Universitetet i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program/specialization: Information Technology – Automation and Signal Processing	Spring semester, 2016 Open
Author: Vegard Brattland	…… (signature author)
Instructor: Tormod Drengstig Supervisor(s): Tormod Drengstig and Ivar Austvoll	
Title of Master's Thesis: Image Processing and Analysis of Leaf Movement in Mimosa Pudica Norwegian title: Bildebehandling og Analyse av Bladbevegelse I Mimosa Pudica	
ECTS: 30	
Subject headings: Image processing, Black box model estimation, Optical flow, Mimosa Pudica, Leaf movement	Pages: 85 + attachments/other: 23 Kristiansand, 15th of June Date/year

[This page is intentionally left blank]

Abstract

This thesis has the aim of develop and improve methods used in an ongoing research project at the University of Stavanger, where the plant *mimosa pudicas* response to change in its illuminative conditions is being examined. The intention is to use these methods to analyse the *mimosa pudicas* responses, as well as exploring the possibility of relating them to a control theoretic viewpoint.

An image processing routine, using the HSV color model and triangle intensity threshold segmentation, was developed to segment time-lapse image series of the *Mimosa Pudica* - quantifying the plants image pixel count as a metric. The result was used to estimate the ARX, ARMAX and Box Jenkins *black box* models, with the plants illuminative condition and image pixel count being used as the systems input and output. A routine for a second metric was also developed, using the method of Farneback to estimate dense optical flow, describing the plants movement orientation in the image plane.

The automatic non-parametric image segmentation routine performance was evaluated on 7 test images, obtaining a DICE similarity coefficient of 0.932 ± 0.0175. Several ARX, ARMAX and Box Jenkins models was estimated without being able to capture the systems dynamics. The routine describing *mimosa pudicas* movement orientation was during the experiment found to give a seemingly correct description of a small time-lapse sequence.

The successfully development of a automatic non-parametric image segmentation routine will remove the need of manual intervention. The observed dynamics in the *mimosa pudica* indicates that the changes in the illuminative condition could be working as a disturbance on the plant. The dense optical flow experiments did also shown patterns of the *mimosa pudica* starting its response to the shift in the illuminative condition, before the actual shift occurred. This suggests that a complex structure could lie behind the plants response to the illumination regimes.

Acknowledgements

This thesis marks the end of my studies in the field of Information Technology - Automation and Signal Processing, at the University of Stavanger. I would first like to thank my supervisor Professor Tormod Drengstig for his help and guidance during the last semester. I would also like to thank Dr Ivar Austvoll for his guidance and feedback on the image processing and motion estimation parts of the thesis .

My last thank goes to my spouse, friends and work-colleagues for their patience and support - which has kept me going the past five years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The work presented in this thesis has its origin from an ongoing research project at the Institute of Data and Electronics (IDE) and Centre for Organelle Research (CORE) at the University of Stavanger, and deals with responses observed in the plant *mimosa pudica* during changes in its illuminative condition.

The observation of the *mimosa pudica* plant over an extensive period of time reveals that the plant opens and closes its leafs based on its illuminative condition, as displayed in figure 1. These responses are described in the literature [24] as being related to the plants circadian rhythm, which is a periodic, self-sustaining and temperature compensating [26] biological process.

Figure 1: *Mimosa Pudica* opening and closing leafs over 24 hours

Further research on the plants responses led to the construction of an experimental environment, where a camera was installed to captured the *mimosa pudicas* response to change in its illuminative conditions. These response was quantified by an image processing routine, where the amount of image pixels belonging to the *mimosa pudica* was used as a metric.

Experiments were then performed where a pattern of periodic illumination, imitating the daily period of day and night, followed by a constant period with no changes in the illumination - resulted in a oscillatory response of the *mimosa pudica*. This is from the literature known to be the plants *free-running* period, originating from its self-sustaining attribute, where an approximatly 24 hour rhythm continues to run during the constant illumination conditions [34].

One of the interesting observations that was found while studying the literatures description of the *free-running* period, is that the *free-running* period often was illustrated to continue its oscillation with a constant amplitude. This description was different from the observations that was done during the experiments at IDE, where the metric that indicated the plants response is displayed in figure 2. This figure reveals an oscillatory response that is slowly decreasing towards a steady state, which is seemingly similar to the underdamped step response known from the field of control theory.

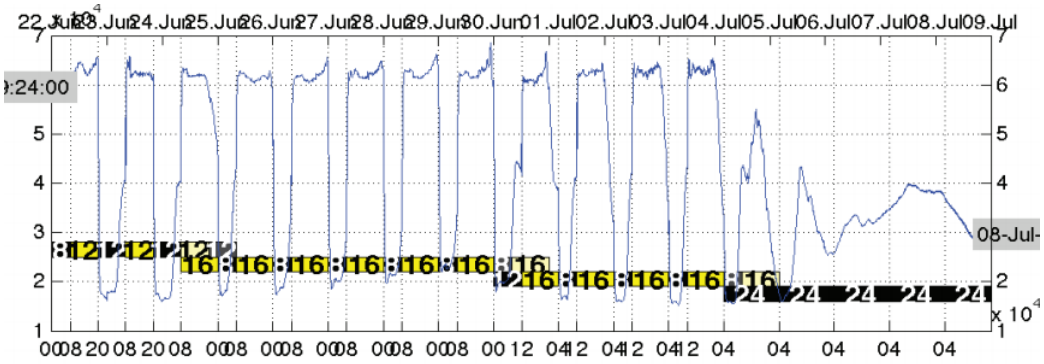


Figure 2: Oscillatory response, preliminary experiment

This observation raised the questions; Is the *mimosa pudicas* response purely biological, or can the plants responses be related to a control theoretical viewpoint, and is it possible to estimate mathematical models that can describe the system?

1.1 Approach and limitations

The scope of this thesis includes the continuation of the preliminaries work that have been done at IDE. The first part involves the improvement of the experiments image processing routine, which quantify the responses by using the amount of image pixels belonging to the *mimosa pudicas* in each image as a metric of its response. This operation is as today performed by a routine that utilizes the RGB color model and the fully implemented Otsu's Matlab function in its image processing. The challenge with the routine is currently that three different threshold limits must be adjusted, which require user to perform a trail and error search to obtain the optimal parameters. The intention is to re-develop the method to a automatic non-parametric routine.

The proposed idea is then to use the results of the image processing method as a metric in the estimation of linear *black box* models. It have been chosen to limit the experiments to include three different models, which is the linear ARX, ARMAX and Box Jenkins models. These estimation is performed by considering the illuminative condition as the systems input and the plants response, found by the image processing, as the systems output.

It has also been chosen to include motion estimation in the scope of the thesis, motivated by the recent published article *Tracking Rhythms in Plants, an automated leaf movement analysis program for circadian period estimation* [12]. The authors of this article presents a method for vertical movement estimation of plants. It will in this thesis be will look at the possibility of develop an alternative metric, where one is interested in the possibility of quantifying the plants horizontal movement in the image plane.

1.2 Organization

It have been chosen to organize the thesis as following

Chapter 2 - Experimental Setup
This chapter contains a brief introduction to the botanical terminology, material and methods used in the thesis experiments.

Chapter 3 - Image Processing
This chapter contains a brief introduction to the botanical terminology, material and methods used in the thesis experiments.

Chapter 4 - System Identification
This chapter contains an introduction to the ARX, ARMAX and Box Jenkins black box models, and a description of the models goodness evaluation.

Chapter 5 - Motion Estimation
The chapter explains the theoretical foundation behind the Farneback method for estimating optical flow.

Chapter 6 - Implementation
This chapter contain a brief and general explanation of the most important Matlab functions developed for the experiments in the thesis.

Chapter 7 - Experiments
This chapter includes a detailed description about the experiments performed in the thesis.

Chapter 8 - Results and Discussion
This chapter contains discussion and evaluation of the results obtained in the experiment chapter.

Chapter 9 - Conclusion
This chapter contains the conclusion of the work performed in the thesis.

Appendix
Contains the Matlab code developed and used in the thesis.

Chapter 2

Experimental Setup

The purpose of this chapter is to present the experimental setup and introduce the essential botanical terminology to the reader. The first section is intended to give an introduction of the *mimosa pudica*, which is the plant used in the experiment. It is then given a brief explanation of plant movement in the following section, before the chapter is ended with a presentation of the experimental environment.

2.1 Mimosa Pudica

The plant used in the thesis experiments is the *mimosa pudica*, known as the *common sensitive plant*, *sensitive plant*, et alia, shown in figure 3. The *mimosa pudica* originates from the tropical parts of America and is a perennial plant (can live several years) that usually grows to be about 15 - 45 cm tall.

Figure 3: Mimosa Pudica

The plants in the experiment is first grown for 3 - 4 weeks, before they are moved to the experimental environment that is introduced in section 2.3. The young plants will usually consist of two to four compound leafs that contain between 7 and 10 pairs of leaflets arranged in a bi-pinnate (feather like) pattern. These leaflets is observed to be in the color range of dark greenish to yellowish with an oblong shape.

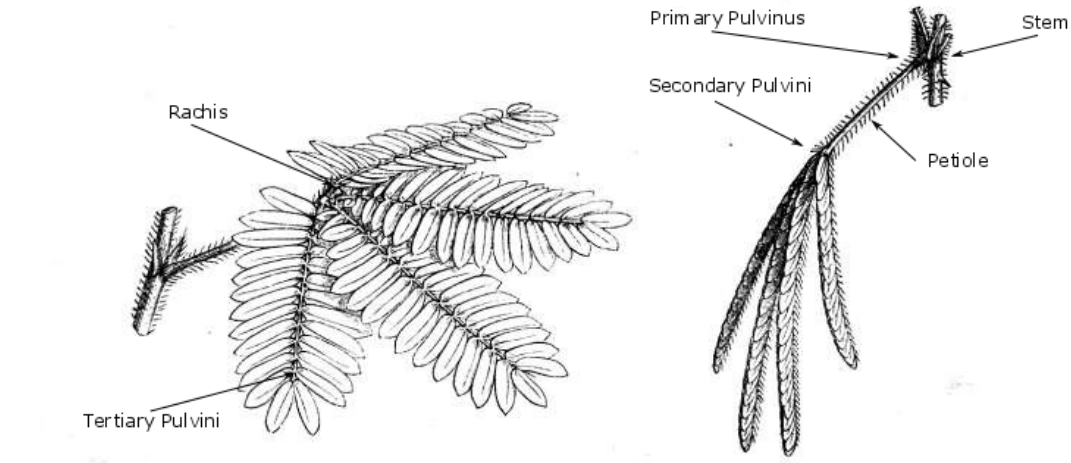


Figure 4: Mimosa Pudica stem and leaf description [37]

One of the favourable properties of the *mimosa pudica* from a experimental viewpoint is the *mimosa pudicas* rapid response to stimulus. These responses originate from the plants different pulvinus, which can be described as motor organs that causes movement by increasing or decreasing the tugor pressure of the pulvinus cells [38]. The *mimosa pudica* has three different pulvinis, which includes a primary pulvinus at the joint of the petiole and stem, a secondary pulvinus located in the joints between the petiole and rachis, and tertiary pulvinus found at the base of each leaflet. The different pulvinis has also been proven to respond differently [11] , where an example is that the primary and tertiary pulvinus of the *mimosa pudica* responds to touch or vibration - while the secondary pulvinus does not.

2.2 Movement In Plants

Motion in plants is a challenging subject as the movements can be classified as repetitious and rhythmic, or provoked - which again can be separated into *nastic* and *tropic* movements [24]. The difference between the latter two is that *tropic* movement depends on the direction of the stimulus, where a familiar example could be a flowers movement towards the direction of sunlight. *Nastic* movements is opposite and does not depend on the direction of the stimulus, which e.g. can be observed as the leaf contract as a result of change in the illumination with the onset of darkness. The motion that is observed can also be said to be either *non-reversible*, where the movement for example is a result of plant growth, or *reversible*, which is a movement that originate from the plants pulvinus.

2.3 Experimental Environment

The *mimosa pudica* plants that are used in the experiment is first grown for 3 to 4 weeks until their height are approximately 3 - 5 centimetres. The plant that is chosen to be used in the experiment is then moved to the experimental environment, which is inside a closed container made up by white colored chip boards. When the plant has been placed inside the container, a blueish A4 sized paper is attached around the *mimosa pudicas* lower stem, only revealing the shoot system of the plant. This is shown in figure 5 and has the intention of making an uniform background that simplifies the image processing.

Figure 5: The mimosa pudica inside the chipboard box

The experiment that is performed in the thesis has the intention of examine the response of the *mimosa pudica* during changes in its illuminative conditions. These responses is provoked by using two different illumination regimes, named *light* regime and *dark* regime, as stimulus. The illumination sources used in the *light* regime consist of 4 strips of 9 light emitting diodes, while the *dark* regime uses 1 strip of 9 light emitting diodes with five of the diodes covered. The light emitting diodes strip produce 0.7 watts with a color temperature of 4000 K per diode and is attached to the container wall.

The experiments with each plant will usually be running for several days where the stimulus is changed with a periodic pattern or kept constant in a certain regime. The plants movement will in this period be recorded as a 3 to 5 minute interval time-lapse series by a Logitech C615 full HD 1080p web-camera that is installed vertically above the *mimosa pudica*. The camera is installed on two metal rails inside the container and is shown together with the light emitting diode strips in figure 6.

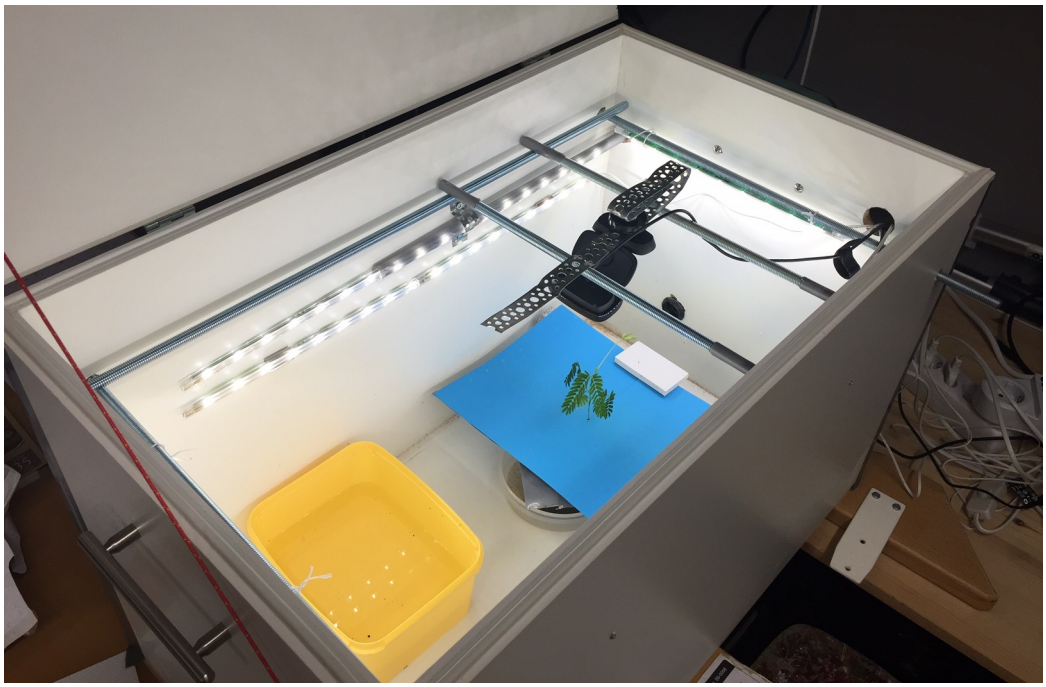


Figure 6: The *Mimosa Pudica* experimental environment

2.4 Running Experiments

The experiment procedure used during the experiments is usually executed by first exposing the *mimosa pudica* for a periodic interval of the *light* and *dark* regimes. The periodic interval is run for a chosen duration of time, before there is either a

- change in the *light* or *dark* illumination regime duration, changing or shifting the periodic interval.
- change into a constant *light* or *dark* illumination regime.

The intention of doing changes in the stimulus of the *mimosa pudica* is to observe the different provoked responses. These responses is captured and continuously saved to a computer as long as the experiment last, which could be everything from days to weeks to months. It has been chosen to only run one experiment at the time, where it is usually chosen to start a new experiment when the plant either wither or grow too large for the experimental setup.

Chapter 3

Image Processing

The purpose of the image processing section in the thesis is to find a method which can identify and count the image pixels that are a member of the *mimosa pudica* plant. The first part examines and compare different color models by looking at the individual components. The next step is to seek a suitable segmentation method to segment the image based on properties from the chosen color model.

3.1 Color Models

The time-lapse series in the thesis consist of images taken every third or fifth minute over a period of several days, and includes challenging conditions like alternating illumination, object movements causing change in local shading and foreground object with varying color and shape. This emphasize the need to evaluate and investigate different color models to examine which representation that lay the best foundation to the task of segmenting the foreground object from the background. The color models that has been chosen to be examined is the well known RGB, the color specification based HSV and the color difference based CIELAB.

3.1.1 RGB

The RGB is a color model where the obtainable colors can be described as a weighted combination of the three primary colors red, green and blue. The color space, which is a geometric representation of the gamut, is usually represented as the unit cube shown in figure 7 - with the primary colors red, green and blue located at the coordinates $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$.

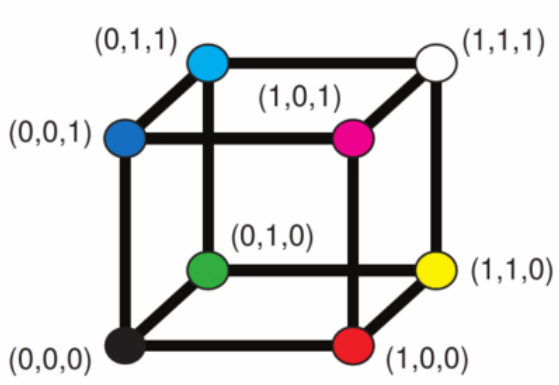


Figure 7: RGB cube [35].

The weights used to describe a colors content of the primaries will in the thesis be represented by a vector (I_R, I_G, I_B), where I is the weight or intensity of the primaries, and the subscripts are abbreviations for red, green and blue. Examples of its use is best seen in figure 7 where e.g equal values of (I_R, I_G, I_B) result in a line of gray scales running from black at $(0, 0, 0)$ to white at $(1, 1, 1)$, while the combination of two fully weighed primary colors $(1, 1, 0)$, $(0, 1, 1)$ and $(1, 0, 1)$ gives yellow, cyan and magenta.

3.1.2 HSV

The HSV color model is obtained by doing a non-linear transformation of the RGB color model and is often represented as a hexcone color space. The purpose of the HSV color model is to provide an easier way to describe color specification in computer graphics [16] by using the terms *hue*, *saturation* and *value*.

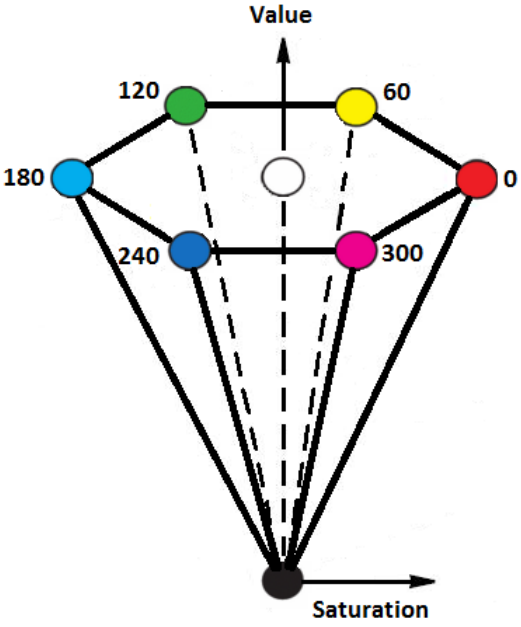


Figure 8: HSV hexcone.

The HSV color space is shown in figure 8 where the different colors can be specified by a vector (hue, sat, val). The *hue* component is represented as an angle $\in$ [0 360], relative to red. The angular value is describing how the color appears to be similar to the RGB primary colors, which are placed with an angular spacing of 120° in their natural order. The *value* component is represented by the line from black to white in the center of the hexcone and is said to be equivalent to the term *brightness* [10]. This can be compared with the diagonal with gray scales from black to white in the RGB cube and is usually represented as a value $\in$ [0 1]. The *saturation* component is often compared to the purity of the color, where a highly saturated area will seem chromatic in comparison with its associated *brightness* level. *Saturation* is usually represented as a value $\in$ [0 1], where an example of a fully saturated color could be the RGB primaries red, green and blue.

3.1.3 CIELAB

CIELAB is a uniform color model developed by the International Commission on Illumination that was designed to be used for specification of color differences. This means that the difference of two colors in the color space can be described by the euclidean distance between them [10].

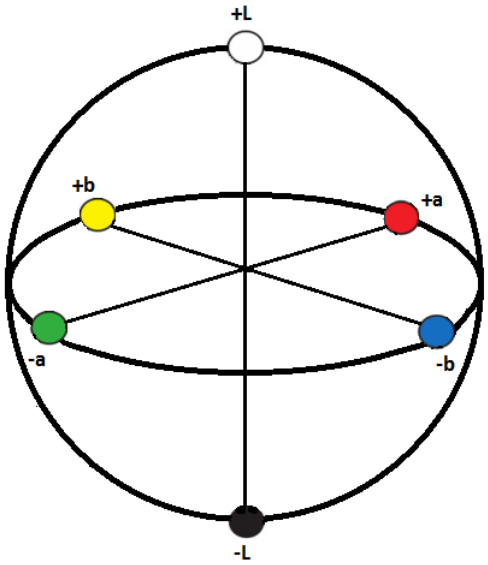


Figure 9: CIELAB color space

An example of a representation of the CIELAB color space is shown in figure 9, where the axis are described by the $L*$, $a*$ and $b*$ components. The $L*$ component is the perceived *lightness* and is represented as the line from black to diffuse white with a value $\in$ [0 100]. *Lightness* can be interpret as a relative *brightness*, and is best explained by using the white partly shaded wall in figure 10 as an example. The wall surface will in terms of *lightness* appear to be uniformly white because the term is judging the brightness on the wall relative to the brightness of similarly illuminated areas. The *brightness* of the wall will not be uniform because it is an absolute measure that in this case is affected by the shades at the corner and the shadows from the trees [21]. The two remaining components in the CIELAB color space is the $a*$ and $b*$ components which describes the colors green-red and blue-yellow content. The obtainable values of these components depends on the color model that the image is converter from [7], and is in the case of converting from RGB with the Matlab command `rgb2lab` found to be a* $\in$ [$-$86.17 80.11] and b*$\in$ [$-$107.85 94.48]

Figure 10: Image lightness and brightness example

3.2 Color Model Comparison

The purpose of this section is to compare and evaluate the three color models representation of the two images in figure 11, taken from the *dark* and *light* regime.

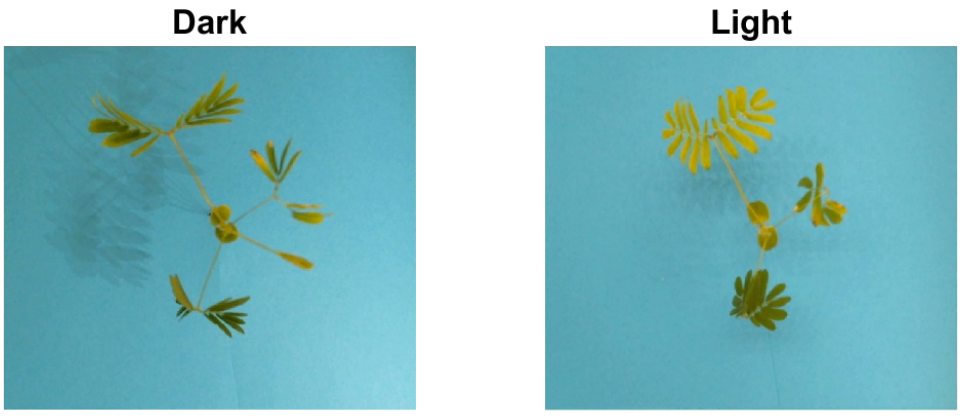


Figure 11: The *dark* regime (left) and the *light* intensity regime (right) used in the color space comparison

It has been chosen to compare the three color models in terms of how their individual components represents useful information about the images, where the important factor will be the individual components ability to provide information that makes it possible to distinguish the background and the foreground. The tools that is used in the comparison is the individual components histograms and the *Color Thresholder* application in the Matlab *Image Processing and Computer Vision* toolbox.

A detailed comparison of the three color models is found in the Color Models Comparison Experiment in Chapter 7.1, with the conclusion that best individual components from each color model is the *blue* component from RGB, the *hue* componet from HSV and the b^* component from CIELAB. The blue RGB component is not chosen since the difference in the *light* and *dark* histogram distribution indicates that it is the component who is most affected by changes in the illumination. Both the *hue* from HSV and b^* from CIELAB seem to provide much of the same result when being evaluated, and there has not been found any other reason for choosing the one over the other except the more intuitive representation of the *hue* component. This is also the reason why the HSV color model will be used in the further image processing of the thesis.

3.3 Image Segmentation

Image segmentation is in many occasions one of the first operations that is performed on the image and can be the foundation for the remaining image processing. The aim of the segmentation is to identify and distinguish regions in an image based on e.g. chosen properties, where an example is to separate a foreground object from a background based on color. One of the challenges with image segmentation is the lack of a universal method [25], and the method must therefore be chosen based on the problem to be solved. It will in this thesis be attempted to implement a method from one of the most basic types of image segmentation techniques, which is the threshold or intensity-based method. This method usually work by doing a comparison of each pixel value (intensity) against one or several threshold values.

3.3.1 Intensity-based Segmentation

The image segmentation used in the thesis is an intensity-based segmentation method that locate a threshold value from the *hue* component histogram. The hue component of the image pixels $\in [0° \ 360°]$ is assigned into 100 histogram bins, meaning that the first bin will contain the number of hue components $\in [0° \ 3.6°]$, the second bin will contain the number of hue components $\in (3.6° \ 7.2°]$ and so on. This arrangement, together with the prior knowledge of an almost uniform blueish background, implies that the background will be seen as a narrow symmetric uni-modal distribution in the histogram. This suggests that a suitable intensity threshold can be found by implementing a triangle based method [40] as shown in figure 12.

