort9	0.852	115	SkeletonDistanceMean1	0.887629		DistTransformRS_Acc2	0.887482
	DistTransformRS_scaledAreaRoot9	0.856		EllipticFourierAbs2	0.886451		DistTransformRS_scaledAreaRoot3	0.891163
30	EllipticFourierAbs46	0.864		EllipticFourierAbs27	0.888513		DistTransformLP_SortScaled4	0.889691
	EllipticFourierAbs63	0.860		EFdistAccScaled2	0.893078	205	EllipticFourierAbs45	0.888807
	EllipticFourierAbs11	0.857		EllipticFourierAbs26	0.891458		DistTransformLP_scaledAreaRoot2	0.88866
	EllipticFourierAbs37	0.860	120	EllipticFourierAbs42	0.887776		EllipticFourierAbs62	0.886451
	EllipticFourierAbs66	0.859		DistTransformLP_Sort9	0.889396		RatioOfPrincipalAxes1	0.888071
35	EFdistAccScaled5	0.858		ExcessGreen1	0.887187		EllipticFourierAbs54	0.889543
	DistTransformLP_AccScaled8	0.858		EllipticFourierAbs22	0.889691	210	EFdistAcc9	0.890869
	DistTransformLP_SortScaled5	0.857		EllipticFourierAbs34	0.889543		EFdistAcc8	0.884094
	EFdistAcc1	0.859	125	EFdist4	0.889985		DistTransformRS_AccScaled6	0.886892
	EFdistAcc2	0.856		EllipticFourierAbs73	0.888365		DistTransformVariance1	0.88542
40	EFdist9	0.857		EllipticFourierAbs69	0.890427		EFdistScaled3	0.882032
	EFdistAcc4	0.861		EFdistVar5	0.892047	215	EllipticFourierAbs61	0.888071
	DistTransformLP_SortScaled7	0.865		EllipticFourierAbs47	0.893225		EFdistAccScaled8	0.887629
	EllipticFourierAbs20	0.865	130	DistTransformRS_Sort10	0.888807		EFdistScaled8	0.884389
	DistTransformLP_SortScaled9	0.858		EllipticFourierAbs77	0.88866		EFdistAcc5	0.886009
45	DistTransformLP_Sort5	0.864		EllipticFourierAbs51	0.883947		EFdistScaled7	0.884683
	DistTransformLP_Sort7	0.866		EFdistVar1	0.892784	220	DistTransformMean1	0.890133
	EllipticFourierAbs67	0.866		EllipticFourierAbs68	0.890869		EFdistAcc7	0.886598
	EFdistAccScaled1	0.866	135	DistTransformRS_Acc9	0.888954		DistTransformLP_Acc3	0.883505
	DistTransformLP_AccScaled7	0.870		ETvar1	0.887923		DistTransformLP_scaledAreaRoot8	0.881738
50	EF_TVH_Mean1	0.863		DistTransformLP_SortScaled10	0.891311		DistTransformRS_Sort1	0.887776
	VarRGB1	0.867		EFdistAccScaled3	0.890722	225	EllipticFourierAbs70	0.886156
	DistTransformLP_Acc5	0.873		SkeletonDistanceVariance1	0.891458		DistTransformRS_Sort7	0.88542
	EFdistVar7	0.867	140	EllipticFourierAbs71	0.887629		AverageChromaticityRed1	0.883947
	DistTransformLP_SortScaled3	0.871		AverageChromaticityBlue1	0.891458		DistTransformRS_Sort6	0.876878
55	CircularVariance1	0.871		DistTransformRS_SortScaled3	0.889838		Eccentricity1	0.880707
	EllipticFourierAbs75	0.871		EllipticFourierAbs8	0.889396	230	DistTransformLP_AccScaled6	0.878351
	DistTransformLP_scaledAreaRoot9	0.869		EllipticFourierAbs5	0.886451		EFdistVar4	0.879971
	EllipticFourierAbs33	0.872	145	EllipticFourierAbs13	0.890869		EllipticFourierAbs76	0.875994
	EllipticFourierAbs3	0.870		MeanDistBetweenHulls1	0.886598		DistTransformRS_Sort2	0.8757
60	DistTransformRS_scaledAreaRoot2	0.865		Rectangularity1	0.891753		DistTransformRS_scaledAreaRoot7	0.878351
	DistTransformLP_scaledAreaRoot10	0.869		HuMoments3	0.8919	235	EFdistAcc6	0.881001
	HuMoments1	0.864		DistTransformLP_AccScaled1	0.89028		EFdistVar2	0.87894
	EFdist7	0.861	150	EFdistAcc3	0.888513		DistTransformRS_SortScaled2	0.874669
	DistTransformRS_SortScaled1	0.864		DistTransformRS_Acc8	0.888365		DistTransformLP_SortScaled2	0.873638
65	EllipticFourierAbs32	0.870		EllipticFourierAbs44	0.888513		DistTransformRS_Sort4	0.869661
	DistTransformLP_Acc7	0.869		EFdistScaled4	0.892636	240	DistTransformRS_Sort5	0.865538
	DistTransformLP_scaledAreaRoot1	0.868		DistTransformRS_Acc3	0.891311		EFdist6	0.865538
	EFdistScaled9	0.866	155	DistTransformLP_scaledAreaRoot3	0.890869		EllipticFourierAbs16	0.869219
	DistTransformRS_AccScaled8	0.875		EllipticFourierAbs50	0.887923		DistTransformRS_Sort3	0.871429
70	EllipticFourierAbs65	0.873		DistTransformRS_Acc10	0.887482		HuMoments2	0.866274
	EFdistAccScaled4	0.871		EllipticFourierAbs24	0.887187	245	DistTransformRS_AccScaled2	0.865685
	EllipticFourierAbs19	0.872		EllipticFourierAbs17	0.889838		HuMoments5	0.859205
	DistTransformLP_AccScaled10	0.870	160	EFdistVar9	0.892784		ObjectPerimeter1	0.853461
	EllipticFourierAbs39	0.869		DistTransformRS_AccScaled1	0.890574		DistTransformRS_AccScaled4	0.85891
75	EllipticFourierAbs52	0.875		EFdist8	0.888513		DistTransformRS_SortScaled8	0.864801
	DistTransformLP_AccScaled5	0.871		EFdist1	0.888071	250	DistTransformRS_SortScaled5	0.866568
	DistTransformLP_Acc10	0.875		EFdistVar6	0.887187		ConvexHullArea1	0.866716
	DistTransformLP_AccScaled9	0.872	165	EllipticFourierAbs74	0.892047		ConvexHullPerimeter1	0.863623
	EllipticFourierAbs72	0.867		EllipticFourierAbs25	0.887187		HuMoments6	0.860383
80	DistTransformLP_scaledAreaRoot7	0.876		EllipticFourierAbs10	0.890574		Compactness1	0.858174
	EFdist3	0.878		EFdistScaled6	0.889102	255	HuMoments8	0.858321
	DistTransformLP_Sort10	0.870		EllipticFourierAbs57	0.892194		HuMoments7	0.862739
	EllipticFourierAbs40	0.870	170	EFdistAccScaled6	0.890133		ObjectArea1	0.860825
	DistTransformLP_scaledAreaRoot5	0.868		DistTransformLP_Acc8	0.887776		HuMoments4	0.858468
85	DistLPCorrelation1	0.872		EllipticFourierAbs21	0.883505		DistTransformRS_Sort8	0.854197
	EllipticFourierAbs59	0.872		EllipticFourierAbs56	0.889396	260	DistTransformRS_SortScaled7	0.854345
	DistTransformRS_Acc5	0.870		EllipticFourierAbs58	0.889249		DistTransformRS_SortScaled6	0.849043

Table J.3: Order of feature importance from forward selection tested on **foliages**. The classification accuracy (CA) indicates the classification rate after the feature has been added to the subset

Appendix K

Recursive feature elimination

Table K.1 shows the features ranged from the most to the least important feature based on recursive feature elimination. tested on plants.

Table K.2 shows the features ranged from the most to the least important feature based on recursive feature elimination tested on cotyledons.

Table K.3 shows the features ranged from the most to the least important feature based on recursive feature elimination tested on foliages.

