CHALMERS

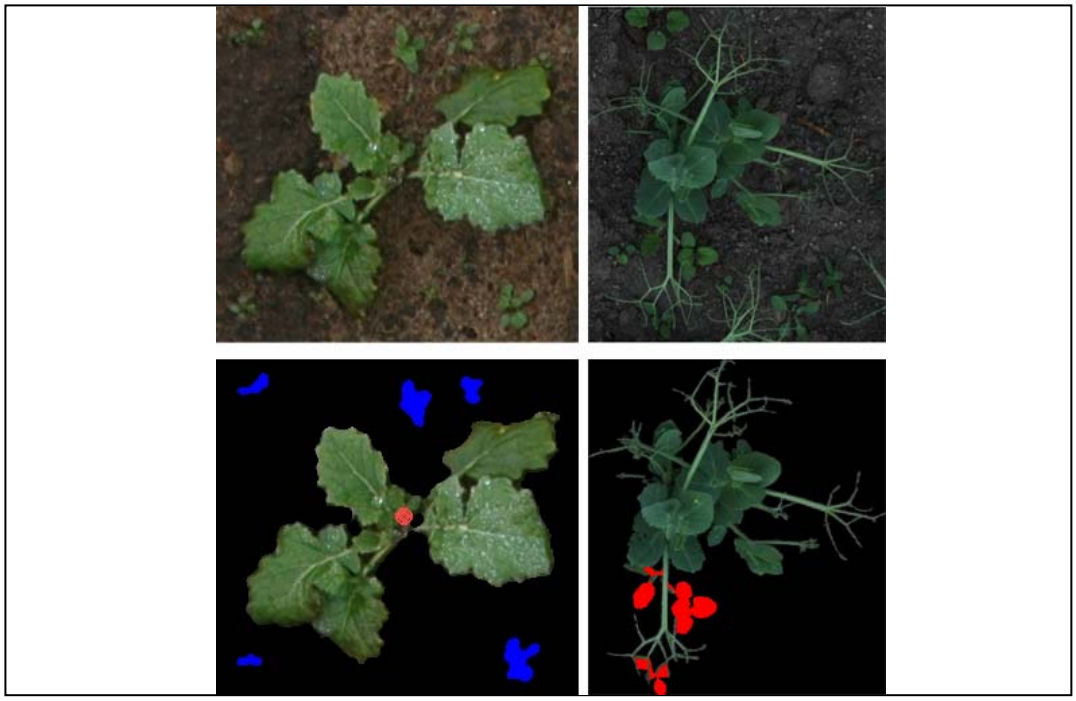


Image Analysis and Morphological Operations as Tools for Weed and Crop Detection

Master of Science Thesis

Martin Nyman

Department of Signals and Systems
Division of Signal Processing and Antennas
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2008
Report No. EX032/2008

Abstract

In modern agriculture unnecessary large doses of herbicides are sprayed on arable crops. Usually a uniform dosage that can handle the largest weed populations is chosen for the entire field. If the dosage could be lowered in areas with small amounts of weeds the same crops could be produced at a lower cost and with less impact on the environment. To be able to adapt the spraying the farmer needs to know the weed distribution over the field. Also the quantity of crop plants in different areas of the field is relevant, for example to study how many of the plants that survive the winter.

The focus of this master's thesis is image analysis with morphological operations for classification of weed and crop in digital photographs of pea and rape cultivations. The number of plants, the number of weeds and the weed covered area are calculated in 51 images of pea crops and 80 images of rape crops. Digital maps of weed and crop distributions can be created from this information for the purpose of precision agriculture.

For segmentation a method of removing grey background is developed. In the classification between crop and weed techniques such as Hough transform, top hat transform, distance transform and colour separation using maximum likelihood ratios are investigated and evaluated.

The results are promising, especially for the images of pea crops. Improvements could be made in several steps of the algorithms and the programs have potential to be useful in real systems. Under conditions with separated plants in the images and adequate lighting, image analysis can be a powerful tool for agricultural applications.

Contents

1 Introduction **1**

1.1 Background 1

1.2 Objectives 2

2 Image Acquisition **2**

3 Segmentation **3**

3.1 Colour Indices 3

3.2 Grey Removal 5

4 Features **7**

4.1 Pea Plants 7

4.2 Rape Plants 8

5 Methods **8**

5.1 Colour Separation Using a Maximum Likelihood Ratio Test 8

5.2 Hough Transform 11

5.3 Distance Transform 13

5.4 Morphological Operations 15

5.4.1 Dilation and Erosion 15

5.4.2 Opening and Closing 16

5.4.3 Top Hat Transformation 17

5.4.4 Greyscale Operations 17

6 Summarising the Algorithms **19**

6.1 Pea Crops 19

6.1.1 Algorithm Track 1 20

6.1.2 Algorithm Track 2 22

6.2 Rape Crops 23

7 Results **27**

7.1 Pea Program Track 1 28

7.2 Pea Program Track 2 33

7.3 Rape Program 36

8 Discussion **43**

9 Conclusions **45**

10 Future Work **45**

A Matlab Code, Pea programs **49**

A.1 Main pea program 1 49

A.2 Segmentation 50

A.3 Measurement box creator 51

A.4 Area detector 53

A.5 Mean area detector and tendril weed detector 54

A.6 Tendril weed detection on single object 55

A.7 Distance mapped skeletons 58

A.8 Tendril removal (second program) 59

B Matlab Code, Rape program **60**

B.1 Main rape program 60

B.2 Leaf vein extraction 61

B.3 Plant centre line markers 62

B.4 Plant centre detector 68

1 Introduction

1.1 Background

When farmers apply herbicides to their crops today they usually choose a uniform dosage on the whole field that is adapted to kill the highest concentration of weeds. But weeds are not evenly distributed over the field. Some parts can be almost free from weeds while other parts are heavily weeded. Herbicides are also somewhat harmful for the crops and limit their growth. Since herbicides can be quite expensive both economic and environmental benefits could be made if the dosage was reduced on the areas with lower weed populations. The same reasoning can be made regarding nutrition of the crops, it would be better to only apply fertilizers where it is needed.

This is the aim of so called precision agriculture. By adapting the treatment of crops and weeds to local conditions the same amount of crop could be achieved to a smaller cost and with less impact on the environment.

The sprayers can be controlled by a digital map of the crop and weed distributions to meet the needs of specific parts of the field. Several studies have been made on this topic using machine vision to form such a map. One problem is that the imaging system has to be designed specifically for different types of arable crops since their visual appearance can be very different.

For crops that grow in uniform rows the detection of crop row positions can facilitate the process. The crops are found in the rows and all plants outside the rows are most likely weeds. *Pérez et al.* used this together with geometrical measurements such as major axis length and area of crop/weed objects to find weeds in cereal fields [7]. Crop row detection has among other techniques been performed with Hough transform and double Hough transform [4].

When the rows are not distinguishable another option is to use morphological operations to enhance features that can be used to separate between crops and weeds. *Soille* used this approach to extract the vein networks appearing in plant leaves to find the plants [8]. Leaf shape examination has also been performed with other methods, for example *Neto et al.* identified plants with elliptic Fourier analysis [6].

The question of how to apply herbicides based on the image information has been treated in different ways. *Tellaeche et al.* divided a field into cells and determined which cells to spray using probability and Bayesian decision theory [10]. Detection and spraying in real time has been done by *Dammer et al.* using an optoelectronic weed sensor. In their study herbicide savings of 24.6 % were achieved without reduction of yield [3].

1.2 Objectives

The goal of this project is to find the information needed to create a distribution map of crop and weed from a number of photographs of pea and rape fields using image analysis with morphological operations. For the images of rape crops the aim is primarily to identify and count the crop plants. In the pea images the main priority is the detection and area calculation of weeds.

2 Image Acquisition

The result of a project of this type relies heavily on the quality of the photo material that is used as input. Ideally we want an image acquisition system that is robust in different lighting and weather conditions. But it is also important to keep the photo acquisition process uncomplicated since the system should be easy to use. By screening off the natural light and using spotlights instead identical lighting conditions could be achieved in all photos but that would mean a lot more equipment had to be carried around on the field.

All photographs used in this project were taken under natural lighting conditions with a Nikon D70 digital camera. The photos are 24-bit RGB images, which mean that they contain red, green and blue spectral information in separate layers. Each image consists of 6.016 Mpixels and every pixel is associated with a 3-dimensional vector (r,g,b) that contains the intensities of the colour bands. The intensity values range between 0 and 255 in a 24-bit image.

For the pea photos the camera was mounted on a 4W-motorcycle that was also equipped with a GPS to keep a log of where each photo was taken. Those images were taken in early May 2007 on a pea field in Skåne. Each image shows approximately 0.38 square meters of the field. The rape photos were taken with the camera mounted on a tripod. This was done in the middle of October 2007 on four different fields near Skara. The sites had been marked and positioned with GPS a couple of days earlier. On each site two photos were taken of areas close to each other to be able to examine local correlation in the crop and weed distributions. A wooden frame that shows one square meter of the field was placed in all photographs of rape crops.

During all image acquisition the camera was pointing directly towards the ground and kept at constant height above the field.

3 Segmentation

The first step in the image treatment is to remove irrelevant information from the images, i.e. soil, stones, twigs and basically everything that is not plants (crop or weed). The complete image I can be characterized into two subparts. $I = P \cup Bg$ where

- P is plants (crop and weed).
- Bg is background (everything else).

Most methods to perform this separation are based on differences in the spectral bands of P and Bg. One obvious difference is that plants contain higher intensities in the green band which can be used to make the discrimination.

3.1 Colour Indices

By calculating an index for each pixel where P-pixels are in one end of the index spectrum and Bg-pixels are in the other the classification can be made with a simple threshold operation. Such indices can be found from performing arithmetic operations with the red, green and blue chromatic layers of the image. The excess green index (ExG) defined by *Woebbecke et al.* [13] is widely used in agronomic applications like this one. The definition is:

$$ExG = 2g - r - b$$

$$r = \frac{R}{R+G+B}, g = \frac{G}{R+G+B}, b = \frac{B}{R+G+B}$$

Where R, G and B are RGB-values between 0 and 255 which is the intensity range of a 24-bit image. Values falling outside that range after the operation are set to be inside the 24-bit span. The result of this operation performed on each pixel is a greyscale image where the plant pixels can be separated from the background with a threshold. (See figure 1).

Another index also introduced by *Woebbecke et al.* [12] is the normalized difference index (NDI) defined as:

$$NDI = \frac{G - R}{G + R}$$

This produces values between -1 and 1 that can be scaled to the range of preferred image format.

A third index suggested by *Marchant et al.* [5] is:

$$F = \frac{r_m}{g_m^A}$$

Where $r_m = \frac{R}{B}$, $g_m = \frac{G}{B}$ and A is a constant calculated from characteristics of the camera. The derivation of this index is based on the physical process of light falling on a surface and being reflected. Starting from Planck's formula for the spectral radiant exitance of a blackbody radiator it can be shown that this index should be invariant to the temperature of the blackbody. This means for example that shadows would not influence the index as long as the reflected light comes from the same surface. The sun is not really a blackbody radiator and the camera filters are not of infinitely narrow bandwidth which is also assumed. But with the use of the CIE daylight model and an assumption of finite bandwidth of the camera the index still holds with only a slight change to the exponent A according to Marchant et al.

All three of these indices lead to grey level images with bimodal histograms from which thresholds can be selected to give the classification between P and Bg. One difference from the two earlier mentioned indices is that the plant pixels end up with a lower F-index than the background pixels. For the ExG and NDI indices the plant pixels appear brighter than the background.

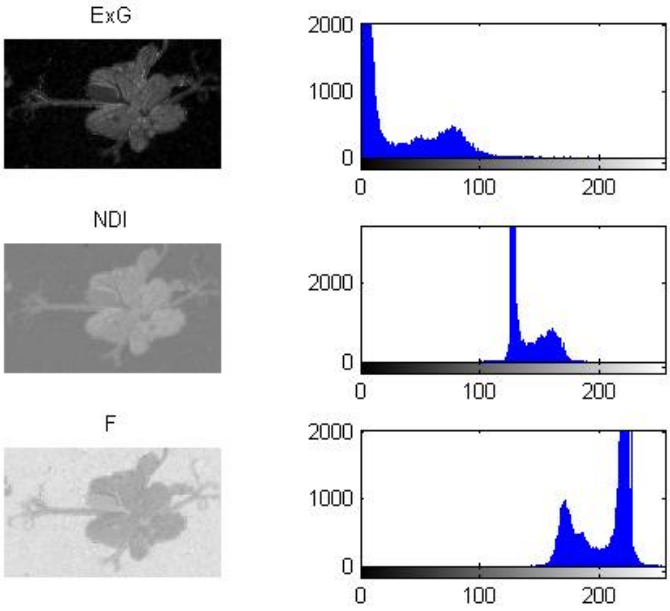


Figure 1: Example of grey level images produced by different segmentation indices and corresponding histograms. In the calculation of F the parameter A was set to 1.6.

3.2 Grey Removal

For the images of pea crops none of the mentioned methods give a perfect classification. Some plant pixels get classified as background or the other way around depending on where the threshold is placed. An example of this can be seen in figure 2 where the Excess Green index and a threshold operation have been applied to an image of a single pea plant. The threshold was selected with Otsu's method which chooses the threshold to minimize the variance between the two classes.

Figure 2: a) Original image **b)** Classified as P using ExG **c)** The complement to image **b**, i.e. the pixels classified as Bg. As can be seen this choice of threshold results in plant pixels ending up in Bg.

To find some characteristics of the pixels that are wrongly classified by the ExG some misclassified plant pixels and some correctly classified background pixels were chosen manually from the image in figure 2c and plotted in red-green-blue space.

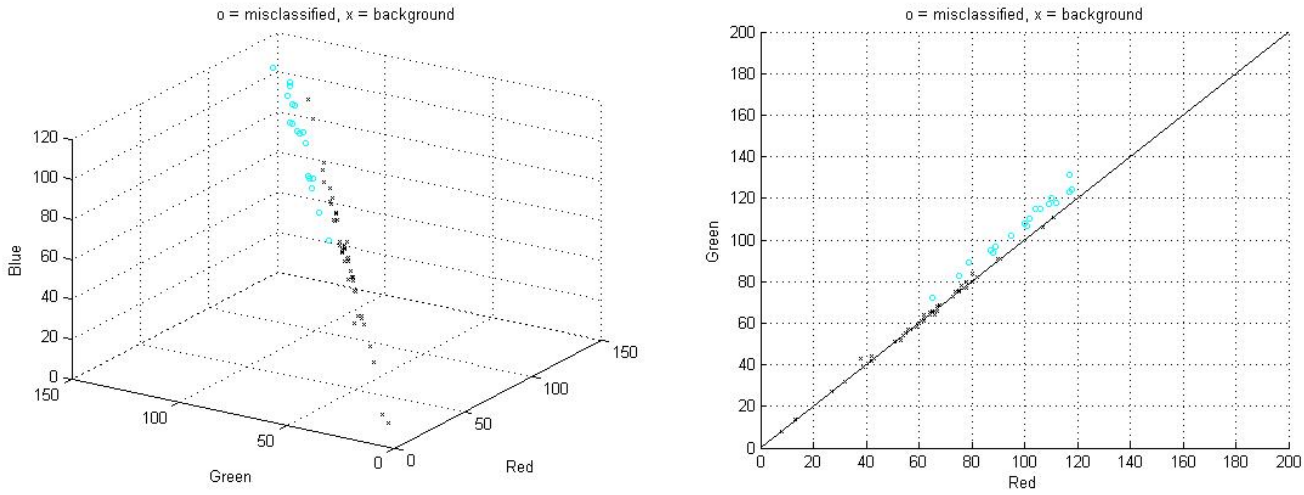


Figure 3: left) 3D scatter plot of some pixels chosen from figure 2c. Black crosses represent Bg-pixels and turquoise rings represent plant pixels wrongly classified as Bg. **right)** Red-green projection of the same pixels. The line shows equal intensity in the red, green and blue bands, i.e. greyscale pixels.

As can be seen in figure 3 the background pixels are lined up closely around the line where red=green=blue, in other words they are rather grey. The misclassified (and also the correctly classified) plant pixels are further away from the line of equal intensities. Based on this observation a new method of classification was set up. For all pixels in the original image the absolute values of green minus red and green minus blue was calculated. These give measurements of the pixels distance to the greyscale line. If both of these distance values are greater than x the pixel is classified as P, if none or only one are greater than x the pixel is classified as Bg. The number x is equivalent to the threshold in the above mentioned index methods and has to be chosen based on the available image material. For the pea images used in this study $x = 7$ has proven to give a very good result. This method of removing pixels of high ”greyness” can be seen as calculating two separate indices and then requiring that a pixel is on the correct side of the threshold in both cases to be classified as P.

$$P \in \{|G - R| > x \ \& \ |G - B| > x\}$$

The method is closely related to the Excess Green index since $(G - R) + (G - B) = 2G - R - B$ but the ”&”-operation in the threshold requirement leads to an improvement, at least with the image material used in this study.

A few background pixels slip through this classification and end up in P as ”grey noise” But as long as the pictures are of high resolution at least the misclassified pixels not in connection with a plant object can be easily removed based on area in the image. The smallest objects of interest (small weed patches) in the used image material have an area of at least 80 pixels. By removing all objects of area less than 35 most of the noise was eliminated. One problem with this removal process is that if a thin part of the plant ”breaks” during the segmentation so that one plant becomes two objects in the segmented image the small part may be removed. But 35-pixel blocks of plants disappearing have only a very small influence on the 6 Mpixel image. An example of the performance of the Grey Removal method can be seen in figure 4 using the same image as in figure 2.

Figure 4: a) Original image **b)** Classified as P using Grey Removal **c)** Classified as Bg using Grey Removal.

The same method was used to remove the background in the images of rape crops, for those images the threshold x was set to 4.

From the segmentation two new images are produced. One binary image with the background set to zero (black) and only ones (white) in the plant objects. In the second image the background is also black but all the original colour information is kept in the plants.

4 Features

With the background removed the next step is to separate the crop from the weeds. I.e. to make a classification into crop C or weeds W where $P = C \cup W$.

In other studies the crop rows has simplified this problem. If the crop rows can be located anything outside the rows can be classified as weeds. The pea and rape plants are also sown in straight lines but when they grow the row pattern is not so easily distinguishable. Fragments of rows can be seen but some of the seeds never produce a plant and the plants that actually are in the images stretch out in all directions and often cover a large part of the distance between the seed rows.

Since the plants are not distributed in a simply recognizable pattern the best approach is to analyse the objects in the images one by one. In this approach some of the concepts from pattern recognition are used. Decision theory for the classification is not implemented in this project however, but the idea of extracting features specific to a certain class is adopted. To make the classification one or several detectors that can analyse an unknown object in the image and determine if it is crop or weed is necessary. The detectors use differences in objects of the two classes to make the distinction. To design the detectors a list of features that distinguish the crop plants from the weeds was compiled.