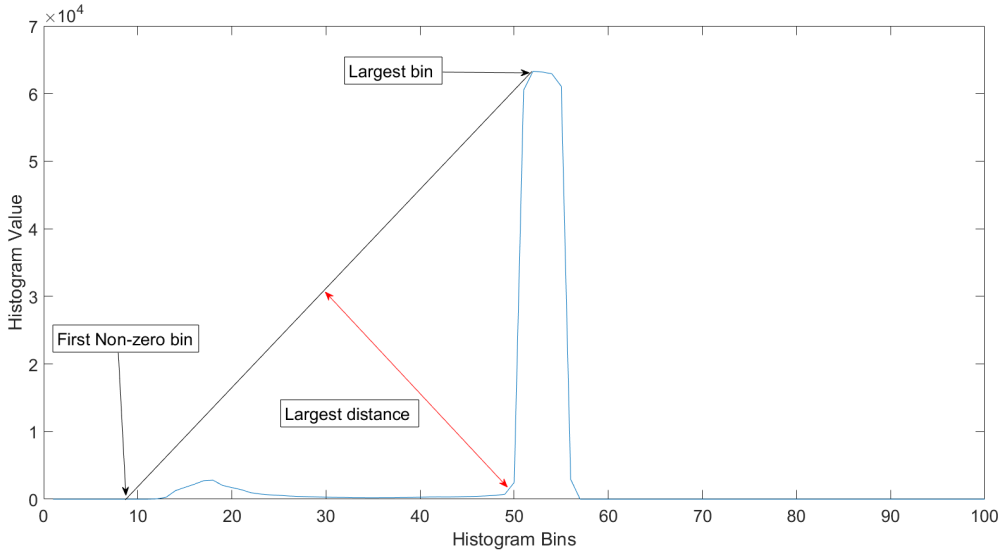


Figure 12: Intensity threshold triangle method

The method starts by locating the (i) largest- and (ii) first non-zero histogram bin by iterating through the histogram data. Their location is then used to calculated parameters for a straight line connecting the two bins, as shown by the black in line figure 12. The next step in the algorithm is to calculate the distance between the blue line and each of the bins that lay in the range between the largest- and first non-zero histogram bin, as shown with the double arrowed red line. The most suitable pixel intensity threshold is then assumed to be represented by the histogram bin with the largest distance to the line.

Chapter 4

System Identification

One of the observed activities seen in the time-lapse series of the *mimosa pudica* is the folding and unfolding of the plant leafs. These activities were made quantifiable by the image segmentation presented in the previous chapter, where the folding and unfolding of leafs is observed to result in a decrease and increase in foreground pixels, respectively. The increase and decrease of pixel count, as seen in figure 13, is in this section considered a measure that indicates the plants response to changes in the illumination regime.

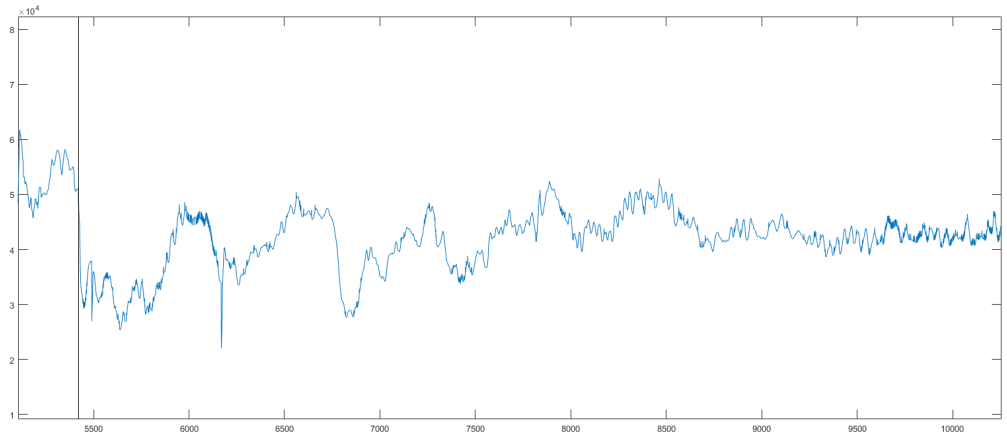


Figure 13: Periodic to constant light regime oscillatory response

The available datasets reveals that the changes from dark to light regime has a tendency to increase the amount of foreground pixels, while a change from light to dark regime tend to cause a reduction. It is also observed that a change from a periodic light regime pattern into a constant illumination seem to cause oscillatory responses. These observations raises the question whether the plants response to the change in its illuminative condition is

purely biological, or if it is possible to relate the oscillatory response to a control theoretic mindset.

The purpose of this section is to first present the parametric *black box* method for modelling the system, where the theoretic foundation for three different versions of the general linear polynomial is presented. This is followed by a section that describes how the model estimations are being evaluated. The experimental results of the model estimation is found in chapter 7.3.

4.1 General Linear Polynomial Model

The system identification that is performed in this chapter is by a parametric method called *black box* modelling, where one assumes that the system that is being modelled is unknown [23]. Mutually for all *black box* models used in this thesis is that they have their basis from the general linear polynomials in equation 4.1, which can be divided into set of deterministic and stochastic rational transfer functions [27] .

$$A(q)y(k) = \underbrace{\frac{B(q)}{F(q)}}_{\text{Deterministic}} u(k) + \underbrace{\frac{C(q)}{D(q)}}_{\text{Stochastic}} e(k) \tag{4.1}$$

The deterministic and stochastic transfer functions is in this case used to describe the dynamics and disturbance of the system of interest [4], and might be best visualized as shown in figure 14, where $u(k)$ is the system input, $y(k)$ is the system output and $e(k)$ is assumed to be white noise.

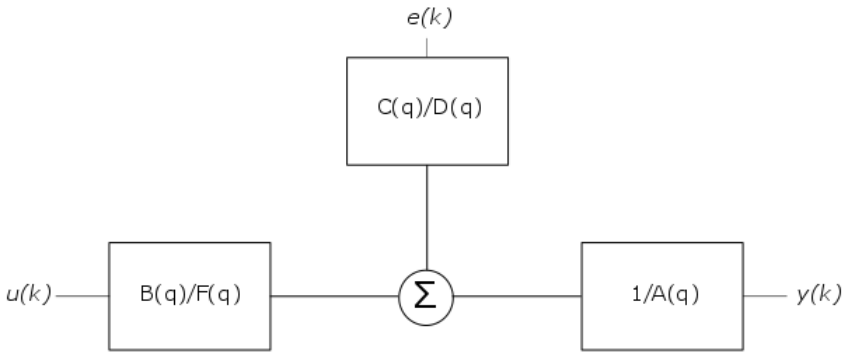


Figure 14: General Linear Polynomial Model

4.1.1 ARX

The *AutoRegressive with eXogenous input* (ARX) model is a simplification of the general linear polynomial model in equation 4.1, and is obtained by setting $C(q) = F(q) = D(q) = 1$. This yields the model shown in equation 4.2, where the *autoregressive* component is given by $A(q)y(k)$, the *exogenous* component is given by $B(q)u(k)$ and a white noise term is given by $e(k)$.

$$y(k) = \underbrace{\frac{B(q)}{A(q)}}_{\text{deterministic}} u(k) + \underbrace{\frac{1}{A(q)}}_{\text{stochastic}} e(k) \tag{4.2}$$

The result of the simplifications in the ARX model is that the deterministic and stochastic transfer functions share the same denominator, implying that system noise and dynamics cannot be modelled independently [22]. This makes the ARX, and the soon to be introduced ARMAX model, most suitable the cases where we experience load disturbances [23].

4.1.2 ARMAX

The *AutoRegressive Moving Average with eXogenous input* (ARMAX) model is obtained by setting $F(q) = D(q) = 1$. The ARMAX model can be viewed as an extension of the ARX model, where the stochastic transfer function in this case includes a moving average model component $C(q)e(k)$. This changes equation 4.1 into equation 4.3, where the deterministic and stochastic transfer functions still have identical denominators.

$$y(k) = \underbrace{\frac{B(q)}{A(q)}}_{\text{deterministic}} u(k) + \underbrace{\frac{C(q)}{A(q)}}_{\text{stochastic}} e(k) \tag{4.3}$$

The benefit of using an ARMAX compared with ARX model is the increased flexibility in the disturbance transfer function, introduced by the moving average component. This makes ARMAX preferrable in the cases where the disturbance cannot be modelled by a single term $e(k)$, while the ARX can in cases of high signal to noise ratio [18] be both suitable and beneficial for a fast estimation of simpler systems.

4.1.3 Box Jenkins

The *Box Jenkins* (BJ) model is obtained by setting $A(q) = 1$, giving the model shown in equation 4.4. It is seen that the deterministic and stochastic set of transfer functions now have different denominators, making it possible to model the systems dynamics and disturbance independently.

$$y(k) = \underbrace{\frac{B(q)}{F(q)}}_{\text{Deterministic}} u(k) + \underbrace{\frac{C(q)}{D(q)}}_{\text{Stochastic}} e(k) \tag{4.4}$$

The different denominator polynomials in the *Box Jenkins* model allows an independent modelling of the systems dynamic and disturbance, thus, making the *Box Jenkin* model favourable in the cases where the disturbance for example is found as measurement noise [23].

4.2 Model Evaluation

The Mathworks Matlab workspace offers a broad range of different methods for the evaluation of a models fit to some data-set, and it has in this thesis been chosen to use the *Akaike's Information Criteria* (AIC) in the initial search of model order combinations. The AIC is a measure that can be used to compare the quality of a set of *black box* models to a particular dataset, where a lower AIC value implies a better model [5].

The AIC formula is shown in equation 4.5, where k is the independent adjustable parameters in the model and L is the models maximum likelihood function [1]. This results in a criterion that favours both models with fewer parameters, in terms of $2k$, as well as the models goodness of fit, in terms of a higher log likelihood $2log(L)$.

$$AIC = 2k - 2log(L) \tag{4.5}$$

The second evaluation method is the normalized root-mean-square error (NRMSE), which is used to compare fit of the different *black box* models against the validation data. The NRMSE is given by equation 4.6, where y_{test} is the test or validation data samples, y_{model} is the estimated output from the *black box* model and μ_{data} is the mean value of the test or validation data samples.

$$fit = 100\left(1 - \frac{\parallel y_{data} - y_{model} \parallel}{\parallel y_{data} - \mu_{data} \parallel}\right) \tag{4.6}$$

Chapter 5

Motion Estimation

The idea of estimating the movement of the leafs in the time-lapse series has its root in a recent published article [12], that uses two frame motion estimation to find the vertical movements of plants in conjunction with circadian period estimation from a 20 minute interval time-lapse series. The experiment performed in the thesis has a slightly different approach, as it is performed an estimation of the *mimosa pudicas* horizontal movement, with a 3 - 5 minute interval time-lapse series.

The motion estimation in the thesis has been chosen to be executed by one of the already implemented dense optical flow algorithms in the Mathworks Matlab 2015b *Computer Vision System Toolbox*. Dense optical flow is described as the computation of an [36] "*independent estimate of motion at each pixel*" and is a measure of pixel movement between two succeeding images [3]. The first of the two available dense optical flow algorithms in Matlab is the Horn-Schunck method, which finds the optical flow under the assumption of the *brightness* and *smoothness* constraint. These constraints imply an assumption of pixel values (intensity) remaining the same after the pixel is being displaced a small distance in the succeeding images, as well as assuming that the velocity of the displacements varies smoothly [15]. The second dense optical flow algorithm is the orientation tensor based method of Farneback, which approximates the image pixel neighbourhoods of two succeeding images as polynomials. The approximated polynomials is then used to find the displacements, under the constraint of slowly varying displacement fields [9].

The algorithm that have been chosen to be implemented for the estimation of motion is the method of Farneback. The method has the benefit of being able to estimate motion at coarser scale due to its pyramid layer

implementation, and thus making it more suitable to estimate the larger displacements associated with e.g. leaf contraction. It also has the benefit of being tensor based algorithm, which is ideal for Matlab implementation in terms of running speeds.

5.1 Method of Farneback

The Gunnar Farneback method is a two-frame motion estimation algorithm that uses polynomial expansion, where a neighbourhood of each image pixel is approximated with the bivariate quadratic polynomial in equation 5.1 , resulting in a local signal model [9].

$$\underline{\boldsymbol{f}}(\underline{\boldsymbol{x}}) \backsim \underline{\boldsymbol{x}}^T \underset{\sim}{\boldsymbol{A}} \underline{\boldsymbol{x}} + \underline{\boldsymbol{b}}^T \underline{\boldsymbol{x}} + c \tag{5.1}$$

The vector $\underline{\boldsymbol{x}}$ is in this case a 1×2 vector containing the running variables x and y, $\underset{\sim}{\boldsymbol{A}}$ is a 2×2 symmetric matrix of unknowns, $\underline{\boldsymbol{b}}$ is a 2×1 vector of unknowns and c is a unknown scalar. Writing it in terms of tensors gives equation 5.2, where the unknown coefficients is defined by the symbol r_i.

$$\boldsymbol{f}(\underline{\boldsymbol{x}}) \backsim \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} r_4 & \frac{r_6}{2} \\ \frac{r_6}{2} & r_5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} r_2 & r_3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + r_1 \tag{5.2}$$

5.1.1 Polynomial Expansion

The quadratic polynomial in equation 5.1 can alternatively be expressed on the form shown in equation 5.3 by multiplying the tensors in equation 5.2

$$\underline{\boldsymbol{x}}^T \underset{\sim}{\boldsymbol{A}} \underline{\boldsymbol{x}} + \underline{\boldsymbol{b}}^T \underline{\boldsymbol{x}} + c = r_1 + r_2 x + r_3 y + r_4 x^2 + r_5 y^2 + r_6 xy \tag{5.3}$$

The idea is that the quadratic polynomial displayed in equation 5.3 is used to approximate a local signal model of *each* image pixels neighbourhood, where the neighbourhood is represented by the $m \times m$ matrix $\underset{\sim}{\boldsymbol{F}}$. The operation starts by arranging the $m \times m$ matrix $\underset{\sim}{\boldsymbol{F}}$ into a $n \times 1$ column vector $\underline{\boldsymbol{f}}$, by stacking the matrix columns in $\underset{\sim}{\boldsymbol{F}}$ from left to right. This makes it possible to see $\underline{\boldsymbol{f}}$ as an element (vector) in a n dimensional vector space C^n. The next step is to define a new subspace of C^n, spanned by a set of 6 linear independent $n \times 1$ basis functions $\underline{\boldsymbol{b}}_i$, kept inn a $n \times 6$ matrix $\underset{\sim}{\boldsymbol{B}}$. The size of $\underline{\boldsymbol{b}}_i$, which in this case is $n \times 1$, indicates that the basis functions are elements in C^n, while their linear independence imply that a set of 6 basis vectors will span the subspace C^6. It is then sought to find the vector in the subspace C^6 that is closes to the neighbourhood vector $\underline{\boldsymbol{f}}$ in the vector space C^n, which

is the projection of the neighbourhood vector $\underline{\boldsymbol{f}}$ onto the column space of $\underset{\sim}{\boldsymbol{B}}$. This is best visualized with coordinate systems of lower dimensions as seen in figure 15, where the closes point on the vertical line with regards to the point P_1 in the two dimensional space, is the projection of P_1 onto the line, indicated by P_2.

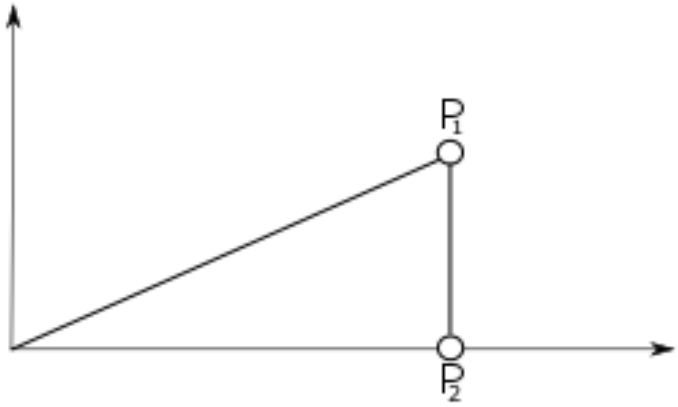


Figure 15: Projection of P_1 onto the vertical line, indicated by P_2

The search for a vector in the subspace C^6 that is closest to the vector $\underline{\boldsymbol{f}}$ in the vector space C^n can alternatively be seen as the least square problem shown in equation 5.4, as you are finding the coefficients r_i to the vector $\mathbf{Br}$ that are minimizing the distance to the vector $\underline{\boldsymbol{f}}$

$$argmin \parallel \underset{\sim}{\boldsymbol{B}}\underline{\boldsymbol{r}} - \underline{\boldsymbol{f}} \parallel \tag{5.4}$$

One of the challenges in a pixel neighbourhood is uncertainties, which can be caused by e.g. missing or corrupt data [8] . The Farneback method is solving this by using a $n \times 1$ vector $\underline{\boldsymbol{c}}$ that specifies the certainty of each image pixel in the neighbourhood vector $\underline{\boldsymbol{f}}$. The result is that missing data from cases like border effects is weighted down with a low or zero valued certainty. In addition, the method is also using an $n \times 1$ applicability vector $\underline{\boldsymbol{a}}$ that indicates the significance of the pixels in the neighbourhood $\underline{\boldsymbol{f}}$. This can be used to e.g. represent a higher importance of certain pixels. With weighting taken into account, the coefficients in the polynomial expansion is solved by finding a column vector $\underline{\boldsymbol{r}}$ that minimizes $\parallel \underset{\sim}{\boldsymbol{B}}\underline{\boldsymbol{r}} - \underline{\boldsymbol{f}} \parallel_{\underset{\sim}{\boldsymbol{W}}}$, where $\underline{\boldsymbol{r}}$ is the weighted least squares solution.

$$argmin \parallel \underset{\sim}{\boldsymbol{B}}\underline{\boldsymbol{r}} - \underline{\boldsymbol{f}} \parallel_{\underset{\sim}{\boldsymbol{W}}} \tag{5.5}$$

The weights are formed by changing the applicability and certainty vectors to diagonal matrices and multiplying them together. Solving equation 5.5 gives the weighted least square solution in equation 5.6

$$\underline{\boldsymbol{r}} = (\underset{\sim}{\boldsymbol{W}}\underset{\sim}{\boldsymbol{B}})^{\dagger}\underset{\sim}{\boldsymbol{W}}\underline{\boldsymbol{f}} \tag{5.6}$$

5.1.2 Displacement Estimation

The polynomial approximation that was performed in the previous subsection resulted in the weighted least square solution $\underline{\boldsymbol{r}}$, which is used in the pixels local model shown in equation 5.7. These local pixel models are now used in the the derivation of the algorithms displacement estimation, as described in Gunnar Farnebacks paper "*Two-Frame Motion Estimation Based on Polynomial Expansion*" [9].

$$\underline{\boldsymbol{f}}(\underline{\boldsymbol{x}}) \backsim \underline{\boldsymbol{x}}^T\underset{\sim}{\boldsymbol{A}}\underline{\boldsymbol{x}} + \underline{\boldsymbol{b}}^T\underline{\boldsymbol{x}} + c \tag{5.7}$$

The example that is used in the derivation will start by assuming that we have a local signal model of a pixel neighbourhood $\underline{\boldsymbol{f}}_{\boldsymbol{1}}(\underline{\boldsymbol{x}})$, that is described by the polynomial coefficients $\underset{\sim}{\boldsymbol{A}}_1$, $\boldsymbol{b}_1$ and c_1. It is then chosen to make a new local signal model $\underline{\boldsymbol{f}}_{\boldsymbol{2}}(\underline{\boldsymbol{x}})$, that is identical to $\underline{\boldsymbol{f}}_{\boldsymbol{1}}(\underline{\boldsymbol{x}})$ but displaced by $\underline{\boldsymbol{d}}$. The displaced local signal model $\underline{\boldsymbol{f}}_{\boldsymbol{2}}(\underline{\boldsymbol{x}})$ can then be formulated as shown in equation 5.8

$$\underline{\boldsymbol{f}}_{\boldsymbol{2}}(\underline{\boldsymbol{x}}) = \underline{\boldsymbol{f}}(\underline{\boldsymbol{x}} - \underline{\boldsymbol{d}}) \backsim (\underline{\boldsymbol{x}} - \underline{\boldsymbol{d}})^T\underset{\sim}{\boldsymbol{A}}_1(\underline{\boldsymbol{x}} - \underline{\boldsymbol{d}}) + \underline{\boldsymbol{b}}_1^T(\underline{\boldsymbol{x}} - \underline{\boldsymbol{d}}) + c_1 \tag{5.8}$$

Multiplying the tensors in equation 5.8, results in equation 5.9

$$\underline{\boldsymbol{f}}_{\boldsymbol{2}}(\underline{\boldsymbol{x}}) \backsim \underline{\boldsymbol{x}}^T\underset{\sim}{\boldsymbol{A}}_1\underline{\boldsymbol{x}} - \underline{\boldsymbol{x}}^T\underset{\sim}{\boldsymbol{A}}_1\underline{\boldsymbol{d}} - \underline{\boldsymbol{d}}^T\underset{\sim}{\boldsymbol{A}}_1\underline{\boldsymbol{x}} + \underline{\boldsymbol{d}}^T\underset{\sim}{\boldsymbol{A}}_1\underline{\boldsymbol{d}} + \underline{\boldsymbol{b}}_1^T\underline{\boldsymbol{x}} - \underline{\boldsymbol{b}}_1^T\underline{\boldsymbol{d}} + c_1 \tag{5.9}$$

The A matrix is as stated earlier a symmetric matrix, which gives it the property $\underset{\sim}{\boldsymbol{A}} = \underset{\sim}{\boldsymbol{A}}^T$ [2]. This makes it possible to rewrite equation 5.9 as equation 5.10

$$\underline{\boldsymbol{f}}_{\boldsymbol{2}}(\underline{\boldsymbol{x}}) \backsim \underline{\boldsymbol{x}}^T\underset{\sim}{\boldsymbol{A}}_1\underline{\boldsymbol{x}} + (\underline{\boldsymbol{b}}_1 - 2\underset{\sim}{\boldsymbol{A}}_1\underline{\boldsymbol{d}})^T\underline{\boldsymbol{x}} + \underline{\boldsymbol{d}}^T\underset{\sim}{\boldsymbol{A}}_1\underline{\boldsymbol{d}} - \underline{\boldsymbol{b}}_1^T\underline{\boldsymbol{d}} + c_1 \tag{5.10}$$

The new local neighbourhood model $\underline{\boldsymbol{f}}_{\boldsymbol{2}}(\boldsymbol{x})$ can then be changed back to the local signal model form as shown in equation 5.7 by defining the new tensors in equation 5.11, 5.12 and 5.13

$$\underset{\sim}{\boldsymbol{A}}_2 = \underset{\sim}{\boldsymbol{A}}_1 \tag{5.11}$$

$$\underline{\boldsymbol{b}}_2 = \underline{\boldsymbol{b}}_1 - 2\underset{\sim}{\boldsymbol{A}}_1\underline{\boldsymbol{d}} \tag{5.12}$$

$$c_2 = \underline{\boldsymbol{d}}^T \underset{\sim}{\boldsymbol{A}}_1 \underline{\boldsymbol{d}} - \underline{\boldsymbol{b}}_1^T \underline{\boldsymbol{d}} + c_1 \tag{5.13}$$

Which result in equation 5.14

$$\underline{\boldsymbol{f_2}}(\underline{\boldsymbol{x}}) \backsim \underline{\boldsymbol{x}}^T \underset{\sim}{\boldsymbol{A}}_2 \underline{\boldsymbol{x}} + \underline{\boldsymbol{b}}_2^T \underline{\boldsymbol{x}} + c_2 \tag{5.14}$$

The result is that equation 5.12 can be used to find the displacement $\underline{\boldsymbol{d}}$ under the assumption that $\underset{\sim}{\boldsymbol{A}}_1$ is invertible, as shown in equation 5.15.

$$\underline{\boldsymbol{d}} = -\frac{1}{2} \underset{\sim}{\boldsymbol{A}}_1^{-1} (\underline{\boldsymbol{b}}_2 - \underline{\boldsymbol{b}}_1) \tag{5.15}$$

The example derived from a single pixel neighbourhood in both images, resulting in equation 5.15, can now be related to each of the individual pixel neighbourhoods in both of the images. The generalization is performed by using the pixels spatial coordinates as indicated by the indexing (x, y). It is also stated in derivation in the article [9] that the ideal case of equation 5.11 is changed to the more realistic approximation in equation 5.16, while the $-\frac{1}{2}(b_2 - b_1)$ term in equation 5.15 is changed to $\triangle b$.

$$\underset{\sim}{\boldsymbol{A}}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\underset{\sim}{\boldsymbol{A}}_1(x, y) + \underset{\sim}{\boldsymbol{A}}_2(x, y)}{2} \tag{5.16}$$

This changes equation 5.15 into equation 5.17

$$\underset{\sim}{\boldsymbol{A}}(\boldsymbol{x}, \boldsymbol{y}) \boldsymbol{d}(x, y) = \triangle \boldsymbol{b} \tag{5.17}$$

It is then attempted to find the displacement $\boldsymbol{d}(x, y)$ that best satisfies equation 5.17. This is equivalent with minimizing equation 5.18, where I is the neighbourhood around the pixel with coordinate (x, y) and $\omega(\triangle x, \triangle y)$ is a Gaussian weight function.

$$\sum_{\triangle x, \triangle y \ \in I} \omega(\triangle x, \triangle y) \parallel \underset{\sim}{\boldsymbol{A}}(x + \triangle x, y + \triangle y) \boldsymbol{d}(x, y) - \triangle \boldsymbol{b}(x + \triangle x, y + \triangle y) \parallel^2 \tag{5.18}$$

Where solving for $\boldsymbol{d}(x, y)$ in equation 5.18 gives equation 5.19

$$\boldsymbol{d}(x, y) = \sum (\omega \underset{\sim}{\boldsymbol{A}}^T \underset{\sim}{\boldsymbol{A}})^{-1} \sum \omega \underset{\sim}{\boldsymbol{A}}^T \triangle \boldsymbol{b} \tag{5.19}$$

Chapter 6

Implementation

The purpose of this chapter is to give a brief presentation of the implemented Matlab routines that are used in the thesis. A simplified overview of the routine work flow is shown in figure 16, where rectangles describe the different parts of the program and arrows indicates the flow of information.