#	Feature	CA	#	Feature	CA	#	Feature	CA
1	ExcessGreen1	0.470		DistTransformLP_Sort Scaled8	0.956	175	DistTransformRS_Acc4	0.945
	DistTransformLP_Sort2	0.690		EFdistAcc2	0.955		DistTransformRS_Acc1	0.945
	Compactness1	0.819	90	EllipticFourierAbs34	0.957		EFdistAcc7	0.946
	AverageChromaticityRed1	0.871		DistTransformLP_Sort Scaled3	0.953		EllipticFourierAbs13	0.944
5	DistTransformRS_Acc Scaled4	0.903		ConvexHullArea1	0.954		EllipticFourierAbs21	0.944
	EFdistAcc Scaled4	0.916		DistTransformLP_scaled AreaRoot5	0.954	180	DistTransformRS_Acc8	0.945
	AverageChromaticityGreen1	0.927		DistTransformRS_scaled AreaRoot10	0.954		DistTransformRS_Sort7	0.944
	Rectangularity1	0.932	95	DistTransformLP_Acc5	0.952		DistTransformLP_scaled AreaRoot3	0.945
	DistTransformLP_Sort Scaled5	0.939		DistTransformLP_Acc Scaled4	0.953		DistTransformLP_Sort10	0.944
10	EFdistAcc Scaled7	0.942		EFdist Scaled1	0.950		HuMoments4	0.945
	EFdist5	0.945		EllipticFourierAbs59	0.951	185	DistTransformLP_scaled AreaRoot4	0.944
	EFdist Scaled3	0.947		DistTransformLP_Sort Scaled6	0.952		DistTransformRS_Acc Scaled8	0.945
	EFdist4	0.945	100	EFdistAcc8	0.950		EllipticVariance1	0.947
	EllipticFourierAbs4	0.948		EllipticFourierAbs7	0.950		DistTransformRS_Sort Scaled5	0.946
15	EllipticFourierAbs58	0.948		EFdistAcc4	0.949		EF_TVH_Mean1	0.946
	DistTransformLP_scaled AreaRoot10	0.950		DistTransformRS_scaled AreaRoot5	0.950	190	DistLPCorrelation1	0.946
	EllipticFourierAbs33	0.950		DistTransformLP_scaled AreaRoot6	0.949		DistTransformRS_Acc Scaled2	0.948
	DistTransformLP_Acc Scaled10	0.948	105	EllipticFourierAbs51	0.949		EllipticFourierAbs29	0.947
	EFdistAcc Scaled3	0.951		AspectRatio1	0.949		EllipticFourierAbs8	0.947
20	Solidity1	0.950		EllipticFourierAbs74	0.949		EllipticFourierAbs67	0.947
	DistTransformRS_Acc6	0.951		DistTransformLP_Acc4	0.949	195	HuMoments3	0.947
	EllipticFourierAbs56	0.952		EFdistAcc3	0.947		DistTransformLP_Acc Scaled9	0.946
	EFdistAcc Scaled5	0.950	110	EllipticFourierAbs63	0.950		DistTransformRS_Acc Scaled6	0.946
	DistTransformLP_scaled AreaRoot1	0.952		DistTransformLP_Acc Scaled7	0.948		EllipticFourierAbs50	0.946
25	EllipticFourierAbs66	0.950		EllipticFourierAbs57	0.949		DistTransformVariance1	0.945
	EllipticFourierAbs25	0.951		SkeletonDistanceMax1	0.949	200	DistTransformLP_Acc Scaled6	0.945
	DistTransformRS_Acc Scaled3	0.952		EFdist Scaled9	0.950		EFdistAcc5	0.943
	EFdist3	0.951	115	DistTransformLP_Sort9	0.951		DistTransformRS_Acc Scaled7	0.944
	EFdist2	0.955		DistTransformLP_Acc2	0.950		DistTransformRS_Sort9	0.945
30	EllipticFourierAbs9	0.952		HuMoments6	0.950		EFdistVar5	0.943
	CircularVariance1	0.954		EFdistVar1	0.949	205	EllipticFourierAbs35	0.943
	EllipticFourierAbs55	0.954		ETvar1	0.950		EllipticFourierAbs49	0.943
	EllipticFourierAbs69	0.953	120	DistTransformRS_Acc2	0.951		DistTransformLP_scaled AreaRoot7	0.943
	EllipticFourierAbs39	0.951		EllipticFourierAbs70	0.950		DistTransformLP_Sort Scaled1	0.944
35	EFdistVar2	0.956		EFdistVar4	0.949		EFdist Scaled8	0.943
	EFdist8	0.954		DistTransformLP_Acc Scaled1	0.950	210	EFdist Scaled2	0.944
	DistTransformLP_Acc Scaled3	0.954		DistTransformRS_Sort Scaled9	0.948		EFdistVar6	0.941
	EllipticFourierAbs24	0.953	125	EllipticFourierAbs72	0.948		FractalDimension1	0.943
	EllipticFourierAbs1	0.955		EllipticFourierAbs18	0.947		ObjectPerimeter1	0.943
40	SkeletonDistanceMean1	0.953		RatioOfPrincipalAxes1	0.946		HuMoments7	0.942
	DistTransformRS_Sort3	0.955		EllipticFourierAbs26	0.945	215	EFdistVar7	0.943
	Convexity1	0.955		EllipticFourierAbs14	0.947		EllipticFourierAbs42	0.942
	EFdistAcc6	0.955	130	EllipticFourierAbs37	0.948		EllipticFourierAbs48	0.942
	VarRGB1	0.957		EllipticFourierAbs11	0.947		EllipticFourierAbs30	0.940
45	EllipticFourierAbs47	0.956		DistTransformLP_scaled AreaRoot8	0.945		DistTransformRS_Acc Scaled9	0.943
	EllipticFourierAbs76	0.956		EllipticFourierAbs75	0.946	220	EllipticFourierAbs60	0.941
	DistTransformLP_Acc7	0.958		DistTransformLP_Sort1	0.945		MinPlantThickness2	0.942
	EllipticFourierAbs23	0.956	135	MeanDistBetweenHulls1	0.944		DistTransformRS_Sort1	0.943
	EFdist Scaled4	0.955		EllipticFourierAbs28	0.945		EllipticFourierAbs22	0.943
50	EllipticFourierAbs12	0.956		DistTransformLP_Acc6	0.946		DistTransformRS_Sort Scaled2	0.942
	EllipticFourierAbs43	0.955		DistTransformRS_scaled AreaRoot6	0.946	225	FormFactor1	0.942
	EllipticFourierAbs20	0.955		DistTransformMean1	0.946		EllipticFourierAbs65	0.943
	EllipticFourierAbs41	0.956	140	EllipticFourierAbs3	0.947		EllipticFourierAbs77	0.943
	EFdistAcc Scaled8	0.957		DistTransformLP_Sort Scaled2	0.947		DistTransformLP_scaled AreaRoot9	0.943
55	DistTransformRS_Sort5	0.956		EFdist9	0.947		EllipticFourierAbs15	0.941
	EllipticFourierAbs40	0.957		DistTransformRS_Acc5	0.947	230	EFdistVar9	0.943
	EFdistVar3	0.957		DistTransformLP_Sort6	0.947		EllipticFourierAbs64	0.944
	EFdistVar8	0.954	145	Sphericity1	0.948		DistTransformLP_Sort4	0.945
	DistTransformLP_Acc1	0.957		DistTransformLP_Acc9	0.947		HuMoments5	0.944
60	EFdistAcc1	0.955		EllipticFourierAbs54	0.948		DistTransformLP_Sort3	0.945
	DistTransformLP_Sort7	0.956		EllipticFourierAbs27	0.946	235	DistTransformRS_scaled AreaRoot1	0.943
	EFdist Scaled7	0.955		DistTransformRS_Acc10	0.947		DistTransformRS_Acc Scaled1	0.943
	DistTransformRS_Sort Scaled3	0.957	150	DistTransformLP_Acc3	0.948		DistTransformLP_Acc Scaled5	0.941
	EllipticFourierAbs71	0.956		EFdist Scaled6	0.946		DistTransformRS_scaled AreaRoot2	0.942
65	DistTransformLP_Sort5	0.956		EllipticFourierAbs36	0.947		EFdist1	0.941
	DistTransformLP_Acc Scaled2	0.957		DistTransformRS_Sort4	0.946	240	EllipticFourierAbs19	0.941
	DistTransformRS_Acc Scaled5	0.958		EllipticFourierAbs6	0.944		DistTransformRS_Sort Scaled1	0.940
	DistTransformLP_scaled AreaRoot2	0.957	155	EllipticFourierAbs10	0.943		DistTransformRS_Acc7	0.942
	EllipticFourierAbs62	0.956		EllipticFourierAbs73	0.946		EllipticFourierAbs52	0.942
70	EllipticFourierAbs53	0.958		DistTransformRS_Sort Scaled6	0.945		DistTransformRS_Sort10	0.942
	EFdist7	0.956		DistTransformLP_Acc10	0.947	245	DistTransformRS_Acc3	0.941
	DistTransformRS_Sort Scaled8	0.958		DistTransformRS_scaled AreaRoot4	0.945		SkeletonDistanceLength1	0.942
	DistTransformRS_Sort2	0.956	160	ConvexHullPerimeter1	0.946		HuMoments8	0.941
	AverageChromaticityBlue1	0.958		ObjectArea1	0.945		DistTransformLP_Acc8	0.940
75	EllipticFourierAbs46	0.960		EFdistAcc Scaled6	0.944		EFdistAcc Scaled1	0.942
	DistTransformLP_Sort8	0.957		EllipticFourierAbs5	0.947	250	DistTransformRS_Sort6	0.940
	EllipticFourierAbs31	0.959		EF_TVH_Max1	0.945		EllipticFourierAbs68	0.941
	EFdist6	0.959	165	DistTransformRS_Acc9	0.946		SkeletonDistanceVariance1	0.940
	EllipticFourierAbs45	0.959		DistTransformRS_scaled AreaRoot7	0.946		DistTransformLP_Sort Scaled7	0.939
80	EllipticFourierAbs61	0.957		Eccentricity1	0.946		EllipticFourierAbs38	0.939
	DistTransformLP_Acc Scaled8	0.957		DistTransformLP_Sort Scaled9	0.943	255	DistTransformRS_scaled AreaRoot3	0.939
	EllipticFourierAbs44	0.957		DistTransformRS_Sort Scaled4	0.946		DistTransformRS_scaled AreaRoot8	0.939
	DistTransformLP_Sort Scaled10	0.955	170	HuMoments2	0.944		HuMoments1	0.938
	EllipticFourierAbs32	0.955		EllipticFourierAbs16	0.944		MinPlantThickness1	0.940
85	DistTransformLP_Sort Scaled4	0.956		DistTransformRS_Sort Scaled7	0.945		EFdist Scaled5	0.940
	EllipticFourierAbs17	0.955		EFdistAcc Scaled2	0.945	260	EllipticFourierAbs2	0.939
	EFdistAcc9	0.955		DistTransformRS_scaled AreaRoot9	0.946		DistTransformRS_Sort8	0.939

Table K.1: Order of feature importance from recursive feature elimination tested on **plants**.