4.1 Pea Plants

- **Area** The weeds are usually a lot smaller than the crop. A simple area detector can be set up by measuring an average area of pea plants and then classify all objects significantly smaller than this value to the weed class. Weeds that are in contact with crop in the image and thus ending up in a larger object is not at all found by this detector though.

- **Tendrils** A pea plant in this growth stage typically has developed two or three tendrils protruding from the plant centre. The thin tendrils resemble antennas as they spread out in different directions. Their shape is characteristic of the pea plants and is not found in any of the weeds.

- **Contour** The shape of the tendrils makes the contour of the pea plant quite different from weed contours. To use this feature the contour could be vectorised and

approximated with some orthogonal polynomial. Perhaps a detector could measure how the coefficients of such a polynomial correlates with coefficients from known objects. The detector would then first have to be ”trained” with contours of known pea and weed objects. This approach has been used in letter recognition [11].

- **Leaves** The leaves in the centre of the pea plant can be recognised using shape or colour criteria.
- **Extent** Extent in this case means the longest straight line that can fit inside an object. This should be larger in the pea plants than in the weeds.
- **Perimeter** The distance around an object is larger for the pea plants, this feature could be used in the same way as the area feature.
- **Colour variance** The pea objects seem to contain a larger colour span than the weed objects. If the colour variance in an object is found to be small it would suggest that it is weed.

4.2 Rape Plants

- **Area, Extent, Perimeter** Also in the images of rape crops the weeds are in general smaller than the crop plants. As long as the weeds are not in connection with plants in the images they can be detected using a size threshold in the same way as for the images of pea crops.
- **Leaf veins** The leaves of the rape plants have bright vein networks that do not appear in the weeds. The vein structure is of the pinnate type which means that a large vein runs through the centre of each leaf and from the central vein other veins branch out in a feather like pattern.

5 Methods

In this section some image analysis methods that are later used in the detectors are introduced.

5.1 Colour Separation Using a Maximum Likelihood Ratio Test

If we want to detect the tendrils (and thus the plant) a good first step would be to remove leaves from the object. The tendrils appear slightly brighter than the leaves

in the images so one approach is to use the colour difference. I.e. to find a plane in RGB-space that separates leaves and tendrils. To study the colours some images were constructed manually in an image editor to contain only leaf pixels and other images were made to contain only tendril pixels. Separate images makes it easy to get the pixels in separate vectors and plot them in RGB-space to see the difference between the two groups without risking misclassification in the plot. An example of this manual separation is shown in figure 5 and the pixels plotted in RGB-space are displayed in figure 6.

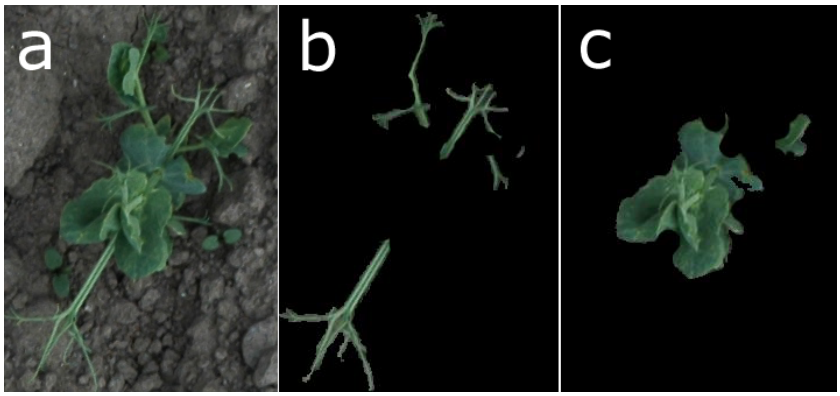


Figure 5: a) Original image of a pea plant. **b)** Tendril pixels extracted manually. **c)** Leaf pixels extracted manually.

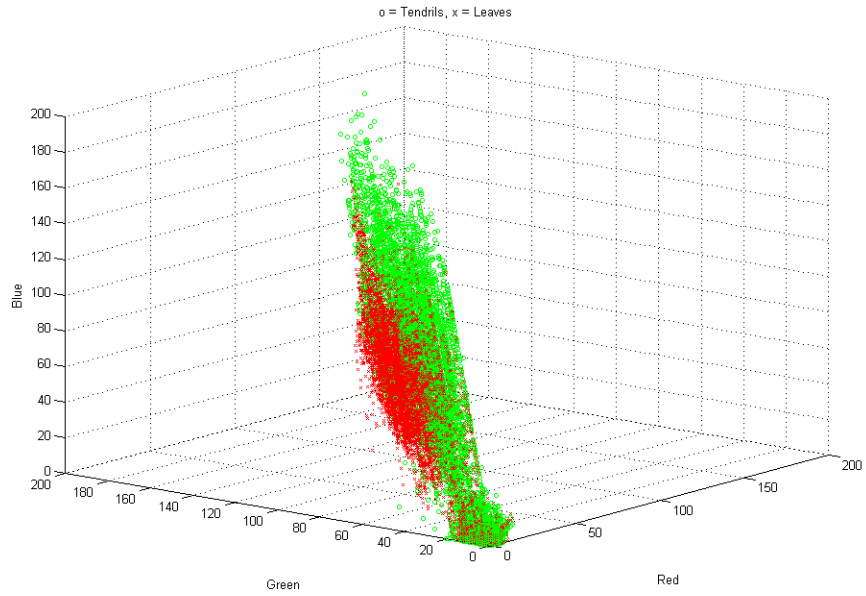


Figure 6: The green rings represent pixels originating from tendrils while the red crosses represent leaf pixels.

As can be seen from figure 6 the two groups are not entirely separated from each other and some pixel values are identical which makes a perfect distinction using only single pixel manipulations impossible. But perhaps could a majority of the groups still be separated by a plane.

If we assume that the two groups are normally distributed this separating plane can be found using a Maximum Likelihood ratio test on the two distributions. A normal (or Gaussian) distribution is described fully by the expectation value and the variance. Since the values in this case are 3-dimensional vectors (RGB) we use an expectation vector and a covariance matrix instead. Using N pixels with RGB-values in row-vectors x these can be estimated as:

$$\text{Expectation vector } \mu = \tfrac{1}{N}\sum_{k=1}^{N} x_k$$

$$\text{Covariance Matrix } \Omega = \tfrac{1}{N}\sum_{k=1}^{N} (x_k - \mu)^T (x_k - \mu)$$

Doing this for the tendril pixels x_T and the leaf pixels x_L separately we obtain two normal distributions where $x_T \sim \text{Norm}(\mu_T, \Omega_T)$ and $x_L \sim \text{Norm}(\mu_L, \Omega_L)$. Now the likelihood density functions can be calculated:

$$f(x|T) = \frac{1}{(2\pi)^{\frac{2}{3}}|\Omega_T|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_T)\Omega_T^{-1}(x-\mu_T)^T}$$

and correspondingly for $f(x|L)$.

For an unknown pixel x a likelihood ratio can be calculated to find if the pixel is most likely in a leaf or in a tendril.

$$\frac{f(x|T)}{f(x|L)} = \frac{|\Omega_L|^{\frac{1}{2}}}{|\Omega_T|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_T)\Omega_T^{-1}(x-\mu_T)^T + \frac{1}{2}(x-\mu_L)\Omega_L^{-1}(x-\mu_L)^T} = \gamma$$

If γ is greater than some value the pixel x is likely to be in a tendril, if γ is smaller than the same value our guess is that the pixel is in a leaf. This expression can be simplified further by applying the natural logarithm on both sides of the equation.

$$\frac{1}{2}\ln\frac{|\Omega_L|}{|\Omega_T|} - \frac{1}{2}(x-\mu_T)\Omega_T^{-1}(x-\mu_T)^T + \frac{1}{2}(x-\mu_L)\Omega_L^{-1}(x-\mu_L)^T = \ln\gamma$$

$$(x-\mu_L)\Omega_L^{-1}(x-\mu_L)^T - (x-\mu_T)\Omega_T^{-1}(x-\mu_T)^T = \alpha$$

Where α is a new constant ($\alpha = 2\ln\gamma - \ln\frac{|\Omega_L|}{|\Omega_T|}$). Using expectation vectors and covariance matrices calculated from the pixels in figure 6 a good value for the constant α can

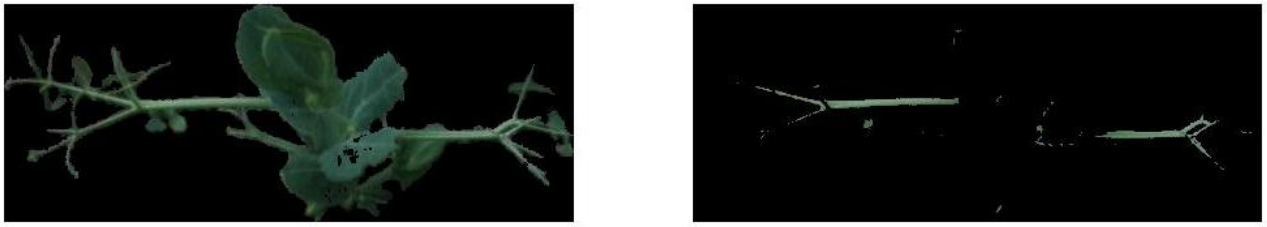


Figure 7: Example of tendrils separated from leaves through maximum likelihood ratio test.

be found through trial and error. The left hand side of the equation is calculated for all pixels x in an object. If the obtained value is greater than α the pixel is classified as tendril, if it is smaller than α it is most likely in a leaf or weed and removed from the image. A difficulty with this method is that the optimal α will vary for different objects depending on lighting conditions in the image etc. And as mentioned before a perfect separation is impossible using only single pixel colour criteria since the pixel groups in figure 6 are intertwined. As can be seen from the example in figure 7 the tendrils are extracted quite well but some bright pixels from leaves and weeds are still present.

5.2 Hough Transform

A Hough transform can be applied to find lines or any other shape that can be formulated in a parameterisation in an image [9]. The idea here was to use it to find the straight lines in tendrils.

Each pixel in an image has x and y coordinates in a Cartesian coordinate system (the row and column position of the pixel). All points connected by a straight line in the image have coordinates (x, y) that satisfy the equation of a straight line $y = kx + m$. I.e. all points in the line have the same values in the parameters k and m. In a parameter space (k, m) every possible straight line in image space is uniquely defined by a point. A Hough transform involves conversion into such a parameter space but since a vertical line in the image would lead to $k \to \infty$ the mentioned parameterisation is not suitable. Instead $\rho = x\cos\theta + y\sin\theta$ is used to describe straight lines and every possible line is represented by a unique point (ρ, θ) in parameter space. An example of this is shown in figure 8, ρ is the distance from the origin to the line along a vector that is perpendicular to the line and θ is the angle between the x-axis and this vector.

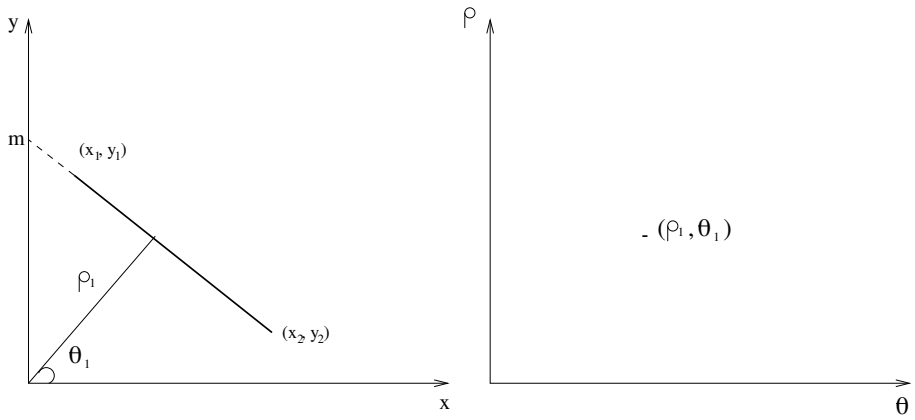


Figure 8: A line in image space (left) and in parameter space (right).

The Hough transform needs an image with uniform background to work. This is to make the distinction between background and foreground easy. A binary image is ideal for this purpose. For every pixel that is not in the background a line test is performed in all possible directions to see if the pixel is in a straight line with another foreground pixel. The parameters (ρ, θ) that correspond to a line connecting with another pixel are saved in an accumulator matrix H. The parameters of the lines that actually appear in the image will show up frequently in H when more pixels are tested since many pixels connect to each other with those same parameters. Parameters that connect only a few pixels will only very seldom be added to H. So when all pixels are tested the lines in the image can be found as the largest accumulated values in H.

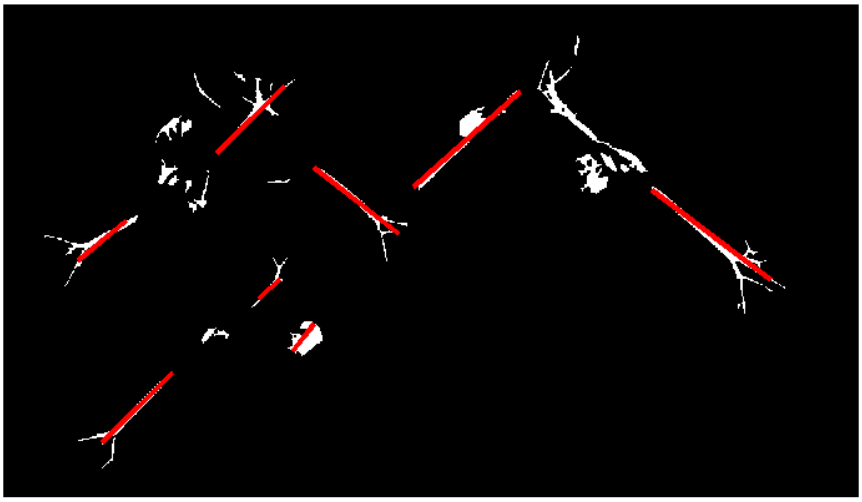


Figure 9: Binary image of pea plants where leaf pixels have been removed using a maximum likelihood ratio test. The red lines show the result of a Hough transform.

Figure 10: The same line markings as in figure 9 shown on the original pea object.

The Hough transform does find lines in pea tendrils but also inside the plant leaves. In figure 9 the transform has been performed after removing leaf pixels with another method. Still some tendrils are not found while a line in a leaf is found as can be seen in figure 10. In this case the tendrils were localised better by the maximum likelihood ratio test alone. Another idea was to use the Hough transform to count the number of tendrils in the image but since multiple lines often are found in the same tendril that did not work very well either. In the end, Hough transform was not used in any of the detectors.

5.3 Distance Transform

Another technique that utilises the thinness in the tendrils compared to the leaves is distance transform combined with skeletonisation. The distance transform is performed on a binary image; every white pixel is assigned with the value of the distance to the closest border. Distance in this case has been chosen as the number of steps through neighbouring pixels to reach the border. Figure 11 shows an example of distance transform applied on an object using 8-connectivity (both edges and corners count as neighbours which means every pixel has eight neighbours). An example of this method applied to an image of a pea plant can be seen in figure 12c.

			1	1			
		1	1	1	1	1	
	1	1	2	2	2	1	
	1	1	2	3	2	1	
		1	2	2	2	1	
		1	1	1	1	1	

Figure 11: A white object with its distance transform values, all squares represent pixels.

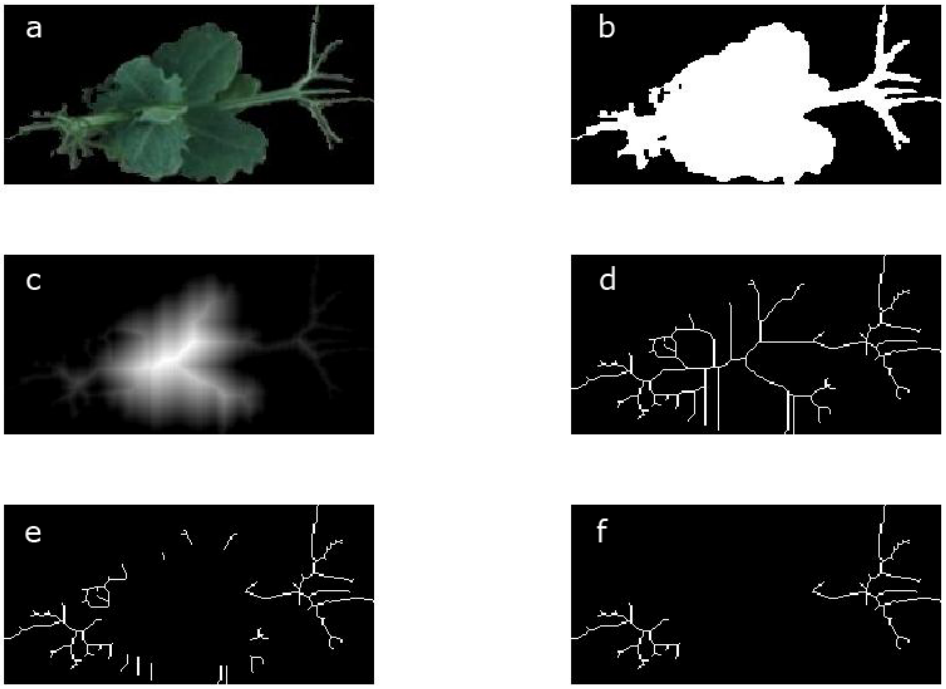


Figure 12: **a)** Original pea object. **b)** Binary pea object. **c)** Distance values shown as intensity. **d)** The skeleton formed from b). **e)** Skeleton combined with distance transform after a threshold operation. **f)** Tendril skeletons obtained after removing the small rests of leaf edges.

The skeleton of an image object is a thinned (one or a few pixels thick) version of its shape that can give a representation of its part structure. One early definition of a skeleton says that it is the medial axis formed by centre points of the largest possible circles inscribed in the object [1]. But skeletons can be constructed in several ways. For instance it can be seen as the local maxima in the distance transformed image. In Matlab (and this project) it is formed by successively removing border pixels without allowing the skeleton to break apart. An example of skeletonisation is shown in figure 12d.

The distance transformed object and the skeleton can be combined to form a skeleton with distance values in all points. By removing large values from this distance skeleton with a threshold the bulky parts of the object disappear, see figure 12e. For the pea objects this can be done to remove the leaves and only keep the skeleton of the tendrils. Some skeleton parts from the edges of leaves remain but can easily be removed with a size threshold as has been done in figure 12f.

5.4 Morphological Operations

The mathematics behind morphological operations on an image is based on the algebra of non-linear operators operating on object shape [9]. In practice they are performed with a structuring element (small matrix) that is moved across every pixel in the image. An output image is produced with pixel values that depend on the structuring element, the neighbourhood in the original image that is covered by the structuring element and the type of operation. Several morphological operations have been used in this project on binary images. A binary image can be described as a two-dimensional set of points (pixels) that are either ”on” or ”off”, (1 or 0). In the following definitions for binary images I is all the pixels in the output image, X is the set of pixels in the input image with value 1 and B is the structuring element.

