Insect Event Extraction in LIDAR Images

using Image Analysis and Convolutional Neural Networks

Ludvig Malmros

Master’s Thesis

Lund University
Department of Mathematics
Faculty of Engineering
Lund University
Sweden
14 June, 2018
Abstract

Insect monitoring has earlier been a manual, tedious and time consuming task that is impossible to do in real time. Thus there exists a need for a real time automatic insect monitoring system for counting and classifying insect for use in scientific research and pesticide spraying control. One approach to automatize this is using LIDAR to detect insects. In this thesis it has been explored how to detect, segment and do merge classification on insect events showing up in large 2D time-range map frames created from a LIDAR optics setup as insects fly through a laser beam.

The suggested extraction method combines simple intensity thresholding techniques and well known edge detection techniques to do the first selection of insect event pixels. A statistical dilation iteration stopping criteria based on background noise is suggested for region growing. Together they create a segmentation method that is used to segment insect events while avoiding accidental splitting and merging of events. Thereafter a trained convolutional neural network is suggested to classify all events that might have been merged, such that they could be discarded instead of being inputted to the species classification system.

Tests and observations indicate that the old segmentation method finds close to all the wanted insect events, but over-segments them drastically in some cases. By dividing the method into one event detection and one border finding part, the suggested extraction method are able to find the same amount of events without increasing the number of splits and merges. At the same time it is able to find event segmentation borders with a higher precision then previously possible.

Tests on the merge classification indicate surprisingly good results for the ability to classify event merges. Creating artificial merges to handle the imbalanced data set shows further improvement, while oversampling does not. Indicated is also that the size compression used does not seem to effect the classification negatively.
Acknowledgements

First of all I would like to thank Fauna Photonics for the opportunity to write my thesis in cooperation with them. I would also like to thank all the employees at Fauna Photonics. Especially I would like to thank Jord Prangsma for his event rating efforts, discussions and useful feedback to the report. A special thanks also goes out to Mikkel Hedegaard Jensen for the help with the setup and discussions concerning the neural network. Naturally a special thanks also goes to my company supervisor Klas Rydhmer and LTH supervisor Magnus Oskarsson for all the useful discussions driving the work forward throughout the entire thesis. Lastly, I would like to thank all the insects that have been kind enough to fly through the LIDAR setup to be captured on 2D time-rang map frames enabling this master thesis.
Acronyms

LIDAR - Light Detection and Ranging
CNN - Convolutional Neural Network
OpenCV - Open Source Computer Vision Library
GUI - Graphical User Interface
WBF - Wing Beat Frequency
SNR - Signal to Noise Ratio
FFT - Fast Fourier Transform
Contents

Abstract i
Acknowledgements ii
Acronyms iii

1 Introduction 1
  1.1 Background and motivation ........................................... 1
  1.2 Data acquisition ...................................................... 2
    Scheimpflug principle .................................................. 2
    Using Scheimpflug LIDAR principle to create 2D time-range maps ...... 3
    Insect event creation ................................................... 3
    Insect events in LIDAR 2D time-range map ............................ 4
  1.3 Problem description .................................................. 5
    General problem and division ......................................... 5
    Merging ............................................................... 6
    Splitting ............................................................. 6
    Different intensity levels ............................................. 7
    Different size, shape and length ..................................... 8
    Other Scheimpflug effects ............................................ 8
    Optical effects ..................................................... 8
    Noise ............................................................... 9
    False positives or too few wing beats to be useful .................. 9
  1.4 Previous work ....................................................... 10
    Previous extraction method ......................................... 10
  1.5 Goals and limitations ............................................... 11
    Further limitations .................................................. 12

2 Theory 13
  2.1 Definitions and measures .......................................... 13
    Definition of segmentation ......................................... 13
    Jaccard index as segmentation performance measure ............... 14
    Receiver operating characteristics(ROC) graphs .................. 14
  2.2 Image convolution and filtering ................................... 15
  2.3 Canny edge detection ............................................... 18
  2.4 Binary morphological dilation .................................... 19
  2.5 Convolutional Neural Network ..................................... 20
    Neurons ........................................................... 20
    Convolutional layer ................................................. 21
    Max pooling layer .................................................. 22
    Fully connected layer ............................................... 23

3 Developed tools 24
  3.1 Visualization and event marking tool ............................. 24
    3.1.1 Detection results and discussion ............................... 27
    Are enough events found by the old method? ....................... 27
Merge and split effect depending on number of dilation iterations ........................................ 29

3.2 Event rating tool ................................................................. 29
   Event rating scale ............................................................... 30

3.2.1 Event rating results and discussion ................................. 31

4 Suggested extraction method .................................................. 34

4.1 Detection and segmentation ................................................. 34

4.1.1 Border segmentation test results and discussion .............. 39
   Problems with a stopping criteria based on background pixels ........................................ 42
   Need for edge data and better gradient filters ......................... 42
   Need for more scalable segmentation measure ......................... 43

4.2 Merge classification ............................................................ 44

4.2.1 Dataset creation and CNN architecture ........................... 44
   Merge data set creation .................................................. 44
   Event image input format ............................................ 45
   Suggested network architecture ........................................ 45

4.2.2 Merge classification test results and discussion .............. 46
   Training and creation of ROC graphs ................................ 46
   Does larger image size improve the classification results . 48
   Improvement using oversampling ...................................... 49
   Improvement using artificial merges .................................. 49
   Miss-classified examples .................................................. 50

5 Summary and Conclusions ....................................................... 54

6 Future improvements ............................................................. 55

   Detection ................................................................. 55
   Segmentation ............................................................. 55
   Classification ............................................................ 55
   Other improvements ....................................................... 56
## List of Figures