#	Feature	CA	#	Feature	CA	#	Feature	CA
1	DistTransformRS_Sort Scaled9	0.590		EF_TVH_Mean1	0.885	175	EllipticFourierAbs19	0.879
	ObjectArea1	0.727		DistTransformRS_Acc3	0.886		DistTransformRS_Acc9	0.882
	DistTransformRS_Sort Scaled2	0.772	90	EllipticFourierAbs46	0.885		EllipticFourierAbs26	0.881
	DistTransformRS_AccScaled4	0.804		EllipticFourierAbs51	0.885		DistTransformRS_Acc10	0.879
5	DistTransformRS_AccScaled2	0.820		DistTransformLP_Acc6	0.887		EF_TVH_Max1	0.880
	VarRGB1	0.837		AverageChromaticityGreen1	0.885	180	EllipticFourierAbs55	0.880
	DistTransformLP_Sort Scaled6	0.845		ConvexHullArea1	0.885		EllipticFourierAbs37	0.878
	EllipticFourierAbs58	0.849	95	DistTransformRS_Sort5	0.887		EllipticFourierAbs73	0.882
	DistTransformLP_Sort8	0.859		DistTransformLP_scaledAreaRoot5	0.885		DistTransformVariance1	0.881
10	DistTransformRS_scaledAreaRoot5	0.865		DistTransformLP_scaledAreaRoot7	0.883		AspectRatio1	0.878
	DistTransformRS_Sort Scaled5	0.868		DistTransformLP_AccScaled8	0.882	185	RatioOfPrincipalAxes1	0.880
	DistTransformMean1	0.872		ExcessGreen1	0.887		DistTransformLP_Acc10	0.881
	DistTransformLP_Sort10	0.871	100	DistTransformRS_scaledAreaRoot10	0.881		EllipticFourierAbs42	0.880
	DistTransformRS_Sort Scaled7	0.874		DistTransformLP_AccScaled6	0.881		DistTransformLP_Sort Scaled9	0.875
15	DistTransformLP_Sort Scaled10	0.877		EllipticFourierAbs48	0.883		EllipticVariance1	0.878
	DistTransformRS_AccScaled8	0.878		EllipticFourierAbs77	0.881	190	EllipticFourierAbs32	0.877
	HuMoments4	0.881		DistTransformRS_SortScaled4	0.882		DistTransformLP_Sort1	0.875
	EllipticFourierAbs62	0.885	105	EFdistVar4	0.881		EllipticFourierAbs44	0.876
	DistTransformLP_Sort Scaled8	0.887		DistTransformRS_scaledAreaRoot4	0.881		DistTransformLP_Acc3	0.876
20	DistTransformLP_scaledAreaRoot4	0.888		EllipticFourierAbs72	0.880		EllipticFourierAbs23	0.875
	EllipticFourierAbs38	0.884		EFdistAccScaled5	0.880	195	EllipticFourierAbs15	0.874
	SkeletonDistanceMax1	0.883		EllipticFourierAbs30	0.881		DistTransformRS_AccScaled9	0.877
	EFdistScaled1	0.885	110	EFdistAcc6	0.877		SkeletonDistanceMean1	0.878
	EllipticFourierAbs43	0.884		DistTransformLP_scaledAreaRoot9	0.877		DistTransformLP_scaledAreaRoot3	0.878
25	DistTransformRS_AccScaled6	0.883		EllipticFourierAbs74	0.880		Convexity1	0.877
	DistTransformRS_Sort Scaled3	0.883		DistTransformRS_Sort1	0.879	200	DistTransformRS_Acc6	0.878
	ObjectPerimeter1	0.884		DistTransformLP_Sort5	0.881		EllipticFourierAbs7	0.879
	EllipticFourierAbs21	0.881	115	DistTransformLP_Sort Scaled1	0.877		HuMoments6	0.877
	DistTransformRS_Acc5	0.882		DistTransformLP_AccScaled2	0.880		DistTransformLP_Acc1	0.879
30	DistTransformLP_Sort6	0.881		DistTransformRS_scaledAreaRoot8	0.878		EllipticFourierAbs16	0.877
	EllipticFourierAbs68	0.882		EllipticFourierAbs12	0.878	205	DistTransformLP_scaledAreaRoot2	0.878
	HuMoments7	0.885		DistTransformRS_Sort6	0.878		EllipticFourierAbs8	0.876
	EFdistAccScaled7	0.886	120	DistTransformLP_Acc9	0.878		DistLPCorrelation1	0.877
	EllipticFourierAbs18	0.885		EllipticFourierAbs47	0.878		EllipticFourierAbs4	0.877
35	EFdistVar5	0.887		DistTransformRS_scaledAreaRoot1	0.878		EFdistAcc9	0.876
	EllipticFourierAbs60	0.887		DistTransformRS_scaledAreaRoot7	0.879	210	EFdistScaled8	0.878
	DistTransformLP_Sort Scaled5	0.886		EFdistAcc3	0.879		EFdist3	0.878
	MinPlantThickness1	0.888	125	EllipticFourierAbs11	0.879		HuMoments8	0.878
	DistTransformRS_scaledAreaRoot6	0.882		EFdistAcc1	0.877		EFdistVar8	0.877
40	EllipticFourierAbs31	0.884		MeanDistBetweenHulls1	0.878		EllipticFourierAbs34	0.877
	DistTransformLP_Sort Scaled3	0.884		EFdistAcc5	0.880	215	EllipticFourierAbs67	0.880
	DistTransformRS_scaledAreaRoot2	0.884		EFdistScaled2	0.878		EFdistAcc2	0.878
	EllipticFourierAbs10	0.883	130	EFdist6	0.880		EllipticFourierAbs64	0.877
	DistTransformLP_Sort2	0.883		EllipticFourierAbs71	0.878		EllipticFourierAbs69	0.878
45	DistTransformRS_Sort Scaled1	0.885		Eccentricity1	0.878		DistTransformLP_scaledAreaRoot10	0.876
	EllipticFourierAbs59	0.885		DistTransformLP_Acc7	0.880	220	EllipticFourierAbs17	0.876
	AverageChromaticityRed1	0.886		EllipticFourierAbs24	0.880		EFdistScaled5	0.876
	ETvar1	0.887	135	CircularVariance1	0.880		EllipticFourierAbs14	0.877
	DistTransformLP_Sort3	0.886		EFdist2	0.878		DistTransformRS_Sort8	0.876
50	DistTransformLP_Sort Scaled4	0.887		DistTransformRS_Acc2	0.879		DistTransformLP_AccScaled9	0.875
	EFdistAcc8	0.888		Sphericity1	0.878	225	DistTransformLP_Acc2	0.875
	EllipticFourierAbs1	0.885		SkeletonDistanceVariance1	0.878		EFdistVar9	0.877
	EFdistAccScaled3	0.887	140	EFdistVar7	0.880		EFdistScaled4	0.877
	EFdistAccScaled6	0.887		DistTransformRS_Sort2	0.881		EFdist1	0.875
55	SkeletonDistanceLength1	0.889		EFdistAccScaled2	0.879		EFdistScaled9	0.876
	EllipticFourierAbs65	0.885		DistTransformLP_Acc5	0.880	230	EllipticFourierAbs5	0.876
	EllipticFourierAbs57	0.885		DistTransformRS_AccScaled5	0.877		EllipticFourierAbs33	0.876
	EFdistVar1	0.885	145	DistTransformRS_scaledAreaRoot9	0.878		EllipticFourierAbs50	0.875
	EllipticFourierAbs13	0.884		EllipticFourierAbs39	0.879		DistTransformLP_Acc8	0.875
60	MinPlantThickness2	0.882		EFdist8	0.879		FractalDimension1	0.875
	DistTransformLP_AccScaled5	0.886		DistTransformLP_Sort4	0.881	235	DistTransformRS_Sort10	0.877
	EllipticFourierAbs28	0.883		DistTransformRS_Acc8	0.880		DistTransformRS_scaledAreaRoot3	0.874
	EFdistAcc4	0.884	150	EllipticFourierAbs49	0.879		DistTransformLP_AccScaled3	0.873
	EllipticFourierAbs66	0.886		DistTransformLP_AccScaled1	0.879		EllipticFourierAbs75	0.874
65	EllipticFourierAbs2	0.887		Rectangularity1	0.879		EllipticFourierAbs36	0.875
	EllipticFourierAbs41	0.886		EFdist9	0.881	240	ConvexHullPerimeter1	0.871
	DistTransformLP_AccScaled4	0.885		EllipticFourierAbs22	0.881		DistTransformRS_Sort Scaled6	0.868
	EFdistScaled6	0.887	155	EllipticFourierAbs40	0.880		EllipticFourierAbs52	0.868
	EFdistVar3	0.886		EFdist5	0.880		EllipticFourierAbs6	0.867
70	EllipticFourierAbs54	0.886		EFdistAcc7	0.880		EllipticFourierAbs3	0.868
	EFdist4	0.885		EFdistScaled7	0.880	245	EllipticFourierAbs29	0.867
	DistTransformLP_Sort Scaled2	0.885		EllipticFourierAbs20	0.880		DistTransformLP_AccScaled7	0.867
	DistTransformRS_AccScaled3	0.885	160	DistTransformLP_Acc4	0.879		HuMoments2	0.868
	DistTransformLP_scaledAreaRoot8	0.885		DistTransformRS_AccScaled1	0.878		EllipticFourierAbs53	0.868
75	EFdistScaled3	0.884		DistTransformRS_Sort3	0.879		EllipticFourierAbs27	0.869
	EllipticFourierAbs25	0.885		DistTransformLP_Sort9	0.881	250	EllipticFourierAbs70	0.868
	DistTransformLP_AccScaled10	0.887		Solidity1	0.880		DistTransformRS_Sort4	0.869
	DistTransformRS_Acc4	0.886	165	DistTransformRS_Sort7	0.879		DistTransformLP_Sort Scaled7	0.868
	EFdistAccScaled4	0.885		EllipticFourierAbs9	0.878		DistTransformLP_scaledAreaRoot6	0.868
80	DistTransformRS_Acc7	0.887		DistTransformLP_scaledAreaRoot1	0.878		EllipticFourierAbs56	0.867
	DistTransformRS_AccScaled7	0.885		DistTransformRS_SortScaled8	0.878	255	DistTransformRS_Sort9	0.867
	EllipticFourierAbs76	0.887		HuMoments1	0.881		Compactness1	0.868
	EFdist7	0.882	170	EFdistAccScaled1	0.880		DistTransformLP_Sort7	0.869
	EllipticFourierAbs63	0.885		FormFactor1	0.880		EllipticFourierAbs35	0.865
85	EFdistVar2	0.887		EFdistVar6	0.880		DistTransformRS_Acc1	0.867
	EllipticFourierAbs45	0.885		EFdistAccScaled8	0.881	260	HuMoments5	0.867
	EllipticFourierAbs61	0.885		HuMoments3	0.880		AverageChromaticityBlue1	0.867

Table K.2: Order of feature importance from recursive elimination tested on **cotyledons**.