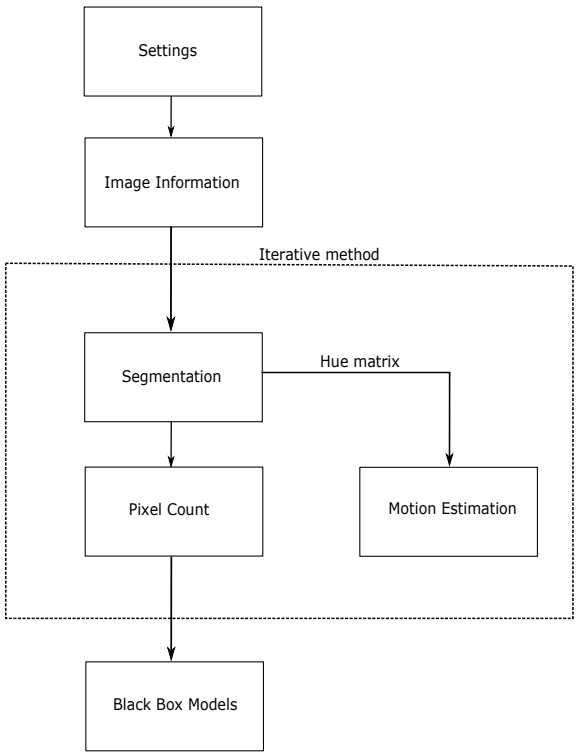


Figure 16: Flow chart of Matlab routines

6.1 Settings and Image Information

The settings and image information part of the routine consist of the initial directory settings and the functions `Mapper`, `BildeInformasjon` and `LightDarkVector`. The initial directory settings is the first part of the program where the image directory main path and the time-lapse series of choosing is selected. These are variables that are fed into the function `Mapper`, which returns a `struct` element that contain all the directory paths used in the thesis.

```
% Folder with the image data
mappe = 'C:/Users/Vegard/Documents/MATLAB/Bilder';

% Image folder and camera direction
bildeDato = '/date_20160130';
kameraRetning = {'/camera1_top','/camera2_buttom'};

% Generating structure with folder and image names
mappeStruktur = Mapper(mappe, bildeDato, kameraRetning);
```

The next function, `BildeInformasjon`, uses the `mappeStruktur` element to gather information about the time-lapse period, including time of start, duration etc. This information, together with illumination regime information from the Matlab file `LightDark.m`, is used as input to the function `LightDarkVector`, which have the role of creating a vector that provides information about the illumination regime at each sample in the time-lapse series.

```
% Loading Dark Light regime file
run(sprintf('%s%s/LightDarkRegime/%s', ...
    mappe,bildeDato,'LightDark.m'));

% Finding timelapse periode and first image offset
[timeLapsePeriode OffsetForsteBilde] = ...
    BildeInformasjon(mappeStruktur);

% Making a Day-Light-Constant vector.
DLCvect = LightDarkVector(DarkLightConstant,...
timeLapsePeriode,OffsetForsteBilde);
```

The returned vector named `DLCvect` is used in the plotting function `Plotting`, described in the last section of the chapter.

6.2 Segmentation

Segmentation of the image is performed by the function `FargeTilSortHvitt`, which uses the image from the time-lapse series as input and returns the segmented image and hue component matrix. The image to be segmented is found by picking the images in the time-lapse series folder in consecutive order. It has also been chosen to crop the images in some cases to reduce the computational complexity in the cases where the background does not contain useful information.

```
1      % Reading path from the mappeStructure file
2      bilde = ...
           imread(sprintf('%s/%s',mappeStruktur.mappeOriginal,...
3              mappeStruktur.infoOriginal(i).name));
4
5      % Cropping image
6      bilde = bilde(1:end-50,1:end-175,:);
7
8      % Segmenting by FargeTilSortHvitt
9      [binBilde picture] = FargeTilSortHvitt(bilde);
```

The `FargeTilSortHvitt` works by first converting the image to the HSV color mode by the Mathworks Matlab `rgb2hsv` function, before the hue component matrix is extracted and the hue pixel values are arranged into a histogram with 100 bins. It is then chosen to smooth the histogram by a moving average filer before it is used as input to the triangle based intensity threshold function `triangle_th`, developed and shared by B. Panneton [30] at the Mathworks file exchange.

```
1      % Using triangle method to find threshold
2      triThreshold = triangle_th(imageHistSmooth,100);
3
4      % Making a segmentation mask
5      hueSegmented = imcomplement(im2bw(hue,triThreshold));
```

The `triangle_th` function returns an intensity threshold based on the method described in section 3.3.1, which is used as the threshold in the Matlab function `im2bw`. The compliment of the `im2bw` function is a binary image where the foreground consist of binary ones and the background consist of binary zeros.

6.3 Plotting

The result of the segmentation is a matrix with either binary ones or zeros, where the total amount of binary ones is found by using the matlab `sum` function. The pixel count and the `DLCvect` is then used as input to the function `Plotting`, which have the purpose of plotting the pixel count of each time-lapse series image.

```
% Plotting pixel count
Plotting(antallHvitePiksler,DLCvect)
```

The `Plotting` functions works by first iterating through the `DLCvect`, where the length and shifts of illumination regimes in stored in the vectors `DLCLength` and `DLCChangeIndex`. This makes it possible to indicate the duration of the *light*, *dark* and *constant* illumination regimes by a yellow, black and red colored bars, as well as emphasize their start and stop by a black vertical line. The necessary information to create the indication bars are found by locating the center of the illumination period, converting the period duration from minutes to hours and finding the bars color.

```
% X-Coordinate for center of indication bar
indBarCenterX = DLCChangeIndex(i)+(DLCLength(i)/2);
% Length of periode in hours
indBarTime = int2str((DLCLength(i)*3)/60);
% Indication bar color
indBarColor = DLCcolorMatrix{DLCvect(DLCChangeIndex(i))};
```

Each of the indication bar is made by the Matlab function `rectangle`, where the `DLCChangeIndex`, `yaxisOffset` and `yaxislimits` variables is used to align the bars to the correct x- and y axis coordinates. The `DLCLength` and `indBarHeight` contains information about the bars length and height, while the `indBarColor` specifies its color.

```
% Plotting Indication bar
rectangle('Position',[DLCChangeIndex(i),yaxislimits(1)-...
yaxisOffset, DLCLength(i),indBarHeight],...
'FaceColor',indBarColor)
```

Much of the same variables is used to plot the rectangular windows indicating

the illumination regimes start and stop, as well as the text describing the regime duration. It was also chosen to add an if-statement changing the text color to white in the cases where the indication bar is black.

```
    % Plotting rectangular Window
    rectangle('Position',[DLCChangeIndex(i),yaxislimits(1),...
        DLCLength(i),windowHeight]);

    % Adding text to indication bar
    if isequal(indBarColor,[0 0 0])
        text(indBarCenterX,yaxislimits(1)-yaxisOffset+1000,...
            indBarTime,'HorizontalAlignment','center',...
            'color','white')
    else
        text(indBarCenterX,yaxislimits(1)-yaxisOffset+1000,...
            indBarTime,'HorizontalAlignment','center');
    end
```

One of the goals of the experiments was to look at the response after a change from periodic to constant illumination regimes, which meant that the change in periodicity had to be highlighted. This was solved by shifting the indication bars down one level at each periodicity change by an offset equal the height of the bar. It was also chosen to plot one of the last *light* and *dark* regime with a fainter color. The result of the `Plotting` function is shown in figure 17.

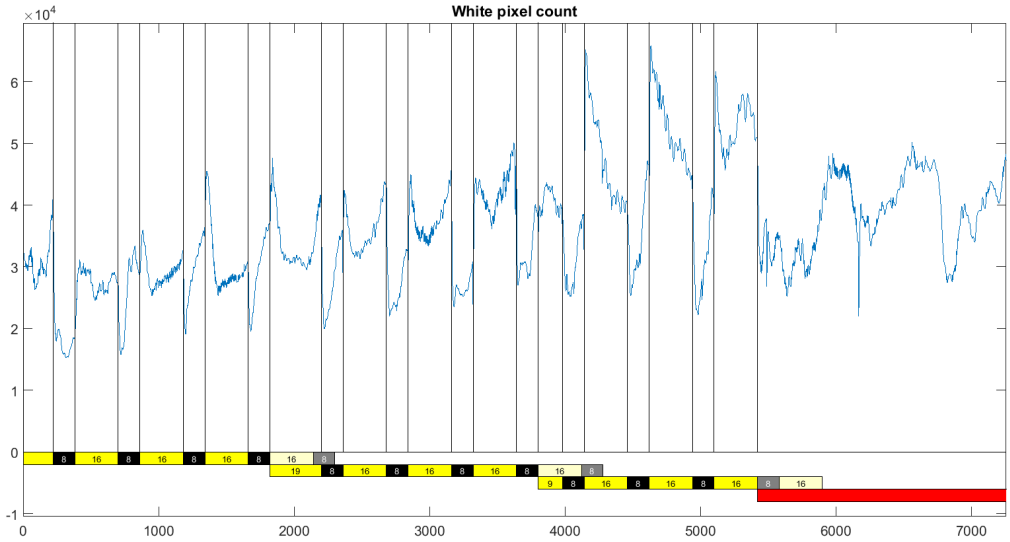


Figure 17: Plot from `Plotting` function

6.4 Motion Estimation

The motion estimation code used in the thesis starts by creating an optical flow object called `opticFlow`.

```
% Creating an optical flow object
opticFlow = opticalFlowFarneback;
```

This object, together with the hue component from the image segmentation, is used as input to the function `estimateFlow`. The `estimateFlow` function calculates the optical flow between two subsequent images and returns the V_x and V_y velocity components and the gradient orientation and magnitude.

```
% Estimating Vx, Vy, Magnitude and Orientation
flow = estimateFlow(opticFlow, bilde);
```

The two variables that were used in the motion estimation experiment is the optical flow orientation and magnitude, stored under the variable names `flowOrientation` and `flowMagnitude`. It was in the Motion Estimation Experiment in chapter 7.4 found that the optical flow orientation could be used if every image gradients first was thresholded by the mean image gradient magnitude. This was performed by making a `logical` mask of the `flowMagnitude` variable, where the optical flow magnitudes above 70% of the mean optical flow magnitude was assigned the value 1 - while the ones below was assigned the value 0 . The `flowMagnitude` mask was then used to change the gradient orientation values in the individual image pixels to `NaN`, in the indexes where `flowMagnitude` was zero valued. This operation was performed after the optical flow orientation had been converted from radians to degrees.

```
  % Only keeping optical flow magnitudes above 70% of mean ...
     magnitude value
    flowMagnitude = flowMagnitude > (meanMag*0.7);

     % Converting from radians and rounding
    degreeMat = round(rad2deg(testvarOri));

    % Converting from [-pi pi] to [0 360]
    lessThanZero = (testvarOri < 0)*360;
    degreeMat = degreeMat + lessThanZero;

```

```
11      % Thresholding, setting values below threshold equal NaN.
12      for k=1:800
13          for j=1:550
14              if flowMagnitude(j,k) == 0
15                  degreeMat(j,k) = NaN;
16              end
17          end
18      end
```

The optical flow orientations was then arranged into a image histogram with 72 bin, where each bin covers a 5° range. The histogram data was also extended by 100° to prevent the discontinuity at 360°, as well as being filtered with a moving average filter by the Matlab `smooth` function. A moving window, ranging over 11 histogram bins, was then made by a `for` loop to locate if there were histogram areas that indicated movement in the image. The histogram values of the bin encapsulated by the window and window index start was saved to the variables `winVal` and `winStart`. The window with the highest value in 11 consecutive histogram bins was then compared with the total amount of the histogram value, where the ratio was calculated and saved to `percentInWindow`.

```
1     % Finding the window with highest values
2       for h=1:length(histoSmooth)-winSize
3           calcWinVal = sum(histoSmooth(h:1:(h+winSize-1)));
4
5           % Saving highest amount and window start
6           if calcWinVal > winVal;
7               winVal = calcWinVal;
8               winStart = h;
9           end
10      end
11
12      % Calculating percentage
13      percentInWindow = winVal/sum(histo.Values)
```

The last operation in the motion estimation experiment was to evaluate and find the peaks for each window. It was chosen that a window had to contain a certain orientation histogram values if it would be classified as an indication of plant movement. If a window contained enough values, the Matlab `findpeaks` function would be used to find the peak value within the chosen range - with the orientation saved to the variable `direction`. It was also chosen to convert the orientation values $\in$ [360 460] to the range [0 100],

as well as setting the `direction` values of the histogram where no windows contained more than 30% of the total histogram values, equal `NaN`.

```
    % Evaluating largest window
    if percentInWindow > winPerThres
        [peakValue location] = ...
           findpeaks(histoSmooth(winStart:1:...
        (winStart+winSize-1)),'MinPeakDistance',9);

        % Finds the peak location
        peakLocation = winStart + location;

        % 72 histogram bins equals 5 degree per bin
        direction(i) = peakLocation*5

        % Changing the directin to range (0 - 359)
        if direction(i) > 359
            direction(i) = direction(i) - 360;
        end
    else
        % If not above threshold, direction equals NaN
        direction(i) = NaN;
    end
```

6.5 Black Box Modeling

The black box modelling was performed by developing the script `modelEst`, which imports the pixel count and illumination regime variables from the Matlab workspace. The pixel count data is first pre-processed by the use of `detrend` and `smooth`, before the test and validation dataset was arranged into `iddata` structures with the variable names `modelTestData` and `modelValData`. The `iddata` structure is the data structure used as one of the input variables to the Matlab `arx`, `armax` and `bj` functions, which is used to estimate the ARX, ARMAX and Box Jenkins models.

```
1  if arxEstimation == 1
2      for na = 1:5
3          for nb = 1:5
4              for nk = 1:5
5              arxSys = arx(modelTestData,[na,nb,nk]);
6              % Saving the best fit (AIC)
7                  if arxSys.Report.Fit.AIC < bestAICARX
8                      bestAICARX = arxSys.Report.Fit.AIC;
9                      bestARXModel = arxSys;
10                 end
11             end
12         end
13     end
14 end
```

The estimation of each black box model is initiated by setting their if statements true, here seen by the example of the ARX model, making it possible to estimate one model at the time. Each model is estimated by an iterative exhaustive search where each obtained *Akaike's Information Criteria* (AIC) score is compared against, the until then, best AIC score, which in the case of ARX is kept in the variable `bestAICARX`. The current model AIC score and order is then saved if the model obtains a better score than the previous calculated models.

The script does also offer two more possibilities, which includes testing the goodness of the model in terms of NRMSE as well as converting the data to SIMULINK compatible variables. The goodness of fit is found by using the Matlab `compare` function, which uses the `modelTestData` or `modelValData` and the estimated model that is being tested, `modelToTest`, as input. The `compare` function used is seen to provide several outputs, where the one that is used is the `fitTest` and `fitVal`, which provides the NRMSE between the

pixel count data and the estimated model.

```
% Comparing the model and test data
 [outputTest fitTest x0] = ...
    compare(modelTestData,modelToTest);

 % Comparing the model and validation data
 [outputVal fitVal x0] = compare(modelValData,modelToTest)

 % Plotting the fit
 figure(3)
 compare(modelTestData,modelToTest)
```

The last option in the script is to make the pixel count vectors Simulink compatible variables. This can be done by making the vector into a $2 \times n$ matrix, where n is the total amount of time lapse images. The change from a vector to a matrix is necessary as Simulink requires time-stamp information in the first matrix column.

```
% Making timestamp vector
ValDataLength = 1:1:length(ValDataInp);
EstDataLength = 1:1:length(EstDataInp);

% Arranging pixel count vector into matrix
dataOutput = [ValDataLength' ValDataOut];
estDataInput = [EstDataLength' EstDataInp];
estDataOutput = [EstDataLength' EstDataOut];
```

Chapter 7

Experiment

7.1 Color Models Comparison

This section describes a detailed comparison and evaluation of the three color models RGB, HSV and CIELAB that was presented in in the Image Processing chapter. The material that are used in the experiment is the individual components histogram representation of the *dark* and *light* regime images displayed in figure 11, as well as use of the Matlab *Color Thresholder* application in the *Image Processing and Computer Vision* toolbox.

The first color model that is being evaluated is the RGB model, with its individual component histograms representation of the *dark* and *light* regime image displayed in figure 18. The upper histogram in figure 18 originates from the red component, I_R, and is observed to have a symmetric two-modal histogram distribution in the *light* regime and a skewed histogram distribution in the *dark* regime. The symmetric two-modal distribution in the *light* regime histogram is by the *color thresholder* application found to have its origin in the blueish background as well as the greenish parts of the *mimosa pudica* - while the yellowish parts of the *mimosa pudica* contains red content in the range [0.5 0.9]. The same tool is used to discover that the skewed distribution in the *dark* regime is the result of uneven illumination of the background, where the red content in the blueish background in the upper left corner is 0.27 - while the red content in lower left corner is 0.55. The green component, I_G, represented by the mid histogram of figure 18 seems to have the same type of challenge as the red component in terms of the yellowish parts of the *mimosa pudica* having of the same amount of I_G as the blueish background, and uneven illumination causing a slight skewness in the *dark* regime histogram distribution. The green component is also serving

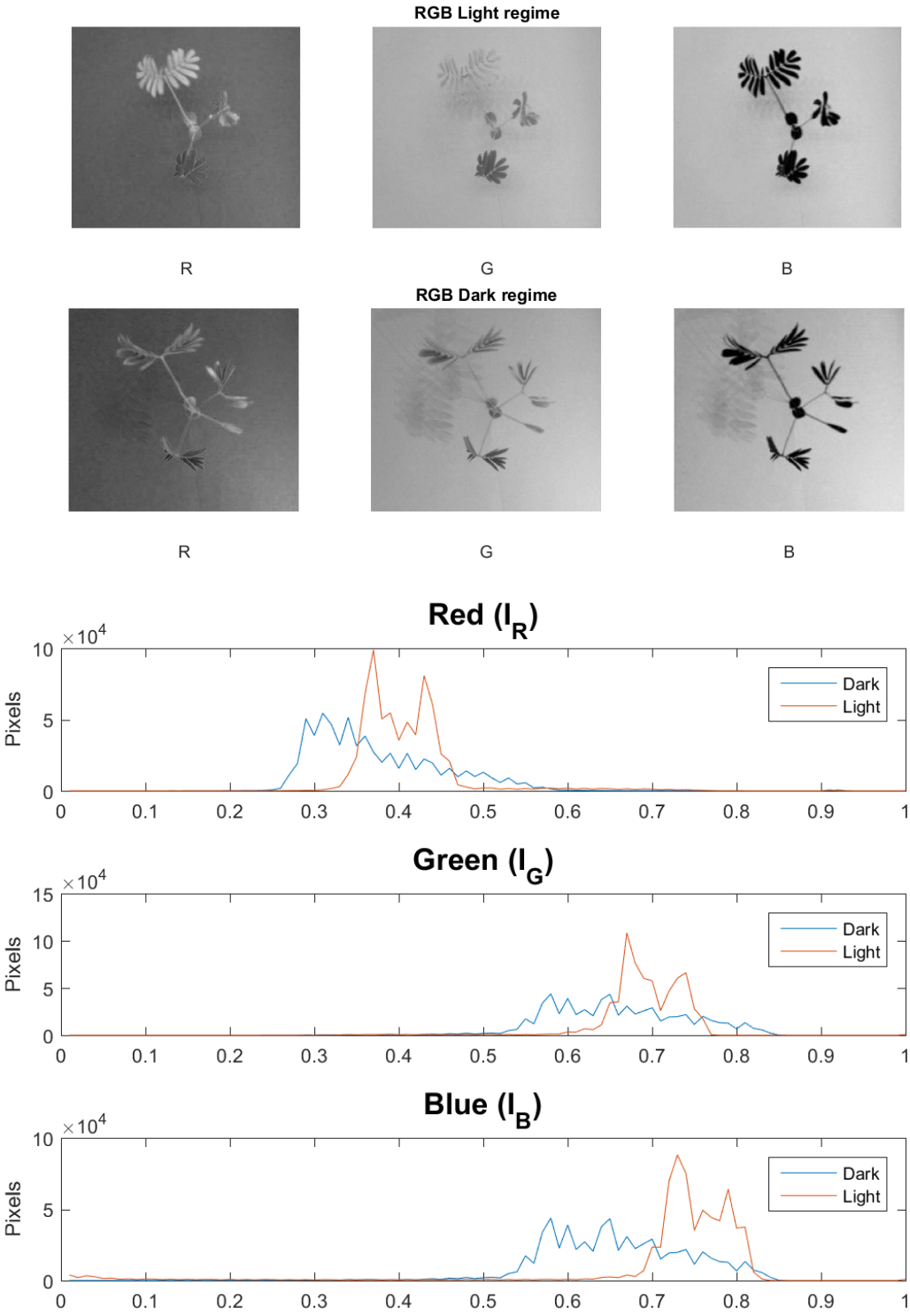


Figure 18: RGB component histograms from *dark* and *light* regime

as a good example of the differences between the RGB color model and our intuition about color. Most people would have thought that the color perceived as greenish in figure 11 would have the highest content of the green component I_G - while very few would actually have thought that the green component content in this case is higher in both the yellowish leafs and the blueish background. The last component is the blue component, I_B, which from the lower histogram in figure 18 is observed to have a symmetric two-modal distribution in the *light* regime - while the distribution in the *dark* regime is wide and symmetric. It is also observed that both histograms has a visible outlier in the range [0 0.1], that is revealed by the *color thresholder* application to be the blue content in the in the center of the *mimosa pudica* leafs. The same tool is used to discover that the blue content in the remaining part of the leafs and in some parts of the petiole is found in the range [0.1 0.5], while the bluish background is located in the range [0.5 0.9] if we include both the *light* and *dark* regime.

The next color model is the HSV model, showing its individual components histograms in figure 19. The first evaluated component that is *hue*, which seems to have a high resemblance when comparing the histograms of the *light* and *dark* regime. This is seen as a symmetric uni-modal short-tailed distribution with basis in the blueish background in the angular value of $180° - 200°$ in both of the histograms. The *Color Thresholder* application also revails a small outlier with a skewed distribution in the angular area of $40° - 85°$, which have its origin from yellowish to greenish parts of the *mimosa pudica*. The range between these two distribution includes the *hue* found in most of the low contrast areas, like the *mimosa pudica* petiole and leaf edges. The next component is *saturation*, which from the mid histogram of figure 19 is seen to have a symmetric uni-modal histogram distribution in the *light* regime, and a skewed uni-modal histogram distribution in the *dark* regime. The skewness in the histogram of the *dark* regime distribution is revealed by the *Color Thresholder* application to be the result of uneven illumination, while the slight rightward shift in the mode of the histogram distribution compared to the histogram of the *light* regime is presumably the result of reduced illumination and thus higher perceived saturation. The use of the *Color Thresholder* application did also unveil that the histogram distributions that are observed in the *dark* and *light* regime are good descriptions of the saturation of the background color. The saturation of the foreground colors seems to be in a broader range as it appears to be slightly declining from a saturation of around 0.9 in the center of the leafs, to a saturation of around 0.3 on the leaf edges and petiole. The last component, *value*, is from the lower histogram in figure 19 seen to have histogram distributions similar

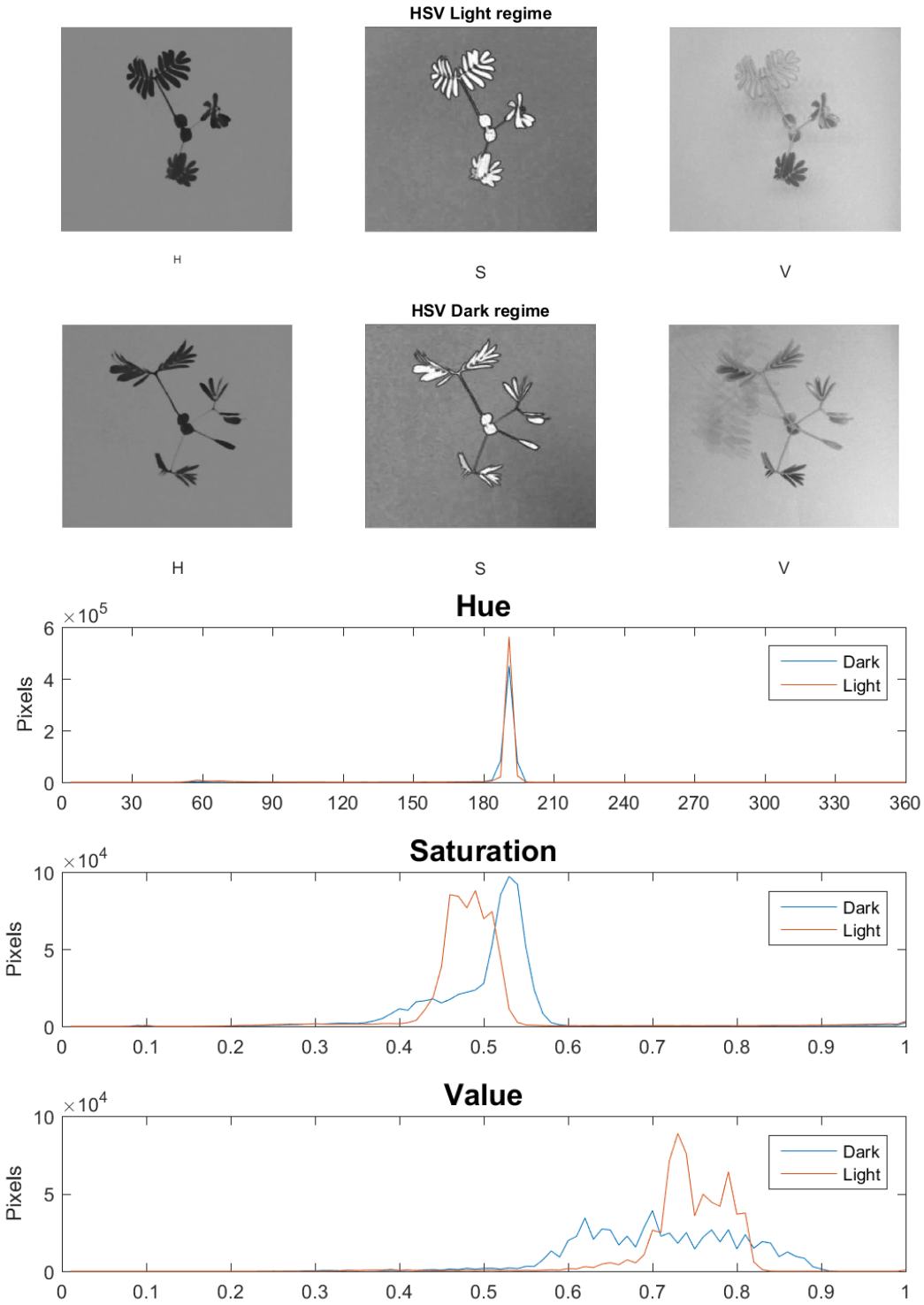


Figure 19: HSV component histograms from *dark* and *light* regime

to the blue RGB component I_B. This is because the non-linear transformation from RGB to HSV defines *value* to be equal to the highest valued RGB component [33] - where the majority of the pixels in this case has its highest content in the blue component. Further observations done in the *Color Thresholder* applications reveals that the *value* of the greenish parts of the *mimosa pudica* seems to be below 0.5 in both the *dark* and *light* regimes, while the yellowish parts of the *mimosa pudica* seems to be in the same *value* range and the background region.