5.4.1 Dilation and Erosion

Dilation can be defined as:

$$X \oplus B = \{p \in I : p = x + b, x \in X, b \in B\}$$

Where p is the set of pixels in the output image that will have value 1 after the operation. Objects in the image get thicker in the directions where the structuring element consists of ones since pixels neighbouring the object will be set to 1 by this operation, see figure 13b.

Erosion can be defined as:

$$X \ominus B = \{p \in I : p + b \in X, b \in B\}$$

The erosion operation $X \ominus B$ will set the pixels p in the output image to 1. The output image will consist only of the pixels at positions where the structuring element fits inside the objects in the original image so this operation makes the objects thinner, see figure 13c.

Figure 13: Example of morphological dilation and erosion operations performed with a structuring element of size 3x3 with value 1 in all positions. **a)** Original image. **b)** Dilation of image a. **c)** Erosion of image a.

5.4.2 Opening and Closing

Erosion may seem like the inverse of dilation but in fact it is not, if an image is eroded and then dilated with the same structuring element the result will not be identical to the original image. Erosion followed by dilation is an opening operation:

$$X \circ B = (X \ominus B) \oplus B$$

This will remove thin parts of the object and can be used to separate an object into several smaller ones, see figure 14b.

Dilation followed by erosion is a closing operation:

$$X \bullet B = (X \oplus B) \ominus B$$

This operation will close holes and gaps in image objects but keep the exterior object borders intact. It can be used to merge objects in the image that are close to each other, see figure 14c.

Figure 14: Example of morphological open and close operations performed with a structuring element of size 3x3 with value 1 in all positions. **a)** Original image. **b)** Opening of image a. **c)** Closing of image a.

5.4.3 Top Hat Transformation

A top hat transformation T is defined as the arithmetic difference between the original image and an opened version of the image.

$$T_B(X) = X \setminus (X \circ B)$$

This will remove parts of the objects that have the same size or are larger than the structuring element. On an image of a hat only the top of the hat would be extracted provided that the structuring element fits in the brim of the hat but not in the top, see figure 15c.

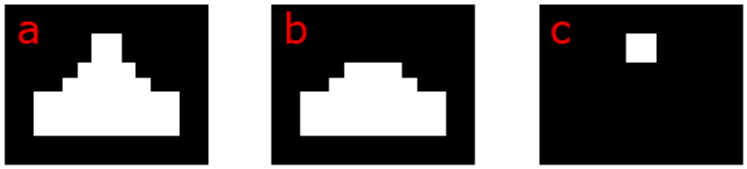


Figure 15: Example of top hat transformation performed with a structuring element of size 3x3 with value 1 in all positions. **a)** Original image. **b)** Opening of image a. **c)** Top hat transformation of image a.

5.4.4 Greyscale Operations

The extension from binary images to greyscale images for morphological operations is more complex [2]. For dilation the structuring element is summed to the original image in the neighbourhood of its current position and the obtained maximum value from that neighbourhood is assigned to the output pixel. For the output pixel with coordinates (u, v) the dilation with a structuring element B covering the pixels in the set $[r, s]$ can be defined as:

$$(X \oplus B)(u, v) = \max_{(r,s)\in B} \{A(u - r, v - s) + B(r, s)\}$$

where A is the set of all pixels in the original image.

Erosion is performed in a similar way, the structuring element is subtracted from the positions in the original image and the minimum value of the current neighbourhood is assigned to the output pixel. With the same notation as above erosion can be defined as:

$$(X \ominus B)(u, v) = \min_{(r,s)\in B} \{A(u + r, v + s) - B(r, s)\}$$

Other morphological operations follow using the same definitions as for binary images but based on the modified dilation and erosion operators.

When used on greyscale images the top hat transformation will extract small features in the image that differ in brightness compared to the rest of the image. In the opening step all structures that can not contain the structuring element are removed. By subtracting the opened image from the original a darker image is obtained where the thin bright structures stand out. An example of this applied to an image of a rape plant can be seen in figure 16c. Through thresholding (with Otsu's method for example) we get a binary image of these features, see figure 16d.

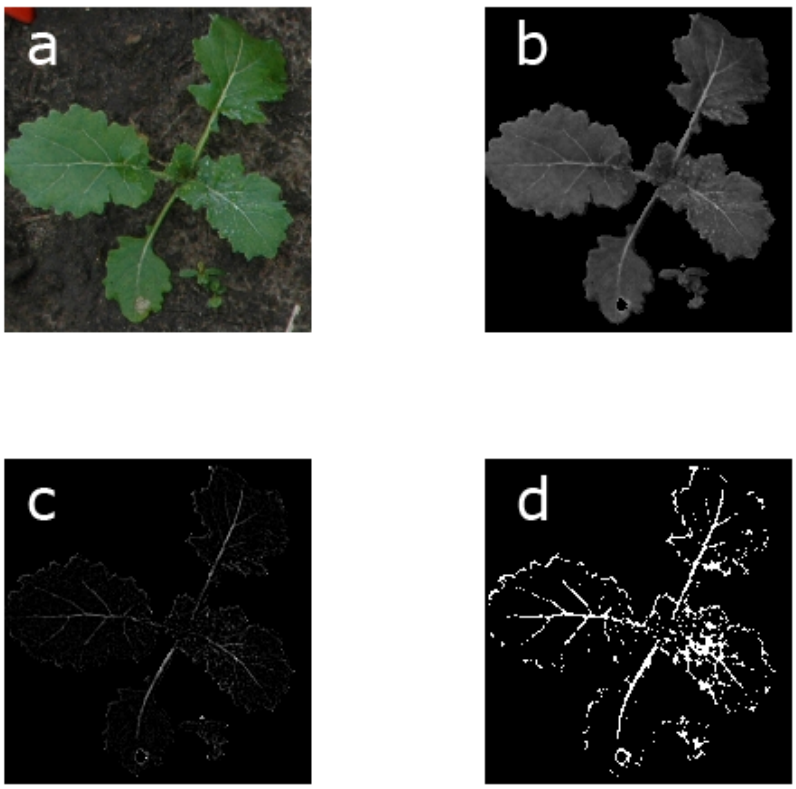
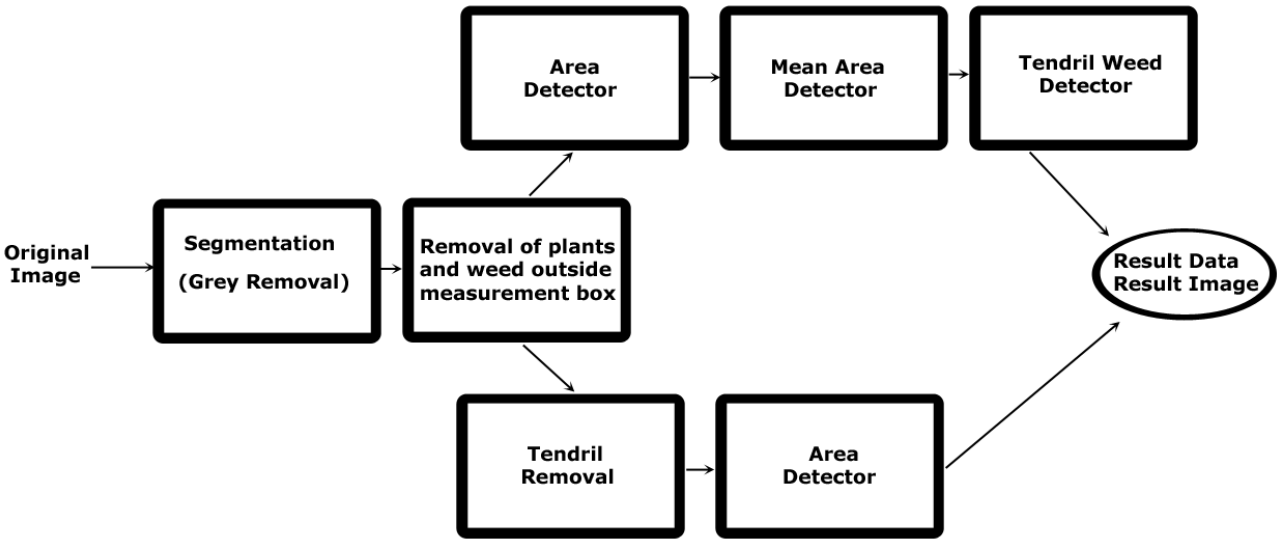


Figure 16: **a)** Original image of a rape plant. **b)** Red channel (after background removal). **c)** Top hat transformation performed on image b (shown here with adjusted intensity). **d)** Binary image obtained from thresholding image c showing leaf veins and other thin bright structures.

6 Summarising the Algorithms

Two image analysis algorithms for pea crops and one for rape crops were developed. In this section the programs that were used to produce the results in section 7 are presented along with the ideas behind them.

6.1 Pea Crops



As already mentioned the algorithm splits up in two separate tracks that will be described below. The two tracks differ mostly in the treatment of tendrils on the pea plants but the first step after segmentation is identical for both methods.

Measurement Box: Objects (pea or weed) that are cut off by one of the image borders have unpredictable properties, their size and shape is not comparable to other objects. So the first step after segmentation is to remove all objects that touch one of the borders. To keep the same statistics of plants per area unit as in the original image the measurements are performed only on objects inside a smaller box in the upper left corner of the image. The idea is to remove objects that are touching the original image borders and keep all objects that touch the other two edges even if they are partly outside of the box. Such objects are to compensate for the removed ones. The box size is determined by the extent D_{max} of the largest object in the image. If the original image has width w and height h the box size is set to $(w - D_{max}) \cdot (h - D_{max})$. This should make it impossible for an object to be inside the box and still touch one of the original image borders. An example is shown in figure 17 where all objects that are completely outside the box or touching one of the image borders have been removed.

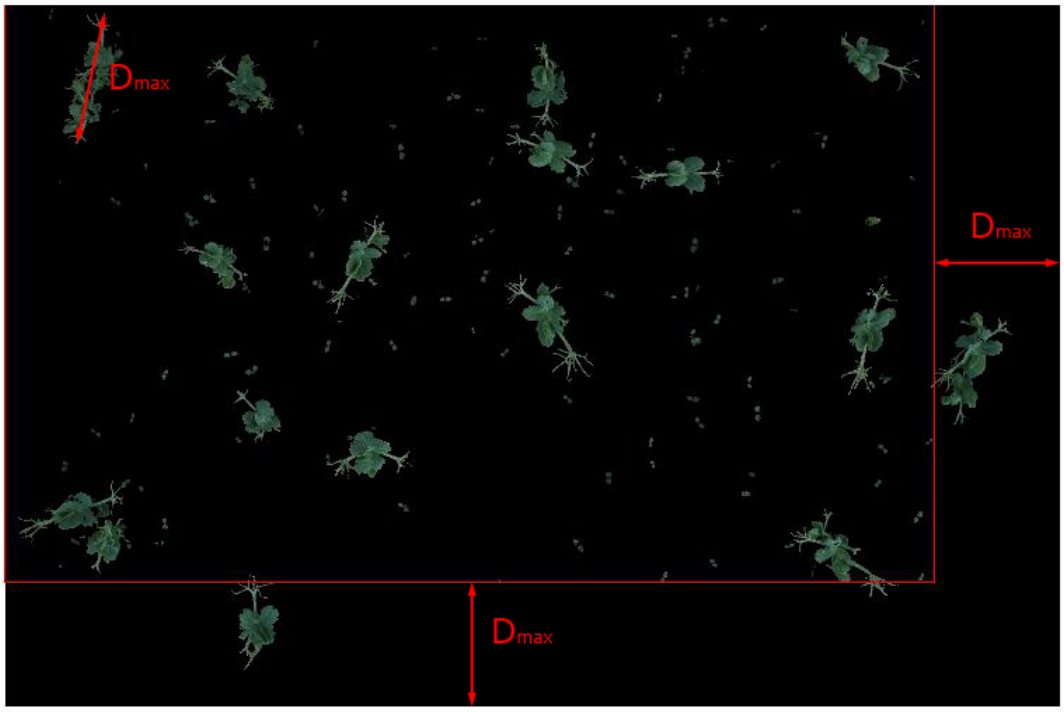


Figure 17: The red lines show the borders of the measurement box whose size is determined by the extent of the largest object.

This method works well as long as the objects in the image are relatively small. In other images pea plants grow so closely together that they form one very large object that causes D_{max} to be huge. In some cases this leads to a measurement box that is so small that no pea objects fit inside, therefore the maximum size of D_{max} has been set to one third of the image height.

6.1.1 Algorithm Track 1

The idea here is to first detect all weeds that are not in connection with a pea plant in the image and then treat the pea plants with possible attached weeds in the last step.

Area detector: The first detector measures the area of all remaining objects in the binary version of the image and classifies all beneath a certain limit as weeds. Objects above the limit are not classified or handled at all in this step. Weed and pea plant sizes vary in the images so no absolute limit exists that can handle all images without misclassification but the limit has been set to never misclassify a pea plant as weed. The

size of the objects has been found to correlate somewhat to the number of objects in the image. Many objects in the image indicate that the growth has been better in this region and both weeds and pea plants are bigger. Of the tested area limits the one that has produced the best result was a linear function of the number of objects that sets the limit value between 3500 and 9125 pixels.

But the box size complicates the relation between size and number of objects. If D_{max} is very small this indicates that the growth in the region has not reached very far and the area limit should be low, but a small D_{max} also leads to a large box which means the number of objects will be high. For this reason an extra condition has been applied to the limit that sets it to 4000 pixels if the number of objects is above 110. A few weed objects still fall outside these conditions but a large majority is detected.

A morphological close-operation is then performed on all detected weed objects. This is done since their stems often disappear in the segmentation which leaves the weed object with a hole in the centre. The quantity of detected weed objects and their total area is then calculated and saved.

Mean Area Detector: As mentioned some free weed objects still remain in some images. These are quite large since they passed the first area limit detector. But the surrounding pea plants are even larger; if the weeds have grown big the peas will have done the same. This second detector measures the area of all remaining objects and calculates the mean value from these measurements. If the area of one object is smaller than 10 % of the mean value it will be classified as weed. In the available image material this detector has found additional weeds without making misclassifications.

Tendril Weed Detector: A lot of the remaining undetected weed is situated near the pea tendrils. This detector was therefore constructed to locate the tendrils and find weeds in connection with them. It requires both a binary image and a colour image of pea objects with connecting weeds since it uses both shape and colour criteria.

The first step is to find the tendril skeletons using the method of distance transform and skeletonisation described in section 5.3. Then a dilation operation is performed around the skeleton to make it 5 pixels thicker. This thickened version of the skeleton is used as a mask that touches the majority of the tendrils and weeds nearby. When subtracting the mask from the original binary object the remaining parts will be the big leaf centre on the pea plant and small weed and tendril parts that has been cut off by the mask. An area measurement is then performed and the small parts are kept and added to the mask while the largest (the leaves) are deleted.

A low estimation of the number of pea plants is also found in this process. Since every pea plant has this large leaf centre the number of thrown away large objects is saved as the number of pea plants. This estimation will be lower than the real quantity because overlapping plants will not be counted correctly.

The tendril mask with the added small parts should now contain only tendrils and

possibly weeds as in the example in figure 18b. To separate weeds from tendrils a maximum likelihood ratio test based on colour is performed (see section 5.1). Apart from a binary mask of weeds figure 18c shows that small tendril parts end up on the wrong side in the separation. After removing those with a size threshold we are left with only weeds. Their area and total quantity is calculated and saved.

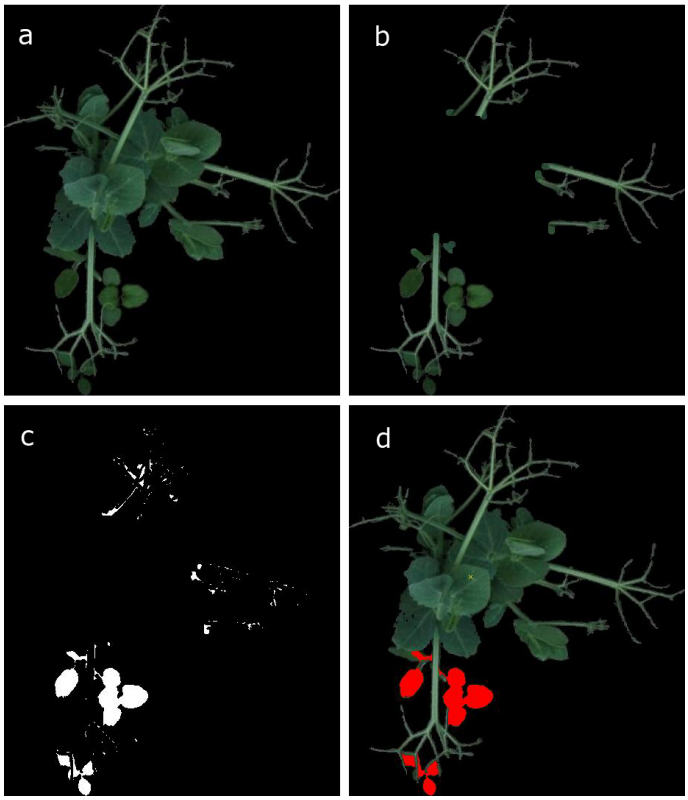


Figure 18: a) Original object containing two pea plants and some weeds. **b)** Tendrils and weed remaining after removal of the leaf centre using dilated tendril skeletons. **c)** Binary image of the weeds and tendril rests after ML ratio colour test. **d)** After removing small objects (tendril rests) image c can be used as a mask to mark the weeds.

6.1.2 Algorithm Track 2

The second approach is to first remove tendrils from the image which will cut free more weeds to become separate objects that can be found based on size.

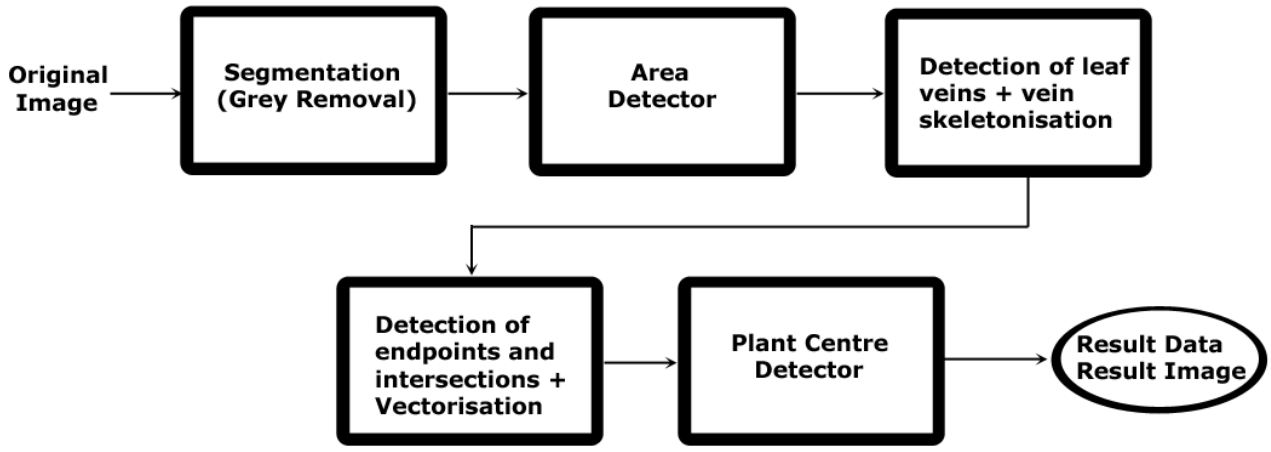
Tendril Removal: Since tendrils are thin and brighter than surrounding leaves and weeds they can be found using the top hat transformation described in section 5.4.3. To do this the red channel is used as greyscale image and a square structuring element of size 10x10 is used in the top hat. There is a significant intensity difference between

tendrils and leaves in all colour channels but the red channel was chosen since the difference between leaves and weeds seems to be smaller there. After setting a threshold using Otsu's method a binary image of tendrils and small artifacts from leaves and weeds is obtained. To remove these artifacts the fact that tendrils are quite long compared to the leaf rests is used. The major axis length is measured on all objects, that is the length in pixels of the major axis of an ellipse approximated around each object. This gives a measurement of the extent of the objects. A length threshold then sorts tendrils from the leaf rests, if the major axis length of an object is shorter than 25% of the maximum major axis length found in the image that object will be deleted. Finally a morphological close operation is performed to tie broken tendril parts back together. The result is an image of (almost) only the tendrils. This is then subtracted from the original image to leave only leaf centres of the plants and weeds.