1. Scheimpflug principle\cite{15} with the object plane along the ground and in angle with the lens and image planes. ........................................... 2
2. Full 2D time-range map frame\cite{15}. ............................................. 3
3. The reflected light from the insect body and its wing beats creating an insect event as it flies through the laser beam\cite{15}. .... 4
4. 2D time-range map of an insect event\cite{15}. ................................. 4
5. The effect of the added noise due to over-segmentation when summing range pixels to do WBF extraction. ..................... 5
6. Three insect events being merged as one none overlapping merge. 6
7. One insect event split into many where the boxes are events found by the old event extraction method. .......................... 7
8. Different intensity level............................................................. 7
9. Different length insect events..................................................... 8
10. Double wing beats................................................................. 9
11. Examples of false positive events that are not useful for insect counting and insect classification. .......................... 9
12. The part of this thesis in relation to the full insect monitoring system. ........................................................... 11
13. Nominator and denominator of Jaccard index........................................... 14
14. ROC curve with confidence interval in true positive rate\cite{17}. .... 15
15. Illustration of one step in an image convolution where the filter h(x,y) is mirrored and positioned in the marked area in image f(x,y) resulting in the value marked in the resulting image g(x,y)\cite{7}. .................................................. 16
16. Examples of common image filters ........................................... 17
17. Examples of a checkerboard image convolved with an averaging filter and Sobel filters in each direction. ........................ 17
18. Before and after image I have been dilated with kernel S. ........ 19
19. Fully connected network showing an input layer connected to the first hidden layer and the second hidden layer connected to the output layer. .................................................. 20
20. Illustration of the neuron showing the inputs x with their corresponding weights w creating a linear combination that is then fed through a non linear activation function. ....................... 21
21. Illustration of a convolutional layer showing the input volume(image) to the left and the output volume to the right. ............. 22
22. Illustration of the operation of a max pooling layer as a down-sampling operation. On the left the change of the volume is illustrated and on the right the down sampling of each individual depth slice is shown. .................................................. 23
23. Visualization tool consisting of four windows. ......................... 24
24. Event split warning marking potential splits with a yellow box. ... 26
25. Event merge warning marking potential merges with a yellow box. 26
26. Events not found warning marking all events manually marked but not found by the extraction method with a black box. ... 27
27. Intensity threshold effect on the number of events found using the old method. Marked is the number of events found with the threshold used in the old extraction method. .......................... 28
28 Event rating tool consisting of five windows containing relevant information to support the raters decisions. 
29 Event rating scale with examples of where some rated events ended up on that scale. 
30 Ratings from the four raters. 
31 Average rating of the event images. 
32 Suggested method for detection and segmentation. 
33 Dilation iteration stopping criteria example showing how the segmentation is grown until the stopping criteria is met. 
34 Jaccard index of 35 good events calculated for the old and the new segmentation method. 
35 An event where the new method showed a substantial improvement. 
36 Illustration of the result when the dilation process can not handle the asymmetry of the event such that unnecessary noise is included on one side of the event to be able to completely fill out the other. 
37 Suggested CNN architecture. 
38 Slight over-fitting during training as can be observed from the difference in the training and testing error at the end of the training. 
39 ROC curves with the true positive rate being merges correctly classified as such and the false positive rate being non merges being classified as merges. 
40 Comparison of a real non crossing merge and artificially created merge. 
41 Merges miss-classified as non merges.
1 Introduction

In this first chapter some background is given together with the motivation and goal of the thesis. Furthermore is a description of the data acquisition included to help the readers understanding of the data used as the starting point for this thesis.

In chapter 2 a theoretical background is given to help the reader understand the algorithms later used in the suggested extraction method. The chapter is divided in such a way that readers already familiar with the used techniques can skip the corresponding theory sections.

In chapter 3 descriptions of the developed visualization and event rating tools are provided, followed by corresponding test results specifically enabled by the tools and discussion about these.

In chapter 4 descriptions of the different parts of the suggested extraction method are provided, followed by corresponding test results for each part and discussion about these.

In chapter 5 the full thesis is summarized and the main conclusions of the thesis is presented.

In chapter 6 the possible next steps in development process of the extraction method that allow potential improvements are presented. Areas of the extraction method that need further exploration are also presented.

1.1 Background and motivation

Years of research have created the foundation for the Copenhagen based company FaunaPhotonics where this thesis was carried out. The research developed from using fluorescence LIDAR to monitor insects[1] into the creation of a "Super resolution LIDAR"[2] which showed how a Scheimpflug LIDAR technique was able to be used for more detailed insect monitoring including wing beat frequency for species identification. The use of Scheimpflug LIDAR to monitor insects is the type of fundamental idea the company FaunaPhotonics later was founded on. Their goal is to automatically count and classify insects in real time for the benefit of easier adapting insect pesticides which would be beneficial for crop growth, environment and the scientific study of insects behavior.

Information about the amount of insects and the species of the insects that are flying in the field at each moment, would allow farmers to make smarter decisions about when to spray such that only harmful insects are killed. This has the potential to result in less environmental impact due to less spraying of chemicals, less impact on beneficial insects such as pollinators and optimize crop growth for the farmers. This could also be seen as an economical incentive for the farmers allowing them to use less pesticide, thus lowering their spraying costs while still protecting the crop from getting damaged. This technology is also useful from an academic point of view. Researcher are able to use the technology for monitoring of insects to better understand their behavior, as in
a recent study where a Scheimpflug LIDAR was used to study the daily flight activities of insects and their predators[28].

To be able to do this the company Fauna Photonics uses a LIDAR setup that generates large amounts of data consisting of large 2D time-range map frames filled with small insect events from insects passing by the equipment. Due to the large amount of data, it would be infeasible to look through it all manually and evermore so impossible to do it in real time. This is why the improvements on the insect event extraction step investigated in this thesis is important and needed, as a step on the way to reach the broader company goal of being able to count and classify insects in real time.

1.2 Data acquisition

Describing the formation of the 2D time-range map frames, this section helps the reader to understand the content represented in the data used in this thesis.

Scheimpflug principle

To create the 2D-data time-range map used in this thesis, a good start is to look at the optical basis for how the range map is created at each time instance. The basis for this is something called the Scheimpflug principle[14]. It is a technique where the image, object and lens planes in the optical system are not parallel, but rather set at an angle relative each other as can be seen in figure 1. This means that objects along the object plane is imaged at different positions on the image plane depending on their distances from the lens. Thus objects closer are imaged at one side of the image plane while objects further away are imaged on the other side. The important thing to note is that the image and object plane are not parallel and the distance on the object plane compared to the position imaged on the image plane is non linear. Further details and equations of the Scheimpflug principle is out of the scope for this thesis and the interested reader is referred to previous work done using this technique, such as Development of a Scheimpflug LIDAR system for atmospheric aerosol monitoring[19], for a deeper theoretical insight.