The last color model is the CIELAB which is shown with its individual component histograms in figure 20. The first component, L*, is observed to have much of the same properties as seen in the *value* component in the HSV color model. Most of the greenish leafs of the *mimosa pudica* has a L* value below 60, while the yellowish parts of the *mimosa pudica* share its value range with the background region.The second component, a*, is displayed by the mid histogram in figure 20. It seems to be a high resemblance in the case of *dark* and *light* regime histograms as both has a symmetrical uni-modal distribution with a slightly longer tail on the right side. Further observations done with the *Color Thresholder* application reveals that the lower negative side of a*, in the range of $[-24\ -15]$, is representing the a* content in the blueish background, the low contrast areas and the parts of the *mimosa pudica* that is perceived to have a darker greenish color. The remaining range, from $[-15\ 0]$, is observed to represent the *mimosa pudica* regions perceived to have a faint greenish to yellowish color. The last component is the b* component, which is shown in the lower histogram of figure 20. Both the *light* and *dark* regime histograms have a high resemblance with a uni-modal short-tailed distribution located on the negative side of the b* component. The *Color Thresholder* application reveals that the uni-modal distribution has its origin from the blueish background and is located in the range around $[-25\ -5]$. This is separated from the b* content in the foreground region, which in both the *dark* and *light* regime is observed to be on the positive side of b* component in the range [0 75].

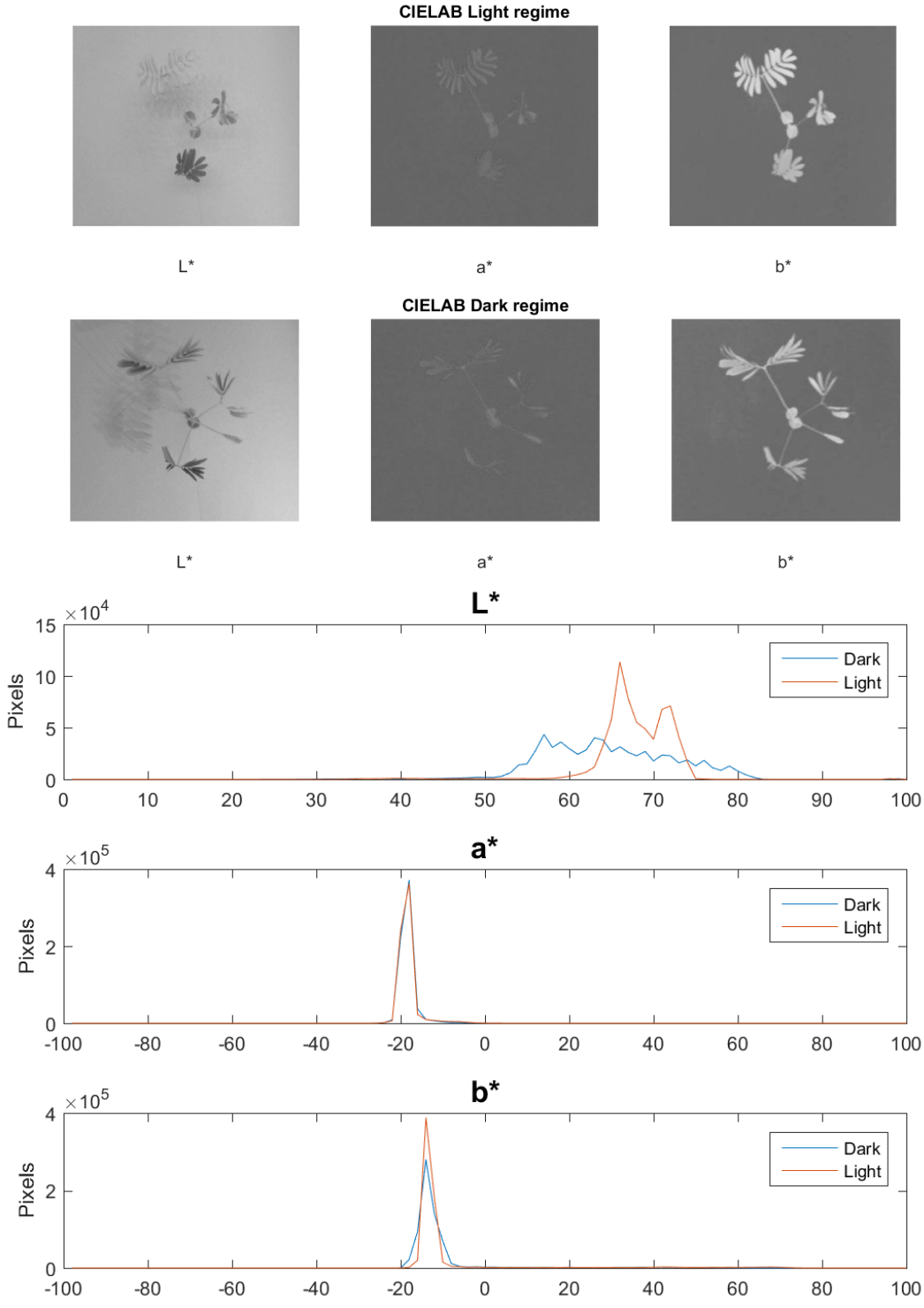


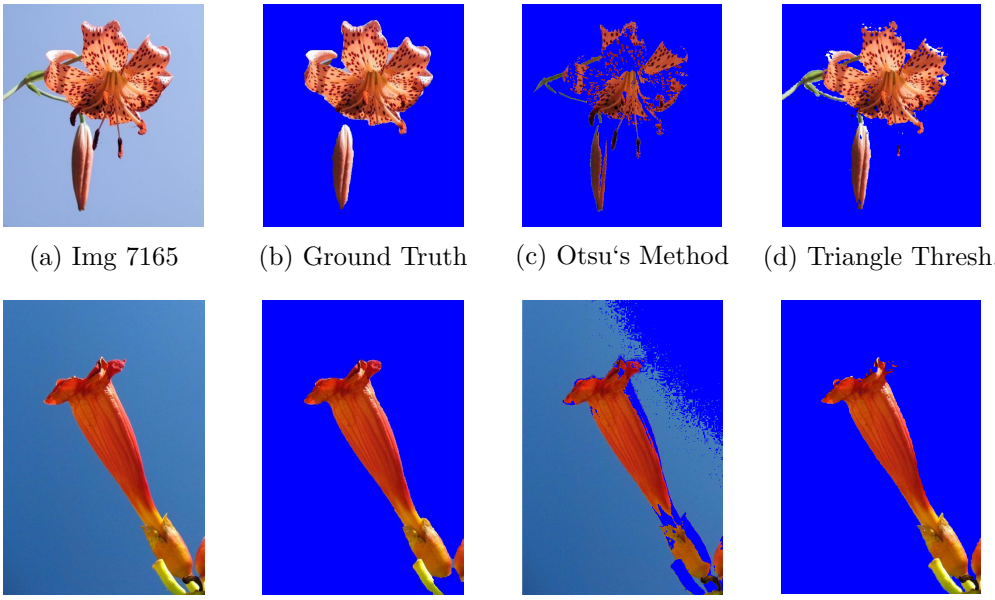
Figure 20: CIELAB component histograms from *dark* and *light* regime

The evaluation of the different color models reveals that best individual components from each color model is the *blue* component from RGB, the *hue* componet from HSV and the b^* component from CIELAB. The blue RGB component is not chosen as the difference in the *light* and *dark* histogram distribution indicates that it is the component who is most affected by changes in the illumination. Both the *hue* from HSV and b^* from CIELAB seem to provide much of the same result when being evaluated, and there has not been found any other reason for choosing the one over the other except the more intuitive representation of the *hue* component. This is also the reason why the HSV color model will be used in the further image processing part of the thesis.

7.2 Image Segmentation Evaluation

The purpose of this section is to present an evaluation of the segmentation routine that was used in the thesis. It has been chosen to first do a subjective evaluation of the segmentation, where the triangle thresholding is compared against the result from segmentation with the automatic and non-parametric Otsu's method [29]. The second part of the section is dedicated to a brief objective evaluation, where the Dice similarity coefficient is calculated for each segmented image.

It has been chosen to use a total of seven images in the evaluation, where two images originates from the Oxford 102 category flower dataset [28], and the remaining five is taken from the available *mimosa pudica* time-lapse series. It was chosen to start the evaluation with the two Oxford 102 category flower images, named 7165 and 7958. These images are displayed in figure 21a and figure 21e and was selected as they fulfil the assumption of a uniform and dominating background color. The two images, 7165 and 7958, did also have available ground truth images that was used to evaluate the segmentation methods performance. The ground truth of image 7165 is displayed in figure 21b and the ground truth of image 7958 is shown in figure 21f.



(a) Img 7165 (b) Ground Truth (c) Otsu's Method (d) Triangle Thresh.

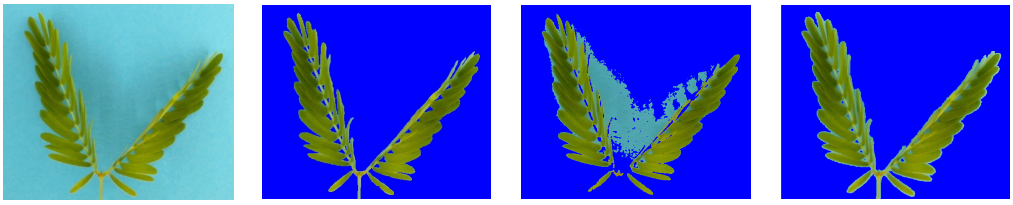
(e) Img 7958 (f) Ground Truth (g) Otsu's Method (h) Triangle Thresh.

Figure 21: Image segmentation - Oxford 102 category dataset

The result of segmenting image 7165 with Otsu‘s method and triangle thresholding is seen in figure 21c and figure 21d. These images clearly shows that the triangle thresholding is most suitable in the case of segmenting the flower from the uniform background. The triangle thresholding manage to segment most of the flower in image 7165 correctly, with the exception of parts at the brightest edges and some of the darker dots at the petals. It should also be noted that the ground truth in figure 21b seem to consist of some of the background pixel, seen as light blue color in the spacing between the petals. These area looks to be segmented correctly by the triangle thresholding in terms of being classified as background pixels, and not foreground. The segmentation performed by the Otsu‘s method is seen to be missing large parts of the flower. This may have its root in the Otsu‘s methods mode of operation, which starts by converting the image from a RGB representation to grayscale. The method is then assuming that the image can be divided into two classes, e.g a foreground and background class, where a threshold is found by minimizing the within-class variance [39]. This is seen from figure 21c to result in a segmentation where the brightest parts of the flower is classified as background pixels - while the darker parts is classified as foreground.

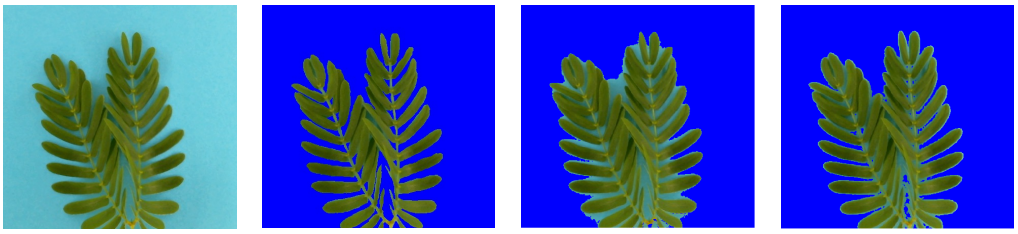
The image named 7958, seen in figure 21e, seem to provide much of the same challenges as image 7165 in figure 21a. The triangle thresholding in figure 21h segments the image with a result that looks to have a high resemblance with the ground truth in figure 21f. It only seem to differ by the triangle thresholding including a small part of the stem in the lower right corner, as well as the segmentation by the triangle thresholding does seem to miss a small part of the upper petal edge. The result of segmentation by Otsu‘s method in figure 21g seem to struggle with the same problems as in the case of segmentation of image 7165, and are in this case miss-classifying large parts of the background as foreground - and some parts of the foreground as background. This ends with a segmentation where its not possible to distinguish the foreground and the background.

One of the challenges with finding suitable test data was that the ground truth often exclude the flower petioles/stems, as in the Oxford 102 category dataset. It was therefore decided to select a total of five images from the available time-lapse series and compose a set of manually segmented images. Three of these images was taken from the *light* regimes and is displayed in figure 22a, figure 23a and figure 24a, with their ground truth displayed in figure 22b, figure 23b and figure 24b respectively. These images were selected to test the segmentation routines performance on instances like local shadows, small spacing between leaflets and occlusion between the compound leafs.



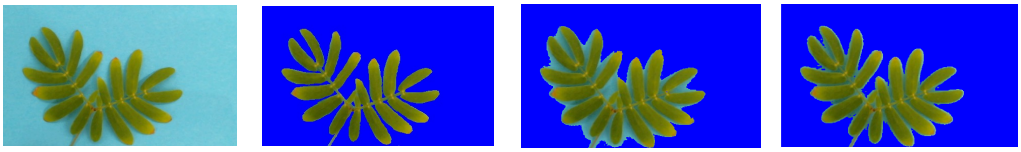
(a) Img 1 (b) Ground Truth (c) Otsu‘s Method (d) Triangle Thresh.

Figure 22: Image segmentation - Test image 1



(a) Img 2 (b) Ground Truth (c) Otsu‘s Method (d) Triangle Thresh.

Figure 23: Image segmentation - Test image 2

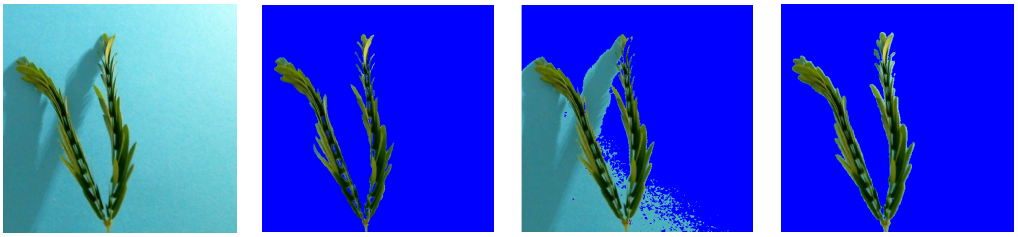


(a) Img 3 (b) Ground Truth (c) Otsu‘s Method (d) Triangle Thresh.

Figure 24: Image segmentation - Test image 3

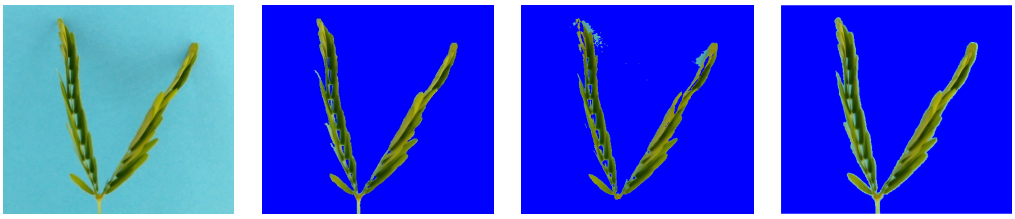
The result of segmentation by the triangle thresholding routine is shown in figure 22d, figure 23d and figure 24d, and shows an overall good performance. It is seen from figure 22d that the shadows does not seem to affect the segmentation, which was expected as the routine perform the segmentation with basis in a hue component image. The triangle thresholding did also manage to segment the individual leaflets with a varying result, where most of the exterior leaflets in figure 23d seems to be classified correctly - while the area between the two compound leafs in the same image seem to include parts of background pixels in the foreground object. The same inclusion of background pixels in the foreground can also be seen in some of the smaller spacing between the leaflets and along the rachis in figure 24d. The same images were then segmented with Otsu‘s method, with the result shown in figure 22c, figure 23c and figure 24c. The result seen in figure 22c show

that the method is more sensitive to local shadows in the images compared with the triangle thresholding. It is also observed from figure 23c that Otsu's method is classifying a larger amount of background pixels in spacing between the leaflets - as foreground. The same is also seen in the spacing between the leaflets and along the contour of the *mimosa pudica* in figure 24c



(a) Img 4 (b) Ground Truth (c) Otsu's Method (d) Triangle Thresh.

Figure 25: Image segmentation - Test image 4



(a) Img 5 (b) Ground Truth (c) Otsu's Method (d) Triangle Thresh.

Figure 26: Image segmentation - Test image 5

The two remaining images is displayed in figure 25a and figure 26a, with their respective ground truth shown in figure 25b and figure 26b. Both images is examples from the *dark* regime and show instances of a folded *mimosa pudica* with and without local shading. The result of segmenting by the triangle thresholding is displayed in figure 25d and figure 26d, and shows good results. It is only possible to see content of the background along the contours of the mimosa pudica and in the small spacings along the rachis. The result of segmenting by Otsu's method is displayed in figure 25c and figure 26c, where figure 25c clearly shows that Otsu's method fails to segment the foreground, i.e. the *mimosa pudica*, from the background. This might be the result of the local shadows making the background non-uniform. The segmentation of image 5 in figure 26c reveals a better result, where the only included parts of the background in the foreground is seen at the upper parts of the compound leafs. It is also observed that the segmentation in figure 26c does not include the *mimosa pudica* stem.

The next step was to perform an objective evaluation of each segmentation by using the Dice similarity coefficient (DCS), shown in equation 7.1, as a metric.

$$QS = \frac{2 \mid X \bigcap Y \mid}{\mid X \mid + \mid Y \mid} \tag{7.1}$$

The Dice similarity coefficient is a metric that describes the similarity of X and Y in the range [0 1], where the coefficients 0 and 1 is the points of non- and full spatial overlap respectively [31]. The quotient of similarity (QS) is found by first multiplying the intersecting samples of binary value 1 in image X and Y by two, before the product is divided by the total amount of samples with binary value of 1 in both images. The calculation of the Dice similarity coefficients is executed in the Matlab workspace with the *DiceScore* function developed by Dr Hanno Scharr [31].

Method	Img 7165	Img 7958	Img 1	Img 2	Img 3	Img 4	Img 5
Triangle	0.8977	0.9354	0.9556	0.8965	0.9272	0.9396	0.9366
Otsu‘s	0.6325	0.2537	0.7741	0.8711	0.8727	0.4391	0.8951

Table 7.1: Dice similarity coefficients - Otsu‘s method and Triangle threshold

The Dice coefficients from the comparison of segmentation is found in table 7.1, where the manually segmented images was used as ground truth. The use of the triangle based thresholding yielded an average quotient of similarity of 0.932 ± 0.0175, while Otsu‘s method resulted in an average quotient of similarity of 0.6805 ± 0.2520. Removing the Oxford 102 category flower datasets, which does not include the flower petiole/stem in the ground truth, changes the result of the triangle based thresholding to an average quotient of similarity of 0.938 ± 0.0109 and Otsu‘s method to an average quotient of similarity of 0.7755 ± 0.1947.

7.3 Black Box Model Estimation

This section describes the experiment where the different *black box* models from chapter 4.1, is used to estimate a model for the oscillations that was observed in the pixel count from the time-series of data. The experiment were started by first partitioning the chosen dataset into a test and validation dataset, where the matlab function `Plotting` from chapter 6.3 is used to displayed the test data in figure 27. It can be seen from the figure 27 that a 24-hour period is divided into 16 hours of *light* regime and 8 hours of *dark* regime, with the exception of the small regime duration change that is seen around the 2000 and 4000 sample area. The intention of the small regime duration changes, called "jetlags", is to observe how the *mimosa pudica* adapts to the shifted 24-hour period that continues after the "jetlag". Each of the 4000 pixel count samples in figure 27 is obtained from the individual segmented 3 minute interval time-lapse images, making the time span of the test data 8.33 days.

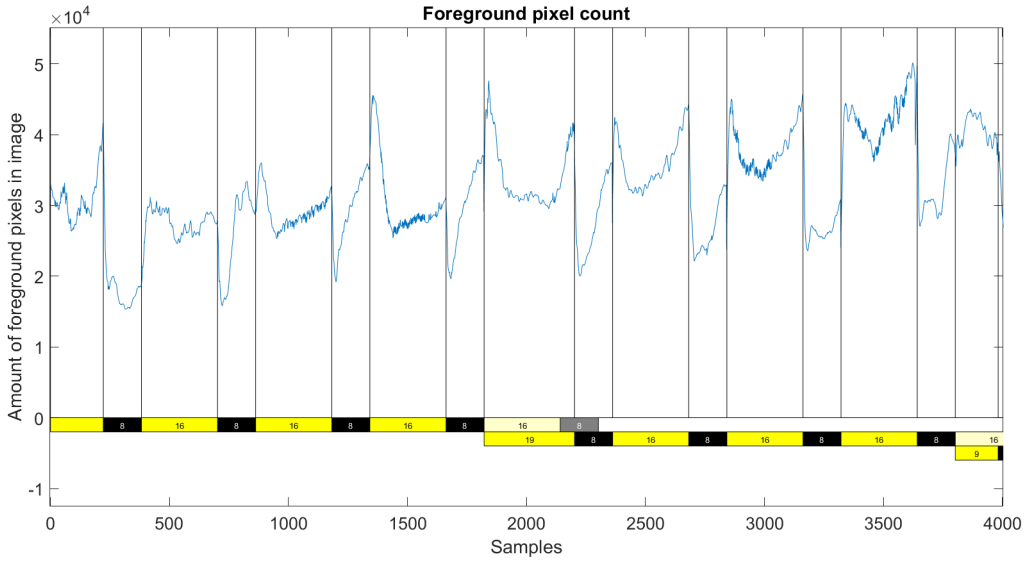


Figure 27: Test dataset - Model estimation

The validation set is displayed in figure 28 and contains 5858 samples, resulting in a dataset with a time span of 12.2 days. The approximately 1000 first samples is seen to come from a 24-hour period divided into 16 hours of *light* regime and 8 hours of *dark* regime, before the illumination is kept constant for 243 hours - indicated by the red bar. The *mimosa pudica* response to the constant stimulus is here seen to result in a underdamped response in the pixel count.

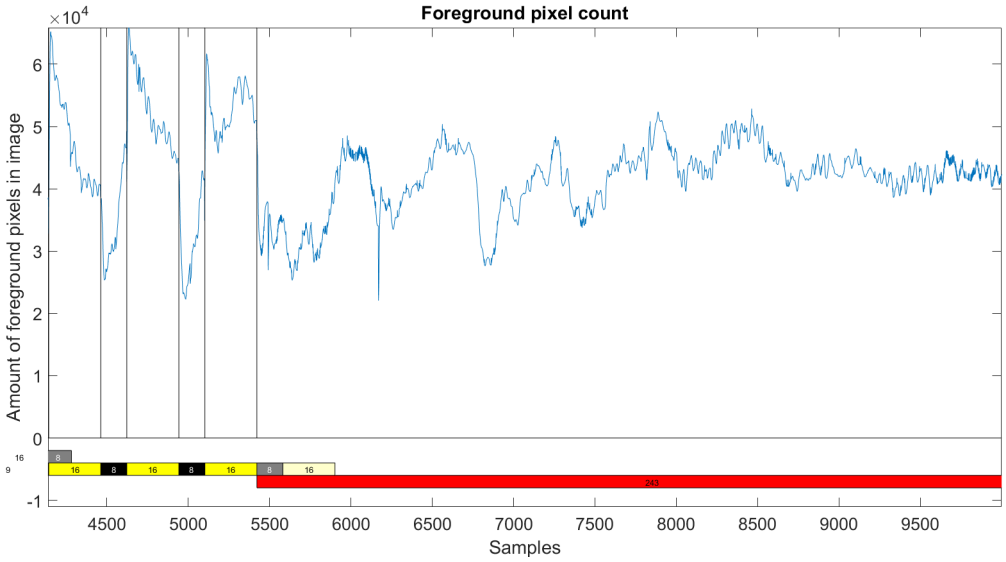


Figure 28: Validation dataset - Model estimation

One of the requirements when using transfer functions to model a dataset is that the datasets cannot have non-stationary effects [20], which includes offsets, trends etc. This was solved by first normalizing the data, making the pixel count $\in$ [0 1], as well as using the Matlab `detrend` function to both remove mean and trends. These operations resulted in the test dataset shown in figure 29 and the validation dataset shown in figure 30.