#	Feature	CA	#	Feature	CA	#	Feature	CA
1	DistTransformRS_SortScaled5	0.537		EllipticFourierAbs55	0.879	175	EllipticFourierAbs75	0.882
	ObjectArea1	0.701		DistTransformRS_scaledAreaRoot6	0.874		ConvexHullPerimeter1	0.876
	DistTransformRS_AccScaled2	0.791	90	DistTransformLP_scaledAreaRoot1	0.878		EllipticFourierAbs74	0.880
	ConvexHullArea1	0.810		DistTransformRS_scaledAreaRoot10	0.880		EllipticFourierAbs46	0.878
5	DistTransformRS_SortScaled7	0.825		EFdist6	0.881		EllipticFourierAbs17	0.877
	HuMoments7	0.828		DistTransformRS_Sort5	0.881	180	HuMoments8	0.879
	EllipticVariance1	0.837		EllipticFourierAbs11	0.882		DistTransformLP_Sort4	0.884
	HuMoments1	0.834	95	EFdistVar5	0.876		EllipticFourierAbs21	0.883
	DistTransformRS_AccScaled4	0.849		DistTransformRS_Acc9	0.878		EFdistVar8	0.879
10	DistTransformRS_AccScaled6	0.848		EllipticFourierAbs54	0.881		DistTransformRS_Acc3	0.878
	HuMoments3	0.850		EllipticFourierAbs72	0.879	185	DistTransformLP_SortScaled3	0.882
	DistTransformRS_Sort7	0.855		EllipticFourierAbs4	0.876		EllipticFourierAbs42	0.880
	HuMoments2	0.867	100	DistTransformLP_AccScaled10	0.880		DistTransformRS_Acc5	0.877
	EllipticFourierAbs67	0.864		EllipticFourierAbs44	0.875		EllipticFourierAbs23	0.882
15	EFdistScaled5	0.863		EllipticFourierAbs28	0.876		EFdistScaled2	0.880
	EllipticFourierAbs30	0.863		EllipticFourierAbs61	0.880	190	EllipticFourierAbs27	0.885
	DistTransformRS_Sort10	0.860		EllipticFourierAbs14	0.877		Rectangularity1	0.881
	DistTransformLP_Sort7	0.863	105	EF_TVH_Mean1	0.881		DistTransformLP_Acc6	0.879
	DistTransformLP_SortScaled10	0.862		DistTransformRS_Acc7	0.879		DistTransformRS_Sort2	0.880
20	EllipticFourierAbs26	0.866		EllipticFourierAbs52	0.881		SkeletonDistanceLength1	0.881
	EFdist7	0.868		EllipticFourierAbs71	0.882	195	DistTransformLP_scaledAreaRoot6	0.878
	EFdistAcc2	0.866		DistTransformLP_Acc1	0.884		DistTransformLP_Sort2	0.878
	EFdistVar9	0.867	110	EllipticFourierAbs64	0.875		EllipticFourierAbs40	0.875
	EFdist5	0.866		EllipticFourierAbs16	0.880		DistTransformRS_Sort1	0.873
25	Eccentricity1	0.861		EllipticFourierAbs10	0.877		DistTransformRS_Sort6	0.878
	EllipticFourierAbs62	0.864		EllipticFourierAbs69	0.877	200	DistTransformRS_scaledAreaRoot3	0.877
	DistTransformMean1	0.861		EllipticFourierAbs7	0.879		DistTransformLP_AccScaled2	0.881
	EllipticFourierAbs25	0.867	115	DistTransformLP_scaledAreaRoot7	0.879		DistTransformLP_scaledAreaRoot10	0.880
	DistTransformRS_Sort4	0.861		HuMoments5	0.881		DistTransformLP_Sort5	0.884
30	DistTransformLP_AccScaled1	0.859		EllipticFourierAbs65	0.881		DistTransformLP_SortScaled5	0.883
	EllipticFourierAbs41	0.864		EllipticFourierAbs36	0.882	205	EllipticFourierAbs70	0.882
	DistTransformRS_SortScaled4	0.860		EllipticFourierAbs13	0.879		EllipticFourierAbs56	0.881
	DistTransformRS_AccScaled5	0.864	120	DistTransformLP_AccScaled5	0.879		EFdist8	0.877
	DistTransformLP_SortScaled4	0.863		DistTransformLP_Sort8	0.881		DistTransformLP_Acc9	0.882
35	EFdistAccScaled3	0.870		DistTransformRS_Acc2	0.881		EFdist1	0.881
	DistTransformLP_Acc2	0.861		EllipticFourierAbs50	0.886	210	EFdistVar7	0.879
	DistTransformRS_Sort8	0.868		EFdist2	0.885		Compactness1	0.883
	EllipticFourierAbs60	0.865	125	EllipticFourierAbs31	0.882		EFdistScaled7	0.881
	DistTransformLP_Acc4	0.866		EFdistScaled8	0.875		DistTransformRS_AccScaled7	0.878
40	DistTransformRS_SortScaled8	0.878		DistTransformLP_Acc3	0.878		EFdistAccScaled4	0.880
	DistTransformLP_SortScaled7	0.872		EllipticFourierAbs18	0.878	215	DistTransformLP_AccScaled6	0.880
	DistTransformRS_Acc6	0.871		EllipticFourierAbs33	0.881		EFdistAcc7	0.880
	EFdistScaled4	0.870	130	FormFactor1	0.881		EllipticFourierAbs6	0.880
	EFdistAcc6	0.872		CircularVariance1	0.878		DistTransformRS_SortScaled9	0.878
45	EFdistVar4	0.873		AspectRatio1	0.884		EllipticFourierAbs43	0.880
	DistTransformLP_AccScaled8	0.874		EllipticFourierAbs73	0.881	220	SkeletonDistanceVariance1	0.882
	DistTransformRS_Sort9	0.871		EFdistAcc9	0.878		DistTransformRS_SortScaled2	0.872
	DistLPCorrelation1	0.872	135	EllipticFourierAbs49	0.881		ExcessGreen1	0.875
	EFdistAcc3	0.870		DistTransformRS_scaledAreaRoot4	0.879		AverageChromaticityRed1	0.876
50	DistTransformLP_scaledAreaRoot4	0.878		DistTransformLP_AccScaled9	0.878		EFdistScaled3	0.873
	EllipticFourierAbs39	0.868		EllipticFourierAbs29	0.881	225	EFdistAccScaled2	0.876
	EllipticFourierAbs22	0.875		EllipticFourierAbs45	0.878		DistTransformRS_Sort3	0.876
	EFdistAccScaled1	0.874	140	DistTransformLP_SortScaled8	0.880		DistTransformRS_scaledAreaRoot1	0.882
	EFdistAccScaled8	0.869		EllipticFourierAbs58	0.883		RatioOfPrincipalAxes1	0.878
55	EllipticFourierAbs24	0.875		EllipticFourierAbs8	0.885		DistTransformRS_SortScaled3	0.881
	MeanDistBetweenHulls1	0.875		EllipticFourierAbs9	0.887	230	EllipticFourierAbs68	0.875
	EllipticFourierAbs57	0.870		DistTransformRS_AccScaled9	0.875		DistTransformRS_Acc10	0.875
	EFdistAcc5	0.874	145	DistTransformLP_Sort3	0.877		EFdist9	0.877
	EFdistAccScaled7	0.871		DistTransformVariance1	0.880		DistTransformLP_AccScaled7	0.878
60	Sphericity1	0.870		DistTransformRS_SortScaled1	0.880		DistTransformLP_Acc8	0.878
	AverageChromaticityGreen1	0.873		EFdistScaled6	0.882	235	EllipticFourierAbs19	0.878
	DistTransformLP_Sort10	0.877		EFdistAccScaled5	0.880		Convexity1	0.875
	EFdistAccScaled6	0.875	150	DistTransformLP_Acc7	0.879		DistTransformLP_SortScaled2	0.874
	DistTransformLP_SortScaled9	0.874		DistTransformLP_Sort1	0.883		DistTransformRS_AccScaled1	0.879
65	DistTransformRS_scaledAreaRoot7	0.875		DistTransformLP_SortScaled1	0.880		DistTransformRS_Acc1	0.876
	DistTransformLP_Sort9	0.871		DistTransformLP_SortScaled6	0.881	240	DistTransformRS_AccScaled8	0.875
	EllipticFourierAbs37	0.876		EllipticFourierAbs63	0.880		DistTransformLP_scaledAreaRoot8	0.872
	DistTransformLP_Acc10	0.876	155	EFdistVar6	0.883		DistTransformRS_scaledAreaRoot8	0.876
	EllipticFourierAbs51	0.873		DistTransformLP_scaledAreaRoot5	0.883		EllipticFourierAbs12	0.871
70	DistTransformRS_scaledAreaRoot2	0.877		DistTransformRS_scaledAreaRoot9	0.877		EllipticFourierAbs38	0.873
	VarRGB1	0.878		EllipticFourierAbs20	0.883	245	HuMoments4	0.868
	DistTransformRS_scaledAreaRoot5	0.874		EllipticFourierAbs5	0.884		EFdistScaled9	0.873
	EFdistVar1	0.876	160	EllipticFourierAbs53	0.883		DistTransformLP_Acc5	0.871
	EllipticFourierAbs76	0.877		Solidity1	0.878		AverageChromaticityBlue1	0.871
75	EllipticFourierAbs77	0.879		EFdistAcc8	0.879		EFdistVar3	0.866
	EllipticFourierAbs34	0.879		EllipticFourierAbs59	0.883	250	DistTransformLP_AccScaled3	0.873
	EllipticFourierAbs35	0.878		EFdistVar2	0.881		EllipticFourierAbs2	0.869
	EllipticFourierAbs66	0.874	165	EFdistAcc4	0.880		ObjectPerimeter1	0.860
	FractalDimension1	0.880		DistTransformRS_Acc4	0.886		DistTransformRS_AccScaled3	0.857
80	EllipticFourierAbs48	0.875		EllipticFourierAbs3	0.885		DistTransformLP_scaledAreaRoot2	0.857
	EllipticFourierAbs47	0.880		EllipticFourierAbs15	0.886	255	SkeletonDistanceMax1	0.859
	EFdistScaled1	0.879		MinPlantThickness1	0.884		DistTransformRS_SortScaled6	0.845
	DistTransformLP_scaledAreaRoot3	0.876	170	DistTransformRS_Acc8	0.876		EllipticFourierAbs32	0.845
	ETvar1	0.876		DistTransformLP_Sort6	0.881		DistTransformLP_AccScaled4	0.849
85	EF_TVH_Max1	0.881		EllipticFourierAbs1	0.883		EFdist4	0.849
	EFdist3	0.878		SkeletonDistanceMean1	0.881	260	DistTransformLP_scaledAreaRoot9	0.848
	MinPlantThickness2	0.881		EFdistAcc1	0.884		HuMoments6	0.846