Figure 1: Scheimpflug principle[15] with the object plane along the ground and in angle with the lens and image planes.
Using Scheimpflug LIDAR principle to create 2D time-range maps

To make use of the Scheimpflug principle a 2048 pixels long array is placed along the image plane. Additionally, a laser beam is created along the object plane such that when an object passes through the laser beam, the backscattered light from the object is detected and in focus on some subset of the pixels in the array. This means that every object that passes through the laser along the object plane within a set range of distances, reflects light that will be in focus on one part of the pixel array. Which pixels that are hit by the focused backscattered light depends on the distance of the object from the lens along the object plane axis.

By sampling the intensity values measured by the pixels in the pixel array at a set sampling speed, a 2D time-range map is created with the range of distances on one axis, the time on the other axis and the intensity value for each time and distance combination. The resulting 2D time-range map used in this thesis was created using an effective sampling rate of 1.75kHz resulting in a 140 megabyte 2D time-range map frame of size 2048 times 17500 pixels per 10 second of measurement, as seen in figure 2.

![Figure 2: Full 2D time-range map frame](image)

Visible in the figure are a great amount of what looks like dots of higher intensity, which are easiest seen by the insect symbol along the range axis. These dots are insects or other objects passing through the laser beam and reflecting back light to the pixel array. Also visible in the figure is a high intensity line that is caused by a large stationary object such as a tree.

Insect event creation

Seen is figure 3 is the creation of one insect event as the insect flies through the laser beam. At each sample time a reflection of backscattered light created by the insect reaches the pixels in the array sensor. Which pixel is hit by the reflection from the insect depends on the distance from the equipment the insect passes the laser. Thus at each sample time a different intensity is registered by
the corresponding pixels. The reflected intensity registered by the pixels depends on the insects wing position at that current point in time. In figure 3 the change in the reflection over time as the insect passes the laser can be seen creating a typical insect event consisting of a body with wing beat peaks super-positioned on top.

Figure 3: The reflected light from the insect body and its wing beats creating an insect event as it flies through the laser beam[15].

Insect events in LIDAR 2D time-range map

Figure 4 shows one such insect event that corresponds to cutting out a small part of the big 2D time-range map in figure 2. In this event image the wing beats are clearly seen as intensity peaks with a body connecting them, which is not always the case for the events created in real world field tests. After these insect events have been extracted, all the range pixel values for each time instance are summed up to create a wing beat profile, from which a wing beat frequency(WBF) can be extracted. This WBF is used as one of the main features in later steps of insect species classification.

Figure 4: 2D time-range map of an insect event[15].

For the rest of the report the big 2D-data time-range map will be referred to as the big or full image and the cutout for each event in the 2D-data time-range map will be referred to as insect event images.
1.3 Problem description

In this section the problem of extracting the insect events is divided into three main subproblems. Furthermore examples of the difficulties of doing the insect event extraction is presented.

General problem and division
The problem of extracting the insect events from the big image can reasonably be divided into three subproblems.

1. **Detection:** The problem of finding all the events that are present in the big image.

2. **Border segmentation:** The problem of deciding the position of the cut line deciding what is considered foreground/event and background. If the border is set too tight one risk missing some part of an insect event or splitting it into many events. Setting the border too wide increase the risk of merges and a lot of extra noise is included in the event as seen in the top part of figure 5. The lower part of the figure shows the extracted insect event when the intensities for all range pixels at each time instance is summed, creating the wing beat profile of the insect event. This profile is used to find the WBF which is more difficult when the segmentation of the insect event includes extra noise as the peaks are less distinct and are smoothed out.

Figure 5: The effect of the added noise due to over-segmentation when summing range pixels to do WBF extraction.
3. **Event classification:**

The problem of being able to say if an event is good enough for the next step in the system, which is insect species classification. This could be classifying the event as good or bad, classify it according to some rating or as focused on in this thesis, classify if the event is actually a merge of two or more events or not.

**Merging**

Occasionally when insect events are captured close to each other the old method used by the company at the beginning of this thesis to extract events, can sometimes result in a merging of insect events not being able to distinguish them as separate events. A merged event has impact on the insect count as many insects are counted as one single event. Additionally, the automatic extraction of wing beat peaks becomes more difficult as the insect wing beat characteristics of the two insect events are combined, thus making the WBF extraction harder or impossible. Figure 6 shows a merge consisting of three insect events.

Merged events could be separated into two main groups. The first group is non overlapping insect events, where the events do not touch, but are merged due to segmentation borders being too big. The second group is overlapping events that merges due to actually having pixels connecting them in the image.

![Figure 6: Three insect events being merged as one non overlapping merge.](image)

**Splitting**

The opposite of the merge problem is when one insect event is split into many, often due to weak or non-existing insect body. This can occur when the segmentation border is set too tight. Figure 7 shows an example of an event that was split into many using the old segmentation method. Each rectangle marks an event found by the old method, even if all the event rectangles actually are many peaks of the same event wrongly being considered as different events.
In the same fashion as a merge, a split would also cause a miss-count of insects. However, a split could probably be considered more of a problem than a merge for two reasons. Firstly, a split often splits an event into more parts than the number of events that are merged, thus creating a higher error on the count. Secondly, after an event is split it is separated into different event images and can be difficult to analyze either manually or automatically. This is due to the surrounding context being lost and the impossibility of visually seeing that the event have been split solely based on each separate extracted event image. For the merge on the other hand it is possible to visually see that the event has been merged solely based on each separate event image. Thus exploring analytical methods to detect it is possible, or taking it even further exploring methods trying to separate the merged event into its parts.