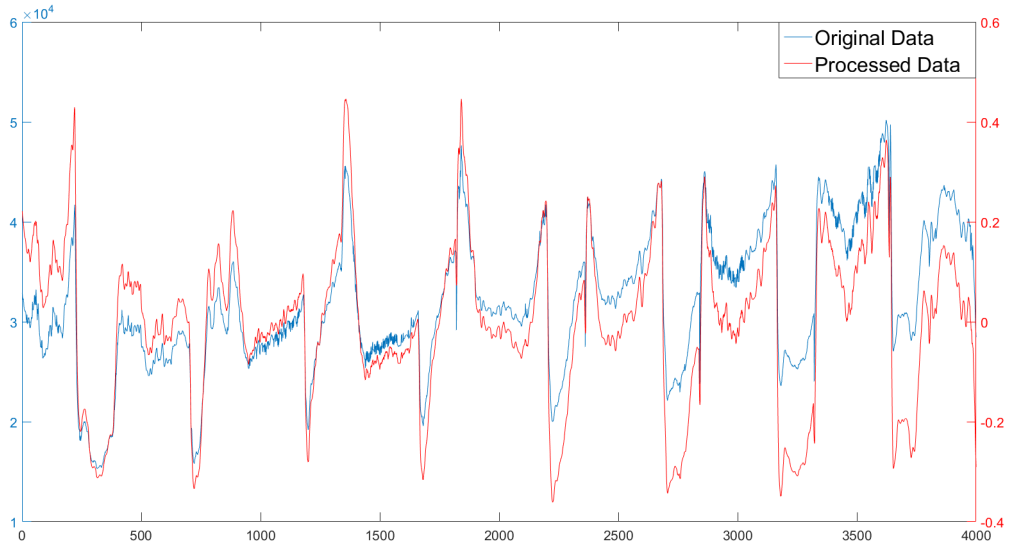


Figure 29: Test dataset, removal of non-stationary effects

The test and validation pixel count data was then arranged as the systems output while the state of the illumination regime, where the light and dark regime was assigned the values 0.1 and -0.1 respectively, was arranged as the systems input. The two datasets were then arranged as `iddent` variables which is a Matlab data format that is compatible with the system identification toolbox

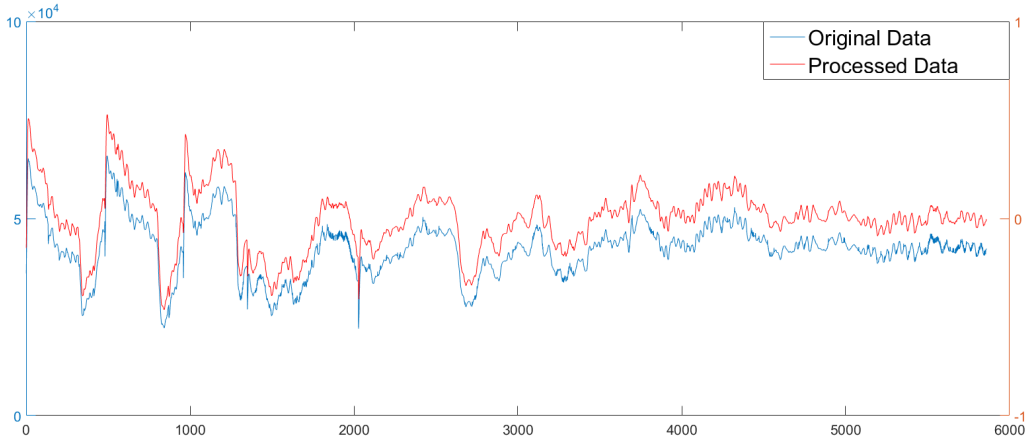


Figure 30: Validation dataset, removal of non-stationary effects

The next step in the experiment was to find the best estimate of the ARX, ARMAX and Box Jenkins models. It was chosen to limit the different model polynomials $A(q), B(q), ..., D(q)$ to have a maximum order of five. The exhausting task of finding the best combinations was solved by an iterative Matlab routine, with the best combinations being the one with lowest AIC values. The ARX, ARMAX and Box Jenkins models were used to estimate a model of the test data in figure 29, where the model was being validated against the dataset in figure 30. The estimated models fit to the test data is displayed in figure 31, where each of the individual models were evaluated to obtain the best AIC score by only consisting of fifth order polynomials. The model with the best NRMSE was the Box Jenkins, which scored a fit of 29.33% to the test data. The same models were then used at the validation dataset, displayed in figure 32, where the Box Jenkins still obtained the highest NRMSE score, with a fit of 19.04%. It is clear from the plot that the models fail to provide a satisfying estimate of the input output relationship of the system.

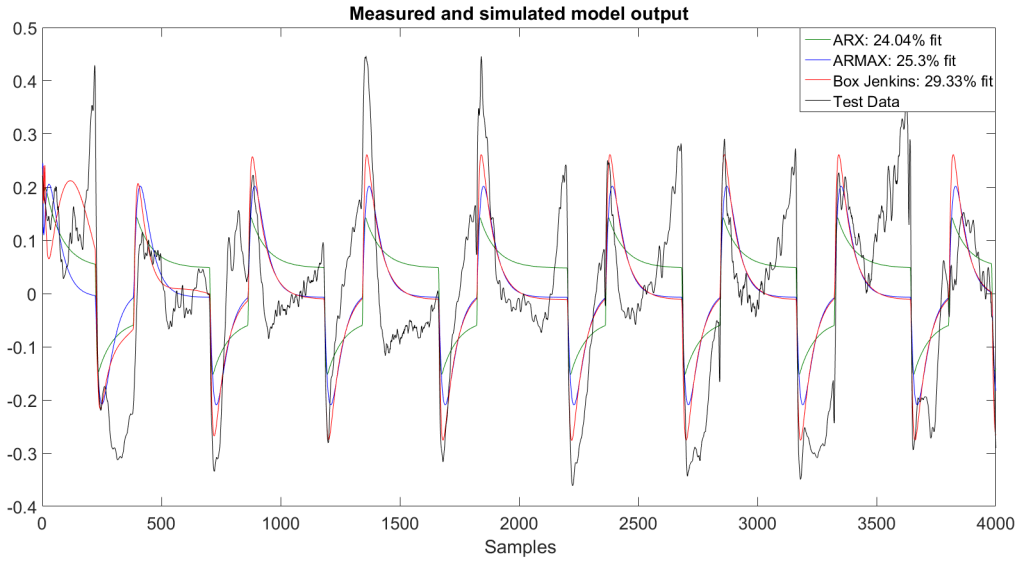


Figure 31: Estimated model fitted to test dataset

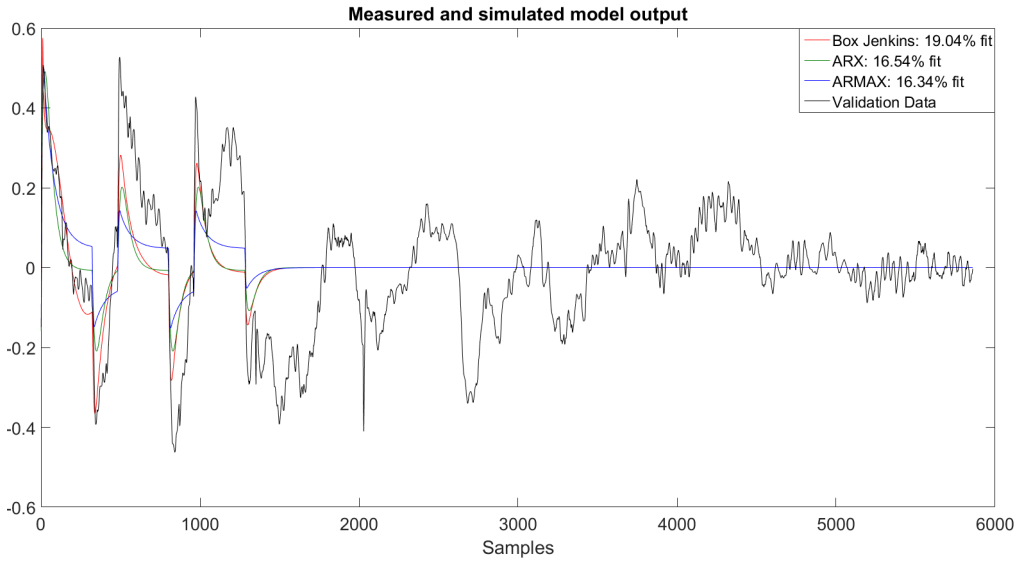


Figure 32: Estimated model fitted to validation dataset

One of the challenges with the test and validation data is the saddled/sloped shaped pixel count, which in most cases were found to be the result of tertiary and/or secondary pulvini angling the leaf in a manner that reduces the surface area. An example of this is shown with the sample range 1800 - 2200 in figure 31, displayed close up in figure 33.

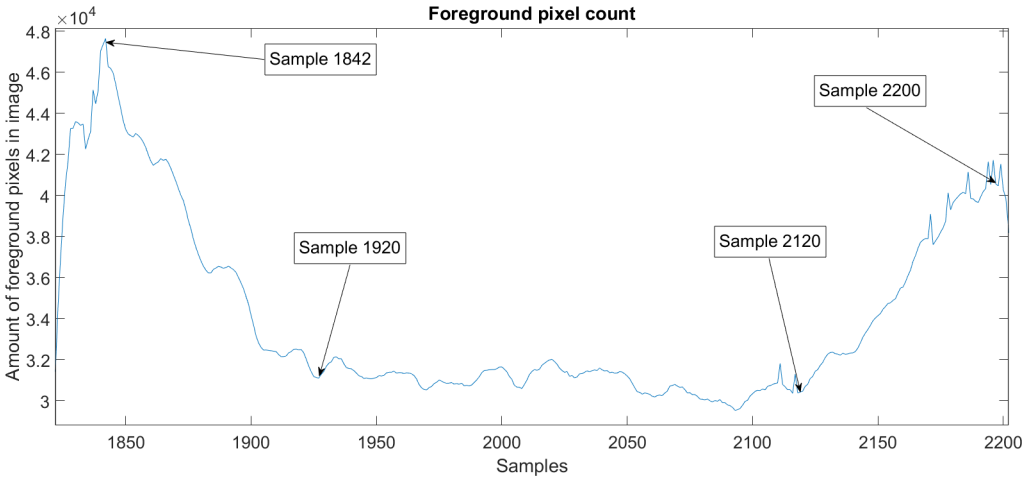
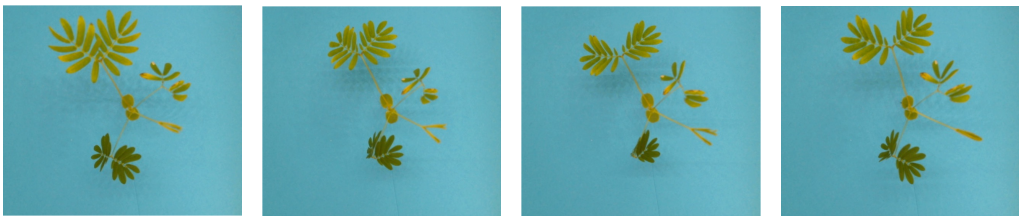


Figure 33: Saddled/sloped pixel count response

The first pixel count peak in the sample range 1800 - 2200 is observed at sample 1842, where it is seen from the leftmost image in figure 34 that the *mimosa pudica* leafs are widespread. The two middle images in figure 34, representing sample 1930 and 2120, is then revealing that the compound leafs are angled downwards for a duration of 200 samples, reducing the total surface area. The count is then slowly increasing between sample 2120 and 2200, with the pixel count reaching its second peak at sample 2200 displayed in the rightmost image in figure 34.



(a) Sample 1842 (b) Sample 1920 (c) Sample 2120 (d) Sample 2200

Figure 34: Images from saddle/sloped response

The saddles/sloped response in the dataset motivated for the inclusion of a new dataset, which would work as test dataset to make new models as well as validation set for the already estimated models. The new dataset is displayed in figure 35 and consists of 5316 samples from a 5 minute interval time-lapse series, making the time span of the data 18.45 days. The data set does also have 3 "jetlags", seen around the 3200, 4200 and 8000 sample point.

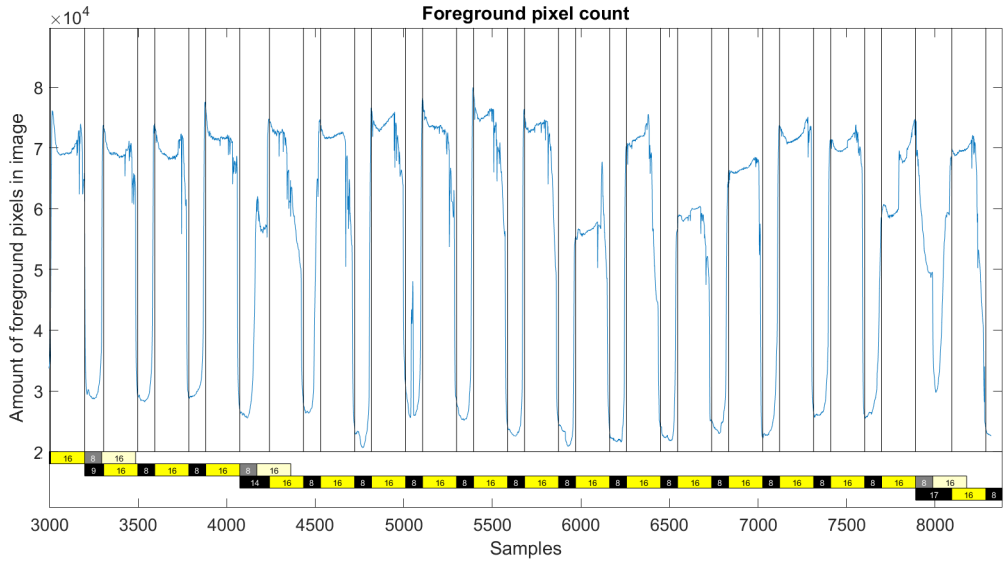


Figure 35: Secondary validation and test dataset

The first operation that was performed was to validate the already estimated models on the new dataset. The result is displayed in figure 36. The overall result is a poor fit, but it is possible to see that the ARMAX model produces an oscillatory response that is slightly shifted compared with the validation data. The delay of the ARMAX model was therefore increased in an attempt to synchronise it with the pixel count response, with the result of an unstable model at a sample delay of 10. The data in figure 35 was then used to estimate new ARX, ARMAX and Box Jenkins models. The result is displayed in figure 37, with the ARX having the polynomial orders $A(q) = 5, B(q) = 3$ and input delay of 3, the ARMAX having the polynomial orders $A(q) = 5, B(q) = 2, C(q) = 5$ and input delay of 2 and the Box Jenkins having the polynomial orders $B(q) = 5, C(q) = 3, D(q) = 5, F(q) = 5$ with input delay of 3. These models were then validated by both the dataset in figure 29 and figure 30, with the result displayed in figure 38 and figure 39.

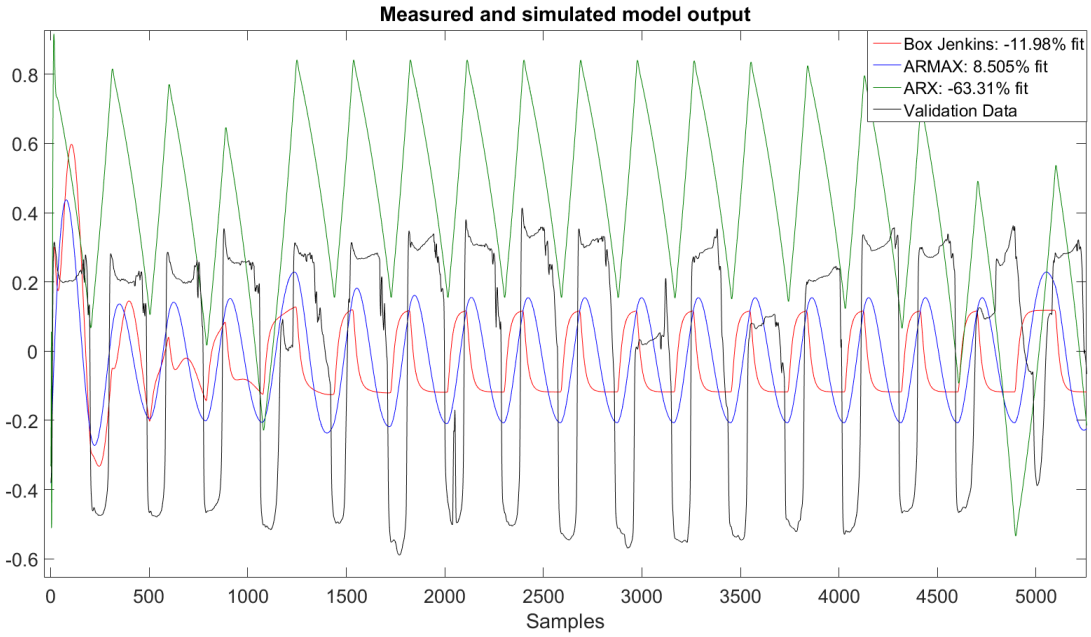


Figure 36: Estimated model fitted to secondary test dataset

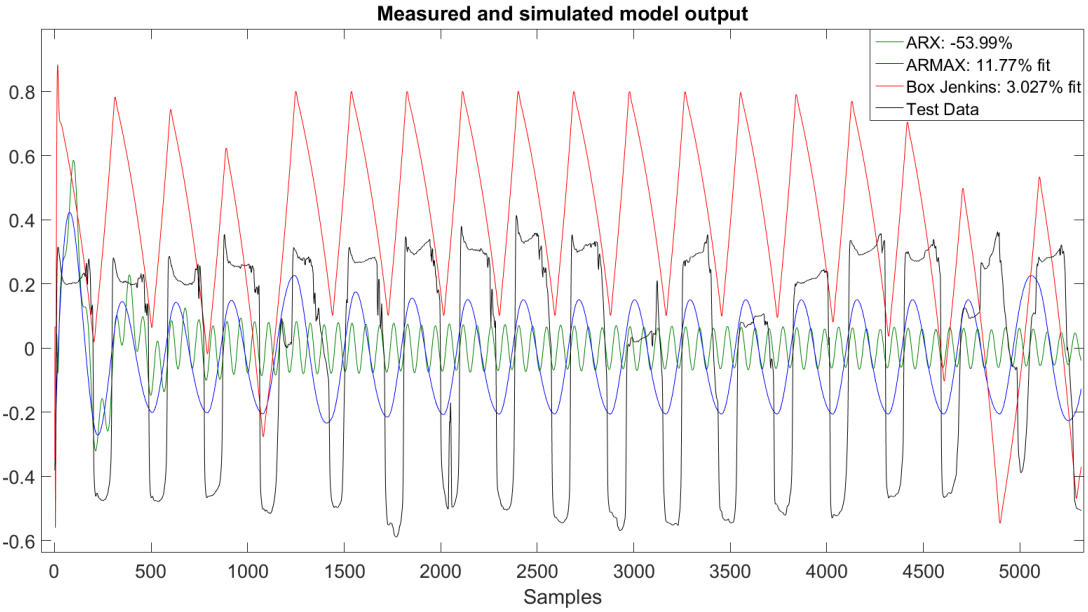


Figure 37: Secondary estimated model fitted to secondary validation dataset

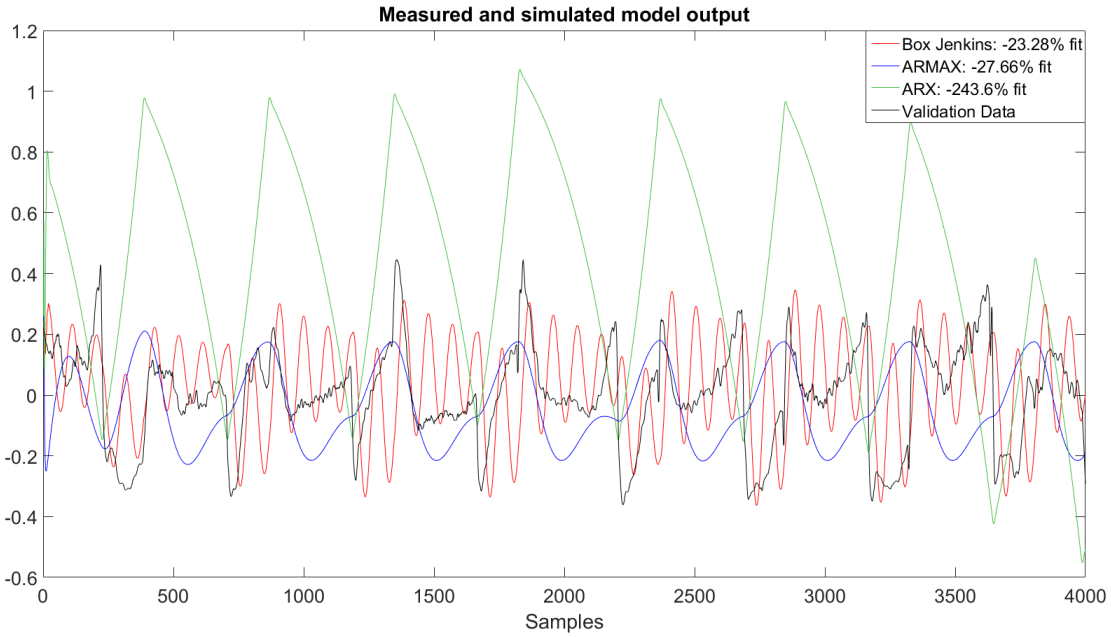


Figure 38: Secondary estimated model fitted to first test dataset

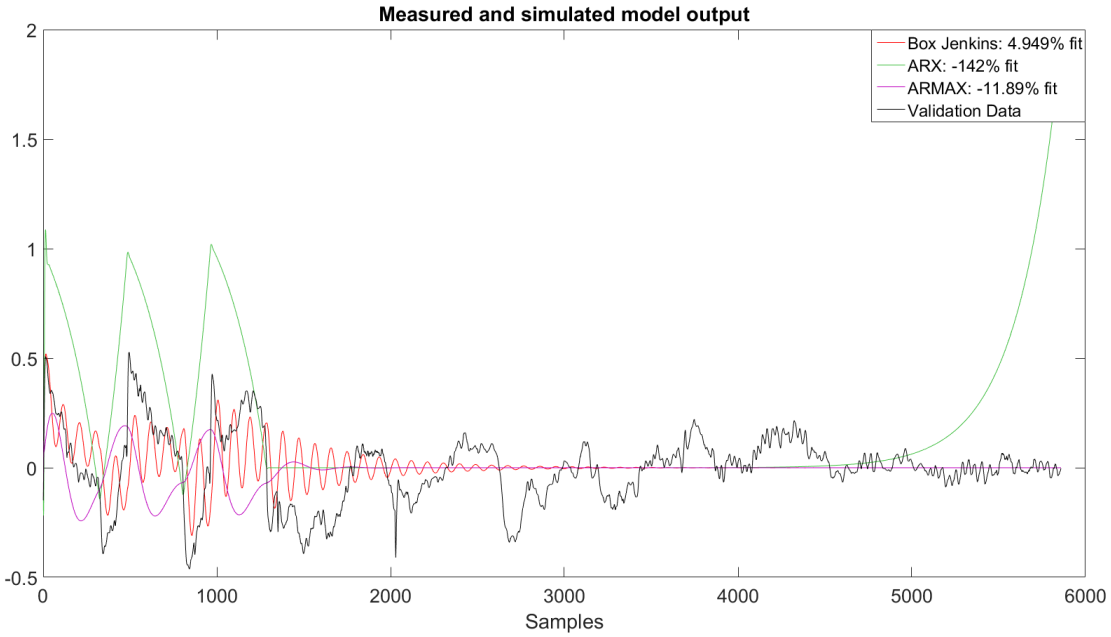


Figure 39: Secondary estimated model fitted to first validation dataset

It is clear from figure 38 and figure 39 that the models estimated from the second dataset does not manage to capture the dynamics of the system in the first dataset. The ARX model provides the worst result in both cases, where it is seen to become unstable during the constant illumination regime in figure 39. The ARMAX model does also struggle in both cases, with an increasing value in the *dark* regime and decreasing value in the *light* regime - which is the opposite of the response in the validation data. The last model is the Box Jenkins which looks to be following the trends of the pixel count, but with a oscillatory response. This response is also seen to be too swift compared with the systems oscillation in constant regime in figure 39.

It was then decided to import the dataset from figure 28 into the Matlab Simulink environment to compare the oscillatory response of the system with a second order transfer function, as shown in equation 7.2.

$$H(s) = \frac{Y(s)}{X(s)} = \frac{K}{(\frac{1}{\omega_n^2})s^2 + 2\frac{\zeta}{\omega_n}s + 1} \tag{7.2}$$

This attempt was based on a trail and error approach with the goal of finding suitable values for the parameters ω, ζ and K, describing systems swiftness, degree of oscillation and gain respectively [32]. As a starting point for the search, it was assumed that a negative step in the input value 0.1 to 0.0 introduced the underdamped response with a period time of 600 samples, displayed in figure 40.