Area Detector: This detector separates pea and weed objects based on their area but a slightly different way to set the area threshold was chosen since the removal of tendrils from the pea plants makes the plants smaller and thus the difference in size from weeds smaller. The area of all objects is measured and then objects with area smaller than 15% of the largest object in the image are classified as weed. The weed objects are also required to be larger than 80 pixels to avoid counting noise from the segmentation as weeds. Just as in the area detector of the first algorithm a close operation is performed on the detected weed objects before the final quantity and area is calculated.

The number of pea plants is calculated as the number of objects larger than or equal to 15% of the area of the largest one.

6.2 Rape Crops



The problem with plants that are cut off by one of the image borders has not been handled at all for the rape images. Often the plants grow so closely together that almost everything would be removed if a measurement box was implemented in the same way as for the pea images. A wooden frame of size 1 m^2 shows which area to analyze in each photo. To remove the frame a rectangular cropping was performed manually on every image. But depending on the orientation of the frame in the image some parts of the measurement areas were lost in the cropping.

As mentioned in the introduction the main focus for the images of rape crops is to find and count the number of plants. Inspired by an article by *Soille* [8] the idea of localising plant centres from information in the leaf veins is used. The amount of leaves on the rape plants vary but the central vein of each leaf always grow out from the plant centre and the branch pattern of other veins in a way look like arrows pointing towards the centre of the plant (see figure 16). But the first step after segmentation is still to detect small weeds not in connection with the plants.

Area Detector: This detector simply classifies all objects with 5000 pixels or less as weeds. The threshold is set quite low but still some leaves of rape plants that get cut off in segmentation are misclassified as weeds. After this step weeds are not handled anymore by this program.

Detection of Leaf Veins: The top hat transform with a 3x3 square structuring element is used to extract the thin leaf veins from the image. Again the red channel was chosen as intensity image to perform the top hat on. The transform also results in a lot of unwanted structures especially from the edges of the plant leaves (see figure 16d). To remove this, a mask of the edges is created using distance transform with a threshold to keep only the six outermost pixels of the leaves. That mask is then subtracted from the top hat transformed image, which removes a large portion of the artifacts. But still quite a lot of structures besides the leaf veins remain in the image as shown in 19b. Another problem that causes errors in later steps of the algorithm is that the leaf veins are not always intact after the top hat transform. In general the leaf veins are longer objects than any of the artifacts so after thresholding to get a binary image, a measurement of major axis length is done on all remaining objects in the transformed image. Anything shorter than 25 % of the longest structure is removed.

Now the image should contain only leaf veins and a closing operation is performed to put broken parts back together. It can be seen in figure 19c that the closing operation is not always enough, a side branch in the pattern from the upper right leaf is not in connection with the central vein.

The final step is to thin the leaf veins down to a thickness of one pixel using skeletonisation as has been done in figure 19d. The image of leaf vein skeletons is then passed on to the next part of the program.

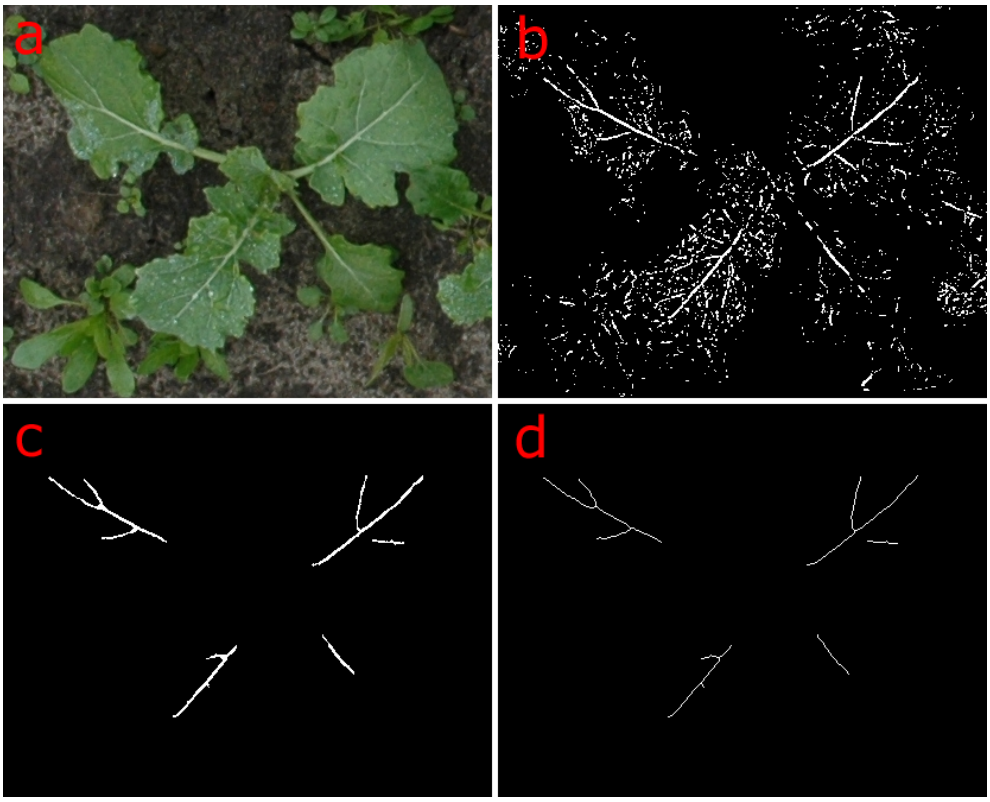


Figure 19: a) Original image of a rape plant and some weeds. **b)** Binary image of the top hat transform performed on the red channel of image a (Leaf edges removed). **c)** Leaf veins extracted from image b based on major axis length. **d)** Leaf vein skeletons.

Endpoints, Intersections and Vectorisation: Every skeleton is now treated separately and the number of skeletons is saved as an estimation of the number of leaves in the image. To find out how the leaf is orientated and in which direction the plant centre is situated information from endpoints and intersections in the leaf vein skeletons is needed. A program written by a previous diploma worker at SIK, Rasmus Nisslert, removes small artifacts on the skeletons (any branch shorter than 6 pixels is deleted) and returns coordinates for all endpoints and intersections on the skeletons.

The next step is to localise the direction of the central vein. The distances between all endpoints in the skeleton are measured and the central vein is simply assumed to be between the two points furthest apart. An example can be seen in figure 20. This method of finding the central vein causes errors sometimes when the leaves are wide or the central vein breaks up in the top hat transform and gets shortened but in most cases it works like it is supposed to. If only two endpoints are found on a skeleton they are assumed to be on the central vein. The plant centre should then be somewhere along that line but the specific direction is impossible to know without side branches.

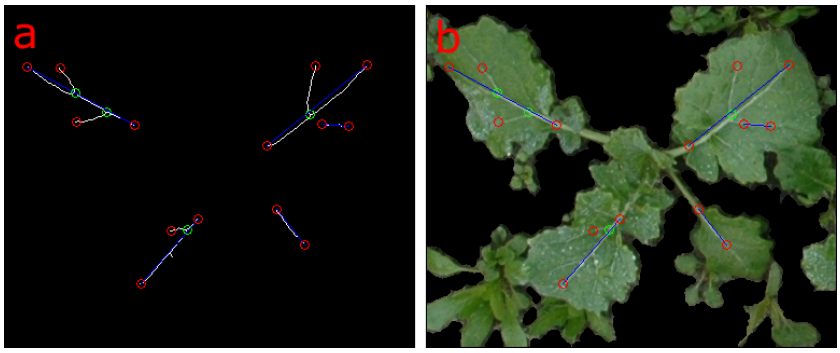


Figure 20: a) The skeletons from figure 19d. Endpoints are marked with red rings, intersections with green rings and central veins with blue lines. In the upper right leaf an error is visible where a side branch has been cut off and is treated as a leaf of its own. **b)** The markings displayed in the original image.

To find out which of the two points on the central vein that is closest to the plant centre the additional endpoints and intersections are used. First the projections of all remaining endpoints and intersections onto the line of the central vein are calculated. For every side branch the intersection point should now be closer to the plant centre than the projected endpoint of that branch. Therefore the endpoint on the central vein that is closer to the mean value of the intersections than the mean value of the projected endpoints is chosen as the one closest to the plant centre.

Some markers are now saved in a separate image like the one in figure 21b where the plant centre search will be performed in the last part of the program. For a skeleton with side branches a 100 pixels long (distance can be chosen) line is drawn from the endpoint closest to the plant centre in the direction of the central vein. This line is given intensity 3. For skeletons without side branches lines of the same length are drawn in both directions. They are given intensity 2 since the probability of those lines going through the plant centre is smaller.

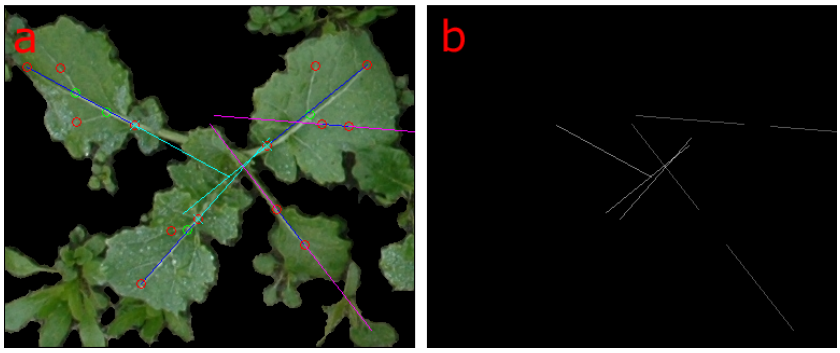


Figure 21: a) Same as figure 20b but with direction markers. Purple lines are from leaves without side branches and turquoise lines are from leaves where the specific direction was detected. **b)** The marker image with weighted intensities that is sent on to the final part of the program.

Plant Centre Detector:A line in the marker image indicate probability of a plant centre being somewhere along that line. Now a summation is performed where each pixel is assigned the sum of all pixels in a certain neighbourhood. The pixel value will then be high where many marker lines intersect or are close to each other. In this way every pixel is treated as a possible plant centre and the value of the pixel after summation will correspond to the probability of an actual plant centre in that position.

The summation is performed in larger and larger neighbourhoods. The first neighbourhood is a 3x3 square, so the pixel will be assigned the sum of its own value and the values of its eight closest neighbours. After the summation all plant centre candidates (values above a certain limit) are saved in a vector and sorted in descending order. A marker is put on the maximum value to indicate a plant centre at that position. Then all lines and plant centre candidate points within a 100 pixel radius around that point are removed. Another centre marker is placed at the remaining highest value candidate and this process is iterated until the candidate vector is empty.

Now the most probable plant centres should be detected and saved and the corresponding lines of those centres are removed. A new summation is then performed in a larger neighbourhood and the process is repeated. The program goes through five neighbourhood sizes in the same manner. They are all squares and their side lengths are 3, 5, 8, 20 and 30 pixels.

The limit for letting a position into the centre candidate vector is the same independent of summation neighbourhood size. The minimum requirement is one line with weight 3 (turquoise in figure 21a) and one line with weight 2 (purple in figure 21a) going through the neighbourhood square. Two purple lines are therefore not enough but all other combinations of lines will result in a plant centre candidate. In fixed numbers the value of a pixel must be larger than 12, 20, 32, 80 and 120 for the respective square sizes.

Finally the number of plants, the number of weeds and some other parameters are saved to a file. Two result images are also saved, one showing the line markers, the detected central leaf veins and the detected plant centre positions. The other one shows only plant centre markings.

7 Results

Here quantitative results produced by the programs from the available image material described in section 2 are presented in tables and graphs. Examples of result images can also be found in this section.

7.1 Pea Program Track 1

Using the program described in section 6.1.1 (the first track of the algorithm) the number of peas, the number of weeds and the total weed covered area were calculated in 51 images. For each image this data was saved to a file and a new image was produced that shows what was actually detected. In these result images the weeds found by the area detector are coloured blue and the weeds found by the mean area detector and the tendril weed detector are coloured red.

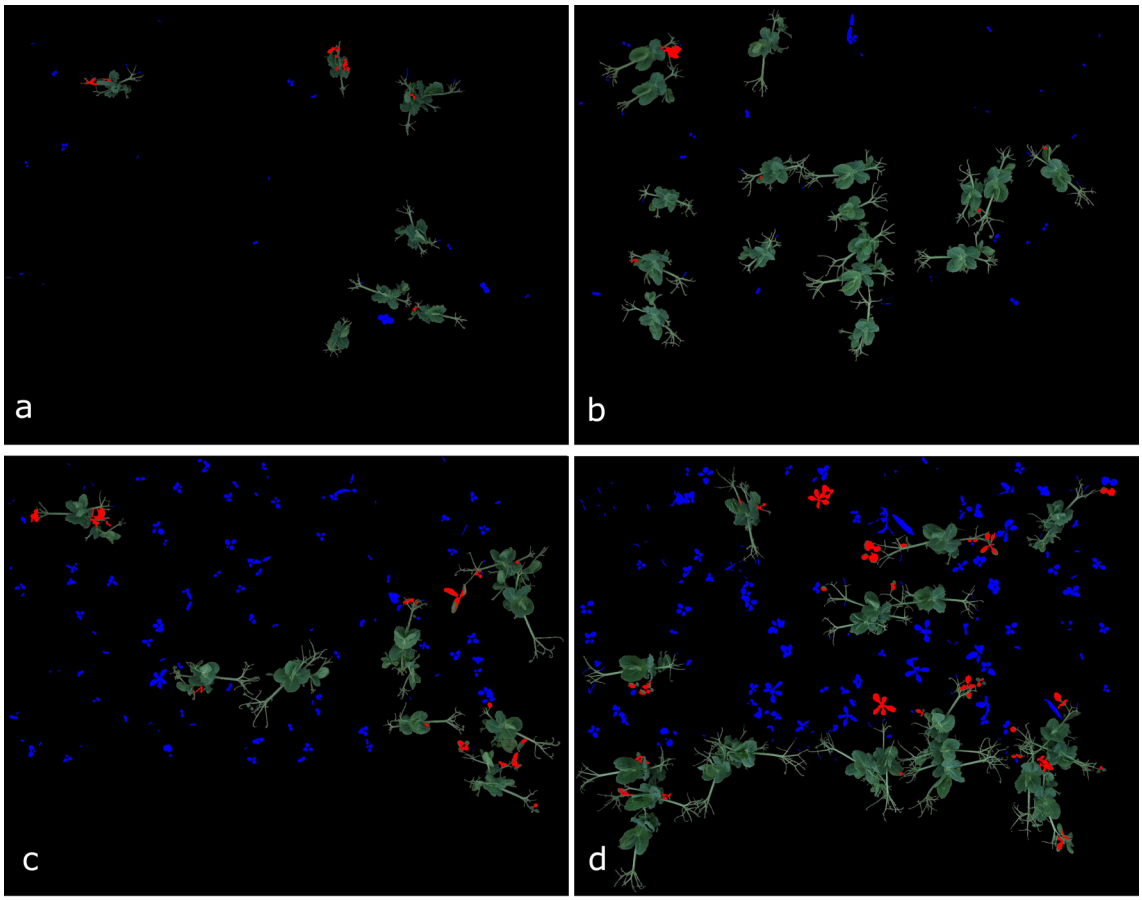


Figure 22: Examples of result images. Weeds classified by the area detector are blue and weeds classified by the mean area detector and the tendril weed detector are red.

Figure 22 shows the result from four images with different densities of weeds and pea plants. As can be seen most of the weeds are found by the area detector. The two large completely red weeds in image d are examples of objects that were missed by the area detector but found by the mean area detector. The rest of the red coloured regions have been classified by the tendril weed detector. Some of those red markings are misclassifications.

Resulting data are presented in the following tables. Areafraction weed is the detected weed area in per cent of the area of the measurement box. The boxfraction is the measurement box area divided by the original image area. When D_{max} reaches it maximum of one third of the image height the boxfraction is at its minimum (0.5187). Since the original images show approximately 0.375 m^2 of the field the last two columns were calculated as quantity / (boxfraction · 0.375).