**Different intensity levels**

Different insect species and other real world phenomena creates insect events with a large intensity variation between the different events. Examples of this can be seen in figure 8 showing one very weak and one relatively strong insect event. In addition to having an intensity difference between events, there also exist a intensity difference within each event. This in-event variation normally consist of two parts. Firstly, there is often a variation in the intensity peaks with lower intensity at the ends of the event and higher intensity in the middle. Secondly, there also exist an intensity variation between the intensity peaks and the insect event body. This makes it more difficult to find e.g. good values for intensity thresholds.
Different size, shape and length
In the data there is a large variation of the size, shape and length of the insect events. This is due to natural reasons such as different insect species, but also an effect of the non linearity of the Scheimpflug principle, where the same object would be imaged differently depending on the distance from the optics at which it is captured. Furthermore, the length of the event and number of wing beat peaks varies greatly and is dependent on time the insect spends in the laser beam. This can be seen in figures 9a compared to figure 9b. There are also variations where some events have no body above the noise level, while others do. This variation of shape, width, length, existence of peak lines and much more makes it more difficult to find one suitable model describing an insect event. Furthermore it makes it more difficult when trying to do some type of event classification, as finding stable and representative features for different types of events is more difficult.

![Figure 9: Different length insect events.](image)

Other Scheimpflug effects
Using the Scheimflug principle to create the LIDAR images results in a non linear relationship between the pixel and the distance represented by each pixel as previously explained. This means that the same insect event will be pictured with a different pixel area depending on at what distance from the laser equipment the insect event is captured and thus get a different size. In addition to this the non linear distance results in a larger concentration of events at shorter distances which possibly could increase the risk of events being merged at that distance.

Optical effects
The optical system used to record the time-range map leads to some added variation in the shapes of the recorded insect events. However, as this thesis take the approach of doing image analysis, the physical reasons behind the effects are not explained and for the sake of this thesis it is enough to conclude that there might exist optical effects that increase the shape variability between insect events. One such example is illustrated in figure 10 showing what looks to be double wing beats, but could potentially be an effect of miss calibrated optics.
Figure 10: Double wing beats.

Noise
The background noise in the images is well defined and is actually even used as a part of the solution to the segmentation problem in the form of a statistical stopping criteria. Its distribution is a zero mean Gaussian with a varying standard deviation which is the same across each big image, but different depending on the time of day and equipment setup. However, even with simply distributed noise, it still makes it more difficult to find an intensity thresholds and is sometime strong enough to drown the weaker part of the event such as the body increasing the risk for splits. Most importantly, it also makes it more difficult to find the edges of the insect events as edge detection techniques often are based on image gradients which are very noise sensitive.

False positives or too few wing beats to be useful
One of the bigger problems when trying to count the insect events is false positive events showing up in the big image. These have high enough intensity levels to be considered an event by the old segmentation method, even if they are created by an object of something other than an insect. An example of this could be a falling leaf or similar passing by the laser. There could also be insect events with bad quality due to few wing beats or being weak and thus not being useful to extract WBF. Consequently not useful for insect classification either. Figure 11 shows a couple of examples of unwanted false positives.

Figure 11: Examples of false positive events that are not useful for insect counting and insect classification.
1.4 Previous work

Some work have previously been done regarding the capture of insect events using LIDAR optics such as a master thesis written by Sandra Török at LTH[27]. Research has also been done on insect classification using LIDAR optics in a lab environment trying to classify insect gender and species[29]. Similar work have also been done at the company for the step after the insect event extraction, doing insect species classification based on lab data. However, the insect event extraction step considered in this thesis, needed between the capturing of the insects and the ability to do insect species classification, has not been explored to the same extent. The company has developed one prototype insect event extraction method, using simple intensity thresholding and fixed border size to automatically detect and segment out each event. However, this method has not been subject to a lot of performance evaluation. Thus different thresholds and if it can even detect all the events in the big image is unknown. The only thing that can be concluded beforehand about the method, is that it tends to over-segment the extracted insect events with too big borders. This is solely based on visual inspection of the events outputted by the method. The need for this evaluation and the development of an improved insect event extraction method is where this thesis has its starting point.

Previous extraction method

The old detection and segmentation method can be divided into three main steps.

1. **Intensity threshold**
   With an intensity threshold based on the standard deviation of the intensity levels in each frame multiplied with a constant, each frame is first thresholded on the intensity level removing weak values considered to be noise.

2. **Ellipse border**
   Each island of connected pixels from the previous thresholding operation is fitted with a fixed size ellipse to define its borders.

3. **Border merge**
   All border ellipses that overlap are combined and consider to belong to the same insect event.
1.5 Goals and limitations

Figure 12 shows an overview of the full insect monitoring system and the parts of the system where this thesis has its focus and limitations. From this perspective the main goal of this thesis is to investigate if it is possible to find and implement a better method for automatically extracting the insect events from the large images using image analysis and machine learning techniques. This mainly consists of three part goals based on the problem division in section 1.3.

1. All events present in the big image should be detected.

2. Based on a manual segmentation of the events used as the golden standard, the developed method should find a segmentation with borders closer to the golden standard compared to the old method.

3. The extracted events should be classified such that only events that are useful in the later insect species classifier are sent as inputs to that classifier. This thesis is limited to exploring the possibility of classifying if an event has been merged or not.

Figure 12: The part of this thesis in relation to the full insect monitoring system.

To be able to reach these goals, some understanding of where the old method, briefly described in section 1.4, works well and where it can be improved is needed. This is to be able to guide the effort of developing a new event extraction method towards the pre-mentioned part goals that has the potential for most improvement. Therefore a part of this thesis consist of the development and implementation of visualization, evaluation and rating tools. These tools also need to enable easier exploration of the data and the extraction methods. However, it should be noted that it is not a part of this thesis to do a thorough
investigation of all the aspects of the old method, but only enough to guide towards the subgoal with the potential of leading to the most improvement.

Further limitations
Some limitations have already been mentioned and to further restrict the thesis from its corner cases the following limitations are also set to the thesis.

- Among the field data used in the thesis there exist big images captured on days with rain. This means that the image are completely overfilled with events created from the rain drops, which makes an extreme case where finding insect events is extremely difficult and is thus not considered in this thesis.

- As described in section 1.2, the created 2D time-range maps are quite large, but consists to the most part of large areas of noise where occasionally relatively small events can be found. This means that there are a lot of pixels containing no useful information that still has to be examined when searching for the insect events. This aspect needs to be considered when developing a method with the long term goal of handling real time insect monitoring. However, finding the most efficient and fastest method for event extraction is not the main focus of this thesis. On the other hand, methods taking unreasonable amounts of time with little possibility of optimization will be still be discarded.