Table K.3: Order of feature importance from recursive elimination tested on **foliages**.

Appendix L

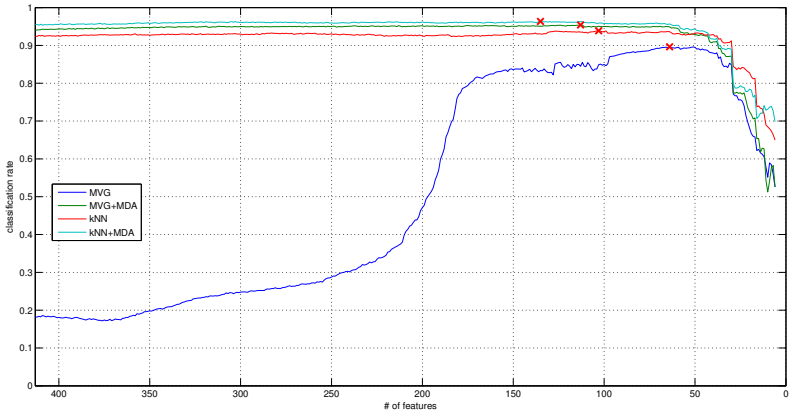
Backward elimination MDA

The highest performing subset of features for the classifiers kNN and MVG with and without MDA are shown in Table L.1.

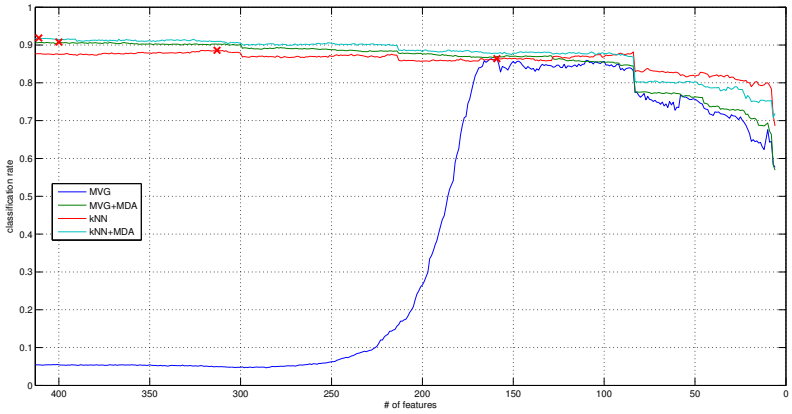
	MVG		MVG+MDA		kNN		kNN+MDA	
	nFeatures	CA	nFeatures	CA	nFeatures	CA	nFeatures	CA
Plant	64	0.8964	113	0.9542	103	0.9387	135	0.9634
Cotyledon	159	0.8638	400	0.9078	313	0.8861	411	0.9187
Foliage	130	0.9069	282	0.9195	143	0.9097	305	0.9285

Table L.1: Classification accuracy (CA) of kNN for the optimal subset in accordance with Recursive Feature Elimination using Multiple Discriminant Analysis.

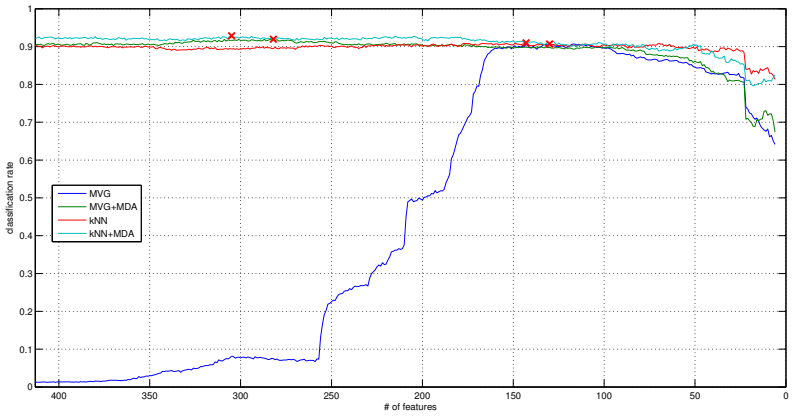
The running classification accuracy when using the MDA backward elimination procedure for kNN, kNN+MDA, MVG and MVG+MDA is seen in figure L.1. The MDA backward elimination procedure works well for plants as the classification accuracies peaks with a low number of features. The procedure do not perform optimal for cotyledon and foliage leaves as the optimal classification accuracies are achieved with a high number of features. The reason for this is presumably that classes are not divided in single Gaussian like clusters.



(a) Classification accuracy for a decreasing number of features for plants.



(b) Classification accuracy for a decreasing number of features for cotyledons.



(c) Classification accuracy for a decreasing number of features for foliage.

Figure L.1: Classification accuracy for a decreasing number of features for plants, cotyledons and foliage leaves for kNN, kNN+MDA, MVG and MVG+MDA using the MDA backward elimination procedure.

Appendix M

Belief Matrix Example

In the three tables below the confusion matrix and the belief matrix of the kNN classifier using $k = 1, 1, 1$ is presented.

		True label							
	Spe.	1	2	3	4	5	6	7	Tot.
Result label	1	242	0	2	0	0	0	0	244
	2	2	185	0	0	1	0	0	188
	3	3	8	316	1	0	1	4	333
	4	1	2	2	567	8	12	3	595
	5	1	3	2	2	691	9	2	710
	6	0	0	0	4	3	232	1	240
	7	1	0	3	1	0	1	120	126

		True label						
	Spe.	1	2	3	4	5	6	7
Result label	1	0.992	0	0.008	0	0	0	0
	2	0.011	0.984	0	0	0.005	0	0
	3	0.009	0.024	0.949	0.003	0	0.003	0.012
	4	0.002	0.003	0.003	0.953	0.013	0.020	0.005
	5	0.001	0.004	0.003	0.003	0.973	0.013	0.003
	6	0	0	0	0.017	0.013	0.967	0.004
	7	0.008	0	0.024	0.008	0	0.008	0.952

Table M.1: An example of a confusion and belief matrix for plants using the kNN classifier using $k = 1$.

		True label							
	Spe.	101	102	103	104	105	106	107	Tot.
Result label	101	164	0	4	1	3	3	7	182
	102	0	0	0	0	0	0	0	0
	103	13	0	799	8	7	2	10	839
	104	5	0	9	318	16	50	6	404
	105	5	0	7	24	1140	41	7	1224
	106	2	0	0	48	24	266	17	357
	107	5	0	13	5	9	24	347	403

		True label						
	Spe.	101	102	103	104	105	106	107
Result label	101	0.901	0	0.022	0.005	0.016	0.016	0.038
	102	0	0	0	0	0	0	0
	103	0.015	0	0.952	0.010	0.008	0.002	0.012
	104	0.012	0	0.022	0.787	0.040	0.124	0.015
	105	0.004	0	0.006	0.020	0.931	0.033	0.006
	106	0.006	0	0	0.134	0.067	0.745	0.048
	107	0.012	0	0.032	0.012	0.022	0.060	0.861

Table M.2: An example of a confusion and belief matrix for cotyledon leaves using the kNN classifier using $k = 1$.