Image	Number of Peas (NoP)	NoP (man. count)	Number of Weeds (NoW)	NoW (man. count)	Area-fraction weed(%)	Box-fraction	NoP $/m^2$	NoW $/m^2$
1	18		131		1.1255	0.6489	74.0	538.3
2	9		83		0.7596	0.5304	45.2	417.3
3	16		137		0.895	0.7254	58.8	503.6
4	18		124		0.694	0.719	66.8	459.9
5	19		97		0.626	0.6825	74.2	379.0
6	16		93		0.7036	0.6434	66.3	385.5
7	16		88		0.5433	0.7098	60.1	330.6
8	7		42		1.0115	0.5187	36.0	215.9
9	11	16	42	25	0.5099	0.5541	52.9	202.1
10	15		41		0.3127	0.5769	69.3	189.5
11	11		55		1.0111	0.5224	56.2	280.8
12	15		40		0.409	0.6516	61.4	163.7
13	20		46		0.7233	0.5187	102.8	236.5
14	22		46		0.3379	0.7077	82.9	173.3
15	12		30		0.3285	0.5503	58.2	145.4
16	16	23	12	6	0.1314	0.5553	76.8	57.6
17	18	21	48	28	0.335	0.6625	72.5	193.2
18	7	8	33	13	0.311	0.5187	36.0	169.7
19	15		74		1.2674	0.5497	72.8	359.0
20	24	28	61	10	0.4482	0.7297	87.7	222.9
21	9		27		0.1665	0.6287	38.2	114.5
22	16		37		0.3792	0.6448	66.2	153.0
23	14		34		0.3359	0.5224	71.5	173.6
24	16	21	118	67	1.052	0.5187	82.3	606.6
25	9		72		0.6783	0.5187	46.3	370.2
26	10	16	80	63	1.8801	0.5187	51.4	411.3
27	14		52		0.9783	0.5422	68.9	255.7
28	15	24	77	47	0.8825	0.6121	65.3	335.5
29	9	12	51	37	0.5987	0.5737	41.8	237.1
30	10	12	43	26	0.4023	0.6088	43.8	188.3

Table 1: Results from the first algorithm for pea crops

Image	Number of Peas (NoP)	NoP (man. count)	Number of Weeds (NoW)	NoW (man. count)	Area-fraction weed(%)	Box-fraction	NoP $/m^2$	NoW $/m^2$
31	10	13	37	29	0.421	0.5515	48.4	178.9
32	9		37		0.5732	0.5187	46.3	190.2
33	9		63		0.509	0.6307	38.1	266.4
34	10	18	61	40	0.8465	0.6016	44.3	270.4
35	10	15	45	18	0.5212	0.5242	50.9	228.9
36	5	8	38	21	0.4244	0.5187	25.7	195.4
37	15	17	40	20	0.3853	0.5187	77.1	205.6
38	6	8	93	70	1.7857	0.5187	30.8	478.1
39	17		139		1.9474	0.5187	87.4	714.6
40	14		154		2.2164	0.5187	72.0	791.7
41	9		127		2.3387	0.5187	46.3	652.9
42	3		119		2.6827	0.5187	15.4	611.8
43	9	12	130	75	2.6022	0.5187	46.3	668.3
44	13	17	113	77	2.5777	0.5187	66.8	580.9
45	8	14	116	86	2.7904	0.5187	41.1	596.4
46	11		163		3.3093	0.5187	56.6	838.0
47	10	14	94	50	1.9553	0.5187	51.4	483.3
48	16		123		2.2813	0.5187	82.3	632.4
49	13		185		4.5436	0.5187	66.8	951.1
50	11		161		3.102	0.5187	56.6	827.7
51	12		202		4.0541	0.5187	61.7	1038.5

Table 2: Results from the first algorithm for pea crops

As can be seen in tables 1 and 2 the number of weeds per square meter in the images ranges between 58 and 1039 so there is a real variation in weed density but in many of the images with high image numbers the Boxfraction has reached its minimum value. This indicates that the method of removing objects with a measurement box is not fully applicable for those images and the statistics of plants/weeds per area unit and also the areafraction weed are unsure there.

To get some indication of the accuracy of the calculations the numbers of pea plants and weeds were counted manually in 20 randomly chosen images (in the same measurement boxes that the program uses). Results from this manual counting is displayed in tables 1 and 2 in the third and fifth columns. The counted result should not be seen as completely correct either since it in some cases can be hard to tell if small objects are a single weed or parts of a bigger weed. Pea plants growing closely together can also cause uncertainty in the counting. But the counted results should be fairly accurate and can at least give a comparison between the program and a human observer, in the plots in figures 23 and 24 each star symbol corresponds to results from one image.

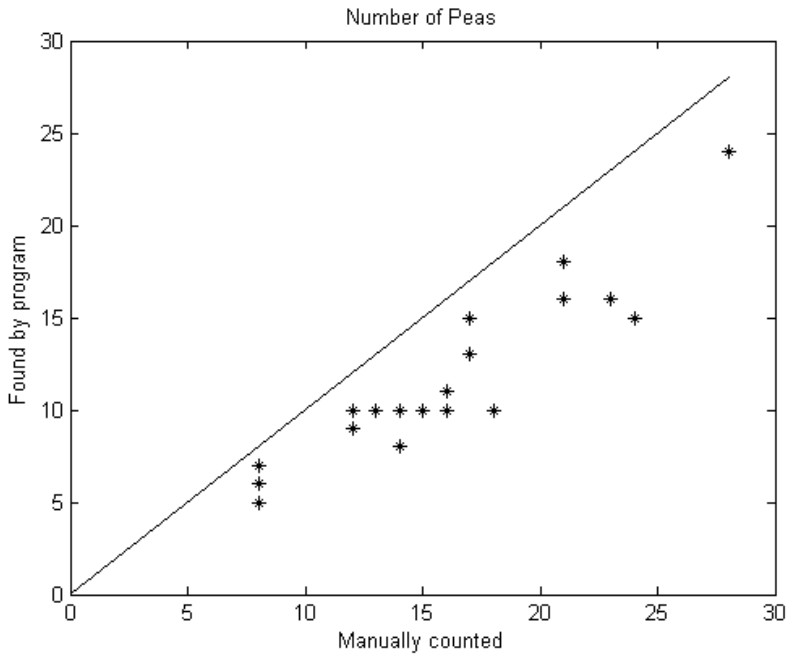


Figure 23: Number of pea plants in 20 images, the straight line shows where the results would be equal.

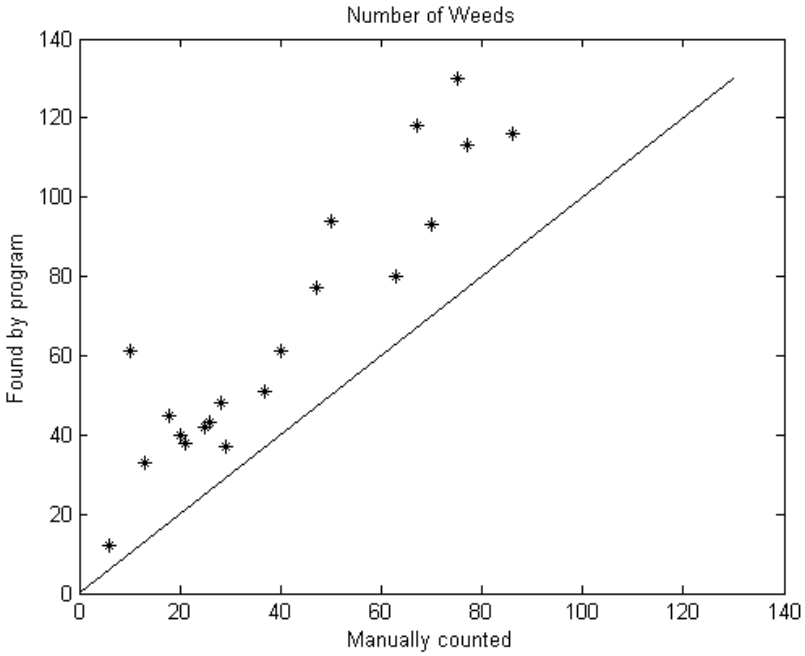


Figure 24: Number of weeds in 20 images, the straight line shows where the results would be equal.

Some error calculations were also performed by manually counting the number of misclassifications in the randomly chosen result images. The ”Missed weeds” in table 3 are weeds or parts of weeds that were not fully marked in red or blue. The column ”Weed misclassifications” displays the share of the objects classified as weeds that were actually part of a pea plant.

Image	Number of Weeds (program)	Peaparts classified as weed	Missed weeds	Weed misclassifications
9	42	5	0	12%
16	12	4	0	33%
17	48	7	0	15%
18	33	7	0	21%
20	61	37	0	61%
24	118	21	1	18%
26	80	2	4	3%
28	77	1	3	1%
29	51	2	1	4%
30	43	2	0	5%
31	37	4	3	11%
34	61	4	2	7%
35	45	16	0	36%
36	38	0	1	0%
37	40	5	1	13%
38	93	3	2	3%
43	130	4	6	3%
44	113	10	10	9%
45	116	2	6	2%
47	94	8	6	9%

Table 3: Error calculations for the first algorithm for pea crops

7.2 Pea Program Track 2

Using the second track of the algorithm, described in section 6.1.2, results were produced from the 20 images that had previously been manually counted to be able to evaluate the result. As before the number of peas, the number of weeds and weed area coverage were calculated in these images. Result images were produced with blue markings on objects classified as weeds.

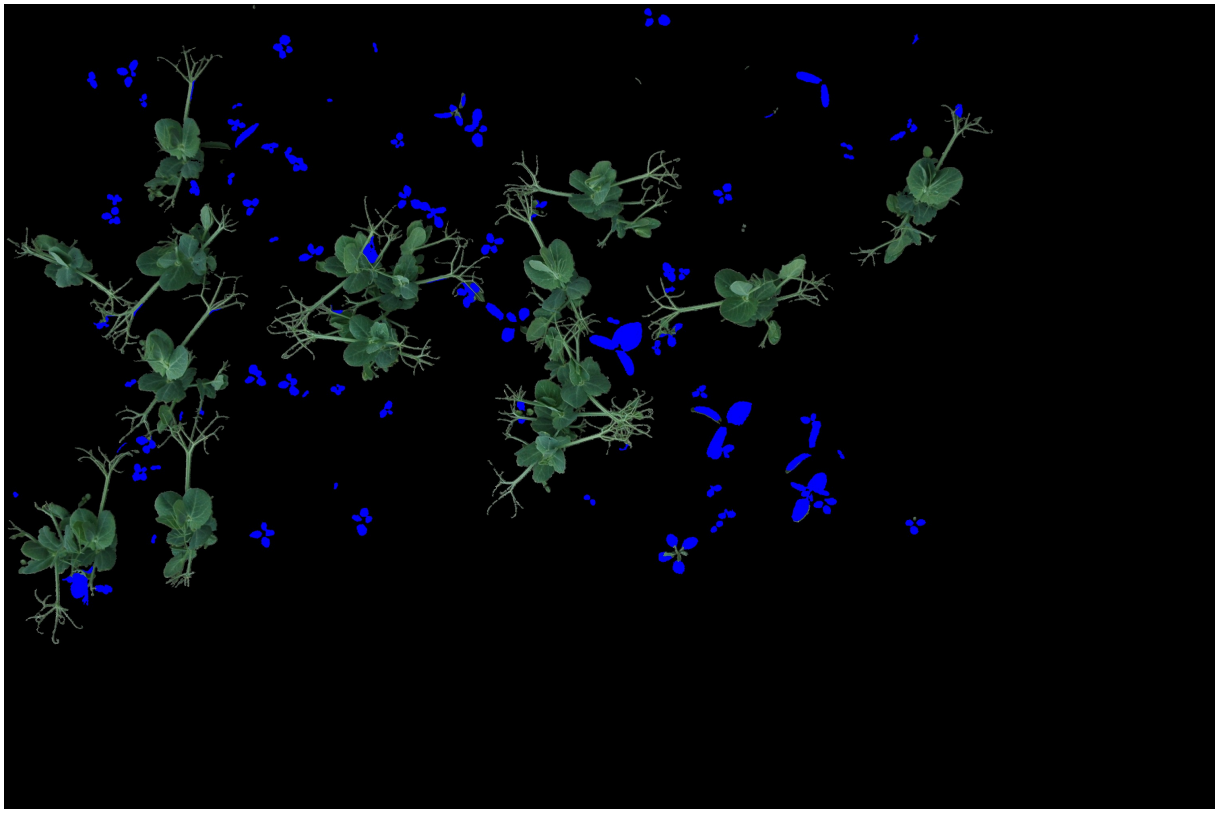


Figure 25: Example of a result image from the second version of the Pea program.

It can be seen in figure 25 that separate weed patches are detected as well as weeds in between tendrils. Some thin parts of weeds are not marked blue since they were removed in the top hat transform along with the tendrils. Those parts are not counted as pea plants by the program but were removed from the image when the area detector made the classification. Another type of misclassification comes from parts of pea plants that are cut off when the tendrils are removed and become small enough to be misclassified as weeds.

Quantitative data are presented in the following table:

Image	Number of Peas (NoP)	NoP (man. count)	Number of Weeds (NoW)	NoW (man. count)	Weed-fraction (%)	Box-fraction	NoP $/m^2$	NoW $/m^2$
9	10	16	36	25	1.0654	0.5541	48.1	173.3
16	18	23	15	6	0.3153	0.5553	86.4	72.0
17	19	21	34	28	0.4107	0.6625	76.5	136.9
18	7	8	16	13	0.2328	0.5187	36.0	82.3
20	26	28	28	10	0.4081	0.7297	95.0	102.3
24	16	21	67	67	1.3285	0.5187	82.3	344.5
26	10	16	85	63	2.1624	0.5187	51.4	437.0
28	16	24	74	47	0.9364	0.6121	69.7	322.4
29	8	12	48	37	0.8218	0.5737	37.2	223.1
30	10	12	41	26	0.4915	0.6088	43.8	179.6
31	13	13	34	29	0.5347	0.5515	62.9	164.4
34	10	18	55	40	1.0107	0.6016	44.3	243.8
35	11	15	24	18	0.2178	0.5242	56.0	122.1
36	6	8	24	21	0.5154	0.5187	30.8	123.4
37	15	17	32	20	0.3788	0.5187	77.1	164.5
38	6	8	76	70	1.7266	0.5187	30.8	390.7
43	8	12	90	75	2.9647	0.5187	41.1	462.7
44	11	17	93	77	2.8507	0.5187	56.6	478.1
45	7	14	90	86	3.6849	0.5187	36.0	462.7
47	9	14	64	50	1.9478	0.5187	46.3	329.0

Table 4: Results from the second algorithm for pea crops

The same analysis of the results as for the first program was done for the data presented in table 4. Figures 26 and 27 show graphs comparing the result with the manual count.

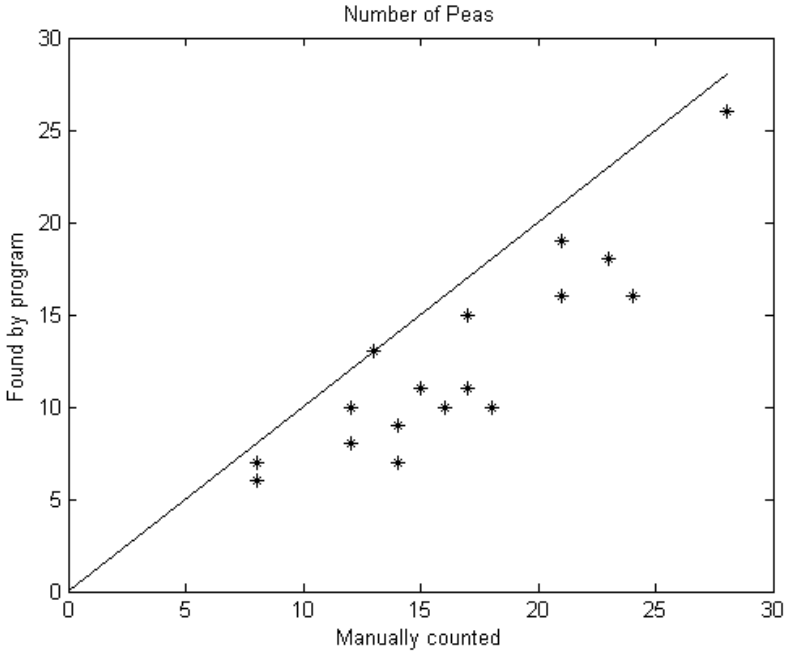


Figure 26: Number of pea plants found by the second program track compared with the manual count.

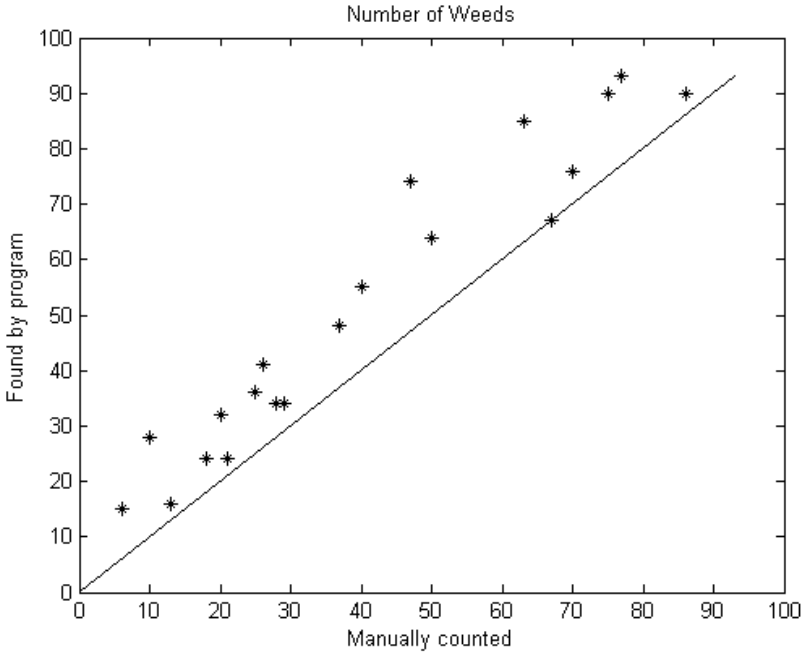


Figure 27: Number of weeds found by the second program track compared with the manual count.

Error calculations were performed also on these result images. The "Missed weeds" in table 5 are weeds or parts of weeds that were lifted out by the top hat. As mentioned they were not counted as pea plants however.

Image	Number of Weeds (program)	Peaparts classified as weed	Missed weeds	Weed misclassifications
9	36	11	3	31%
16	15	6	0	40%
17	34	9	0	26%
18	16	4	0	25%
20	28	15	1	54%
24	67	11	1	16%
26	85	9	3	11%
28	74	5	7	7%
29	48	7	4	15%
30	41	8	3	20%
31	34	5	3	15%
34	55	12	5	22%
35	24	5	4	21%
36	24	6	2	25%
37	32	10	4	31%
38	76	4	6	5%
43	90	5	10	6%
44	93	6	4	6%
45	90	9	18	10%
47	64	4	8	6%

Table 5: Error calculations for the second algorithm for pea crops

7.3 Rape Program

The program described in section 6.2 processed all 80 images of rape crops producing two result images from each as well as quantitative data saved to a file. In the result images plant centres found by the program are marked by dots of five different colours in a range from red to white. The colour of the dot indicates the size of the square used in the summation to find the plant centre. A red dot means the plant centre was found by the smallest square size (highest probability) and a white dot means the largest square was used (smallest probability). Examples of such result images from the four different fields are presented in figure 28. The second type of result image also contains the line markers from the leaves to help identify errors in the program.