- This thesis explores the possibility of classifying the insect event images as being a merge or not. One step further would be to develop a method that can separate the merged events into their consecutive parts once they have been detected. However, this thesis only considers the classification and not the separation of merges.
2 Theory

In this section the basics of the theories used in this thesis for image analysis and neural network classification are described to easier allow the reader to follow the preceding sections. For readers with previous knowledge in image analysis section 2.2-2.4 could likely be skipped and for readers with knowledge in convolution neural networks section 2.5 could likely be skipped. The interested reader can follow the references to get a deeper theoretical insight if desired.

2.1 Definitions and measures

Definition of segmentation

With finding better segmentation borders as one of the goals of this thesis a more rigorous definition of what image segmentation really means is presented. Strictly mathematically an image segmentation is often defined as the following[3]. Let $\mathcal{R}$ represent the entire image. Segmentation is then a method that divides $\mathcal{R}$ into $n$ subsections, $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3,...$ such that,

1. $\bigcup_{i=1}^{n} \mathcal{R}_i = \mathcal{R}$, the union of all regions/segments cover entire $\mathcal{R}$.
2. $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset$ for all $i$ and $j$, $i \neq j$, there exist no overlap of the regions.
3. $P(\mathcal{R}_i) = \text{True}$ for $i = 1, 2, ..., n$ where $P$ is the logical uniformity predicate defined over the points in set $\mathcal{R}_i$
4. $P(\mathcal{R}_i \cup \mathcal{R}_j) = \text{False}$ for $i \neq j$ and $\mathcal{R}_i$ and $\mathcal{R}_j$ are neighboring regions.
5. $\mathcal{R}_i$ is a connected set for each $i = 1, 2, ..., n$.

In less mathematically formal terms this means that,

1. All pixels must be assigned to a region.
2. Each pixel must belong to a single region only.
3. Each region must be uniform.
4. Any merged pair of adjacent regions must be non-uniform.
5. Each region must be a connected set of pixels.

The uniformity predicate mentioned in point 3 and 4 is basically chosen for each segmentation problem defining some similarity within each segmented region. It can often be e.g. color or intensity level properties in an image defining this similarity. For this thesis, it could be edge characteristics and the intensity level for the first steps of segmenting the full image into foreground and background. However, for the later steps finding closer insect event borders and dividing the foreground into separate insect events, this uniformity predicate would be harder to define as each region $\mathcal{R}_i$ would then represent one insect event between which the previous characteristics of similarity often are similar. Additionally, point 5 is discarded for this thesis. The reason for this is easiest explained by an example. In the case of an insect event without a body the subregion $\mathcal{R}_i$ representing this insect event will consist of many intensity islands representing the insect wing beats. These are not connected sets of pixels. However, it should still be considered only one subregion representing the same insect event. To summarize, point 1 and 2 are the ones of biggest interest for this thesis where each region $\mathcal{R}_i$ should represent an insect event and one region $\mathcal{R}_1$ the background in the final insect segmentation.
**Jaccard index as segmentation performance measure**

The Jaccard index or sometimes called intersection over union, is an objective region based similarity measure. In the context of image segmentation techniques it gives a similarity measure between the manually segmented ground truth/golden standard and the segmentation output from the tested segmentation algorithm. Defining the algorithm segmentation result as the set $A$ and the golden standard segmentation as $B$ the Jaccard index is given by\(^{16}\),

$$J(A, B) = \frac{G(A \cap B)}{G(A \cup B)}, \quad 0 \leq J(A, B) \leq 1,$$

where $G(\cdot)$ is the counting operation counting all the pixels in that set. To get a better understanding of equation 1, the set of the nominator in the Jaccard index can be seen in figure 13a and the set of the denominator in figure 13b. In the context of images with a discrete number of pixels where 1 represent foreground and 0 background, this simply means counting the number of pixels of the golden standard and the method segmentation that overlaps and divide the result with the number of pixels in the golden standard and the method segmentation together while including shared pixels only once.

![Venn Diagrams](image)

(a) Intersection  
(b) Union

Figure 13: Nominator and denominator of Jaccard index.

**Receiver operating characteristics(ROC) graphs**

One way to test the performance of machine learning classifiers such as convolutional neural networks is ROC graphs. An interesting introduction to these graphs written by Tom Fawcett\(^{17}\) is the base for this section if not otherwise stated.

ROC graphs are a good tool to visualize the performance of classifiers as they provide a richer measure of classification performance compared to scalar measures such as accuracy, error rate or error cost. For many cases simple classification accuracy have been showed to often be a poor metric for measuring performance. A ROC graph illustrates the relative trade-offs done between benefits (true positives) and costs (false positives). Figure 14 shows an example of a ROC graph.
Any classifier outputting a probability for each of two classes, such as a convolutional neural network with a sigmoid activation function in the output layer used in this thesis, can be used with a threshold to create a binary classifier. If the classifier output is above the threshold, it classifies the output to one class and if below the other. For each possible choice of threshold a different point in the ROC space is created. By varying this threshold and calculating the true positive and false positive rate for a range of discrete threshold values going from 0 to 1, the bended curve seen in figure 14 is created. Also shown in the graph is a confidence interval of the true positive rate. This is created by retraining and retesting the classification model multiple times with different splits of the data into training-and test sets, enabling a more stable comparison between different models. Informally, classifiers resulting in a ROC graph with the curve pushed as much as possible to the upper left corner of the graph, is considered to be a better classifier. Further left means less false positives and further up means more true positives. Consequently, this means that the optimal point corresponds to the upper left corner. A point in that corner has a perfect classification, classifying the tested class fully correct while no data from the other class is wrongly classified as the tested class.

2.2 Image convolution and filtering

Szeliski[7] describes linear image filtering as a local operator which uses the neighboring pixels to determine the final value of the center pixel at each position in the image. Its a weighted combination of neighboring pixels and is a linear filtering operation. To apply a filter, a 2D kernel consisting of the weights deciding the impact of each neighboring pixel on the result is used. Doing linear filtering is then the same as doing a convolution between the kernel and the image.
The equation for the convolution operation is given by\[7,\]
\[g(i, j) = \sum_{k,l} f(k,l)h(i-k, j-l), \] (2)

or written with the convolution operator as,
\[g = f * h, \] (3)

where \(g\) is the filtered image, \(f\) the input image and \(h\) the kernel/filter.