		True label							
	Spe.	201	202	203	204	205	206	207	Tot.
Result label	201	0	0	0	0	0	0	0	0
	202	0	0	0	0	0	0	0	0
	203	0	0	13	5	0	6	0	24
	204	0	0	2	430	8	18	14	472
	205	0	0	0	10	274	20	8	312
	206	0	0	8	13	19	250	9	299
	207	0	0	0	15	8	28	200	251

		True label						
	Spe.	201	202		204	205	206	207
Result label	201	0	0	0	0	0	0	0
	202	0	0	0	0	0	0	0
	203	0	0	0.542	0.208	0	0.250	0
	204	0	0	0.004	0.911	0.017	0.038	0.030
	205	0	0	0	0.032	0.878	0.064	0.026
	206	0	0	0.027	0.043	0.064	0.836	0.030
	207	0	0	0	0.060	0.032	0.112	0.797

Table M.3: An example of a confusion and belief matrix for foliage leaves using the kNN classifier using $k = 1$.

Appendix N

Wrong identifications

Five examples of wrong identifications are shown for each species to present difficulties for the different species[1]. Figure N.1 shows 5 of the 16 Maize (species 1) plants that have been classified incorrectly out of the 250 plant samples. Figure N.2 shows 5 of the 9 Maize (species 1) plants that have been classified incorrectly out of the 198 plant samples. Figure N.3 shows 5 of the 5 Maize (species 2) plants that have been classified incorrectly out of the 459 plant samples. Figure N.4 shows 5 of the 9 Maize (species 3) plants that have been classified incorrectly out of the 577 plant samples. Figure N.5 shows 5 of the 9 Maize (species 4) plants that have been classified incorrectly out of the 706 plant samples. Figure N.6 shows 5 of the 16 Maize (species 5) plants that have been classified incorrectly out of the 274 plant samples. Figure N.7 shows 5 of the 10 Maize (species 6) plants that have been classified incorrectly out of the 340 plant samples.

[1] images of all misidentifikations can be found on the disc attached to the report

(a) Species:1 Result:2 **(b)** Species:1 Result:7 **(c)** Species:1 Result:5 Results:5 105 **(d)** Species:1 Result:3 Results:1 103 **(e)** Species:1 Result:3 Results:3 101

Figure N.1: wrong classifications of species 1.

(a) Species:2 Result:4 **(b)** Species:2 Result:5 **(c)** Species:2 Result:5 **(d)** Species:2 Result:3 **(e)** Species:2 Result:3

Figure N.2: wrong classifications of species 2.

(a) Species:3 Result:4 **(b)** Species:3 Result:1 Results:1 105 **(c)** Species:3 Result:5 **(d)** Species:3 Result:4 **(e)** Species:3 Result:6 Results:101 206

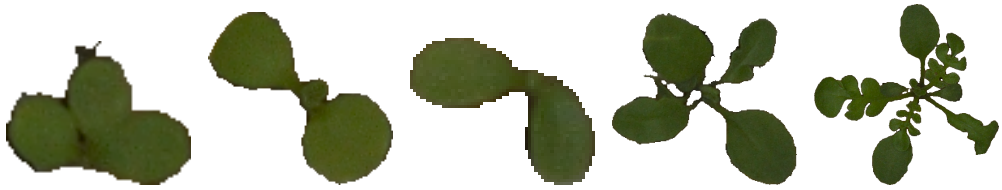
Figure N.3: wrong classifications of species 3.

(a) Species:4 Result:2 **(b)** Species:4 Result:6 Results:6 104 104 206 **(c)** Species:4 Result:6 Results:6 104 **(d)** Species:4 Result:5 Results:4 105 105 206 **(e)** Species:4 Result:6 Results:4 206

Figure N.4: wrong classifications of species 4.

(a) Species:5 Result:4 **(b)** Species:5 Result:4 **(c)** Species:5 Result:7 Results:4 107 **(d)** Species:5 Result:6 Results:5 106 206 **(e)** Species:5 Result:4 Results:4 104 204

Figure N.5: wrong classifications of speices 5.



(a) Species:6 Result:7 **(b)** Species:6 Result:5 Results:6 105 **(c)** Species:6 Result:4 Results:4 104 **(d)** Species:6 Result:7 Results:6 107 207 **(e)** Species:6 Result:4 Results:6 204

Figure N.6: wrong classifications of species 6.

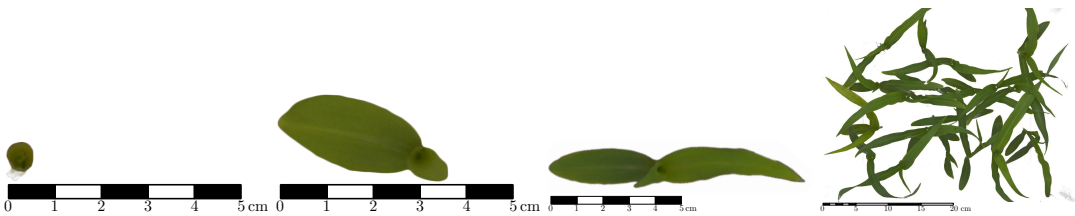
(a) Species:7 Result:2 **(b)** Species:7 Result:4 **(c)** Species:7 Result:3 **(d)** Species:7 Result:6 **(e)** Species:7 Result:4

Figure N.7: wrong classifications of species 7.

Appendix O

Plant samples

The following figures shows selected samples for the 12 species from the dataset.



(a) BBCH 10 (b) BBCH 11 (c) BBCH 12 (d) BBCH 14

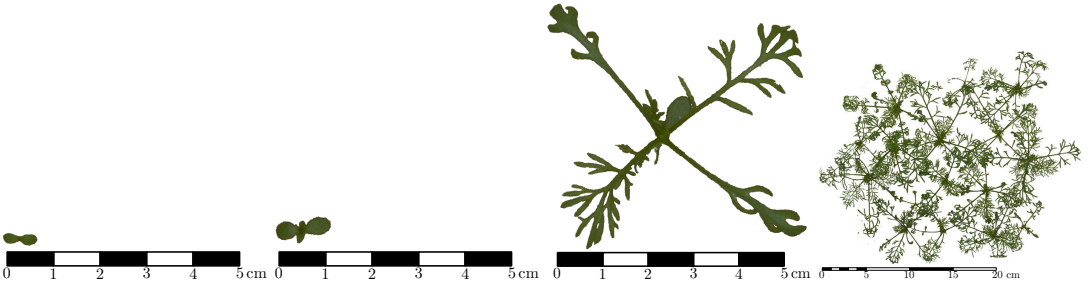
Figure O.1: Species 1: Maize

(a) BBCH 10 (b) BBCH 12 (c) BBCH 13 (d) BBCH 15

Figure O.2: Species 2: Winter Wheat

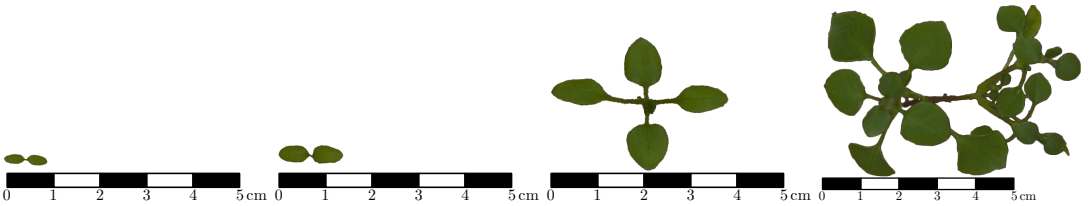
(a)BBCH 12 (b)BBCH 12 (c)BBCH 13 (d)BBCH 14

Figure O.3: Species 3: Sugar beet



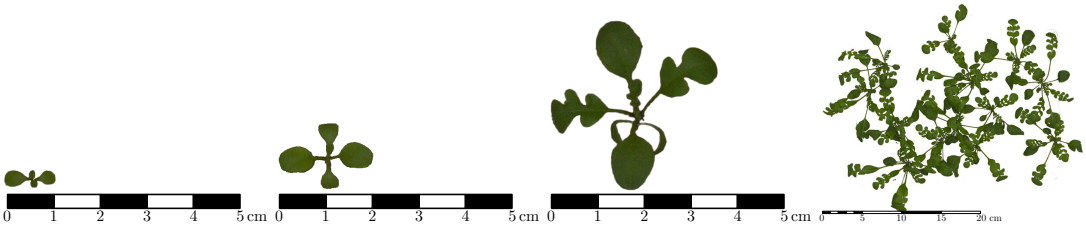
(a)BBCH 12 (b)BBCH 12 (c)BBCH 13 (d)BBCH 14

Figure O.4: Species 4: Scentless mayweed



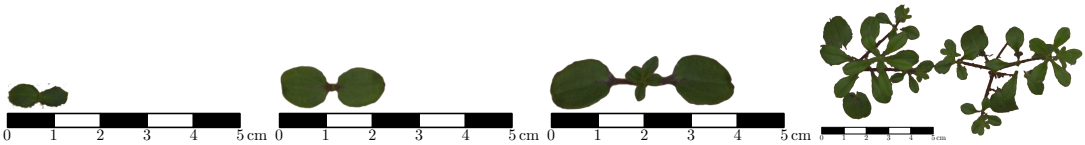
(a)BBCH 12 (b)BBCH 12 (c)BBCH 14 (d)BBCH 23