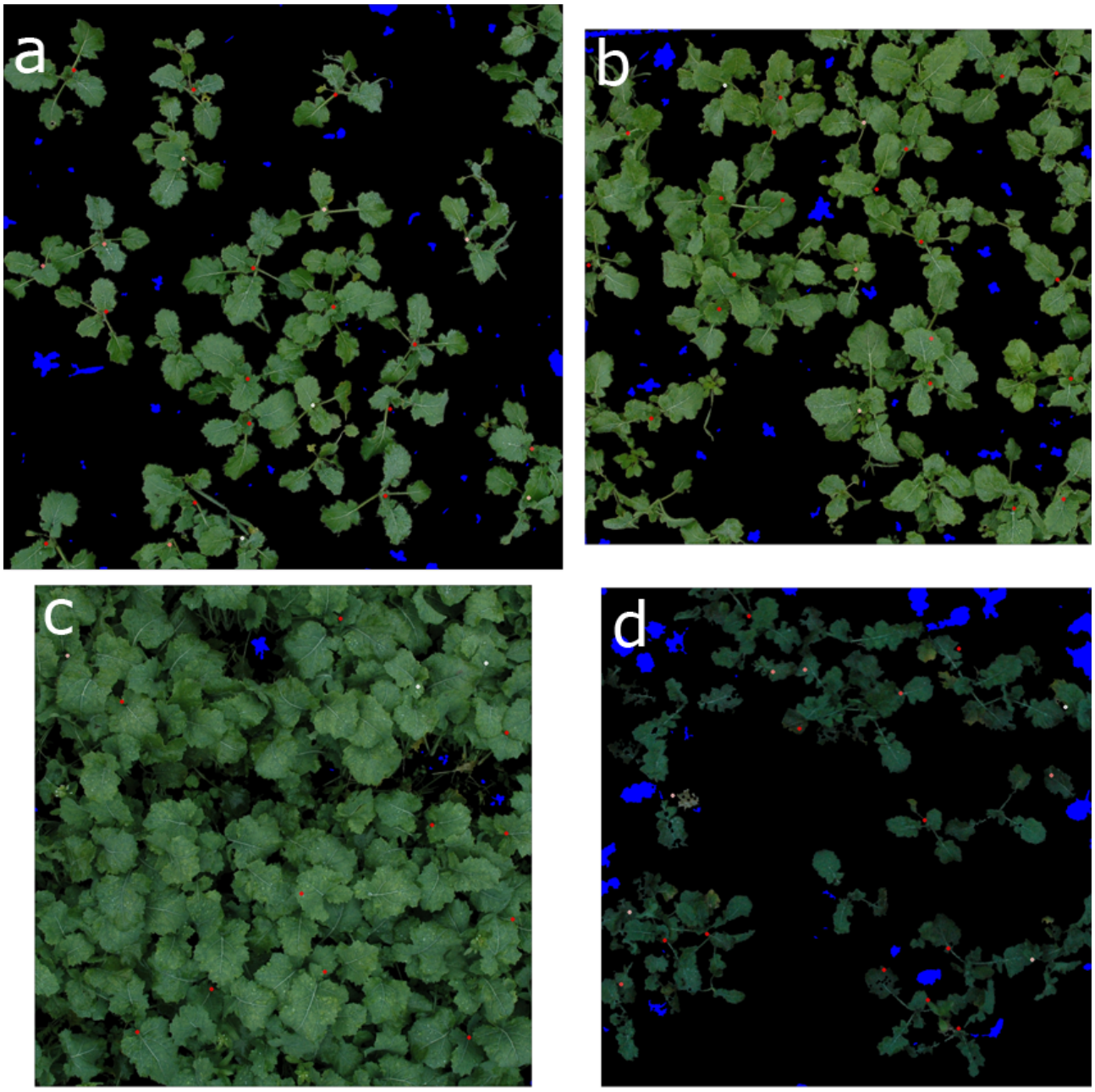


Figure 28: Four result images from the program for rape crops. The photos exemplify conditions on different fields: **a)** Bjertorp 1, **b)** Bjertorp 2, **c)** Axvall, **d)** Ribbingsberg. Blue markings show areas classified as weeds. Red, pink and white dots indicate plant centres. The different sizes of the images are due to the cropping to remove the wooden frame.

Results from the four different fields are given in tables 6, 7, 8, 9, 10 and 11. The first figure in the image number indicates on which of the four fields the photo was taken according to the following key.

- 1 - Bjertorp 1 (skifte 15)
- 2 - Bjertorp 2 (skifte 10)
- 3 - Axvall
- 4 - Ribbingsberg

The next two figures in the image number is the site number and the last figure (1 or 2) shows which of the two squares on the site the photo came from. The image size shows how much of the image that was left after cropping. As size reference the first image (1011) was chosen since the frame was well aligned in that photo, this made it possible to extract the full square meter in the cropping. The manual count of rape plants was performed on the field. This count includes all plants and is sometimes higher than the number of plants actually visible in the photos. The number of leaf vein skeletons gives an estimation of the number of leaves in the image. The areafraction weed is the total detected weed area in percent of the image size.

Image	Image Size (m^2)	Number of Rape plants (NoR)	NoR (man. count)	Number of leaf vein skeletons	Number of Weeds (NoW)	Area-fraction weed$(\%)$	NoR $/m^2$	NoW $/m^2$
1011	1.00	23	36	112	65	0.76	23.0	65.0
1012	0.85	17	38	80	65	1.38	20.1	76.7
1021	0.88	17	30	80	120	2.15	19.3	136.2
1022	0.86	10	37	55	145	2.19	11.6	168.7
1031	0.84	9	31	54	96	1.52	10.7	114.2
1032	0.84	9	36	44	114	2.29	10.7	135.9
1041	0.88	6	23	33	61	1.43	6.8	69.2
1042	0.89	2	29	21	112	2.14	2.2	125.9
1051	0.81	9	44	59	89	1.84	11.1	109.3
1052	0.87	14	43	78	48	1.07	16.0	54.9
1061	0.83	16	46	92	74	2.03	19.3	89.3
1062	0.87	21	40	105	96	2.13	24.2	110.6
1071	0.84	18	34	100	117	2.54	21.4	139.1
1072	0.86	11	30	62	143	4.33	12.8	165.9

Table 6: Results from the program for rape crops, photos from Bjertorp 1

Image	Image Size (m^2)	Number of Rape plants (NoR)	NoR (man. count)	Number of leaf vein skeletons	Number of Weeds (NoW)	Area-fraction weed$(\%)$	NoR $/m^2$	NoW $/m^2$
1081	0.89	21	43	103	92	1.60	23.6	103.2
1082	0.86	17	40	90	171	4.44	19.8	198.9
1091	0.81	5	24	26	182	3.36	6.2	225.8
1092	0.81	4	37	24	131	2.86	4.9	161.3
1101	0.87	11	43	81	66	1.45	12.6	75.5
1102	0.86	14	50	80	140	2.71	16.4	163.6
1111	0.87	21	67	135	81	2.02	24.1	93.1
1112	0.85	14	31	69	147	3.60	16.4	172.6
1121	0.89	15	43	93	142	2.52	16.9	159.7
1122	0.86	12	29	71	187	4.17	14.0	218.0
1131	0.91	8	36	55	93	1.50	8.7	101.7
1132	0.84	16	29	75	99	1.49	19.1	118.3
1141	0.90	7	31	47	360	4.05	7.7	397.9
1142	0.86	9	39	67	323	3.03	10.4	373.8
1151	0.89	21	47	113	38	0.51	23.6	42.6
1152	0.79	15	34	86	26	0.86	19.0	32.9
1161	0.82	17	50	95	43	0.94	20.8	52.5
1162	0.81	14	36	84	32	0.56	17.3	39.6
1171	0.84	9	29	57	344	6.06	10.7	410.0
1172	0.86	5	28	50	306	5.29	5.8	355.0
1181	0.86	5	21	34	281	4.32	5.8	326.0
1182	0.90	9	29	47	348	4.90	9.9	384.6
1191	0.87	12	32	73	89	1.32	13.8	102.5
1192	0.90	19	37	96	54	1.39	21.2	60.1

Table 7: Results from the program for rape crops, photos from Bjertorp 1

Image	Image Size (m^2)	Number of Rape plants (NoR)	NoR (man. count)	Number of leaf vein skeletons	Number of Weeds (NoW)	Area-fraction weed$(\%)$	NoR $/m^2$	NoW $/m^2$
2011	0.81	15	42	94	70	1.20	18.6	86.8
2012	0.86	16	42	79	67	1.05	18.6	77.9
2021	0.83	24	53	110	53	1.10	29.1	64.2
2022	0.82	15	54	102	92	1.68	18.3	111.9

Table 8: Results from the program for rape crops, photos from Bjertorp 2

Image	Image Size (m^2)	Number of Rape plants (NoR)	NoR (man. count)	Number of leaf vein skeletons	Number of Weeds (NoW)	Area-fraction weed$(\%)$	NoR $/m^2$	NoW $/m^2$
2031	0.84	17	53	98	128	0.75	20.3	152.6
2032	0.79	22	47	128	48	0.43	27.9	61.0
2041	0.82	19	42	110	32	0.64	23.2	39.1
2042	0.82	18	54	104	28	0.66	21.8	34.0
2051	0.84	12	46	72	113	2.14	14.3	134.7
2052	0.81	16	50	82	67	1.51	19.8	82.9
2061	0.80	8	30	47	72	1.39	10.0	90.2
2062	0.75	23	62	124	124	1.09	30.6	164.9
2071	0.81	2	41	28	252	2.35	2.5	310.3
2072	0.73	11	41	63	197	2.17	15.1	270.5
2081	0.74	4	47	29	185	3.26	5.4	250.0
2082	0.73	8	30	41	92	2.18	10.9	125.3
2091	0.80	10	54	69	61	1.80	12.5	76.3
2092	0.77	14	52	103	86	1.02	18.1	111.2

Table 9: Results from the program for rape crops, photos from Bjertorp 2

Image	Image Size (m^2)	Number of Rape plants (NoR)	NoR (man. count)	Number of leaf vein skeletons	Number of Weeds (NoW)	Area-fraction weed$(\%)$	NoR $/m^2$	NoW $/m^2$
3011	0.79	14	188	112	10	0.14	17.8	12.7
3012	0.76	14	94	105	3	0.02	18.5	4.0
3021	0.77	23	142	126	15	0.21	30.0	19.5
3022	0.78	19	88	112	12	0.14	24.4	15.4
3031	0.76	31	114	157	6	0.15	40.6	7.9
3032	0.77	25	128	157	15	0.13	32.3	19.4
3041	0.83	25	124	170	18	0.16	30.2	21.8
3042	0.76	35	124	206	29	0.55	45.8	38.0
3051	0.79	34	190	208	14	0.12	43.2	17.8
3052	0.75	33	130	201	7	0.06	43.9	9.3
3061	0.71	23	148	144	43	0.91	32.3	60.3
3062	0.77	32	100	195	37	0.56	41.6	48.1

Table 10: Results from the program for rape crops, photos from Axvall

Image	Image Size (m^2)	Number of Rape plants (NoR)	NoR (man. count)	Number of leaf vein skeletons	Number of Weeds (NoW)	Area-fraction weed$(\%)$	NoR $/m^2$	NoW $/m^2$
4011	0.77	9	35	74	156	2.90	11.8	203.8
4012	0.79	13	40	95	50	0.73	16.5	63.3
4021	0.81	7	32	61	28	0.55	8.6	34.6
4022	0.74	9	27	90	144	0.73	12.2	194.7
4031	0.79	6	42	65	20	0.25	7.6	25.2
4032	0.78	7	47	106	59	1.04	9.0	76.1
4041	0.79	6	19	32	486	2.85	7.6	618.5
4042	0.76	6	26	43	663	3.31	7.9	872.6
4051	0.76	4	52	62	20	0.11	5.3	26.3
4052	0.76	18	41	121	45	0.54	23.8	59.4
4061	0.77	19	41	143	38	2.27	24.6	49.1
4062	0.77	8	37	59	155	5.56	10.3	200.3

Table 11: Results from the program for rape crops, photos from Ribbingsberg

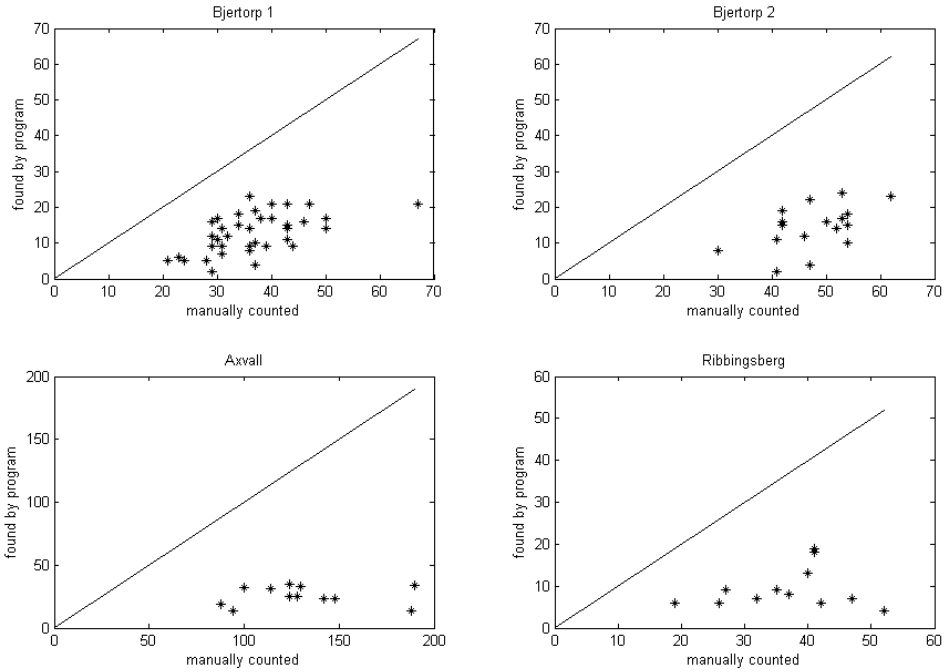


Figure 29: Manual count performed on the field plotted versus the number of plants found by the program. The four graphs show data from the separate fields.

Figure 29 shows a comparison between the result from the program and the manual count of plants on the four fields. For these graphs the full image material could be used in the comparison since all sites had been manually counted.

In some sites the plant density is very high, especially at the Axvall field as can be seen in the manually counted numbers in table 10. In figure 30 the plant+weed area fraction (the total area of plants and weeds in the image divided by total image area) is plotted versus the number of plants from the manual count (for all 80 images).

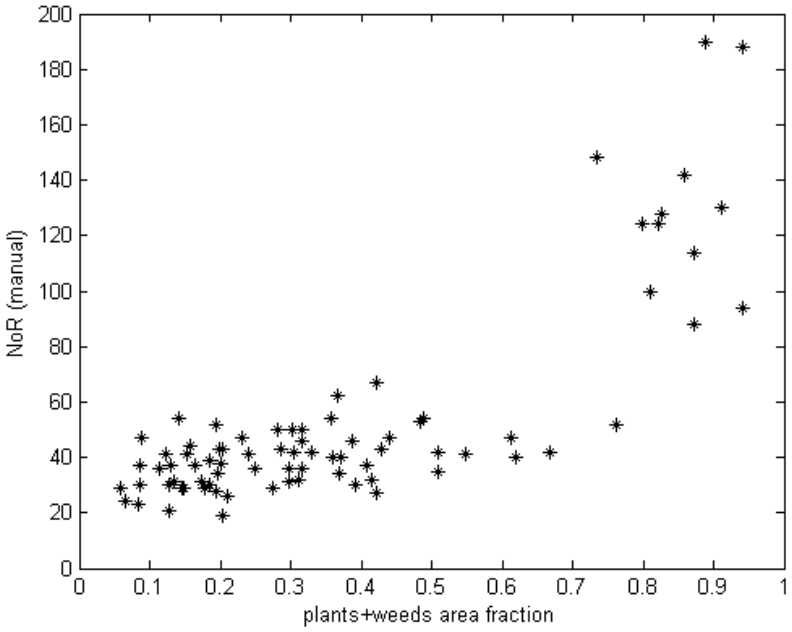


Figure 30: The number of plants as a function of vegetation coverage in the images.

By subtracting the number of plants found by the program from the manually counted numbers the dependency of vegetation coverage for the program error can be studied. See the plot in figure 31.

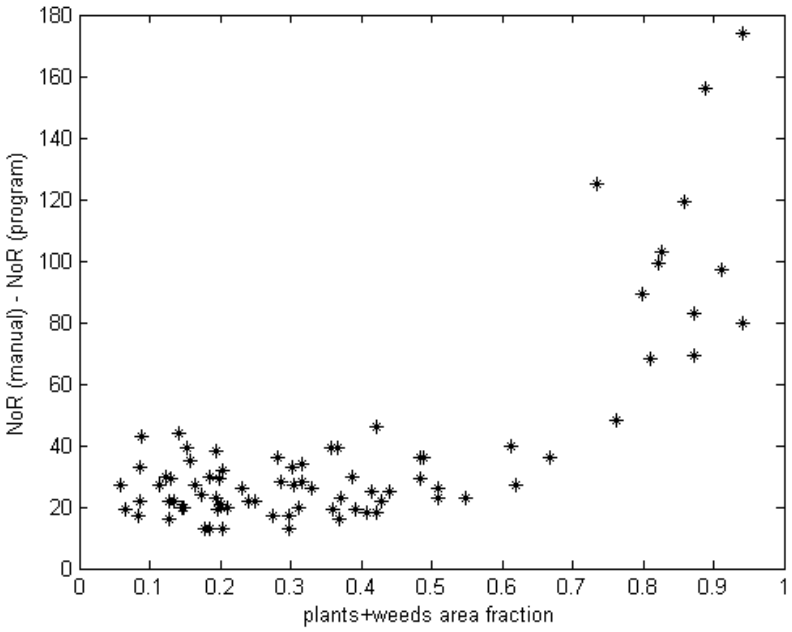


Figure 31: The difference in the plant quantity result as a function of vegetation coverage in the images.

8 Discussion

Pea program 1: In figures 23 and 24 a trend can be seen in the points towards the line that marks the desired result. The numbers of pea plants found by the program are in all cases lower than those found in the manual count. This was to be expected, the number of plants in the program are taken as the number of leaf centres that are removed by the tendril weed detector. Each pea plant has such a leaf centre but if two plants overlap they will only be counted as one. For example, the two pea plants in figure 18 will be seen as one.

Figure 24 shows that the number of weeds is overestimated in all images by the program. Probably this is mostly due to those weeds that split into several objects in the segmentation. Some errors are also made by the area detector where parts of pea tendrils that have been cut off in the segmentation are wrongly classified as weeds. An additional source of error in the weed quantity calculation is that weeds covered by tendrils sometimes will be counted as two separate weeds since they form separate objects on each side of the tendril. An example of this can be seen in figure 18 c and d. All these factors lead to overestimation in the number of weeds. In the error estimation calculated on 20 randomly chosen images (table 3) the misclassification rate of weeds vary from 0 to 61% with a mean value of 13%.

The covered weed area (areafraction weed) is very hard to compare to a correct result in a quantitative way. The only available method of evaluation is to study where the coloured areas in the result images are situated. From looking at those images it seems like the tendril weed detector is the one that makes most misclassifications. Since it separates weeds from tendrils based on colour the lighting in the image is an important factor. In darker images the tendrils are less bright and in those images a lot of errors are made. This causes errors not only in covered area calculations but also in weed and plant quantity. The high misclassification rates in certain images are mostly due to this problem. For this method to work in a more stable way uniform lighting conditions in all images would be preferable. In the brighter images very few errors are made and the value of the weed covered area should be fairly correct. The area based detectors seem to function well also in darker images.

Pea program 2: From table 5 it can be seen that the misclassification rate for this program is in general slightly higher than for the first version of the program. But the comparison with the counted reference in figures 26 and 27 indicates that the program performs at least equally well as the first algorithm in quantitative numbers. Some errors occur in the top hat transformation as weeds or parts of weeds are lifted out before the area based classification. But the high misclassification rates in some images are mostly caused by pea parts that get cut off when the tendrils are removed and then get classified as weeds. This type of errors also affects the covered weed area.