Informally, the convolution takes a mirrored filter and puts its center pixel at each position in the image. At each such filter position each of the filter pixels are multiplied with the correspondingly overlapped image pixels. The resulting values of these multiplications are then summed to give a final output value at that position. This calculation for one position of the filter in the image is illustrated in figure 15. Here the kernel \(h(x, y)\) overlap the marked area in image \(f(x, y)\) and the operation as previously explained gives the output at the position marked in the output image \(g(x, y)\). This is repeated for each position in the image according to equation 2. Additionally, one can note that the filtered output image \(g(x, y)\) has a smaller image size than the input image \(f(x, y)\) in figure 15. To get the same image output size as the input, padding is normally used. Padding means adding one or more extra borders of pixels around the input image. The number of extra pixel borders needed depend on the filter size. The simplest padding is just adding border pixels with the value 0 called zero padding.

There exist a wide variety of different filters that can be used depending on the application. Some common ones are averaging filter seen in fig 16a, the simplest approximative Gaussian filter seen in fig 16b and the two Sobel filters seen in figure 16c and figure 16d.
Figure 16: Examples of common image filters

(a) Averaging filter
(b) Gaussian filter
(c) Sobel filter in x direction
(d) Sobel filter in y direction

The averaging kernel and the Gaussian kernel smooth the image removing high frequency noise. The averaging kernel basically takes the average of the current pixels and its neighbors at each position in the input image. The Gaussian kernel is a discretization of the continuous Gaussian function given by

\[ h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}, \]  

(4)

where \( \sigma \) is the standard deviation while \( u \) and \( v \) are the coordinates in the 2D filter. An example of this is the blurred image in figure 17b created from the convolution operation in equation 2 of the averaging filter and the original checkerboard image in figure 17a. In this case the Gaussian kernel gives a very similar result and is thus not shown separately.

Figure 17: Examples of a checkerboard image convolved with an averaging filter and Sobel filters in each direction.

The Sobel kernels perform an approximative 2D spatial gradient measure in an image. Typically it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image. Sobel combines a Gaussian smoothing and a differentiation in the same kernel. Thus these two operations is combined in the same convolution with the Sobel kernels. The results of applying the Sobel kernels can be seen in figure 17c and figure 17d respectively.

Defining the the image \( I \) one get the gradient approximations in respective direction,

\[ G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I, \]  

(5)
\[ G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I, \] (6)

from the convolution of the two Sobel kernels. From this the gradient magnitude is given by,

\[ G = \sqrt{G_x^2 + G_y^2}, \] (7)

and the gradient direction are calculated with equation,

\[ \theta = \arctan \left( \frac{G_y}{G_x} \right). \] (8)

These can then be used in gradient based edge detection techniques such as Sobel[21] directly using gradient magnitude or Canny[11] using both gradient direction and gradient magnitude as later introduced in section 2.3.

Moving away from the restrictions of the linear convolution operation also non linear filters exist. One such example is bilateral filtering which tend to be more edge preserving than normal Gaussian filters[20]. However, they also have longer execution time as they do not use the convolutional operation which can be efficiently optimized in implementation using the Fast Fourier Transform[20].

2.3 Canny edge detection

There exist a lot of different edge detection algorithms. The Canny edge detection algorithm[11] is one of the most commonly used edge detectors in practice[12]. It consists of four main steps.

1. **Filter to remove noise**
   To remove the noise from the image as this greatly impact the detection of edges the first step of of the Canny edge detector exist of smoothing the image with a Gaussian kernel as explained in section 2.2

2. **Filter to detect edges**
   After the image has been smoothed an edge kernel can be used to mark out all possible edges. However, a more common approach is to combine the smoothing in point 1 with the edge detection in this point into one step using Sobel filters as explained in section 2.2.

3. **Non maxima suppression of edges**
   After step 2 the image consists of the absolute value of the gradient and the gradient direction in each pixel given by equation 7 and equation 8 respectively. For each pixel only the pixel or its neighbors with the highest gradient absolute value along the gradient direction is kept as an edge pixel. This is called non maximum suppression as only the pixels with the strongest gradient magnitude along the gradient direction are considered to belong to the edge thus thinning the edges.

4. **Upper and lower edge threshold**
   The remaining pixels are then grouped such that all pixels with a gradient magnitude greater than the the upper threshold is considered strong edge pixels. All pixels with a gradient magnitude value greater than the lower
threshold but lower than the upper is considered weak edge pixels. Lastly, the pixels with a gradient magnitude lower then the lower threshold is considered non edge pixels. Thus, all strong edge pixels are considered edge pixels together with all weak edge pixels which has a 8/4-connected neighbor with a strong edge pixel while the rest will remain non edge pixels.

2.4 Binary morphological dilation

In fairly general and mathematical terms dilation is explained as a set-theoretical operation on binary images by Forsyth[8] as the following.

Assume two binary images $I$ and $S$ where each image is a representation of a set of elements belonging to a finite grid. In each image, pixels with value 1 are elements belonging to the set and pixels with value 0 does not. Let $S_p$ be the image obtained by shifting the center of image $S$ to the pixel $p$. The dilation of the set $I$ by $S$ is then defined as the new set,

$$I \oplus S = \{p : S_p \cap I \neq \emptyset\}.$$  \hspace{1cm} (9)

In less mathematical rigorous terms, dilation can be seen as a stamping operation where $S$ can be seen as a binary kernel/stamp of any size that is stamped on all value 1 pixels in image $I$ to thicken or fill out the image $I$. The dilation operation is illustrated in figure 18 where the binary image $I$ in (a) have been dilated with a 3x3 kernel $S$ resulting in (b) where black represent the original image $I$ and the gray all pixels added through the dilation.

![Figure 18: Before and after image I have been dilated with kernel S.](image)
2.5 Convolutional Neural Network

In recent years neural networks have become very popular, showing good results in many different areas of image analysis being able to learn features which would be hard to design by hand. One example is a network created from a collaboration between Google DeepMind and the University of Oxford resulting in a lip reading system able to beat professional lip readers[13]. This subsection gives a short introduction to the different layers in a convolution neural network to help the reader unfamiliar with convolutional neural networks to understand the underlying structures of the event classification done in this thesis. A multitude of literature can be found about neural networks. Thus this section is focused on the parts important to understand this thesis and is, if not otherwise stated, based on a good lecture series for the course CS231n: Convolutional Neural Networks for Visual Recognition taught at Stanford[10].