Figure O.5: Species 5: Chickweed



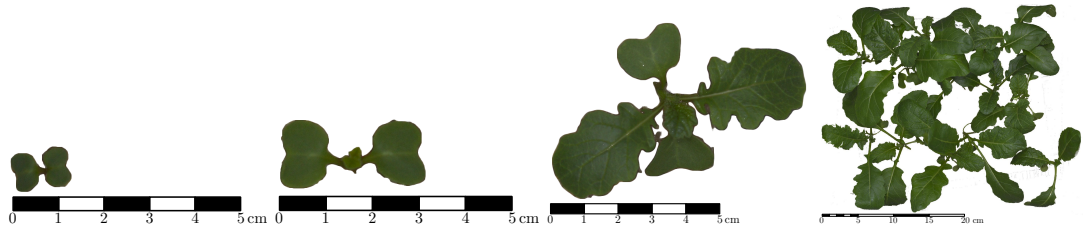
(a)BBCH 14 (b)BBCH 14 (c)BBCH 17 (d)BBCH 20

Figure O.6: Species 6: Shepherd's-purse



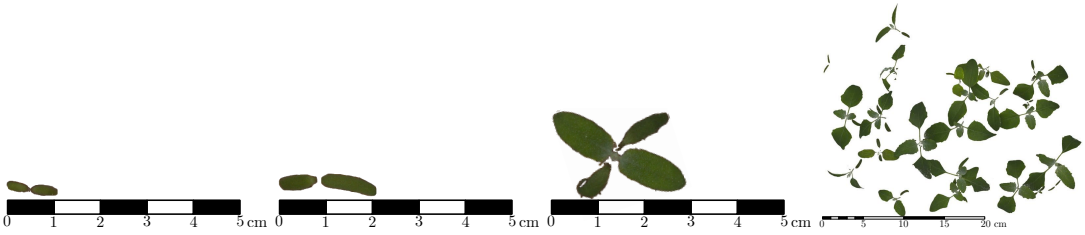
(a)BBCH 12 (b)BBCH 12 (c)BBCH 13 (d)BBCH 33

Figure O.7: Species 7: Cleavers



(a)BBCH 12 (b)BBCH 12 (c)BBCH 14 (d)BBCH 33

Figure O.8: Species 9: Charlock

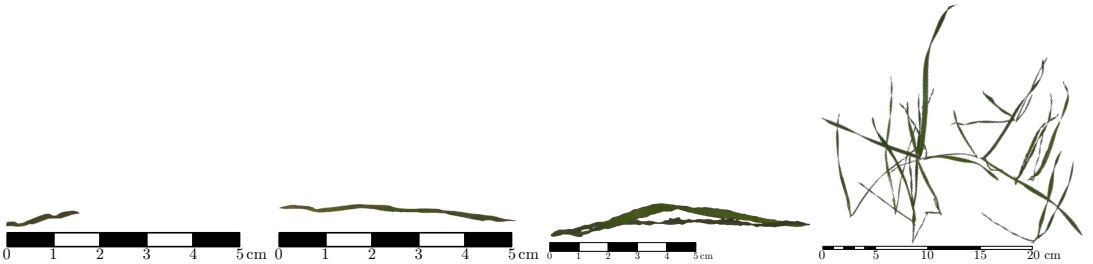


(a)BBCH 12 (b)BBCH 12 (c)BBCH 14 (d)BBCH 16

Figure O.9: Species 10: Fat Hen

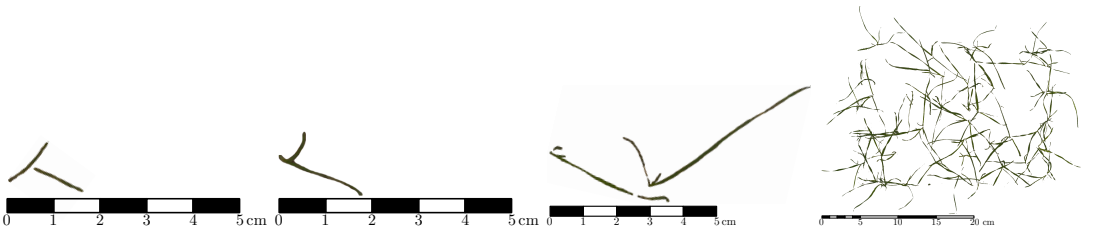
(a)BBCH 2 **(b)**BBCH 2 **(c)**BBCH 3 **(d)**BBCH 19

Figure O.10: Species 11: Cranes-bill



(a)BBCH 11 **(b)**BBCH 11 **(c)**BBCH 12 **(d)**BBCH 14

Figure O.11: Species 13: Black grass



(a)BBCH 12 **(b)**BBCH 12 **(c)**BBCH 13 **(d)**BBCH 15

Figure O.12: Species 14: Loose Silky-bent

Bibliography

[1] L. Neumeister, *Pesticide use reporting*. PAN Germany, 2002.

[2] E. C. Oerke, "Crop losses to pests," *The Journal of Agricultural Science*, vol. 144, pp. 31–43, 2 2006.

[3] D. Slaughter, D. Giles, and D. Downey, "Autonomous robotic weed control systems: A review," *Computers and Electronics in Agriculture*, vol. 61, no. 1, pp. 63 – 78, 2008.

[4] J. Hodgson, "The nature, ecology, and control of canada thistle," 1968.

[5] "Directive 2002/91/ec of the european parliament and of the council." Official Journal of the European Communities, December 2002.

[6] J. L. Helga Willer, "The european market for organic food 2011," 2013.

[7] N. D. Tillett, T. Hague, A. C. Grundy, and A. P. Dedousis, "Mechanical within-row weed control for transplanted crops using computer vision," *Biosystems Engineering*, vol. 99, no. 2, pp. 171–178, 2008.

[8] T. Bakker, *An autonomous robot for weed control: design, navigation and control*. Wageningen Universiteit, 2009.

[9] L. N. Jørgensen, E. Noe, A.-M. Langvad, P. Rydahl, J. E. Jensen, J. E. Ørum, and H. Pinnschmidt, *Vurdering af Planteværn Onlines økonomiske og miljømæssige effekt*, vol. 115 of *Bekæmpelsesmiddelforskning fra Miljøstyrelsen*. Miljøstyrelsen, 2007.

[10] T. W. Berge, "Site-specific weed management (sswm)," *Nordic Association of Agricultural Scientists*, vol. 7, no. 9, pp. 95–99, 2011.

[11] F. Poulsen Engineering ApS, "Visionweeding." `http://visionweeding.com/`. [Online; accessed 16-12-2013].

[12] E. Graglia, "Importance of herbicide concentration, number of droplets and droplet size on growth of Solanum nigrum L, using droplet application of glyphosate," in *XIIeme Colloque International sur la Biologie des Mauvaises Herbes*, pp. 527–533, 2004.

[13] S. Christensen, H. T. Søgaard, P. Kudsk, I. Lund, M. Nørremark, E. S. Nadimi, and R. N. Jørgensen, "Intelligent weed control technologies," 2008.

[14] H. T. Søgaard, "Weed classification by active shape models," *Biosystems Engineering*, vol. 91, no. 3, pp. 271 – 281, 2005.

[15] C. Xia, J.-M. Lee, Y. Li, Y.-H. Song, B.-K. Chung, and T.-S. Chon, "Plant leaf detection using modified active shape models," *Biosystems Engineering*, vol. 116, no. 1, pp. 23 – 35, 2013.

[16] T. M. Giselsson, "Real time crops and weeds classification from top down images of plant canopies," tech. rep., University of Southern Denmark, Faculty of Engineering, 2010.

[17] T. M. Giselsson, H. S. Midtiby, and R. N. Jørgensen, "Seedling discrimination with shape features derived from a distance transform," *Sensors Journal*, 2013.

[18] J. C. Neto, G. E. Meyer, D. D. Jones, and A. K. Samal, "Plant species identification using elliptic fourier leaf shape analysis," *Comput. Electron. Agric.*, vol. 50, pp. 121–134, Feb. 2006.

[19] D. J. Hearn, "Shape analysis for the automated identification of plants from images of leaves," *Taxon*, vol. 58, no. 3, pp. pp. 934–954, 2009.

[20] J.-X. Du, X.-F. Wang, and X. Gu, "Shape matching and recognition base on genetic algorithm and application to plant species identification," in *Advances in Intelligent Computing* (D.-S. Huang, X.-P. Zhang, and G.-B. Huang, eds.), vol. 3644 of *Lecture Notes in Computer Science*, pp. 282–290, Springer Berlin Heidelberg, 2005.

[21] B. Åstrand, *Vision based perception for mechatronic weed control.* PhD thesis, Chalmers University of Technology 2005, 2005.

[22] T. Hague, J. Marchant, and N. Tillett, "Ground based sensing systems for autonomous agricultural vehicles," *Computers and Electronics in Agriculture*, vol. 25, no. 1–2, pp. 11 – 28, 2000.

[23] M. R. Ehsani, S. K. Upadhyaya, and M. L. Mattson, "Seed location mapping using rtk gps," *Transactions of the ASAE*, 2004.

[24] R. B. Brown and S. D. Noble, "Site-specific weed management. sensing requirements - what do we need to see?," *Weed Science Society of America*, 2005.

[25] "Growth stages of mono-and dicotyledonous plants," tech. rep., Federal Biological Research Centre for Agriculture and Forestry, 2001.

[26] Planteværn online, "Crop - growthstages." `http://www.ipmdss.dk/cp/SeasonPlan/CropScale.asp?ID=djf&Language=en`, 2013. [Online; accessed Nov-2013].

[27] B. Åstrand and A.-J. Baerveldt, "An agricultural mobile robot with vision-based perception for mechanical weed control," *Autonomous Robots*, vol. 13, pp. 21–35, jul 2002.

[28] R. C. Gonzalez and R. E. Woods, *Digital image processing*. Upper Saddle River, N.J.: Prentice Hall, 2008.

[29] L. Tang, L. F. Tian, and B. L. Steward, "Color image segmentation with genetic algorithm for in-field weed sensing," *Agricultural and Biosystems Engineering*, 2000.