The first algorithm is more stable in its performance, especially in calculating weed covered area, but the second is less sensitive to bad lighting conditions since it does not use any method that is only based on colour. With more development the second method might be more reliable and then it would be the preferable of the two because of the robustness to lighting conditions.

Rape program: Figure 29 shows that the program for rape crops underestimates the number of plants in all images. One of the reasons for this is that the reference counting used was performed on the fields and includes plants that may be too small to be considered relevant and also plants that are hidden under leaves which mean they are not visible in the photos. That plants cover each other can be seen in figure 30. The jump in the trend that can be seen around 70% area coverage can be explained by overlapping of plants when the quantity is high on a small area. This also means that a smaller part of each plant is visible in each photo. In figure 31 it is evident that the program error in plant quantity is a lot higher for images with high vegetation density.

The sowing of the Axvall field was performed a couple of weeks earlier than on the other fields. Therefore those plants are larger and denser as can be seen in figure 28c. The photos from the Ribbingsberg field were taken at sunset, figure 28d is for example darker than the others. Darker images cause more errors in the segmentation and also in later steps of the algorithm. The conditions on the fields at Bjertorp were better with plants at an early growth stage and better lighting in the images. Figure 29 also indicates that the quantitative result were more accurate on those fields.

9 Conclusions

Perhaps the programs developed in this project are not ready to be used in real applications in their current state but the results still show that image analysis can be a useful tool for these types of tasks.

The performance depends a lot on the images used as input. When the plants are separated from each other in the images the results have been shown to be better. Also the lighting conditions are important to be able to make a reliable analysis. By acquiring all photos when the plants are small and under uniform lighting much would be gained.

Top hat transform has been the most useful of all image analysis methods evaluated in this project. It has proven to be very good for extracting thin features such as leaf veins or tendrils. Many types of crops have thin structures that could be used to identify the plant so top hat transform could probably be a good tool also in other crops.

10 Future Work

The results from the programs for pea crops indicate that the methods are not that far from a possible implementation in a real system. The second track of the algorithm for pea crops was put together rather quickly towards the end of the project and improvements could probably be made. The main reason for misclassifications in the second program is parts of pea plants being cut off when the tendrils are removed. Perhaps this could be solved by localising the large leaf centres of the plants and perform a closing operation with objects nearby.

The program for rape crops could also be improved in a number of ways. Firstly a method of performing a precise cropping should be developed if a wooden frame is to be put in the photos. For extracting the leaf veins *Soille* [8] describes another type of top hat transformation using a rank-max opening that possibly could give a better result. With the top hat used here side branches in the leaf vein pattern are often not obtained and thus the specific direction of the leaf is impossible to detect. To find the central vein in the pattern more sophisticated ways than measuring the distance between endpoints can certainly be used. In cases when this step goes wrong marker lines are sometimes obtained pointing in directions perpendicular to the leaf which is bound to cause errors in the search for plant centres. Perhaps the weighting of probability markers and the size of summation filters could also be optimized. So given more time for development this program can potentially detect rape plants in a much more reliable way.

Acknowledgment

I would like to thank my supervisor at SIK, Niklas Lorén, for all help and ideas and for always seeing things from a positive point of view. Thank you also to my examiner Lennart Svensson at Chalmers for great ideas and support. Thomas Börjesson at Lantmännen, Johan Nilsson, Anders Larsolle and Lena Engström at SLU all gave me good input which I am grateful for. Thanks also to everyone at SAM, SIK. It has been great working with you all.

References

[1] H. Blum. Biological Shape and Visual Science. *Journal of Theoretical Biology*, 38:205–287, 1973.

[2] M. L. Comer and E. J. Delp. Morphological Operations for Color Image Processing. *Journal of Electronic Imaging*, 8(3):279–289, 1999.

[3] K. H. Dammer and G. Wartenberg. Sensor-based Weed Detection and Application of Variable Herbicide Rates in Real Time. *Crop Protection*, 26:270–277, 2007.

[4] Ch. Gée, J. Bossu, G. Jones, and F. Truchetet. Crop/weed Discrimination in Perspective Agronomic Images. *Computers and Electronics in Agriculture*, 60(1):49–59, 2008.

[5] J. A. Marchant and C. M. Onyango. Shadow-invariant Classification for Scenes Illuminated by Daylight. *Journal of the Optical Society of America*, 17(11):1952–1961, 2000.

[6] J. C. Neto, G. E. Meyer, D. D. Jones, and A. K. Samal. Plant Species Identification Using Elliptic Fourier Leaf Shape Analysis. *Computers and Electronics in Agriculture*, 50:121–134, 2006.

[7] A. J. Pérez, F. Lopez, J. V. Benlloch, and S. Christensen. Color and Shape Analysis Techniques for Weed Detection in Cereal Fields. *Computers and Electronics in Agriculture*, 25:197–212, 2000.

[8] P. Soille. Morphological Image Analysis Applied to Crop Field Mapping. *Image and Vision Computing*, 18:1025–1032, 2000.

[9] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Brooks/Cole Publishing Company, 1993.

[10] A Tellaeche, X. P. Burgos-Artizzu, G. Pajares, and A. Ribeiro. A Vision-based Method for Weeds Identification through the Bayesian Decision Theory. *Pattern Recognition*, 41(2):521–530, 2008.

[11] R. K. Tetouev. Contour Recognition Based on Spectral Methods. Solution of the Problem of Choice of the Start-Point. *Pattern Recognition and Image Analysis*, 17(2):243–251, 2007.

[12] D. M. Woebbecke, G. E. Meyer, K. Von Bargen, and D. A. Mortensen. Plant Species Identification, Size and Enumeration Using Machine Vision Techniques on Near-Binary Images. *SPIE Optics in Agriculture and Forestry*, 1836:208–219, 1992.

[13] D. M. Woebbecke, G. E. Meyer, K. Von Bargen, and D. A. Mortensen. Color Indices for Weed Identification Under Various Soil, Residue, and Lighting Conditions. *Transactions of the ASAE*, 38(1):259–269, 1995.

A Matlab Code, Pea programs

A.1 Main pea program 1

```
clear all
close all
clc

global bildnamn WeedsizeArea WeedTendArea NumWeedA NumWeedT NumPea
global BoxArea BoxFraction colim AreaWeedim TendWeedim

first = true;
for i = 9179:9229
    bildnamn = ['../Artor/DSC_' num2str(i) '.JPG'];
    disp(['Running image DSC_' num2str(i) '.JPG']);
    Ia = imread(bildnamn);

    s = size(Ia);
    %Remove background
    [binim, colim] = Segment(Ia);

    %Remove edges and objects outside the "box"
    [NonEdgeim, BoxArea] = Edgerm(binim);
    BoxFraction = BoxArea/(s(1)*s(2));

    %Detect weedobjects
    [WeedsizeArea, NumWeedA, AreaWeedim] = AreaDetector(NonEdgeim);
    WeedFraction = WeedsizeArea/BoxArea;

    disp(['Detected weed coverage:' num2str(100*WeedFraction) ...
        '%, Number of Weeds:' num2str(NumWeedA)]);

    %Form binary image of the remaining pea objects
    binim = imsubtract(NonEdgeim, AreaWeedim);
    binim8 = im2uint8(~binim);
    %Form colour image of the remaining pea objects
    colim(:,:,1) = imsubtract(colim(:,:,1),binim8);
    colim(:,:,2) = imsubtract(colim(:,:,2),binim8);
    colim(:,:,3) = imsubtract(colim(:,:,3),binim8);

    %Detect weed near tendrils
```

```
    [WeedTendArea, NumWeedT, NumPea, TendWeedim] =...
        TendrilWeedDetector(binim, colim);
    WeedFraction = WeedTendArea/BoxArea;
    disp(['Detected Weed coverage: ' num2str(100*WeedFraction)...
        '%, Number of Weeds:' num2str(NumWeedT)]);

    WeedFraction = (WeedsizeArea+WeedTendArea)/BoxArea;
    TotNumWeed = NumWeedA+NumWeedT;
    disp(['Total Weed coverage: ' num2str(100*WeedFraction) ...
        '%, Number of Weeds:' num2str(TotNumWeed)]);
    disp(['Number of Peas:' num2str(NumPea) ...
        ', Boxfraction: ' num2str(100*BoxFraction) '%']);

    %Save calculated data and save a result image
    if first
        savedata(1);
        first = false;
    else
        savedata;
    end
    saveimage(true);
end
```

A.2 Segmentation

```
function [segbin, segcolor] = Segment(im)

%SEGMENT removes background from uint8 image.
%[segbin, segcolor] = Segment(im) returns a binary image segbin
%and a color image segcolor, both with background set to zero.
%
%segbin = Segment(im) returns only the binary segmented image

gray1 = imsubtract(im(:,:,2),im(:,:,1));
gray2 = imsubtract(im(:,:,2),im(:,:,3));
s = size(im(:,:,1));
segbin = false(s);

for i=1:s(1)
    for j=1:s(2)
        if gray1(i,j) > 7 && gray2(i,j) > 7
```

```
            segbin(i,j) = true;
        end
    end
end

% Remove small noise (<35 pixels)
L = bwlabel(segbin);
data = regionprops(L,'Area');
area = [data.Area];

brus = find(area < 35);
tabort = ismember(L,brus);
segbin = imsubtract(segbin,tabort);
segbin = im2bw(segbin);

% remove background from color image
segcolor = zeros(size(im));
segcolor = im2uint8(segcolor);
uint8bin = im2uint8(~segbin);

segcolor(:,:,1) = imsubtract(im(:,:,1),uint8bin);
segcolor(:,:,2) = imsubtract(im(:,:,2),uint8bin);
segcolor(:,:,3) = imsubtract(im(:,:,3),uint8bin);
```

A.3 Measurement box creator

```
function [binim_out, DmaxArea] = Edgerm(binim_in)

%EDGERM removes all objects that are touching the edge or are
%completely outside the measurement box in a binary image.
%
%binim_out = Edgerm(binim_in) returns a binary image with
%all such objects removed.
%
%[binim_out, DmaxArea] = Edgerm(binim_in) also returns the total
%area of the measurement box.

%Create binary mask of edges
s = size(binim_in);
edgeframe = false(s);
edgeframe(1:end,1) = true;
```

```
edgeframe(1:end,end) = true;
edgeframe(1,1:end) = true;
edgeframe(end,1:end) = true;

%Label all objects in the image
[L nO] = bwlabel(binim_in);

touch = L.*edgeframe;
edgeobj = unique(touch);
edgeobj = edgeobj(2:end);

%Remove objects touching the edge
remove = ismember(L,edgeobj);
binim_out = imsubtract(binim_in,remove);
binim_out = im2bw(binim_out);

%Find Dmax
data = regionprops(L,'MajorAxisLength');
MAL = [data.MajorAxisLength];
Dmax = round(max(MAL));
if Dmax > min(s)/3
    Dmax = round(min(s)/3);
    disp('Warning: Dmax is too big and therefore reduced')
end

%Create mask for Dmax-box
box = false(s);
box(1:s(1)-Dmax,1:s(2)-Dmax) = true;
touch = L.*box;

boxobj = unique(touch);
boxobj = boxobj(2:end);
allobj = 1:nO;
nonboxobj = setdiff(allobj,boxobj);

%Remove objects outside the box
remove = ismember(L,nonboxobj);
binim_out = imsubtract(binim_out,remove);
binim_out = im2bw(binim_out);

%Calculate Area of the Dmax box
DmaxArea = (s(1)-Dmax)*(s(2)-Dmax);
```

A.4 Area detector

```
function [WeedArea, NumWeed, Weedbinim] = AreaDetector(binim)

%AREADETECTOR classes all objects below an area limit in a binary
%image as weed. The limit is set between 3500 and 9125 pixels
%depending on the number of objects in the image.
%
%[WeedArea, NumWeed, Weedbinim] = AreaDetector(binim) returns the
%total detected weed area, the number of detected weed objects and
%a binary image with only weed objects.
%
%WeedArea = AreaDetector(binim) returns only the detected weed area.

%Measure area of all objects
disp('Running area weed detector');
[L nO] = bwlabel(binim);
data = regionprops(L,'Area');
area = [data.Area];

%Area Limit is set to a value between 3500 and 9125 depending on
%the number of objects
if nO <=110
    limit = abs(nO-35)*75+3500;
else
    limit = 4000;
end
small = find(area < limit);
Weedbinim = ismember(L,small);

%Repair broken weedobjects with a morphological close operation
se = strel('disk',5);
Weedbinimclosed = imclose(Weedbinim,se);

%Count the number of weeds and calculate their total area
[L nO] = bwlabel(Weedbinimclosed);
data = regionprops(L,'Area');
area = [data.Area];
WeedArea = sum(area);
NumWeed = nO;
```

A.5 Mean area detector and tendril weed detector

```
function [WeedArea, NumWeed, NumPea, Weedim] = ...
    TendrilWeedDetector(binim, colim)

%TENDRILWEEDDETECTOR detects weed near tendrils. Arguments should be a
%binary image and a  color image of the same objects.
%
%WeedArea = TendrilWeedDetector(binim, colim) returns the area of the
%detected weed.
%
%[WeedArea, NumWeed, NumPea, Weedim] = TendrilWeedDetector(binim, colim)
%also returns the number of detected weeds, the number of detected pea
%objects and a binary image with the detected weed.

disp('Running tendril weed detector');
s = size(binim);
%Label remaining objects
[L nO] = bwlabel(binim);
data = regionprops(L,'Area');
area = [data.Area];
MeanArea = mean(area);
WeedA = zeros(1,nO);
WeedN = zeros(1,nO);
PeaN = zeros(1,nO);

Weedim = binim;

for o=1:nO
    disp(['Treating object ' num2str(o) ' of ' num2str(nO)]);
    objind = [];
    objind = find(L==o);
    %Object indices in the full image
    [x1 y1] = ind2sub(s,objind);

    %Mean Area detector
    if area(o)/MeanArea < .1
        disp('Mean Area detector activated');
        WeedA(o) = area(o);
        WeedN(o) = 1;
        for i = 1:length(objind)
            Weedim(objind(i)) = 1;
```

```
        end
    else
        %Object indices in separate object box
        x2 = x1-(min(x1)-1);
        y2 = y1-(min(y1)-1);

        %Create box containing only one binary object
        binobj = false(max(x2),max(y2));
        %Create box containing only one colour object
        colobj = zeros(max(x2),max(y2),3);
        colobj = im2uint8(colobj);

        for i = 1:length(objind)
            binobj(x2(i),y2(i)) = binim(x1(i),y1(i));
            colobj(x2(i),y2(i),1) = colim(x1(i),y1(i),1);
            colobj(x2(i),y2(i),2) = colim(x1(i),y1(i),2);
            colobj(x2(i),y2(i),3) = colim(x1(i),y1(i),3);
        end

        %Send single object to subroutine ObjWeedArea
        [WeedA(o), WeedN(o), PeaN(o), weedmask] = ...
            ObjWeedArea(binobj, colobj);

        for i = 1:length(objind)
            Weedim(x1(i),y1(i)) = weedmask(x2(i),y2(i));
        end
    end

end
Weedim = im2uint8(Weedim);
WeedArea = sum(WeedA);
NumWeed = sum(WeedN);
NumPea = sum(PeaN);
```

A.6 Tendril weed detection on single object

```
function [WeedArea, NumWeed, NumPea, swbw] = ...
    ObjWeedArea(binobj, colobj)

%OBJWEEDAREA detects weed near tendrils of a single pea object.
%Arguments should be a binary image and a colour image of the
```

```
%same object.
%
%WeedArea = ObjWeedArea(binobj, colobj) returns the pixel area
%of the detected weed.
%
%[WeedArea, NumWeed, swbw] = ObjWeedArea(binobj, colobj) also
%returns the number of detected weeds and a binary image
%containing only the detected weed.

%Normal distribution parameters
mys = [32.2768 41.7997 34.3680];
lambdas = 1e3*[1.4655 1.8423 1.5582;
               1.8423 2.3569 1.9707;
               1.5582 1.9707 1.6654];
myo = [20.1220 33.3492 21.9074];
lambdao = 1e3*[0.3792 0.5992 0.4127;
               0.5992 1.0045 0.6667;
               0.4127 0.6667 0.4736];

objSize = size(binobj);

%Initiate images
swbw = false(objSize);
swbwkomp = false(objSize);
swc = zeros(size(colobj));
swc = im2uint8(swc);

%Find tendril skeleton (subroutine)
swbw = DMtendril(binobj,13);
%Disregard Distance information
swbw = im2bw(swbw);

%Dilate around skeleton
se = strel('disk',5);
swbw = imdilate(swbw,se);

%Subtract leaf centre from the binary object
swbw = imsubtract(binobj,~swbw);
swbw = im2bw(swbw);

%The complement of the original binary image
swbwkomp = imsubtract(binobj, swbw);
```

```
[L nO] = bwlabel(swbwkomp);
data = regionprops(L,'Area');
area = [data.Area];

%Throw away large objects (leaves)
tbindex = find(area < .2*max(area));
tpindex = find(area >= .2*max(area));
NumPea = length(tpindex);
swbwkomp = false(objSize);
swbwkomp = ismember(L,tbindex);

%Put back cut off parts to the tendrils
%The object should now contain only tendrils and surrounding weeds
swbw = imadd(swbw, swbwkomp);
swbw8 = im2uint8(~swbw);

swc(:,:,1) = imsubtract(colobj(:,:,1),swbw8);
swc(:,:,2) = imsubtract(colobj(:,:,2),swbw8);
swc(:,:,3) = imsubtract(colobj(:,:,3),swbw8);

swc = im2double(swc);
%Clear swbw and perform ML-test between tendrils and weeds
swbw = false(objSize);
x = zeros(1,3);
for i = 1:objSize(1)
    for j = 1:objSize(2)
        x = [swc(i,j,1),swc(i,j,2),swc(i,j,3)];
        if sum(x(1,:))>0.1 && (x-mys)*inv(lambdas)*(x-mys)'-...
                (x-myo)*inv(lambdao)*(x-myo)'<-0.3437
            swbw(i,j)=true;
        end
    end
end

swbw = imfill(swbw,'holes');
%Remove the outermost pixels (to separate objects)
swbw = bwdist(~swbw,'cityblock');
swbw = swbw >1;


[L nO] = bwlabel(swbw);
data = regionprops(L,'Area');
area = [data.Area];
```

```
%Remove small parts (tendril parts)
tbindex = find(area <= 100);
swbwkomp = false(objSize);
swbwkomp = ismember(L,tbindex);
%Binary mask of the detectd weed
swbw = imsubtract(swbw,swbwkomp);
%Dilate one pixel back
se2 = strel('square',2);
swbw = imdilate(swbw,se2);

%Calculate total area of the detected weed
[L nO] = bwlabel(swbw);
data = regionprops(L,'Area');
area = [data.Area];
WeedArea = sum(area);
NumWeed = length(area);
```