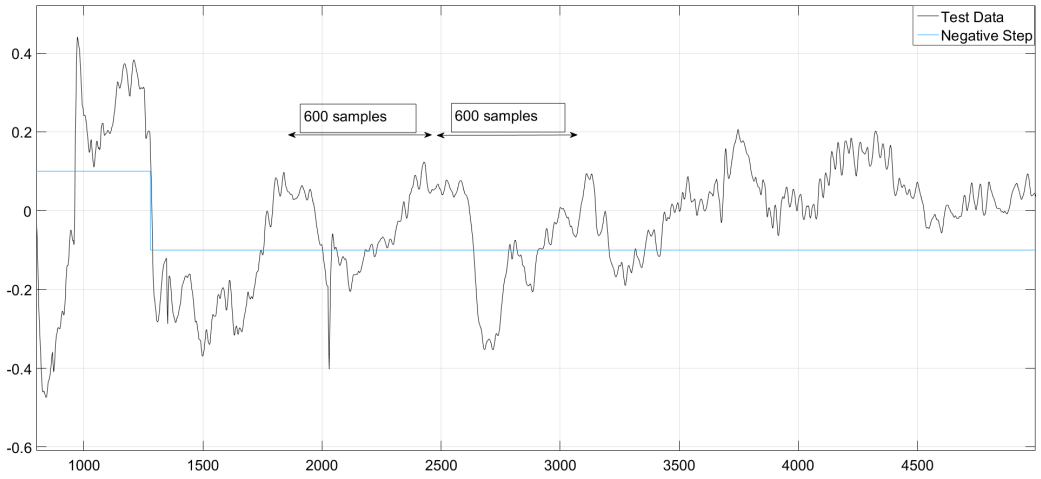


Figure 40: First validation dataset step response

The underdamped response of the system implies that $0 < \zeta < 1$, which means that the systems rise-time/system speed relationship is given by the approximation in equation 7.3 [13]

$$T_r \approx \frac{1.5}{\omega_n} \tag{7.3}$$

The challenge by using equation 7.3 to find ω_n was in this case to locate the correct T_r. It was instead chosen to use equation 7.4 to obtain a value for ω_n, as the period of the oscillation already was known to be 600 samples from figure 40.

$$\omega_n = \frac{2\pi}{T_p} \tag{7.4}$$

The result of using a period of 600 samples, which equals 30 hours, gave a $\omega_n = 0.0104$. This did also make it possible to find the approximated rise-time of 144.23 samples, or approximatly 7.22 hours, by equation 7.3. Several combination were then tried to find a suitable ζ and K, where a $\zeta = 0.075$ and $K = 2$ gave the underdamped response displayed in figure 41, and the transfer function equation 7.5.

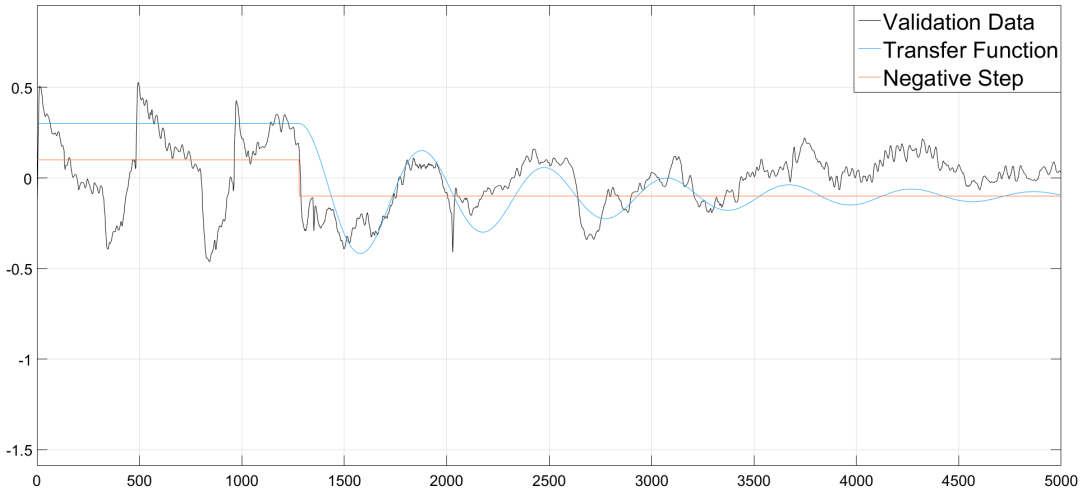


Figure 41: Transfer function fitted to first validation dataset step response

Two interesting observations were seen during the analysis of the oscillations and the estimation of the transfer function. The first observation is that the period of the underdamped response in the *constant* regime has a longer duration (30 hours) than in the periodic *light* and *dark* regime (24 hours).

The second observation is that the swiftness of the system, i.e. seen in the rise time, also seem to change during the *constant* regime - where it clearly becomes slower. These observations could imply that the periodic *light* and *dark* regime works as a disturbance on the plant, forcing the widespread and closing of leafs, while the oscillations seen in the *constant* regime is suggested to be the plants *free-running* period, with a period time of 30 hours.

$$H(s) = \frac{Y(s)}{X(s)} = \frac{2}{9000s^2 + 15s + 1} \tag{7.5}$$

It was then chosen to do one last test where the estimated transfer functions response were validated by the test dataset from figure 27 and figure 35. The validation and illumination regime data was imported into Simulink, with the illumination regime being used as input to the transfer function - as displayed in figure 42.

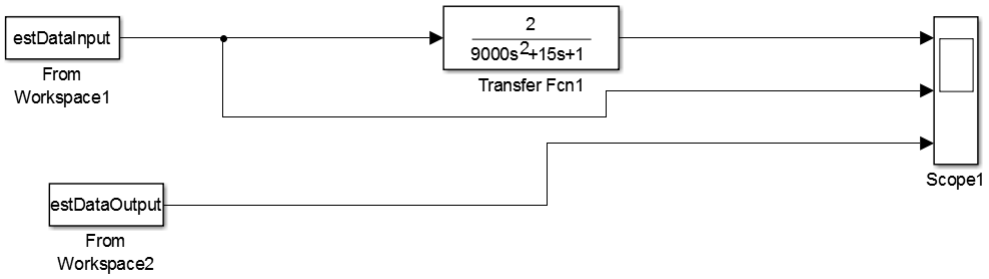


Figure 42: Transfer function model in Simulink

The comparison of the estimated models output and the validation data is displayed in figure 43 and figure 44. The figures reveal that the transfer function model is seen to have a sinusoidal like response, where maximum peak value of the estimated system in most cases is occurring during the *dark* illumination regime and the minimum peak value in most cases are found in the *light* illumination regime. This is suggested to originate from the systems difference in swiftness, seen from the transfer functions slower rise time and time to first peak, causing the transfer function model to lag behind the response of the validation data.

The final conclusion is that the transfer function model does not manage to model either of the systems in the validation datasets.

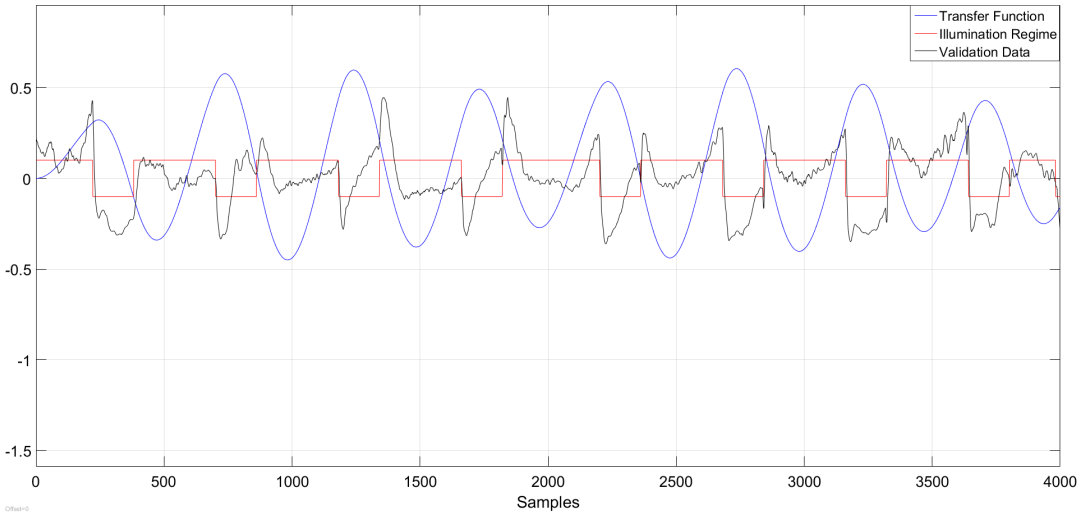


Figure 43: Transfer function fitted to first test dataset

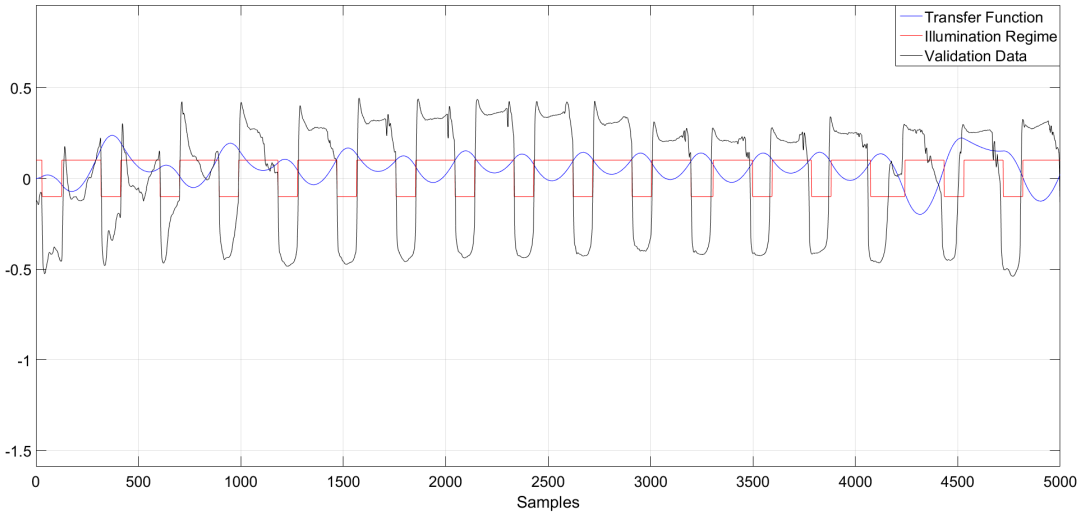


Figure 44: Transfer function fitted to secondary test/validation dataset

7.4 Motion Estimation Evaluation

The intention of the motion estimation evaluation section is to examine the possibility of quantifying the plants movement into a metric usable in the analysis of the *mimosa pudica* responses. To simplify the examination, a dataset of a *mimosa pudica* with only two compound leafs have been used, which is assumed to result in fewer motion gradients. This dataset was also used as the secondary dataset in the black box experiment section, and was seen to have a levelled response in the *dark* regime and a slightly sloped and saddled response in the *light* regime. The levelled response in the *dark* regime is observed to be the result of almost no widespread of the individual *mimosa pudica* leaflets. The response observed in the *light* regime contains more activity, with the saddle/sloped response originating from (i) occlusion of the leaflets, and (ii) tertiary and/or secondary pulvini angling the individual leaflets upwards and thus reducing the leaflet area.

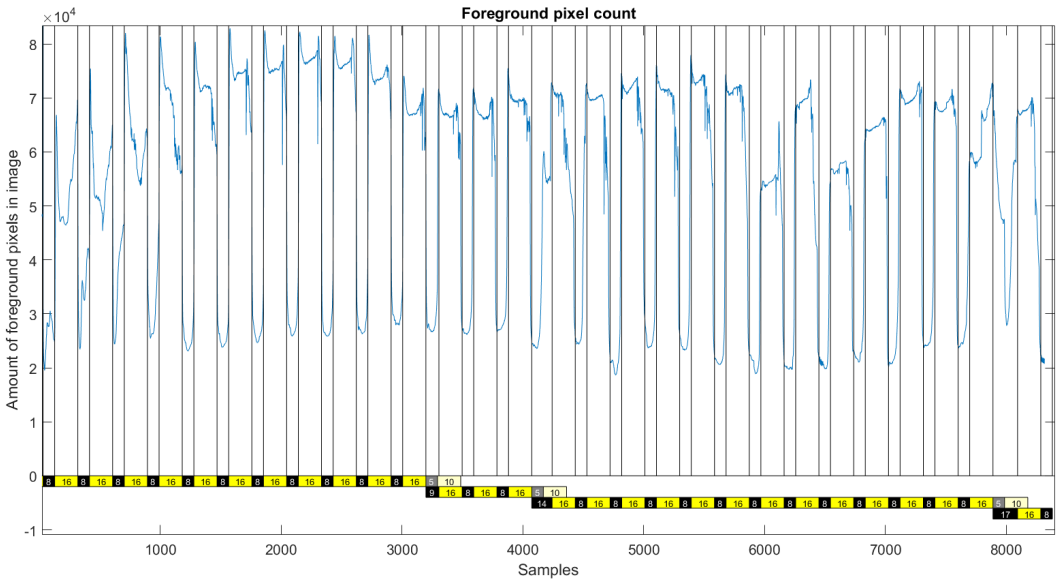


Figure 45: Pixel count data from images used in motion estimation

The two variables that were selected for examination in this experiment was the optical flow vectors orientation and magnitude. Optical flow was in chapter 5 described as a measure of pixel movement between to succeeding images, and can be exemplified by figure 46 where the red arrows indicates the optical flow vectors.

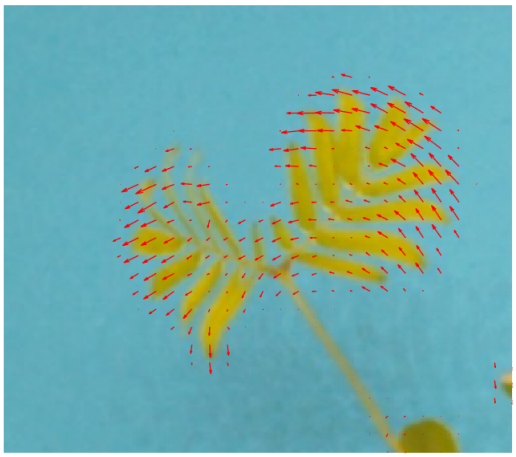
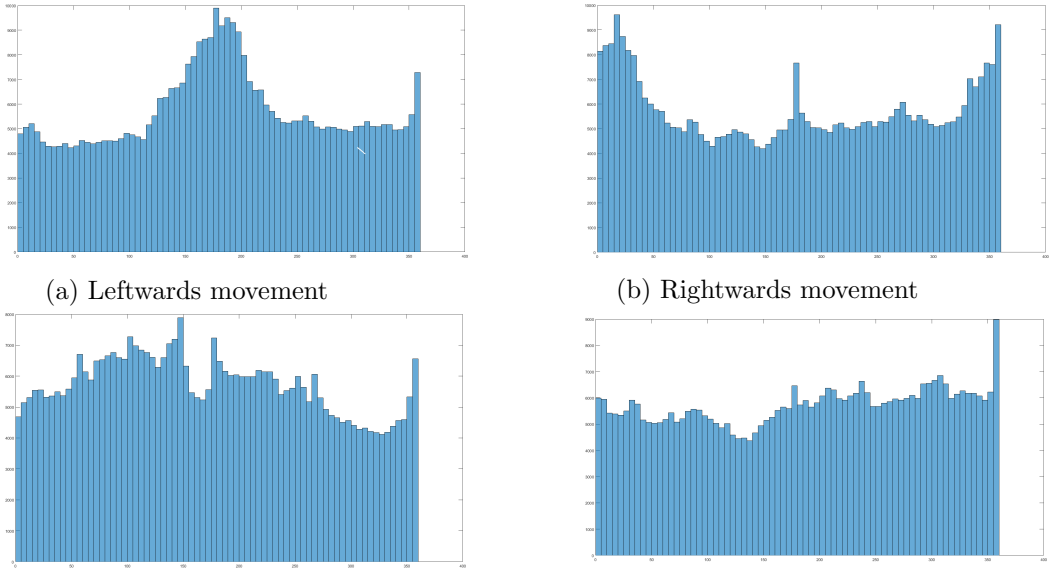


Figure 46: Optical flow vectors

It was chosen to start with the optical flow orientation, which was converted from radians into an angular value $\in [0° \ 360°]$ and arranged into histograms with 72 bins - where each bin represents an orientation range of 5°. This is displayed in figure 47, where the individual histogram bins is arranged along the x-axis and the histogram bin values is displayed along the y-axis.



(a) Leftwards movement

(b) Rightwards movement

(c) Opening

(d) Closing

Figure 47: Four cases of leaf movement

The histograms displayed in figure 47 reveals four different cases of movement, with figure 47a and figure 47b showing leftwards and rightwards movement of the compound leafs, and figure 47c and figure 47d shows the opening and closing of the compound leaf during the change of illumination regimes. The figures reveals that the leftwards and rightwards movement produces what looks like a Gaussian shaped histogram envelope, while the opening and closing of the compound leafs seem to consist of a more even spread among the orientations.

The optical flow magnitudes were then examined by first studying the mean magnitude of the optical flow. The mean magnitude is displayed in figure 48, and reveals that the majority of the large magnitudes is found, but not limited, to be around the shift of the *light* and *dark* regime and vice versa. The large magnitudes was during inspection found mostly to be associated with movements such as in the widespread and closing of the compound leafs, while the smaller magnitudes were in most cases originating from the slow and prudent leaf movement in the image plane.

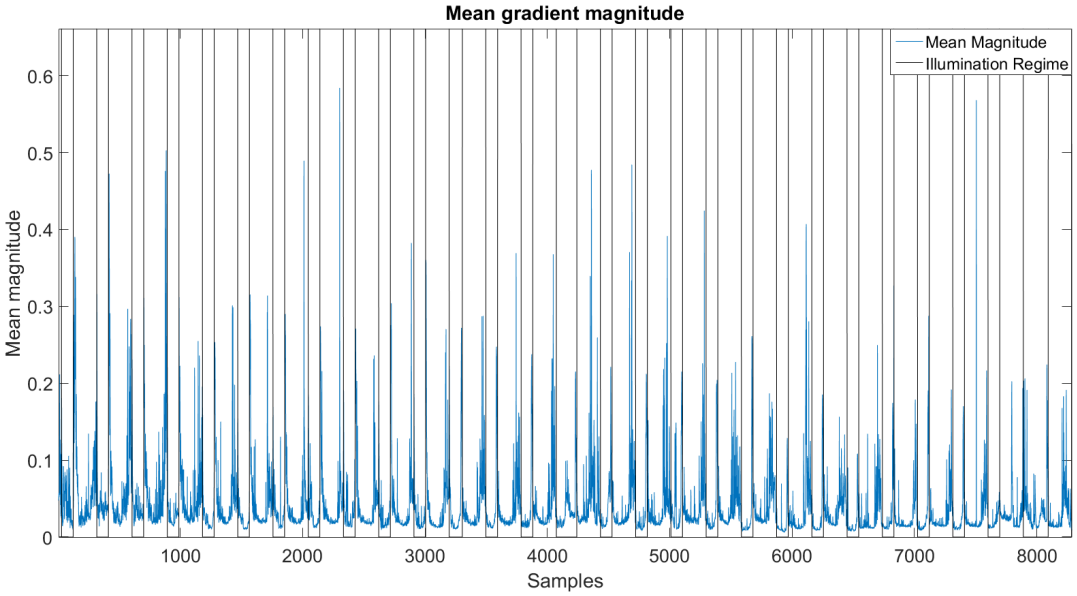


Figure 48: Mean gradient magnitude

One of the observations that motivated for further experimenting was the Gaussian like shape of the optical flow orientation histograms, which seemed to occur when the plant moved in a certain direction. It was therefore suggested that if the histograms bins within a limited area, e.g. 11 histogram bins, contained a chosen percentage of the total value of all the histogram

bins, e.g. 50 percent, then the peak value within the limited area would be the direction of the leafs movement. This suggestion can be exemplified by figure 49, where the 11 histogram bins inside the red rectangle contains 97 341 of the 440 000 histogram orientations. The rectangle is in this case encapsulating 22% of the total amount of orientations, with the peak value indicating a movement towards $\approx 170°$.

The fact that the Farneback method is a dense optical flow method meant that each individual image pixel contributed with values to the orientation histogram, which also included orientation information from for example the background pixels. This meant that the orientation histogram most likely contained excess information, which could (i) lead to an erroneous interpret of the orientation direction and (ii) made the amount of data to be processed substantially larger.

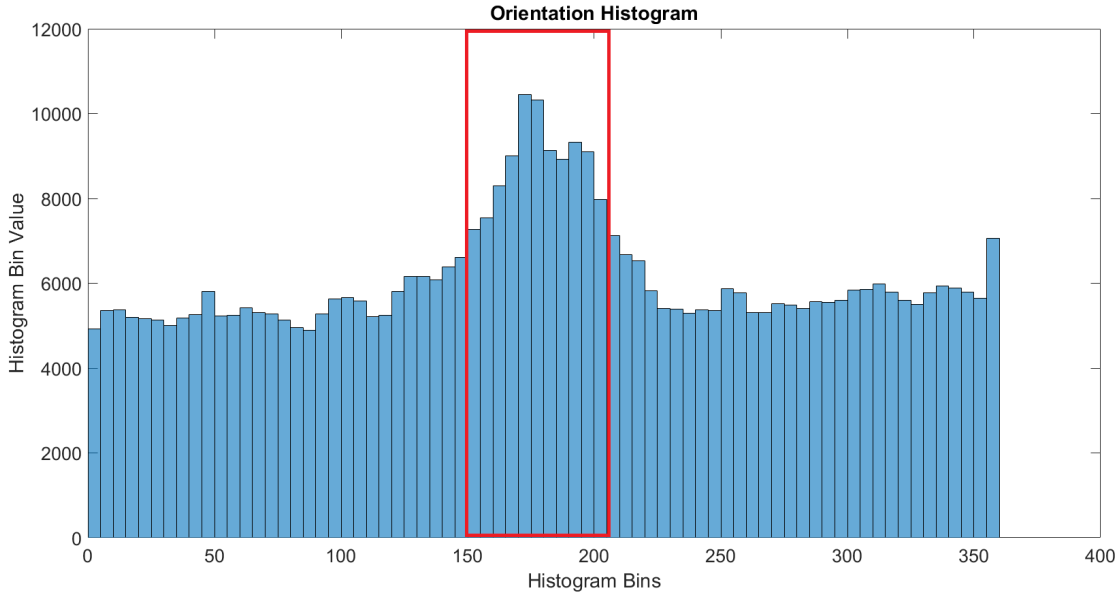


Figure 49: Motion estimation, direction estimation

It was therefore chosen to threshold each optical flow vector by using the optical flow vectors mean magnitude, where only vectors with a magnitude above 70% of the optical flow vectors mean magnitude was taken into account. The result is shown in figure 50, which is the optical flow vector orientations from the same two subsequent images as figure 49, with the red rectangle now containing 22192 of the 35074 histogram orientations - which is approximately 63% of the total amount of orientation values.

The result of thresholding the optical flow vector orientation values by the optical flow vectors mean magnitude was here seen to reduce the histogram bin values of the histogram bins which was assumed to not contribute with information about plants movement direction. The total percentage of orientation values inside the red rectangle is from figure 49 to figure 50 seen to change from 22% to 63%, which is an increase of 2.86 times.

The next step was to develop a method that would locate the orientation direction. The two main challenges that had to be solved was the possibility of multiple peaks, which can be seen in figure 50, as well as compensation for the discontinuity at the 0° and 360° orientation. This was solved by first performing a moving average filtering routine on the dataset, before the 20 first histogram bins were added to the end, extending the histogram to [0° 460°] - where the result of these operations is displayed in figure 51.

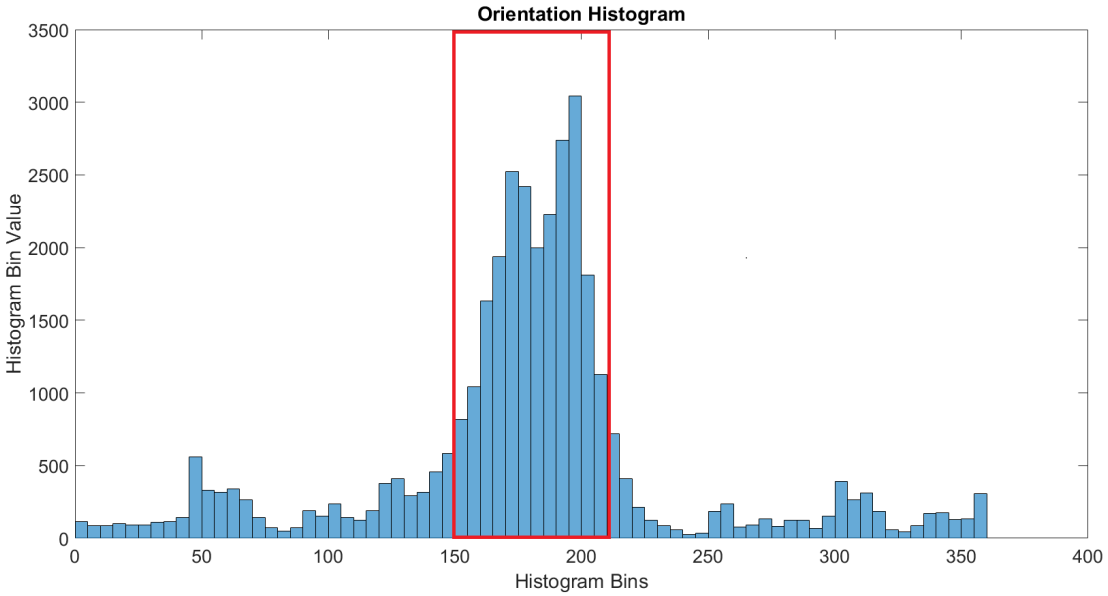


Figure 50: Motion estimation, improved direction estimation

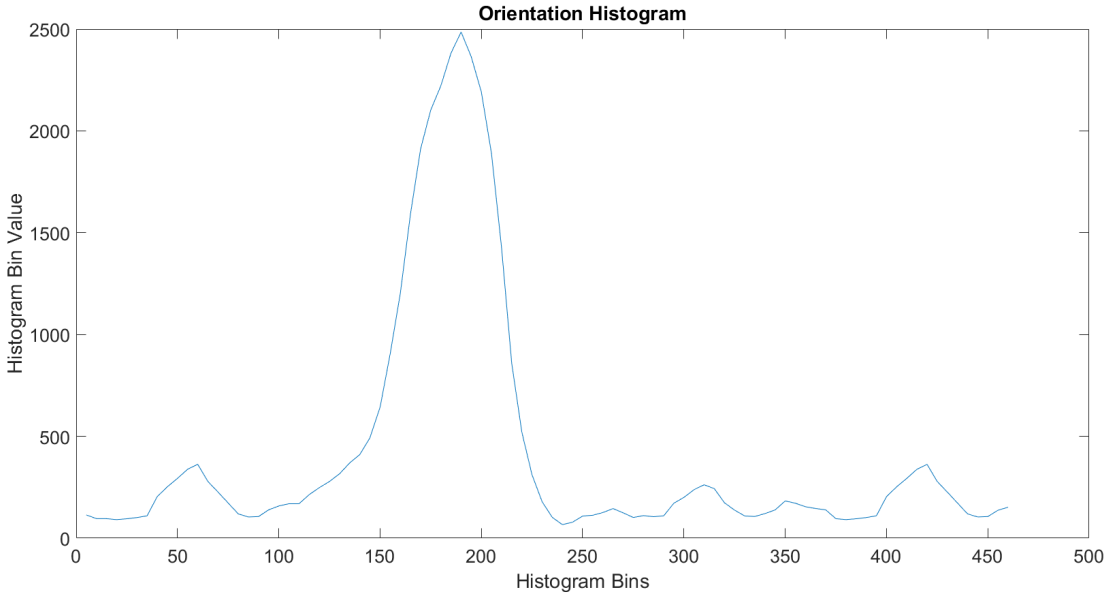


Figure 51: Motion estimation, smoothed and extended histogram

It was at this point chosen to run tests with a small number of images, where the purpose was to see if it was possible to quantify the observed movements. The sequence used is displayed in figure 52 and contains 24 images. It was seen from playing the sequence in Matlab that the plant wiggling slightly back and forth in image 1 to image 5, but with a prudent movement towards the right. The plant does then change its direction around image 7 where it starts to move towards the left side. This leftwards movement continues the remaining sequence.