[30] W. D. M., M. G. E., V. B. K., and M. D. A., "Color indices for weed identification under various soil, residue, and lighting conditions," *Transactions of the ASAE*, vol. 38, no. 1, pp. 259–269, 1995.

[31] G. E. Meyer and J. C. Neto, "Verification of color vegetation indices for automated crop imaging applications," *Computers and Electronics in Agriculture*, vol. 63, no. 2, pp. 282 – 293, 2008.

[32] M. Weis, C. Gutjahr, V. R. Ayala, R. Gerhards, C. Ritter, and F. Schölderle, "Precision farming for weed management: techniques," *Gesunde Pflanzen*, vol. 60, no. 4, pp. 171–181, 2008.

[33] F. Toselli and J. Bodechtel, *Imaging Spectroscopy: Fundamentals and Prospective Applications*. EUR 12898, Springer, 1992.

[34] A. T. Nieuwenhuizen, *Automated detection and control of volunteer potato plants*. Wageningen University, 2009.

[35] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 9, no. 1, 1979.

[36] G. E. Meyer, J. C. Neto, D. D. Jones, and T. W. Hindman, "Intensified fuzzy clusters for classifying plant, soil, and residue regions of interest from color images," *Computers and Electronics in Agriculture*, 2004.

[37] J. a. C. Neto, G. E. Meyer, and D. D. Jones, "Individual leaf extractions from young canopy images using gustafson-kessel clustering and a genetic algorithm," *Comput. Electron. Agric.*, vol. 51, pp. 66–85, Apr. 2006.

[38] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2009.

[39] D. Graves and W. Pedrycz, "Fuzzy c-means, gustafson-kessel fcm, and kernel-based fcm: A comparative study," in *Analysis and Design of Intelligent Systems using Soft Computing Techniques* (P. Melin, O. Castillo, E. Ramírez, J. Kacprzyk, and W. Pedrycz, eds.), vol. 41 of *Advances in Soft Computing*, pp. 140–149, Springer Berlin Heidelberg, 2007.

[40] R. Babuška, *Computational Intelligence in Modeling and Control*. Delft University of Technology, 2009.

[41] D. E. Gustafson and W. C. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," *Decision and Control including the 17th Symposium on Adaptive Processes, 1978 IEEE Conference on Date 10-12 Jan. 1979*, pp. 761–776, 1979.

[42] A. Gupta and R. Bowden, "Fuzzy encoding for image classification using gustafson-kessel algorithm," in *ICIP*, pp. 3137–3140, IEEE, 2012.

[43] B. Minasny, "Fuzzy k-means," 2013.

[44] H. Midtiby, T. Mosgaard Giselsson, and R. Jørgensen, "Estimating plant stem emerging points (pseps) of sugar of beets in early growth stages," *Biosystems Engineering*, vol. 111, no. 1, pp. 83–90, 2012.

[45] G. Cerutti, L. Tougne, J. Mille, A. Vacavant, and D. Coquin, "Understanding Leaves in Natural Images - A Model-Based Approach for Tree Species Identification," *Computer Vision and Image Understanding*, vol. 117, pp. 1482–1501, Oct. 2013.

[46] R. Szeliski, *Computer Vision: Algorithms and Applications*. New York, NY, USA: Springer-Verlag New York, Inc., 1st ed., 2010.

[47] S. Seitz, "Computer vision: Projective geometry." `http://courses.cs.washington.edu/courses/cse576/08sp/lectures/Projective.ppt`, 2008.

[48] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. New York, NY, USA: Cambridge University Press, 2 ed., 2003.

[49] F. P. Kuhl and C. R. Giardina, "Elliptic fourier features of a closed contour," *Computer Graphics and Image Processing*, vol. 18, pp. 236–258, March 1982.

[50] J. S. Cope, D. P. A. Corney, J. Y. Clark, P. Remagnino, and P. Wilkin, "Plant species identification using digital morphometrics: A review.," *Expert Syst. Appl.*, vol. 39, no. 8, pp. 7562–7573, 2012.

[51] M. Peura and J. Iivarinen, "Efficiency of simple shape descriptors," in *In Aspects of Visual Form*, pp. 443–451, World Scientific, 1997.

[52] M. Persson and B. Åstrand, "Classification of crops and weeds extracted by active shape models," *Biosystems Engineering*, vol. 100, no. 4, pp. 484 – 497, 2008.

[53] G. E. Meyer, T. Mehta, M. F. Kocher, D. A. Mortensen, and A. Samal, "Textural imaging and discriminant analysis for distinguishing weeds for spot spraying," *American Society of Agricultural Engineers*, 1998.

[54] J.-X. Du, X. Wang, and G.-J. Zhang, "Leaf shape based plant species recognition.," *Applied Mathematics and Computation*, vol. 185, no. 2, pp. 883–893, 2007.

[55] G. Toussaint, "Solving geometric problems with the rotating calipers," *IEEE MELECON '83*, 1983.

[56] D. Dimitrov, C. Knauer, K. Kriegel, and G. Rote, "On the bounding boxes obtained by principal component analysis," in *Proc. 22nd European Workshop on Computational Geometry*, (Delphi, Greece), pp. 193–196, 2006.

[57] J. C. Russ, *The Image Processing Handbook, Sixth Edition.* Taylor & Francis, 2011.

[58] Wikipedia, "Moment (mathematics) — Wikipedia, the free encyclopedia," 2013. [Online; accessed 2-December-2013].

[59] C.-C. Chen, "Improved moment invariants for shape discrimination.," *Pattern Recognition*, vol. 26, no. 5, pp. 683–686, 1993.

[60] I. Badami, "Hu moments of order 3," 2013.

[61] O. M. Bruno, R. de Oliveira Plotze, M. Falvo, and M. de Castro, "Fractal dimension applied to plant identification," *Information Sciences*, vol. 178, no. 12, pp. 2722 – 2733, 2008.

[62] R. Bowles, "Legendre polynomials." 2017.

[63] S. Wu, "Legendre polynomial fitting," 2008.

[64] G. Blanchard, O. Bousquet, and P. Massart, "Statistical performance of support vector machines," tech. rep., 2004.

[65] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, Fourth Edition.* Academic Press, 4th ed., 2009.

[66] Wikipedia, "Support vector machine — Wikipedia, the free encyclopedia." http://en.wikipedia.org/wiki/Support_vector_machine, 2013. [Online; accessed May-2013].

[67] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," tech. rep., Department of Computer Science, National Taiwan University, 2003.

[68] J. Milgram, M. Cheriet, and R. Sabourin, ""One Against One" or "One Against All": Which One is Better for Handwriting Recognition with SVMs?," in *Tenth International Workshop on Frontiers in Handwriting Recognition* (G. Lorette, ed.), (La Baule (France)), Université de Rennes 1, Suvisoft, Oct. 2006. http://www.suvisoft.com Université de Rennes 1.

[69] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. Wiley, 2 ed., 2001.

[70] H. Vafaie and I. F. Imam, "Feature Selection Methods: Genetic Algorithms vs. Greedy-like Search," 1994.

[71] R. Gutierrez-Osuna, "sequential feature selection." http://research.cs.tamu.edu/prism/lectures/pr/pr_l11.pdf, 2011.

[72] H. Vafaie and K. D. Jong, "Genetic algorithms as a tool for feature selection in machine learning," in *ICTAI*, pp. 200–204, Society Press, 1992.

[73] J. C. W. Debuse and V. J. Rayward-smith, "Feature Subset Selection within a Simulated Annealing DataMining Algorithm," *J. Intell. Inf. Syst.*, vol. 9, no. 1, pp. 57–81, 1997.

[74] J. L. Rodgers and W. A. Nicewander, "Thirteen ways to look at the correlation coefficient," *The American Statistician*, vol. 42, no. 1, pp. pp. 59–66, 1988.

[75] N. Drakos and R. Moore, "Mutual information." http://nlp.stanford.edu/IR-book/html/htmledition/mutual-information-1.html. Accessed: 2013-12-11.

[76] F. Song, D. Mei, and H. Li, "Feature selection based on linear discriminant analysis," in *Proceedings of the 2010 International Conference on Intelligent System Design and Engineering Application - Volume 01*, ISDEA '10, (Washington, DC, USA), pp. 746–749, IEEE Computer Society, 2010.

[77] D. Ruta and B. Gabrys, "An overview of classifier fusion methods," *Computing and Information Systems*, 2000.

[78] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Mach. Learn.*, vol. 29, pp. 131–163, Nov. 1997.

[79] J. Myers, K. Laskey, and K. A. DeJong, "Learning bayesian networks from incomplete data using evolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference* (W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, eds.), pp. 476–485, 1999.

[80] C. Demirkesen and H. Cherifi, "Fusing image representations for classification using support vector machines," *Image and Vision Computing New ...*, 2009.

[81] L. Chen and H. Tang, "Improved computation of beliefs based on confusion matrix for combining multiple classifiers," *Electronics Letters*, vol. 40, no. 4, pp. 1–2, 2004.

[82] E. K. Chong and S. H. Zak, *An Introduction to Optimization*. Wiley Series in Discrete Mathematics and Optimization, Wiley, 2011.

[83] H. Do, H. F. Silverman, and Y. Yu, "A real-time srp-phat source location implementation using stochastic region contraction(src) on a large-aperture microphone array.," *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on date 15-20 April 2007*, pp. 121–124, 2007.

[84] T. S. Sørensen, "Nonlinear total variation based noise removal algorithms," 2012.

[85] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Phys. D*, vol. 60, pp. 259–268, Nov. 1992.

[86] G. Gilboa, "Total variation denoising."

ISBN 978-87-997533-0-7