Neurons

The basis of a neural network is neurons built up in layers connected with weights, which are learned during a training process to learn patterns from training data. These neurons are connected with an input layer, a couple of hidden layers and an output layer as seen in figure 19.

Figure 19: Fully connected network showing an input layer connected to the first hidden layer and the second hidden layer connected to the output layer.

Figure 20 shows one of these neurons with the inputs $x_i$, weights $w_i$ and bias $b$. These are together forming an linear combination $\sum_i w_i x_i + b$ which is then set as input to a non linear activation function $f(\cdot)$ creating the final output of the neuron to be $f(\sum_i w_i x_i + b)$. A popular activation function for the hidden layers is ReLu given by $f(x) = \max(0, x)$. Another activation function mainly used in the neurons in the output layer when doing binary classification is the sigmoid function. It is given by $\sigma(x) = 1/(1 + e^{-x})$ and squash the output values between $[0,1]$. This can then be interpreted as the probability of the input belonging to one of the two output classes such that the probability of class 1 is given by $P(y_i = 1 \mid x; w)$ and correspondingly the probability of the other class is given by $P(y_i = 0 \mid x; w) = 1 - P(y_i = 1 \mid x; w)$. 
Convolutional layer

The first part of a convolutional neural network consists of one or more convolutional layers. One advantage of the convolutional layer is its use of shared weights, which improves scalability and makes it more usable for image classification. Simplified one could say that the convolutional layer are the feature extractor, learning weights for a group of filters such that they do not have to be designed by hand allowing for the possibility of finding structures in the data which is difficult to find otherwise.

Figure 21 shows how a convolutional layer takes a volume as input and outputs a volume activation map. The figure shows an example where the input consists of a $32 \times 32$ image with 3 color channels. However, it could just as well have been the an activation map volume from a previous convolutional layer with the format $[W_1 \times H_1 \times D_1]$.

Each neuron in the output activation map is connected to a local region in the input volume of filter size $F$ in width and height and in full in depth. Assuming a filter size of $5 \times 5$ for the example in figure 21 which has an input volume with size $[32 \times 32 \times 3]$, each neuron in the output is connected with $5 \times 5 \times 3 = 75$ weight parameters plus 1 bias parameter. These weights are shared such that they are the same for all neurons in the same depth slice of the output volume.

The size of the output volume, given by $[W_2 \times H_2 \times D_2]$, is decided by a couple of hyper parameters. The depth $D_2$ is decided by the choice of number of filters $K$. The output width $W_2$ and height $H_2$ depends on a combination of the filter size $F$ and stride $S$ and the amount of zero padding $P$ according to the input output specification below. In this setting stride means the step size of the filter when moving across the input and the padding is the extra border of zeros added as explained in section 2.2.
Figure 21: Illustration of a convolutional layer showing the input volume (image) to the left and the output volume to the right.

Input:
Volume of size $W_1 \times H_1 \times D_1$

Output:
Volume of size $W_2 \times H_2 \times D_2$ where,
$W_2 = (W_1 - F + 2P)/S + 1$
$H_2 = (H_1 - F + 2P)/S + 1$
$D_2 = K$

Hyper parameters:
$K$ - Number of filters
$F$ - Filter size
$S$ - Stride
$P$ - Amount of zero padding

Max pooling layer
The max pooling layer does down sampling of the activation maps from the previous layer. The most common way to do max pooling is using a 2x2 pooling filter with a stride of 2 as illustrated in figure 22. The pooling operation is run independently on each slice in the input volume as seen in the left half of the figure. The right half of the figure shows the operation on one of the slices. The first filter position is the upper left corner (red) containing a maximum value of 6. This value is the output from that position. The filter then moves two steps to the right (stride 2) to the upper right corner (green). Again the max value is added to the output. This continues throughout the entire depth slice and is repeated for all the depth slices.
Figure 22: Illustration of the operation of a max pooling layer as a down-sampling operation. On the left the change of the volume is illustrated and on the right the down sampling of each individual depth slice is shown.

Input: $W_1 \times H_1 \times D_1$

Output: Volume of size $W_2 \times H_2 \times D_2$ where:
\[
W_2 = \frac{(W_1 - F)}{S} + 1
\]
\[
H_2 = \frac{(H_1 - F)}{S} + 1
\]
\[
D_2 = D_1
\]

Hyper parameters:
$F$ - Filter size
$S$ - Stride

**Fully connected layer**
As the convolution layer does not output a classification but rather a bunch of features, a fully connected network is normally added to do the final classification of the input volumes/images. A fully connected layer is basically what the name suggests, a layer in the neural network where each neuron in the layer has a connection to all neurons in the adjacent layers as was previously illustrated in figure 19.
3 Developed tools

Tools were developed to give a deeper understanding of the data, to enable creation of needed datasets and to evaluate the effects caused by changes in the extraction method. Additionally, they were also used to guide the method development towards the parts with most potential for improvement. In this section two of these tools are presented together with some test results enabled fully or partly by these tool.

3.1 Visualization and event marking tool

Searching for many small insect events manually in big images to be able to create data sets and evaluate the performance of any extraction method is difficult for a multitude of reason. Firstly, it is very difficult to get an overview of the events one is looking for due to the large image size. This makes it difficult to systematically go through and mark the positions and bounding boxes of the events. These bounding boxes are important to manually count the insect events in each big image frame, such that one can conclude if all events that should be detected actually are detected. Secondly, when the events have been extracted by the tested method, it is impossible to access if an event was split or not, solely based on the information in the extracted event. Thus an overview with bounding boxes is again useful to be able to detect these splits. Lastly, there are no simple way of getting a global overview of the positions of all the found events. This could be very useful when accessing if some problems are more frequent in some parts of the image than other parts, as the non linearity of the Scheimpflug principle might make the extraction problem more difficult in some parts rather then others.

To improve on the previously mentioned problems, an interactive visualization program was developed and implemented. The visualization program was written as an interactive GUI in python with the help of the matplot libray[31]. The GUI consists of 4 main windows as shown in figure 23.