A.7 Distance mapped skeletons

```
function DMtendril = DMtendril(Im,tjock)
%DMstipel returns distancemapped skeleton of tendrils. Argument
%should be a binary pea object

se = strel('rectangle', [2 2]);
Im = imclose(Im,se);

%Calculate distance from each plant pixel to closest border
D = bwdist(~Im,'cityblock');
skel = bwmorph(Im,'skel',Inf);
%Mask with ones in the entire object except its skeleton
skelmask = immultiply(D,~skel);
%Distancemap-skeleton
DM = imsubtract(D,skelmask);
%Save only the thin parts of the skeleton
kant = DM >= 1 & DM <= tjock;

[L nO] = bwlabel(kant);
data = regionprops(L,'Perimeter');
perimeter = zeros(1,nO);
perimeter = [data.Perimeter];
```

```
small = find(perimeter < 250);
tabort = ismember(L,small);
%Remove small parts (parts of leaves)
kant = imsubtract(kant,tabort);
%Distancemap of (hopefully) only tendril skeletons
DMtendril = kant.*DM;
```

A.8 Tendril removal (second program)

```
function tendrilbin = tendrils(Ia)

% TENDRILS uses a to hat transform to localise tendrils
% in an image of pea plants.
% tendrilbin = tendrils(Image) returns a binary image
% of only tendrils.


disp('Running tendril detector');
%Intensity image (red channel)
int = Ia(:,:,1);

%Tophat filter
TH = imtophat(int,strel('square',10));

%Create binary image
level = graythresh(TH);
BW = im2bw(TH,level);
BW = imclose(BW,strel('disk',2));

%Measure Major axis length on all objects
L = bwlabel(BW);
data = regionprops(L,'MajorAxisLength');
MAL = [data.MajorAxisLength];

%Remove short objects
small = find(MAL<.25*max(MAL));
tabort = ismember(L,small);
tendrilbin = imsubtract(BW,tabort);
tendrilbin = imclose(tendrilbin,strel('disk',8));
tendrilbin = im2bw(tendrilbin);
```

B Matlab Code, Rape program

B.1 Main rape program

```
clear all
close all
clc

global imnum imsize NumWeed WeedArea TotArea NumLeaves
global Dirfound NumPlants

%Set parameters showimage and saveimage to true or false
showimage = true;
saveimage = false;

%Bjertorp 1
first = true;
for i = 101:119
    for j = 1:2
        imnum = 10*i+j;
        imname = ['../CroppadRaps/Bjertorp1/' num2str(imnum) '.jpg'];
        disp(['Running image ' num2str(imnum) '.jpg']);
        Im = imread(imname);

        s = size(Im);
        imsize = s(1)*s(2);

        %Remove background
        [binim, colim] = Segment(Im);

        %Detect weedobjects
        [WeedArea, NumWeed, TotArea, AreaWeedim] = AreaDetector(binim);
        WeedFraction = WeedArea/imsize;

        disp(['plant+weed coverage: ' num2str(100*TotArea/imsize) '%']);
        disp(['Detected weed coverage: ' num2str(100*WeedFraction)...
            '%, Number of Weeds: ' num2str(NumWeed)]);
```

```
        %Binary image of the remaining rape objects
        binim = imsubtract(binim, AreaWeedim);
        binim8 = im2uint8(~binim);
        %Color image of the remaining rape objects
        colim(:,:,1) = imsubtract(colim(:,:,1),binim8);
        colim(:,:,2) = imsubtract(colim(:,:,2),binim8);
        colim(:,:,3) = imsubtract(colim(:,:,3),binim8);
        %Mark detected weeds blue
        Weed8 = im2uint8(AreaWeedim);
        colim(:,:,3) = imadd(colim(:,:,3),Weed8);

        %Find leaf vein skeletons
        vskelim = Leafveins(colim);

        %Find plant centre probability markers
        [Markerim, Refim, NumLeaves, Dirfound] =...
            findmarkers(vskelim,colim,showimage,saveimage);

        %Find plant centers
        NumPlants =...
            findcenters(Markerim,Refim,colim,showimage,saveimage);

        if first
            savedata(1);
            first = false;
        else
            savedata;
        end
    end
end
```

B.2 Leaf vein extraction

```
function skelim = Leafveins(Ia)

%LEAFVEINS finds leaf veins in a colour image using a top
%hat transform.
%skelim = Leafveins(Image) returns a binary image
%of thinned (skeletonised) leaf veins.
```

```
%Intensity image (red channel)
int = Ia(:,:,1);

%Create mask of Leaf edges
bin = im2bw(int,.9/255);
D = bwdist(~bin,'cityblock');
middle = D > 6;
edge = imdilate(middle,strel('disk', 6));
edge = imsubtract(edge,middle);
edge = im2uint8(edge);

%Tophat filter
TH = imtophat(int,strel('square',3));

%remove edges
TH = imsubtract(TH,edge);

%Create binary image
level = graythresh(TH);
BW = im2bw(TH,level);

%Measure Major axis length
L = bwlabel(BW);
data = regionprops(L,'MajorAxisLength');
MAL = [data.MajorAxisLength];

%Remove short objects
small = find(MAL<.25*max(MAL));
tabort = ismember(L,small);
veinbin = imsubtract(BW,tabort);
veinbin = imclose(veinbin,strel('disk',8));

%Skeltonisation
skelim = bwmorph(veinbin,'thin',Inf);
```

B.3 Plant centre line markers

```
function [Markerim, Referenceim, NumLeaves, Dirfound] =...
    findmarkers(skelim, colim, show, save)
```

```
% FINDMARKERS produces an image with plant centre probability
% line markers from a binary image of leaf vein skeletons.
% function [Markerim, Referenceim, NumLeaves, Dirfound] =...
%    findmarkers(skelim, colim, show, save)
% returns Markerim with the marker lines, Referenceim which
% is an image of the same size containing reference numbers for
% all markers. Output data NumLeaves is the number of leaf vein
% skeletons in the image and Dirfound is the number of skeletons
% where the precise direction was found. Input skelim should be a
% binary image of leaf vein skeletons. Colim should be the original
% image of plants (for display). Parameters show and save should
% be set to true or false depending on if the result should be
% displayed and/or saved to file.


global vectorim

s = size(skelim);
%Put one pixel thick frame around the colour image
colimframe = zeros(s(1)+2,s(2)+2,3);
colimframe = im2uint8(colimframe);
colimframe(2:s(1)+1,2:s(2)+1,:) = colim;
vectorim = colimframe;
%Put one pixel thick frame around the skeleton image
skelimframe = false(s+2);
skelimframe(2:s(1)+1,2:s(2)+1) = skelim;
if show
    figure, imshow(colimframe)
    hold on
end

s = size(skelimframe);
z = s(1)*s(2);
%Create image where line markers will be saved
Markerim = zeros(s);
Markerim = im2uint8(Markerim);
%Create Reference image
Referenceim = zeros(s);
Referenceim = im2double(Referenceim);
%Parameter mindist determines the length of the line markers
mindist = 100;
Dirfound = 0;
```

```
[L NumLeaves] = bwlabel(skelimframe);
disp([’Number of leaves: ’ num2str(NumLeaves)])
data = regionprops(L,’BoundingBox’);

%This loops goes through all skeletons and treats them separately
for o = 1:NumLeaves
    %-------------------------------------------------------------
    % Create frame (1 pixel thick) around single skeleton object
    %-------------------------------------------------------------
    BB = data(o).BoundingBox;
    xMin = floor(BB(2));
    xMax = xMin+BB(4);
    yMin = floor(BB(1));
    yMax = yMin+BB(3);

    skelobj = false(fliplr(BB(3:4))+3);
    skelobj(2:end-1,2:end-1) = skelimframe(xMin:xMax,yMin:yMax);

    %-------------------------------------------------------------
    % Get end points and intersections
    %-------------------------------------------------------------
    endPoints = getEndPoints(skelobj);
    intersections = getIntPoints(skelobj);

    %-------------------------------------------------------------
    % Convert end and intersection points to Java vectors
    %-------------------------------------------------------------
    if (~isempty(endPoints))
        endVector = javaMethod(’createPointVector’,’GelTree’,...
            endPoints(:,1)-1,endPoints(:,2)-1);
    else
        endVector = java.util.Vector;
    end
    if (~isempty(intersections))
        intVector = javaMethod(’createPointVector’,’GelTree’,...
            intersections(:,1)-1,intersections(:,2)-1);
    else
        intVector = java.util.Vector;
    end

    %-------------------------------------------------------------
    % Remove branches shorter than minBranchSize
    %-------------------------------------------------------------
```

```
minBranchSize = 5;
I_clean = javaMethod('cleanSkeleton','GelTree',skelobj,...
    endVector,intVector,minBranchSize,1,size(skelobj,1),...
    1,size(skelobj,2));

%Find end points and intersections
ep = getEndPoints(I_clean);
ip = getIntPoints(I_clean);
%Convert to coordinates in the original image
ep(:,1) = ep(:,1)+yMin-2;
ep(:,2) = ep(:,2)+xMin-2;
ip(:,1) = ip(:,1)+yMin-2;
ip(:,2) = ip(:,2)+xMin-2;
%Vectors where distances of projected points to central vein
%endpoint will be stored
edist = 0;
idist = 0;

if size(ep,1)>1
    %Find the central leaf vein (longest vein)
    maxlen = 0;
    for j=1:size(ep,1)-1
        for k=j+1:size(ep,1)
            len = norm(ep(j,:)-ep(k,:));
            if len > maxlen
                maxlen = len;
                p(1,:) = ep(j,:);
                p(2,:) = ep(k,:);
                longind = [j k];
            end
        end
    end

    v = p(2,:)-p(1,:); %Direction vector of central vein
    ep(max(longind),:) = [];    %Remove central vein from ep
    ep(min(longind),:) = [];
    %If possible, gather information to find vein orientation
    if size(ep,1)>0
        %Projections of endpoints onto central vein
        for j = 1:size(ep,1)
            w = ep(j,:)-p(1,:);
            a = dot(v,w)/norm(v)^2;
            edist(j) = norm(a*v);
```

```
    end
    %Projections of intersection points onto central vein
    for j = 1:size(ip,1)
        w = ip(j,:)-p(1,:);
        a = dot(v,w)/norm(v)^2;
        idist(j) = norm(a*v);
    end
end

%Display/save result
if show
    plot(p(:,1),p(:,2))     %Central vein (blue)
    plot(p(:,1),p(:,2),'or')    %Central vein endpoints (red)
end
if save
    %Draw blue line on central vein
    ind = drawline(fliplr(p(1,:)),fliplr(p(2,:)),s);
    vectorim(ind) = 0;
    vectorim(ind+z) = 0;
    vectorim(ind+2*z) = 255;
    %Mark endpoints of central vein with red rings
    radius = min([5 p(1,2)-1 s(1)-p(1,2) p(1,1)-1 s(2)-p(1,1)]);
    vectorim(:,:,1) = MidpointCircle(vectorim(:,:,1),...
        radius, p(1,2), p(1,1), 255);
    vectorim(:,:,2) = MidpointCircle(vectorim(:,:,2),...
        radius, p(1,2), p(1,1), 0);
    vectorim(:,:,3) = MidpointCircle(vectorim(:,:,3),...
        radius, p(1,2), p(1,1), 0);
    radius = min([5 p(2,2)-1 s(1)-p(2,2) p(2,1)-1 s(2)-p(2,1)]);
    vectorim(:,:,1) = MidpointCircle(vectorim(:,:,1),...
        radius, p(2,2), p(2,1), 255);
    vectorim(:,:,2) = MidpointCircle(vectorim(:,:,2),...
        radius, p(2,2), p(2,1), 0);
    vectorim(:,:,3) = MidpointCircle(vectorim(:,:,3),...
        radius, p(2,2), p(2,1), 0);
end
t = mindist/norm(v);
%Determine direction of the leaf if possible
if sum(edist)>0 && sum(idist)>0 &&...
        abs(mean(idist)-mean(edist))>3
    Dirfound = Dirfound+1;
    if mean(idist)<mean(edist)
        %p(1,:) was found to be closest to plant center
```

```
            q = round(p(1,:)-t*v);
            ind = drawline(fliplr(p(1,:)),fliplr(q),s);
            Markerim(ind) = Markerim(ind) + 3;
            Referenceim(ind) = o;
            if save
                %Draw cyan colored line in result image
                vectorim(ind) = 0;
                vectorim(ind+z) = 255;
                vectorim(ind+2*z) = 255;
            end
            if show
                plot(p(1,1),p(1,2),'xc','MarkerSize',10)
                plot([p(1,1);q(1)],[p(1,2);q(2)],'c')
            end
        else
            %p(2,:) was found to be closest to plant centre
            q = round(p(2,:)+t*v);
            ind = drawline(fliplr(p(2,:)),fliplr(q),s);
            Markerim(ind) = Markerim(ind) + 3;
            Referenceim(ind) = o;
            if save
                %Draw cyan colored line in result image
                vectorim(ind) = 0;
                vectorim(ind+z) = 255;
                vectorim(ind+2*z) = 255;
            end
            if show
                plot(p(2,1),p(2,2),'xc','MarkerSize',10)
                plot([p(2,1);q(1)],[p(2,2);q(2)],'c')
            end
        end
    else
        %No direction found
        q = round(p(1,:)-t*v);
        ind = drawline(fliplr(p(1,:)),fliplr(q),s);
        Markerim(ind) = Markerim(ind) + 2;
        Referenceim(ind) = o;
        if save
            %Draw magenta colored line in result image
            vectorim(ind) = 255;
            vectorim(ind+z) = 0;
            vectorim(ind+2*z) = 255;
        end
```

```
            if show
                plot([p(1,1);q(1)],[p(1,2);q(2)],'m')
            end

            q = round(p(2,:)+t*v);
            ind = drawline(fliplr(p(2,:)),fliplr(q),s);
            Markerim(ind) = Markerim(ind) + 2;
            Referenceim(ind) = o;
            if save
                %Draw magenta colored line in result image
                vectorim(ind) = 255;
                vectorim(ind+z) = 0;
                vectorim(ind+2*z) = 255;
            end
            if show
                plot([p(2,1);q(1)],[p(2,2);q(2)],'m')
            end
        end
        if show
            drawnow
        end
    end
end
```

B.4 Plant centre detector

```
function NumPlants = findcenters(Markerim, ref, colim, show, save)

% FINDCENTERS performs a search for probable plant centres in an
% image of line markers.
% NumPlants = findcenters(Markerim, ref, colim, show, save) returns
% NumPlants as the number of detected plant centres. Input Markerim
% should be an image with line markers. ref should be an image of the
% same size containing reference numbers for the lines. Colim should be
% the original image of plants (for display). Parameters show and save
% should be set to true or false depending on if the result should be
% displayed and/or saved to file.

global imnum vectorim

s = size(colim);
```

```
%Put one pixel thick frame around the colour image
centreim = zeros(s(1)+2,s(2)+2,3);
centreim = im2uint8(centreim);
centreim(2:s(1)+1,2:s(2)+1,:) = colim;

%Create image that will be used to remove marker lines
touch = zeros(size(ref));
%Matrix where plant center positions will be saved
centers = zeros(1,2);

%Size of summation squares
sumsize = [3 5 8 20 30];
divider = zeros(1,length(sumsize)+1);
%Intensity requirement after summation
req = [12 20 32 80 120];
%Colour arguments for the show command
farg = (['oy'; 'or'; 'om'; 'oc'; 'ob']);
%Colour arguments for the save command
probcolor = [0 64 127 191 255];
%Vector where centre dot colour will be stored for each centre
centcolor = 0;
%Parameter mindist determines the radius of the circle in
%which markers and centre candidates will be removed
mindist = 100;

%This loop goes through the summation sizes
for l=1:length(sumsize)
    markersum = filter2(ones(sumsize(l)),Markerim);
    %Find plant centre candidate positions
    index = find(markersum>req(l));
    %Sort candidates in descending order
    [blaj tempindex] = sort(markersum(index),'descend');
    index = index(tempindex);

    %Go through list of candidates and put plant centre markings
    %or remove the candidate from list.
    while length(index)>0
        tooclose = false;
        [a b] = ind2sub([s(1) s(2)]+2,index(1));
        for d = 2:size(centers,1)
            %Check if the current point is too close to a previous
            %centre point
            if norm(centers(d,:)-[a b])<mindist
```

```
                tooclose = true;
                index(1) = [];
                break
            end
        end
        if ~tooclose
            %Decide radius of removal circle (mindist if not too
            %close to the image border).
            r = min([mindist a-1 s(1)-a b-1 s(2)-b]);
            for j = 0:r
                touch = MidpointCircle(touch, j, a, b, 1);
            end
            %Save plant centre position and dot colour
            centers(end+1,:) = [a b];
            centcolor(end+1,:) = probcolor(l);
            index(1) = [];
        end
    end

    %Remove lines close to the detected plant centres
    touch = touch.*ref;
    remove = ismember(ref,nonzeros(unique(touch)));
    remove = im2uint8(remove);
    Markerim = imsubtract(Markerim,remove);
    divider(l+1) = size(centers,1)-1;
end

centers(1,:) = [];
centcolor(1) = [];
%Number of plant centres found
NumPlants = size(centers,1);
disp(['Number of plants: ' num2str(NumPlants)])

%Display and/or save result image
if show
    plot(centers(:,2),centers(:,1),'oy','Markersize',4)
    title(['Number of plant centers: ' num2str(size(centers,1))])

    figure, imshow(centreim)
    hold on
    for l = 1:length(sumsize)
        plot(centers(divider(l)+1:divider(l+1),2),...
            centers(divider(l)+1:divider(l+1),1),...
```

```
            farg(l,:),'Markersize',4);
    end
    title(['Number of plant centers: ' num2str(size(centers,1))])
end
if save
    for i = 1:size(centers,1)
        for j = 1:6
            radius = min([j centers(i,1)-1 s(1)-centers(i,1)...
                centers(i,2)-1 s(2)-centers(i,2)]);
            vectorim(:,:,1) = MidpointCircle(vectorim(:,:,1),...
                radius, centers(i,1), centers(i,2), 255);
            vectorim(:,:,2) = MidpointCircle(vectorim(:,:,2),...
                radius, centers(i,1), centers(i,2), 255);
            vectorim(:,:,3) = MidpointCircle(vectorim(:,:,3),...
                radius, centers(i,1), centers(i,2), 0);
            centreim(:,:,1) = MidpointCircle(centreim(:,:,1),...
                radius, centers(i,1), centers(i,2), 255);
            centreim(:,:,2) = MidpointCircle(centreim(:,:,2),...
                radius, centers(i,1), centers(i,2), centcolor(i));
            centreim(:,:,3) = MidpointCircle(centreim(:,:,3),...
                radius, centers(i,1), centers(i,2), centcolor(i));
        end
    end
    imtitle = [num2str(imnum) 'Vectors.jpg'];
    imwrite(vectorim, imtitle, 'jpeg', 'Quality',100);
    imtitle = [num2str(imnum) 'Centers.jpg'];
    imwrite(centreim, imtitle, 'jpeg', 'Quality',100);
end
```