Three different tests were run, where the parameter that differs them was the required percentage of orientation values in the rectangular window. This parameter was set to calculate the value of the histogram bins that is encapsulated by the rectangular window, and compares it with the total value of the entire histogram. The ratio between the two is then used to evaluate if there value inside the rectangular window is high enough to define a movement direction.

The first run was parametrized to have a required ratio of 0.5, with the results displayed in table 7.2. The high amount of NaN (Not a Number) indicated that no movement direction was found, as the ratio might have been too high. It was therefore chosen to run the same experiment with a ratio of 0.4.

(a) Img 1

(b) Img 2

(c) Img 3

(d) Img 4

(e) Img 5

(f) Img 6

(g) Img 7

(h) Img 8

(i) Img 9

(j) Img 10

(k) Img 11

(l) Img 12

(m) Img 13

(n) Img 14

(o) Img 15

(p) Img 16

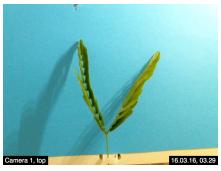
(q) Img 17

(r) Img 18

(s) Img 19

(t) Img 20

(u) Img 21



(v) Img 22

(w) Img 23

(x) Img 24

Figure 52: Plant movement sequence

Img 1	**Img 2**	**Img 3**	**Img 4**	**Img 5**	**Img 6**	**Img 7**	**Img 8**
NaN	NaN	NaN	NaN	NaN	NaN	210°	NaN
Img 9	**Img 10**	**Img 11**	**Img 12**	**Img 13**	**Img 14**	**Img 15**	**Img 16**
185°	NaN	205°	NaN	NaN	NaN	180°	185°
Img 17	**Img 18**	**Img 19**	**Img 20**	**Img 21**	**Img 22**	**Img 23**	**Img 24**
180°	NaN	NaN	NaN	35°	NaN	195°	185°

Table 7.2: Motion estimation, ratio 0.5

Running the experiment with a 0.4 ratio is displayed in table 7.3 and reveals a lower amount of NaN values. The increase of direction values when going from a ratio of 0.5 to 0.4 motivated to attempt one last run with a 0.3 ratio, to investigate if it was possible to obtain even more orientations. The result is displayed in table 7.4 and it is here seen that only 3 of 24 values is classified as NaN. It is possible to see from table 7.4 that the the orientation in image 1 and 4 indicates a rightwards movement towards $340° - 355°$, while most of the remaining images seem to consist of leftwards movements in the range $155° - 185°$. These readings are according to the observations that was seen during the inspection of the images in Matlab.

Img 1	**Img 2**	**Img 3**	**Img 4**	**Img 5**	**Img 6**	**Img 7**	**Img 8**
340°	NaN	NaN	NaN	NaN	230°	210°	160°
Img 9	**Img 10**	**Img 11**	**Img 12**	**Img 13**	**Img 14**	**Img 15**	**Img 16**
185°	NaN	205°	5°	NaN	NaN	180°	185°
Img 17	**Img 18**	**Img 19**	**Img 20**	**Img 21**	**Img 22**	**Img 23**	**Img 24**
180°	NaN	165°	155°	35°	NaN	195°	185°

Table 7.3: Motion estimation, ratio 0.4

Img 1	**Img 2**	**Img 3**	**Img 4**	**Img 5**	**Img 6**	**Img 7**	**Img 8**
340°	NaN	280°	355°	230°	230°	210°	160°
Img 9	**Img 10**	**Img 11**	**Img 12**	**Img 13**	**Img 14**	**Img 15**	**Img 16**
185°	180°	205°	5°	165°	160°	180°	185°
Img 17	**Img 18**	**Img 19**	**Img 20**	**Img 21**	**Img 22**	**Img 23**	**Img 24**
180°	NaN	160°	155°	35°	NaN	195°	185°

Table 7.4: Motion estimation, ratio 0.3

It was then chosen to run the motion estimation on the whole dataset of 8316 images, with the ratio parameter set at 0.3. The `NaN` values that was returned by the routine was removed, and the empty vector orientation

entries was filled by using the Matlab `interp1` function to interpolate values. The result is displayed in figure 53 and reveals a some what chaotic plot.

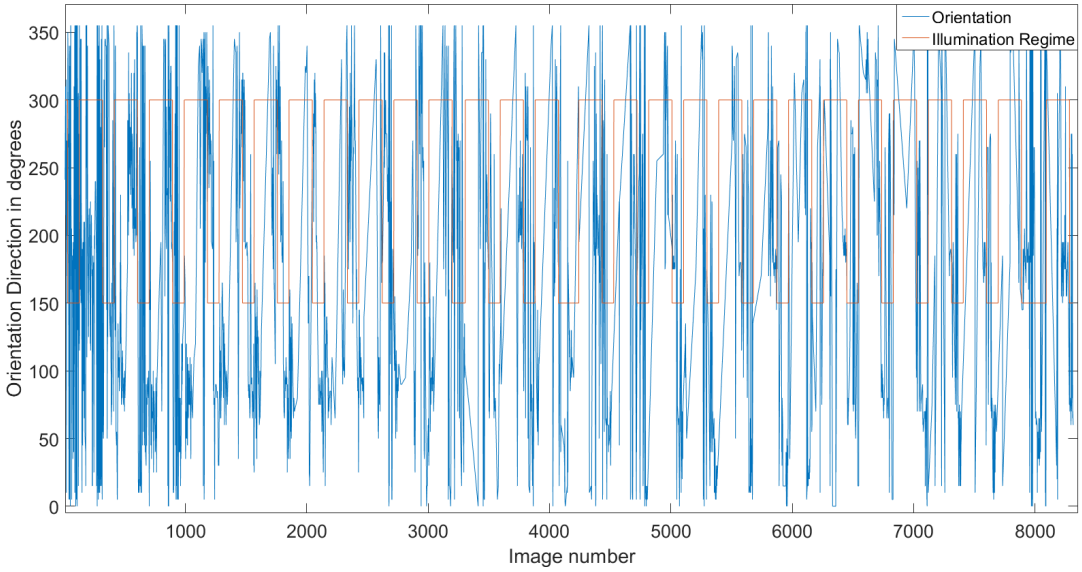


Figure 53: Orientation detection, whole dataset

It was then chosen to do a closer inspection of the plot in figure 53, displayed in figure 54, which shows what looks to be a pattern in the plants orientation. The orientation value is found to move towards $300° - 360°$ before the shift from the *dark* to the *light* illumination regime, and is seemingly switching its direction to towards $100° - 50°$ before the shift from *light* to *dark* illumination regime. A frame by frame inspection of the sequence around the illumination shift was performed to look for the origin of the pattern, where the movements were found to originate from what seems to be the *Mimosa Pudicas* primary pulvini. The observation revealed that the primary pulvini seemed to pull the compound leafs upwards before the shift to the *dark* illumination regime, which was interpret by the routine as a movement with an orientation of $300° - 360°$. The opposite was seen during the shift to the *light* regime, where the compound leafs were forced downwards, resulting in a movement with an orientation of $100° - 50°$.

The same frame by frame inspection was done in the evaluation of the noisy areas that is seen around the shift of illumination regimes in figure 54. It was not possible to find a specific reason to why the values is observed to fluctuate, but it is suggested to be related to the opening and closing of

the *Mimosa Pudicas* individual leaflets - which includes a large amount of optical flow vectors going in various directions.

The last observation that is seen from figure 54 is the straight sloped line which occurs around the middle of each *light* illumination regime. This is the result of the interpolation that was performed to fill in the empty values in the orientation vector, and indicates that the *Mimosa Pudicas* compound leaf does not seem to move that much during this period- which was also confirmed by playing and observing the time-lapse sequence.

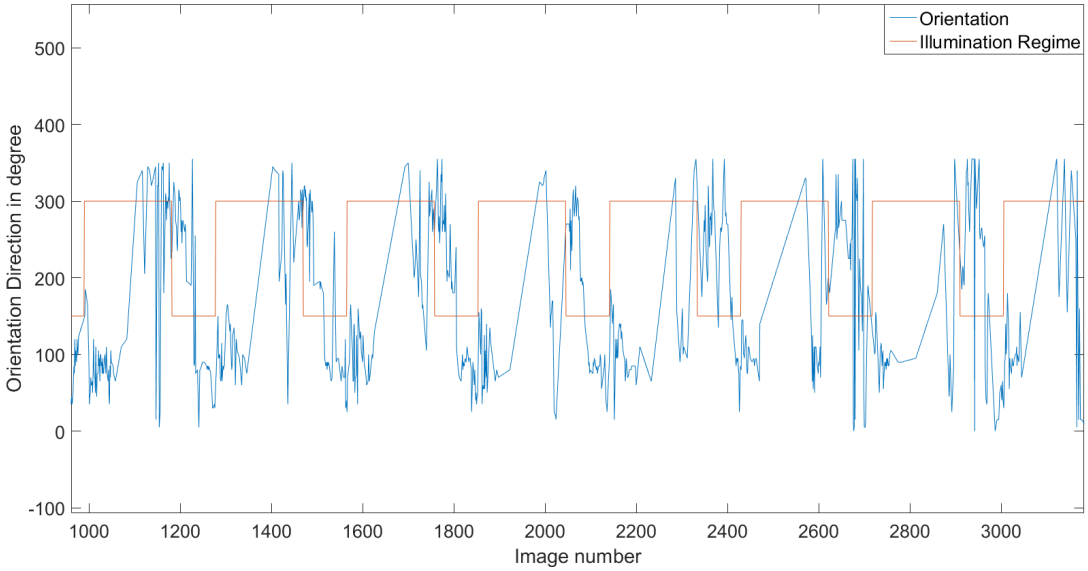


Figure 54: Orientation detection, partition of dataset

Chapter 8

Results and Discussion

The purpose of this chapter is to present and discuss the results from the image processing, black box modelling and motion estimation experiments. It has been chosen to present the results in three individual sections - where the results is obtained from the experiments performed in chapter 7.

8.1 Image Processing

The scope of the image processing part of the thesis has consisted of both choosing a suitable color representation, as well as finding a routine for the segmentation of the foreground, i.e. the *mimosa pudica*, from the background

The evaluation of color space involved the examination of the three colors models; RGB, HSV and CIELAB. Experiments in chapter 7.1 revealed that both the *hue* component from HSV and the b^* component from CIELAB provided a good foundation for the segmentation of the foreground object. This was based on the clear separation of the foreground and background pixels in both components histograms - as well as experiments showing that *hue* and b^* was among those components who were least susceptible to illumination changes. It was chosen to use the *hue* component in the segmentation routine, as the component was found to have a more intuitive representation.

The clear separation of the foreground and background pixels in the *hue* component histogram motivated for the use of a triangle thresholding segmentation routine. The segmentation routine was in section 7.2 tested on seven different images. Five of the images included different instances from the available time-lapse series of the *mimosa pudica*, while the two remaining images came from the Oxford 102 category dataset. The result of the

segmentation is displayed in table 8.1, where the segmentation performed by the triangle threshold method obtains a mean Dice similarity coefficient of 0.932 ± 0.0175 as an overall score on all the images. These results were compared against a segmentation performed by Otsu's method, whom achieved a Dice similarity coefficient of 0.6805 ± 0.2520 on the same dataset.

Method	Img 7165	Img 7958	Img 1	Img 2	Img 3	Img 4	Img 5
Triangle	0.8977	0.9354	0.9556	0.8965	0.9272	0.9396	0.9366
Otsu's	0.6325	0.2537	0.7741	0.8711	0.8727	0.4391	0.8951

Table 8.1: Dice similarity coefficients - Otsu's method and Triangle threshold

8.2 Black Box Modelling

The purpose of the system identification in the thesis was to estimate ARX, ARMAX and Box Jenkins models describing the pixel count and illumination regime relationship of the *mimosa pudica*. These models were then supposed to be used to evaluate the pixel count responses in a control theoretic viewpoint.

The ARX, ARMAX and Box Jenkins model were chosen with basis in the obtained AIC scores. The three models were then compared by their achieved NRMSE fit on several validation datasets, where a Box Jenkins model obtained the highest overall fit of 19.04%. The low NRMSE score was interpret as a clear indication that the estimated models failed to capture the system dynamics.

One of the challenges with estimating the black box models were that the swiftness of the system seemed to change when the illumination regime was altered from periodic to constant. It was therefore chosen to look at the possibilities of obtaining the system characteristics by fitting the oscillatory response to a second order transfer function. The result was the underdamped system described in equation 8.1.

$$H(s) = \frac{Y(s)}{X(s)} = \frac{2}{9000s^2 + 15s + 1} \tag{8.1}$$

The obtained second order transfer function were then validated against two datasets. This experiment revealed that there was a difference in both the (i) swiftness and the (ii) period of the second order system, compared with the systems used as validation. This was an interesting observations, as

one of the validation datasets were captured from the same *mimosa pudica* plant as the test data used to estimate the transfer function. This could imply that the periodic *light* and *dark* regime works as a disturbance on the plant, forcing the widespread and closing of leafs, while the oscillations seen in the *constant* regime is suggested to be the plants *free-running* period, with a period time of 30 hours.

It should not go unmentioned that there are several factors that could have affected the estimation of good black box models. One of the problems is assumed to be related to the tertiary and/or secondary pulvinus downward angling of the compound leafs. This was seen to result in a temporary reduction in the leaf surface area, which was observed in the pixel count plot as a saddle/slope shaped curve. This is likely to affect the possibility of obtaining a good and correct description of the system dynamics.

The second challange with the test data used to estimate the models in the experiments was that it was either periodic or constant, with the periodicity only changing a few times during each experiment. The stimulus used to provoke the *mimosa pudica* response should ideally have parts where the illumination regime were a pseudo random binary sequence [14], i.e. the changes in the illumination regime should have had both random changes and durations. This would most likely make it possible to capture the dynamics of the system in a better manner.

The failure of modelling the response motived for a further investigation on the topic of modelling responses in plants. The best description was found in the recently published *How to stop a biological clock: Point of singularity* [6] by W.Engelmann and K.H.Witte, whom referred to the work of Johnsson and Karlsson on feedback models for biological rhythms [17, 19]. Their work dealt with oscillations in the Kalanchoe flower, where a 24 hours rhythm is modelled by a non-linear feedback model. The success of modelling the Kalanchoe flower with a more complex model could imply that a linear *black box* model is not adequate for the task of modelling the *mimosa pudicas* response to changes in its illuminative conditions.

8.3 Motion Estimation

The purpose of the motion estimation performed in the thesis was to look at the possibility of quantifying a metric whom could describe the observed movements of the *mimosa pudica* in the image plane. It was chosen to cal-

culate the optical flow by using of the method of Farneback, described in chapter 5, which also comes as a fully implemented Matlab function. The two variables that was chosen for examination was the optical flow vectors magnitude and orientation.

Evaluation of the optical flow orientation revealed that the movement of the compound leafs in a certain direction create an almost Gaussian like curve in the optical flow orientation histogram plot. The examination of the optical flow magnitudes indicated that the majority of the large magnitude were linked to the opening and closing of the *mimosa pudica* leafs, while the smaller magnitudes in most cases originated from the slow and prudent compound leaf movement.

A ratio based window search method was developed in chapter 7.4, that calculated the ratio between the orientation values of the histogram bins inside a moving window and the total amount of orientation values of all the histogram bins. Three test were performed where the best result was evaluated to be when the search window contained more than 30% of the total orientation values in the histogram. This was based on a test sequence of 24 images, where the method seemingly classified the same orientation directions as the ones that was seen during the manual inspection of the dataset.

The routine was also tested on the whole dataset of 8316 images, where a pattern in orientation direction around the shift of the illumination regime was suggested to be linked to the primary pulvinus. The observations could imply that the *Mimosa Pudicas* response to the illumination shift starts before the actual shift is taking place.

One of the challenges by estimating plant movement in this thesis was the large amount of compound leafs in each dataset. The method evaluates the overall movement in the image plane, which mean that smaller movements performed by the individual compound leafs could be missed. The second problem of examining the orientation with several compound leafs in the image frame is that movements can potentially repeal each other by moving in opposite orientation, and thus evening out the histogram curve.

A suggested solution to the mentioned challenges could be to reduce the amount/size of the compound leafs in the image plane. An example of this can be seen in the experiment performed in the recent published article *Tracking Rhythms in Plants, an automated leaf movement analysis program for circadian period estimation* [12], where vertical movement was successfully

estimated. A second solution could be to implement tracking of the individual leafs, where several methods for tracking leafs are suggested in the article *Leaf segmentation in plant phenotyping: a collation study* [31]. This could potentially allow the estimation of optical flow in time-lapse series with multiple compound leafs.

Chapter 9

Conclusion

The intention of the thesis as stated in the introduction was to (i) improve the image processing method used in the preliminary work, (ii) examine the *mimosa pudicas* responses to change in its illuminative conditions and evaluate the possibility of estimate mathematical models that could describe the responses, and (iii) look at the possibility quantify the observed movement in the time-lapse series.

The preliminary work that was performed at IDE was executed with a segmentation routine that required the users intervention. Several color models have been thoroughly evaluated and a new automatic non-parametric image processing routine has been developed - obtaining a DICE similarity score of 0.932 ± 0.0175 on test data. It is therefore suggested that the new method replace the preliminary routine in the further work.

It was not managed to estimate any black box models that described the input/output relationship of the *mimosa pudica*. The experiments did reveal that both the systems (i) period and (ii) swiftness changed during the underdamped response in the *constant* illumination regime, indicating that the *light* and *dark* regime could work as disturbances on the plant. The period duration of the oscillation in the *constant* regime was also found to be 30 hours, which deviates from the approximately 24 hour period that is suggested in the literature. This is only based on experiments with one plant, and additional experiments is therefore necessary to support this claim.

A method estimating of the *mimosa pudicas* movement orientation has been developed. The method was seen to obtain seemingly good results during test with a short image sequence. Experiments did also reveal what was suggested to be a pattern of movement in the *Mimosa Pudicas* primary pul-

vini, where the shift of the illumination regimes seemed to induce movements in certain directions.

Both the case of changing system dynamics when going from periodic to constant illumination regime, as well as the repetitious patterns seen prior of the illumination changes, could indicate that there is a complex structure behind the observed responses of the *Mimosa Pudica*. This structure has not been found during this thesis, but preliminary methods have been improved and new routines have been developed. These methods can, and will hopefully, be used in the further examination of the responses of the *Mimosa Pudica*.

Bibliography

[1] Hirotugu Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, 1974.

[2] Howard Anton. *Elementary linear algebra*. John Wiley & Sons, 2010.

[3] John L Barron and Neil A Thacker. Tutorial: Computing 2d and 3d optical flow. *Imaging Science and Biomedical Engineering Division, Medical School, University of Manchester*, 2005.

[4] Pierre Borne, Dumitru Popescu, Florin Gheorghe Filip, and Dan Stefanoiu. *Optimization in Engineering Sciences: Exact Methods*. John Wiley & Sons, 2013.

[5] Kenneth P Burnham and David R Anderson. *Model selection and multimodel inference: a practical information-theoretic approach*. Springer Science & Business Media, 2003.

[6] Wolfgang Engelmann and Karl-Heinz Witte. How to stop a biological clock: Point of singularity. 2016.

[7] Mark D Fairchild. *Color appearance models*. John Wiley & Sons, 2013.

[8] Gunnar Farnebäck. Polynomial expansion for orientation and motion estimation. 2002.

[9] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image analysis*, pages 363–370. Springer, 2003.

[10] David A Forsyth and Jean Ponce. A modern approach. *Computer Vision: A Modern Approach*, pages 88–101, 2003.

[11] Jörg Fromm and Walter Eschrich. Transport processes in stimulated and non-stimulated leaves of mimosa pudica. *Trees*, 2(1):7–17, 1988.

[12] Kathleen Greenham, Ping Lou, Sara E Remsen, Hany Farid, and C Robertson McClung. Trip: Tracking rhythms in plants, an automated leaf movement analysis program for circadian period estimation. *Plant methods*, 11(1):1, 2015.

[13] Finn Haugen. Praktisk reguleringsteknikk. *Tapir forlag*, 2003.

[14] Finn Haugen. *Dynamiske Systemer: modellering, analyse og simulering.* Akademika Forlag, 2012.

[15] Berthold K Horn and Brian G Schunck. Determining optical flow. In *1981 Technical symposium east*, pages 319–331. International Society for Optics and Photonics, 1981.

[16] George H Joblove and Donald Greenberg. Color spaces for computer graphics. In *ACM siggraph computer graphics*, volume 12, pages 20–25. ACM, 1978.

[17] A Johnsson and HG Karlsson. A feedback model for biological rhythms: I. mathematical description and basic properties of the model. *Journal of theoretical Biology*, 36(1):153–174, 1972.

[18] Jer-Nan Juang and Minh Q Phan. *Identification and control of mechanical systems.* Cambridge University Press, 2001.

[19] HG Karlsson and A Johnsson. A feedback model for biological rhythms: Ii. comparisons with experimental results, especially on the petal rhythm of kalanchoë. *Journal of theoretical biology*, 36(1):175–194, 1972.

[20] Karel J Keesman. *System identification: an introduction.* Springer Science & Business Media, 2011.

[21] Frederick AA Kingdom. Lightness, brightness and transparency: A quarter century of new ideas, captivating demonstrations and unrelenting controversy. *Vision Research*, 51(7):652–673, 2011.

[22] Lennart Ljung. System identification: theory for the user. *Englewood Cliffs*, 1987.

[23] Lennart Ljung. Black-box models from input-output measurements. In *Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE*, volume 1, pages 138–146. IEEE, 2001.

[24] Stefano Mancuso and Sergey Shabala. *Rhythms in plants*. Springer, 2007.

[25] Oge Marques. *Practical image and video processing using MATLAB*. John Wiley & Sons, 2011.

[26] C Robertson McClung. Plant circadian rhythms. *The Plant Cell*, 18(4):792–803, 2006.

[27] National Instruments. *LabVIEW 2014 Control Design and Simulation Module Help*.

[28] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[29] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[30] Dr B. Panneton. Gray image thresholding using the triangle method. `http://www.mathworks.com/matlabcentral/fileexchange/28047-gray-image-thresholding-using-the-triangle-method/content/triangle_th.m`. Accesses: 2016-04-05.

[31] Hanno Scharr, Massimo Minervini, Andrew P French, Christian Klukas, David M Kramer, Xiaoming Liu, Imanol Luengo, Jean-Michel Pape, Gerrit Polder, Danijela Vukadinovic, et al. Leaf segmentation in plant phenotyping: a collation study. *Machine vision and applications*, pages 1–22, 2015.

[32] Dale E Seborg, Duncan A Mellichamp, Thomas F Edgar, and Francis J Doyle III. *Process dynamics and control*. John Wiley & Sons, 2010.

[33] Alvy Ray Smith. Color gamut transform pairs. *ACM Siggraph Computer Graphics*, 12(3):12–19, 1978.

[34] David E Somers. The physiology and molecular bases of the plant circadian clock. *Plant Physiology*, 121(1):9–20, 1999.

[35] Maureen Stone. *A field guide to digital color*. CRC Press, 2013.

[36] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[37] Paul Hermann Wilhelm Taubert. Leguminosae. 1887.

[38] Minoru Taya. Bio-inspired design of intelligent materials. In *Smart structures and materials*, pages 54–65. International Society for Optics and Photonics, 2003.

[39] Xiangyang Xu, Shengzhou Xu, Lianghai Jin, and Enmin Song. Characteristic analysis of otsu threshold and its applications. *Pattern recognition letters*, 32(7):956–961, 2011.

[40] GW Zack, WE Rogers, and SA Latt. Automatic measurement of sister chromatid exchange frequency. *Journal of Histochemistry & Cytochemistry*, 25(7):741–753, 1977.

Appendix

Matlab Code

Main script

```
% Vegard Brattland, University of Stavanger, 2016
% Thesis: Imaga Processing and Analysis of Leaf Movement...
% in Mimosa Pudica
% Main script in Matlab code

clear all -except Keep

% Creating an optical flow object
opticFlow = opticalFlowFarneback('NeighborhoodSize',15,...
'FilterSize',25);

% Folder with the image data
mappe = 'C:/Users/Vegard/Documents/MATLAB/Bilder';

% Image folder and camera direction
bildeDato = '/date_20160315';
kameraRetning = {'/camera1_top','/camera2_buttom'};

% Calling function to generate structure with folders and ...
    imagenames
mappeStruktur = Mapper(mappe, bildeDato, kameraRetning);

% Loading the dark, light and constant information
run(sprintf('%s%s/LightDarkRegime/%s', ...
    mappe,bildeDato,'LightDark.m'));

% Calling the BildeInformasjon function.
[timeLapsePeriode OffsetForsteBilde] = ...
    BildeInformasjon(mappeStruktur);

% Making a vector which indicates day or night.
```

```
29 DLCvect = ...
       LightDarkVector(DarkLightConstant,timeLapsePeriode,...
30 OffsetForsteBilde);
31
32 % Making plant direction vector
33 direction = [];
34
35 % Looping to segment images and estimation orientation
36 for i=1:numel(mappeStruktur.infoOriginal)
37
38     % Reading time-lapse images
39     bilde = ...
           imread(sprintf('%s/%s',mappeStruktur.mappeOriginal,...
40             mappeStruktur.infoOriginal(i).name));
41
42     % Cropping image
43     bilde = bilde(1:end-50,1:end,:);
44
45     % Segmentation routine
46     [binBilde picture hue] = FargeTilSortHvitt(bilde);
47
48     % Calculating the optical flow
49     flow = estimateFlow(opticFlow,hue);
50
51     % Count the amount of binary ones in the images
52     antallHvitePiksler(i) = sum(sum(binBilde));
53
54     % Arranging flow orientations and magnitude in variables
55     flowOrientation = flow.Orientation;
56     flowMagnitude = flow.Magnitude;
57
58     % Calculating mean optical flow magnitude
59     meanMag = mean(mean(flowMagnitude));
60
61     % Only keeping larger magnitudes
62     flowMagnitude = flowMagnitude > (meanMag*10);
63
64     % Masking out the low magnitudes
65     flowOrientationMask = flow.Orientation.*flowMagnitude;
66
67     % Optical flow orientation and magnitude assigned to ...
           new variables.
68     testvarOri = flow.Orientation;
69     testvarMag = flow.Magnitude;
70
71     % Converting orientation from radians and rounding answer
72     degreeMat = round(rad2deg(testvarOri));
73
74     % Converting orientation from [-180 180] to [0 360]
```

```
75      lessThanZero = (testvarOri < 0)*360;
76      degreeMat = degreeMat + lessThanZero;
77
78
79      % Setting zero magnitudes equal NaN.
80      for k=1:800
81          for j=1:550
82              if flowMagnitude(j,k) == 0
83                  degreeMat(j,k) = NaN;
84              end
85          end
86      end
87
88      % Making a orientation histogram
89      histo = histogram(degreeMat,72);
90
91      % Total amount of orientation values
92      totOrient = sum(histo.Values);
93
94      % Arranging the histogram values to variable
95      histoValues = histo.Values;
96
97      % Extends orientation histogram
98      histoValues = [histoValues histoValues(1:20)];
99
100     % Smoothing the orientation histogram
101     histoSmooth = smooth(histoValues,'moving');
102
103     % Window size and larges window value variables.
104     winSize = 11;
105     winVal = 0;
106
107     % Finding the window with highest values
108     for h=1:length(histoSmooth)-winSize
109         calcWinVal = sum(histoSmooth(h:1:(h+winSize-1)));
110
111         % Saving highest amount and window start
112         if calcWinVal > winVal;
113             winVal = calcWinVal;
114             winStart = h;
115         end
116     end
117
118     % Calculating percentage
119     percentInWindow = winVal/sum(histo.Values);
120
121     % If above 30%, find orientation
122     winPerThres = 0.3;
123
```

```
124     if percentInWindow > winPerThres
125         [peakValue location] = ...
               findpeaks(histoSmooth(winStart:1:...
126         (winStart+winSize-1)),'MinPeakDistance',9);
127 
128         if isempty(location) == false
129 
130             % Finds the peak location
131             peakLocation = winStart + location;
132 
133             % 72 histogram bins equals 5 degree per bin
134             direction(i) = peakLocation*5;
135 
136             % Changing the directin to range (0 - 359)
137             if direction(i) > 359
138                 direction(i) = direction(i) - 360;
139             end
140         else
141             direction(i) = NaN;
142         end
143 
144     else
145         % If not above threshold, direction equals 0
146         direction(i) = NaN;
147     end
148 
149 % Iteration count
150 i
151 
152 
153 end
154 
155 % Plotting the result
156 Plotting(antallHvitePiksler,DLCvect)
```

Function : Mapper

```
% Vegard Brattland, University of Stavanger, 2016
% Thesis: Imaga Processing and Analysis of Leaf Movement...
% in Mimosa Pudica
% ************************************************************
% This function returns a structure with the necessary folders
% and image information
% ************************************************************


function [mappeStruktur] = Mapper(mappe, bildeDato, ...
    kameraRetning)

% Making path to folders with original and cropped images
mappeOriginalBilde = ...
    [mappe,bildeDato,kameraRetning{1,1},'/images'];
mappeBeskjartBilde = ...
    [mappe,bildeDato,kameraRetning{1,1},'/cropped'];

mappeOriginalBildeTo = ...
    [mappe,bildeDato,kameraRetning{1,2},'/images'];
mappeBeskjartBildeTo = ...
    [mappe,bildeDato,kameraRetning{1,2},'/cropped'];

% Original and cropped image names
originaleBilderEn = dir([mappeOriginalBilde,'/2016*.jpg']);
beskjarteBilderEn = dir([mappeBeskjartBilde,'/crop*.jpg']);

originaleBilderTo = dir([mappeOriginalBilde,'/2016*.jpg']);
beskjarteBilderTo = dir([mappeBeskjartBilde,'/crop*.jpg']);

% Arranging structure with folder information
mappeStruktur = struct('mappeOriginal',mappeOriginalBilde,...
    'mappeCropped',mappeBeskjartBilde,...
    'infoOriginal',originaleBilderEn,...
    'infoCropped',beskjarteBilderEn,...
    'mappeOriginal_2',mappeOriginalBildeTo,...
    'mappeCropped_2',mappeBeskjartBildeTo,...
    'infoOriginal_2',originaleBilderTo,...
    'infoCropped_2',beskjarteBilderTo);

end
```

Function : BildeInformasjon

```
% Vegard Brattland, University of Stavanger, 2016
% Thesis: Imaga Processing and Analysis of Leaf Movement...
% in Mimosa Pudica

% This function is inspired from a Matlab file recieved from
% Prof. Tormod Drengstigs at the University of Stavanger

% ********************************************************
% This function returns information used in plotting
% ********************************************************

function [timeLapsePeriodeMin,OffsetForsteBilde] = ...
    BildeInformasjon(mappeStruktur)

% Calculate timeinterval for time-lapse series
[sorterDato, sortidx] = ...
    sort([mappeStruktur.infoOriginal.datenum]);

% Making structure with image information
originaleBilder = mappeStruktur.infoOriginal(sortidx);

% Finding date to first and last image in name column
forsteBilde=originaleBilder(1).name;
sisteBilde=originaleBilder(end).name;

% Finds year, month, date, time, minute and second on ...
    laste image
yyyy = eval(sisteBilde(1:4));
mm = eval(sisteBilde(6:7));
dd = eval(sisteBilde(9:10));
HH = eval(sisteBilde(12:13));
MM = eval(sisteBilde(15:16));
SS = eval(sisteBilde(18:19));
datoSisteBilde = datenum(yyyy,mm,dd,HH,MM,SS);

% Finds year, month, date, time, minute and second on ...
    first image
yyyy = eval(forsteBilde(1:4));
mm = eval(forsteBilde(6:7));
dd = eval(forsteBilde(9:10));
HH = eval(forsteBilde(12:13));
MM = eval(forsteBilde(15:16));
SS = eval(forsteBilde(18:19));
datoForsteBilde = datenum(yyyy,mm,dd,HH,MM,SS);
midnattForsteBilde = datenum(yyyy,mm,dd,00,00,00);
```

```
42
43 % Differance between first midnight and first image
44 OffsetForsteBilde = (datoForsteBilde - midnattForsteBilde)*24;
45
46 % Time-lapse periode in days
47 timeLapsePeriode = datoSisteBilde - datoForsteBilde;
48
49 % Time lapse period in minutes
50 timeLapsePeriodeMin = timeLapsePeriode*24*60;
51
52 end
```

Function : LightDarkVector

```
1  % Vegard Brattland, University of Stavanger, 2016
2  % Thesis: Imaga Processing and Analysis of Leaf Movement...
3  % in Mimosa Pudica
4
5  % ************************************************************
6  % This function returns a Dark, Light and...
7  % Constant illumination state vector.
8  % ************************************************************
9
10
11
12 function [DLCindicator] = ...
       LightDarkVector(DarkLightConstant,...
13 timeLapsePeriode,OffsetForsteBilde)
14
15 % Transposing and making colum vector with ...
       DarkLightConstant elements
16 DLCvect = DarkLightConstant';
17 DLCvect = DLCvect(:);
18
19 % Making a reference matrix where 1 = night, 2 =day and 3 ...
       = constant
20 DLCref = [1 2 3];
21
22 % Padding making equal shape as DLCvect
23 DLCref = padarray(DLCref,[(length(DLCvect)/3)-1 ...
       0],'replicate','pre')';
24 DLCref = DLCref(:);
25
26 % DLC Indicator where 1 = night, 2 = day, 3 = constant
27 DLCindicator = [];
```

```
28
29 for i=1:length(DLCvect)-1
30     % Length of current light regime in minutes
31     currDLClength = round(DLCvect(i)*60);
32
33     % Increasing length of regime vector if the new light ...
           regime is
34     % longer than 1 minute
35     if currDLClength > 0
36        DLCindicator = ...
              [DLCindicator;ones(currDLClength,1)*DLCref(i)];
37     end
38 end
39 % Removing the samples between midnight and first image. ...
       Offset is
40 % in the unit [hours] and is converted to minutes.
41 DLCindicator = DLCindicator(OffsetForsteBilde*60:end);
42
43 % The images is taken every 3 minutes. Downsampling, ...
       keeping every third
44 % sample
45 DLCindicator = downsample(DLCindicator,3);
46
47 end
```

Function : FargeTilSortHvitt

```
1  % Vegard Brattland, University of Stavanger, 2016
2  % Thesis: Imaga Processing and Analysis of Leaf Movement...
3  % in Mimosa Pudica
4
5  % ************************************************************
6  % This function returns a segmented time-lapse image, ...
7  % the hue component and the image that was used as input
8  % ************************************************************
9
10
11 function [hueSegmented ,picture, hue] = ...
       FargeTilSortHvitt(picture)
12
13 % Converting from RGB to HSV colorspace
14 hsv = rgb2hsv(picture);
15
16 % Seperating the hue component.
17 hue = hsv(:,:,1);
```

```
18
19 % Assigning hue component to histogram
20 imageHistogram = imhist(hue,100);
21
22 % Smoothing the data
23 imageHistSmooth = smooth(imageHistogram,'moving');
24
25 % Using triangle method to find threshold
26 triThreshold = triangle_th(imageHistSmooth,100);
27
28 % Making a mask
29 hueSegmented = imcomplement(im2bw(hue,triThreshold));
30
31
32 end
```

Function : triangle_th [30]

```
1  function [level]=triangle_th(lehisto,num_bins)
2  % Triangle algorithm
3  % This technique is due to Zack (Zack GW,   Rogers WE,
4  % Latt SA (1977), "Automatic measurement of sister
5  % chromatid exchange frequency", J. Histochem.
6  % Cytochem. 25 (7): 741  5 3 , )A line is constructed between
7  % the maximum of the histogram at (b) and the lowest (or
8  % highest depending on context) value (a) in the histogram.
9  % The distance L normal to the line and between the line
10 % and the histogram h[b] is computed for all values from
11 % a to b. The level where the distance between the
12 % histogram and the line is maximal is the threshold value
13 % (level). This technique is particularly effective
14 % when the object pixels produce a weak peak in the histogram.
15
16 % Use Triangle approach to compute threshold (level)
17 % based on a 1D histogram (lehisto). num_bins levels
18 %  gray image.
19
20 % INPUTS
21 % lehisto :   histogram of the gray level image
22 % num_bins:   number of bins (e.g. gray levels)
23 % OUTPUT
24 % level   :   threshold value in the range [0 1];
25 %
26 % Dr B. Panneton, June, 2010
27 % Agriculture and Agri-Food Canada
```

```
% St-Jean-sur-Richelieu, Qc, Canad
% bernard.panneton@agr.gc.ca


%   Find maximum of histogram and its location along the x ... axis
    [h,xmax]=max(lehisto);
    xmax=round(mean(xmax));   %can have more than a single ... value!
    h=lehisto(xmax);

%   Find location of first and last non-zero values.
%   Values<h/10000 are considered zeros.
    indi=find(lehisto>h/1000000);
    fnz=indi(1);
    lnz=indi(end);

%   Pick side as side with longer tail. Assume one tail is ... longer.
    lspan=xmax-fnz;
    rspan=lnz-xmax;
    if rspan>lspan  % then flip lehisto
        lehisto=fliplr(lehisto');
        a=num_bins-lnz+1;
        b=num_bins-xmax+1;
        isflip=1;
    else
        lehisto=lehisto';
        isflip=0;
        a=fnz;
        b=xmax;
    end

%   Compute parameters of the straight line from first ... non-zero to peak
%   To simplify, shift x axis by a (bin number axis)
    m=h/(b-a);

%   Compute distances
    x1=0:(b-a);
    y1=lehisto(x1+a);
    beta=y1+x1/m;
    x2=beta/(m+1/m);
    y2=m*x2;
    L=((y2-y1).^2+(x2-x1).^2).^0.5;

%   Obtain threshold as the location of maximum L.
    level=find(max(L)==L);
    level=(a-1)+mean(level);
```

```
73
74 %   Flip back if necessary
75     if isflip
76         level=num_bins-level+1;
77     end
78
79     level=level/num_bins;
```

Function : Plotting

```
1  % Vegard Brattland, University of Stavanger, 2016
2  % Thesis: Imaga Processing and Analysis of Leaf Movement...
3  % in Mimosa Pudica
4
5  % ************************************************************
6  % This function plots the white pixels with a color bar ...
       indicating
7  % night or day
8  % ************************************************************
9
10
11
12
13 function [] = Plotting(antallHvitePiksler,DLCvect)
14
15 % The derivative will indicate a change in Dark Light ...
       Constant periode.
16 % This means that there is a shift from day to night, or ...
       night to day.
17 changeInDLC = abs(diff(DLCvect));
18
19 % Making a matrix of color string. Yellow indicates day, ...
       black indicates
20 % night and red indicates constant.
21 DLCcolorMatrix = {[0 0 0],[1 1 0],[1 0 0]};
22
23 % Finding locations of the where there is a change in Dark ...
       Light Constant
24 % periode.
25 [DLCVal DLCChangeIndex] = findpeaks(changeInDLC);
26
27 % Shifting all index one step forward so the index is ...
       located at the
28 % start of the new dark light periode.
29 DLCChangeIndex = DLCChangeIndex + 1;
```

```
30
31 % Finding the length of the different Dark Light Constant ...
      periodes.
32 DLCLength = diff(DLCChangeIndex);
33
34 % A change in periode occurs when the previous periodeis ...
      shorter or longer.
35 % A periode change can only occur from the third DLC shift ...
      because the
36 % first DLC periode might be shorter during start-up
37
38 % Empty matrix with index of locations with a change of ...
      periode
39 DLCregimeChange = [];
40
41 % Checking if there is a change in periode length. ...
      Comparing night with
42 % night and day with day.
43 for i=3:length(DLCLength)
44     % Comparing length of night periode with night, and ...
         day with day.
45     if DLCLength(i) ~= DLCLength(i-2)
46         DLCregimeChange = [DLCregimeChange;DLCChangeIndex(i)];
47     end
48 end
49
50 % Downsample by 2 to prevent shifing until next change in ...
      periode
51 DLCregimeChange = downsample(DLCregimeChange,2);
52
53 % Making a figure for the white pixel plot
54 figure(1)
55 plot(antallHvitePiksler)
56 title('White pixel count')
57 ylabel('Amount of white pixels in image')
58 xlabel('Samples')
59
60 % Finding the range of the y-axis for DLC rectangle insertion
61 yaxislimits = get(gca,'ylim');
62
63 % ************************************************
64 % This loop is windowing each DLC periode. It is plotted
65 % colored rectangles beneath each window indicating the length
66 % of the periode and if it is dark, light or constant. A
67 % change in the periode length will result in a downward
68 % shift in the indicators
69
70 % WINDOW: The first rectangle that is plotted is the
71 % window. One window contains one DLC periode.
```

```
% Window is starting at the lower y-axis limit and
% streches to the upper limit. The bounds of the
% windows is defined by the length of the DLC periodes.

% INDICATION BAR: The second rectangle is the indication
% bar. It is located beneath the window. It is colored
% yellow for day, black for night and red for constant.
% A change in the DLC periode length will be indicated
% by a downward shift of the indicator bar. The
% indication bar has a height of 2000 pixels.

% TEXT: Adding text to the indication bar. A IF condition
% chooses white text if night and black text else. The
% text is placed in the middle of the indication bar.

% Defining the height of the window and indication bar.
windowHeight = yaxislimits(2) - yaxislimits(1);
indBarHeight = 2000;

% Y-axis offset for the plotting of indication bars.
yaxisOffset = 2000;

for i=1:length(DLCChangeIndex)-1

    % X-Coordinate for center of indication bar
    indBarCenterX = DLCChangeIndex(i)+(DLCLength(i)/2);
    % Length of periode in hours
    indBarTime = int2str((DLCLength(i)*5)/60);
    % Indication bar color
    indBarColor = DLCcolorMatrix{DLCvect(DLCChangeIndex(i))};

    % Plotting of the first window and indication bar.
    if i == 1
       % Plotting Window
        rectangle('Position',[0,yaxislimits(1),...
            DLCChangeIndex(1),windowHeight]);

        % Plotting Indication bar.
        rectangle('Position',[0,yaxislimits(1)-yaxisOffset,...
        DLCChangeIndex(1),indBarHeight],'FaceColor',...
        DLCcolorMatrix{DLCvect(1)})
    end

    % Checking if there is a change in DLC periode.
    if ismember(DLCChangeIndex(i),DLCregimeChange) == true

        % Calculating changing index for the faint color bars.
        periodeChangeIndex = [DLCChangeIndex(i) ...
            DLCChangeIndex(i)+...
```

```
120             DLCLength(i-2)];
121
122         % Plotting faint indication bars.
123         for j=1:2
124
125             % Changing to faint color
126             indBarColorFaint = ...
                    DLCcolorMatrix{DLCvect(DLCChangeIndex(i-3+j))};
127             % Finding center of the indication bar
128             shiftindBarCenterX =  ...
                    periodeChangeIndex(j)+(DLCLength(i-3+j)/2);
129             % Converting from minutes to hours
130             shiftindBarTime = ...
                    int2str((DLCLength(i-3+j)*5)/60);
131
132             % Changing color.
133             if isequal(indBarColorFaint,[0 0 0])
134                 % Grey
135                 indBarColorFaint = [0.5 0.5 0.5];
136             else
137                 % Faint yellow
138                 indBarColorFaint = [1 1 0.80];
139             end
140
141             % Plotting Indication bar
142             rectangle('Position',[periodeChangeIndex(j)...
143             ,yaxislimits(1)-yaxisOffset, DLCLength(i-3+j),...
144             indBarHeight],'FaceColor',indBarColorFaint)
145
146             % Adding text to indication bar.
147             if isequal(indBarColorFaint,[0.5 0.5 0.5])
148                 text(shiftindBarCenterX,yaxislimits(1)-...
149                 yaxisOffset+1000,shiftindBarTime,...
150                 'HorizontalAlignment','center','color','white')
151             else
152                 text(shiftindBarCenterX,yaxislimits(1)-...
153                 yaxisOffset+1000,shiftindBarTime,...
154                 'HorizontalAlignment','center');
155             end
156
157         end
158
159         % Increasing the yaxisOffset
160         yaxisOffset = yaxisOffset + 2000;
161     end
162
163     % Plotting Window
164     rectangle('Position',[DLCChangeIndex(i),yaxislimits(1),...
165         DLCLength(i),windowHeight]);
```

```
166
167     % Plotting Indication bar
168     rectangle('Position',[DLCChangeIndex(i),yaxislimits(1)-...
169     yaxisOffset, DLCLength(i),indBarHeight],'FaceColor',...
170     indBarColor)
171
172     % Adding text to indication bar
173     if isequal(indBarColor,[0 0 0])
174         text(indBarCenterX,yaxislimits(1)-yaxisOffset+1000,...
175             indBarTime,'HorizontalAlignment','center','color',...
176             'white')
177     else
178         text(indBarCenterX,yaxislimits(1)-yaxisOffset+1000,...
179             indBarTime,'HorizontalAlignment','center');
180     end
181
182 end
183
184 end
```

Function : modelEst

```
1  % Vegard Brattland, University of Stavanger, 2016
2  % Thesis: Imaga Processing and Analysis of Leaf Movement...
3  % in Mimosa Pudica
4
5  % ********************************************************
6  % This function is used to estimate black box models
7  % ********************************************************
8
9
10 % Choice of model
11 arxEstimation = 0;
12 armaxEstimation = 0;
13 bjEstimation = 0;
14 testModel = 0;
15 simulinkModel = 1;
16
17 % Experiment test data
18 EstDataOut = antallHvitePiksler(1:4000)';
19 EstDataInp = DLCvect(1:4000);
20
21 % Experiment validation data
22 ValDataOut = antallHvitePiksler(1:4000)';
23 ValDataInp = DLCvect(1:4000);
```

```
24
25 % Normalization of test data
26 norm_EstDataOutput = (EstDataOut - min(EstDataOut))/...
27     (max(EstDataOut) - min(EstDataOut));
28
29 % Normalization of validation data
30 norm_ValDataOutput = (ValDataOut - min(ValDataOut))/...
31     (max(ValDataOut) - min(ValDataOut));
32
33
34 % Changing test dataLight = 0.5, and Dark = -0.5
35 for i = 1:length(EstDataInp)
36     if EstDataInp(i) == 1
37         EstDataInp(i) = -0.1;
38     end
39     if EstDataInp(i) == 2
40         EstDataInp(i) = 0.1;
41     end
42 end
43
44 % Changing val data Light = 0.5, and Dark = -0.5
45 for i = 1:length(ValDataInp)
46     if ValDataInp(i) == 3
47         ValDataInp(i) = 0;
48     end
49     if ValDataInp(i) == 1
50         ValDataInp(i) = -0.1;
51     end
52     if ValDataInp(i) == 2
53         ValDataInp(i) = 0.1;
54     end
55 end
56
57 % Detrending the test,validation and full dataset
58 EstDataOut = detrend(norm_EstDataOutput);
59 ValDataOut = detrend(norm_ValDataOutput);
60
61 % Smoothing data
62 EstDataOut = smooth(EstDataOut,'moving');
63 ValDataOut = smooth(ValDataOut,'moving');
64 ValDataOutMov = smooth(ValDataOut,125,'moving');
65
66 % Making iddata file
67 modelTestData = iddata(EstDataOut,EstDataInp,1);
68 modelValData = iddata(ValDataOut,ValDataInp,1);
69
70 % AIC initial
71 bestAICARX = 1;
72 bestAICARMAX = 1;
```

```
bestAICBJ = 1;

%Plotting the data
figure(4)
lengthVect = 1:1:length(EstDataOut);
plotyy(lengthVect,antallHvitePiksler(1:4000),lengthVect,...
EstDataOut)

figure(5)
lengthVect = 1:1:length(ValDataOut);
plotyy(lengthVect,antallHvitePiksler(1:4000),lengthVect,...
ValDataOut)



% Estimating ARX
if arxEstimation == 1
    for na = 1:5
        for nb = 1:5
            for nk = 1:5
            arxSys = arx(modelTestData,[na,nb,nk]);
            % Saving the best fit (AIC)
                if arxSys.Report.Fit.AIC < bestAICARX
                    bestAICARX = arxSys.Report.Fit.AIC;
                    bestARXModel = arxSys;
                end
            end
        end
    end
end

% Estimating ARMAX
if armaxEstimation == 1
    for na = 1:5
        for nb = 1:5
            for nc = 1:5
                for nk = 1:5
                    armaxSys = ...
                       armax(modelTestData,[na,nb,nc,nk]);
                % Saving the best fit (AIC)
                    if armaxSys.Report.Fit.AIC < bestAICARMAX
                        bestAICARMAX = ...
                           armaxSys.Report.Fit.AIC;
                        bestARMAXModel = armaxSys;
                    end
                end
                nc
            end
            nb
```

```
        end
        na
    end
end
% Estimating Box-Jenkins
if bjEstimation == 1
    for nb = 1:5
        for nc = 1:5
            for nd = 1:5
                for nf = 1:5
                    for nk = 1:5
                     bjSys = ...
                         bj(modelTestData,[nb,nc,nd,nf,nk]);
                         % Saving the best fit (AIC)
                         if bjSys.Report.Fit.AIC < bestAICBJ
                             bestAICBJ = bjSys.Report.Fit.AIC;
                             bestBJModel = bjSys;
                         end
                    end
                end
            end
            nc
        end
        nb
    end

end

% Testing model of choice
if testModel == 1
    modelToTest = bj(modelTestData,[5 5 5 5 1])

    % Comparing the model and test data
    [outputTest fitTest x0] = ...
        compare(modelTestData,modelToTest);

    % Comparing the model and validation data
    [outputVal fitVal x0] = compare(modelValData,modelToTest)

    % Plotting the fit
    figure(3)
    compare(modelTestData,modelToTest)
end

if simulinkModel == 1
    % Making timestamp vector

    K = 2;
    w0 = 0.01;
```

```
167     zeta = 0.075;
168
169     ValDataLength = 1:1:length(ValDataInp);
170     EstDataLength = 1:1:length(EstDataInp);
171
172     dataOutput = [ValDataLength' ValDataOut];
173     estDataInput = [EstDataLength' EstDataInp];
174     estDataOutput = [EstDataLength' EstDataOut];
175     ValDataOutputMov = [ValDataLength' ValDataOutMov];
176
177     save('simuModData.mat','dataOutput','estDataInput',...
178     'estDataOutput','ValDataOutputMov')
179
180     save('modelData.mat','modelTestData','modelValData')
181 end
```