Figure 23: Visualization tool consisting of four windows.
In the top left is an overview window in which the entire large image is shown to give an broader overview. In the top right corner is a zoomed in window representing the area in the overview window marked by a black box. The lower images are visualizing the current selected event where the right image is the normal cut out of the event, while the left plot is a 3D surface plot of the event.

1. **Overview window**
   The overview window allows the user to move through the entire big image systematically in steps or navigate over it directly with a mouse and zoom tools. In this window previously manually marked events can be displayed with red boxes and can then be compared with blue boxes representing events found by the segmentation method.

2. **Zoom window**
   The zoom window shows the cut out of the current step when systematically going through the entire image to manually mark new events. It is also in this window the red bounding boxes for manually marking events can be created. By clicking event boxes, the content of that particular bounding box is shown in the lower windows.

3. **3D event window**
   The 3D event window shows the 3D surface plot of the selected event. This can be useful as the event intensities had a large range and varied a lot which can make it difficult to determine the event borders from the 2D event image alone. It can also be helpful enabling a complementary view of the selected event.

4. **2D event window**
   The 2D event window simply shows a cut out of the currently selected event.

Two important and useful features of the visualization tool is its merge and split warning system. These use manually marked event boxes(red) and the event boxes(blue) marked by the segmentation method. If the manual event box is overlapped by two or more boxes found by the method, it will give a warning of a potential split(yellow) as shown in figure 24. When an event box found by the method is overlapped by two or more manually found event boxes, it is considered a merge warning(yellow) as shown in figure 25. Additionally, events not found by the method, meaning a red box not overlapped by any blue box, can be marked with black boxes as can be seen in figure 26.
Figure 24: Event split warning marking potential splits with a yellow box.

Figure 25: Event merge warning marking potential merges with a yellow box.

The split warning was of particular importance, as splits can not be detected manually once the event images have been extracted, unlike the merges. Also important to note is that these warnings are based on how detected event boxes and manual ground truth boxes overlap. This is not sufficient to claim that an actual merge or split has occurred and has to be checked manually to access if they truly are merges or splits. Hence, it is only a warning system. Although using the warning system limits the search space of finding merges and splits.

On top of this there is also a feature marking the ground truth boxes as red or green depending on if they have been rated good or bad, according to a rating scheme later explained in section 3.2.
3.1.1 Detection results and discussion

Are enough events found by the old method?
To indicate the detection performance of the old method a first test was run, counting the number of events found as a function of the intensity threshold parameter of the old method. This was done for a couple of big image frames out of which one example is showed in figure 27. The general look for the rest of the plots follows the same trends as in the example. They all start with zero events found for the lower thresholds. This is the case when the threshold is low enough, such that the whole image is considered to be the same event. The zero event range is followed by a rapid increase in the number of events that later turn into a rapid decrease, together creating a hill. This behavior can likely be explained by a lot of noise being considered separate events at a certain point, leading to a rapid increase that peaks when most of the small noise peaks are considered separate events. The following fast decrease could then be when more noise starts ending up below the intensity threshold. This fast decrease ends in a more slowly decreasing line along which the true number of events could be found. With the intensity threshold set to 6 in the old method, 288 events where found in this example image as marked by a circle in figure 27. This can be compared to 316 manually marked events found using the visualization tool. This means that 91 percent of the manually marked events were found. Repeating the process for a total of four big image frames from different setups and/or time of the they when recorded, 90 percent of the manually marked events were found on average. Using the not found events warning in the visualization tool showed in figure 26, it was easily realized that a majority of the events not found had a very weak intensity. Furthermore it could be questioned if they should have been marked as events in the first place and would very unlikely be considered useful for later processing steps. The remaining difference could possibly be explained by the presence of merges and splits effecting the count.
Figure 27: Intensity threshold effect on the number of events found using the old method. Marked is the number of events found with the threshold used in the old extraction method.

The presented indications of the detection performance at current threshold values was considered good enough, to be able to focus the development effort towards the segmentation instead. However, something that can not be concluded from the plot is how big of a problem splits and merges might be.
Merge and split effect depending on number of dilation iterations

Enabled by the merge and split warnings, a very limited but indicative test was done to see the effects of switching the old border selection step explained in section 1.4 with a dilation using a 3 times 3 kernel repeated for a set number of dilation iterations. The results are shown in table 1 and can be compared to the old method resulting in 10 splits and 9 merges in the same big image which contained 231 manually found events. The first thing to note is that there is

<table>
<thead>
<tr>
<th>Nr iterations</th>
<th>Nr splits</th>
<th>Nr merges</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>45</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>19</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 1: Effect of number of dilation iterations

a trade-off between the number of splits and merges, such that more dilation iterations gives more merges but less splits. This makes sense considering more iterations give bigger borders and thus a higher risk of merging two events. The second thing to note is the big difference in the number of split and merges between the different number of iterations. However, in addition to this (not seen in the table) the number of dilation steps needed to give visually good segmentation boarders varied a lot between the events in the same tested image. This means two things. Firstly, the number of iteration need to be adaptable to each separate event to get desirable boarders. Secondly, it is not possible to find one set number of iterations for the full image to avoid splits and merges.

Based on this reasoning the suggested segmentation method, described in detail in section 4.1, was designed in two separated steps. One step finding the events, thus deciding which insect wing peaks belonged to which event, with the task of regulating the splits and merges and one step finding more precise borders. In addition to this the number of dilation iteration needed to be adaptive to each event.

3.2 Event rating tool

To ensure that the events extracted by the tested method actually had the possibility of being useful for the later stage of insect classification, a rating of the manually marked events was conducted. This way focus could be put on making the segmentation method work well on useful events with characteristics such as not being false positives, to weak or have other undesired properties.

The rating was done by four persons at FaunaPhotonics, all familiar with the appearance of the event images. A total of 547 manually cut out events from two different big images created at two different field tests was rated according to a scale from 0 to 5. Crossing events was rated with a 0, while all other events was given a value between 1-5 depending on the following criteria:

1. Existence of WBF
2. Number of clear peaks
3. Visible insect event body
4. Signal to noise ratio

To help the expert raters to evaluate the rating criteria when giving their rating scores, a rating program was created. For each event the plots shown in figure 28 was shown to the raters to base there judgment on. This consists of five windows containing useful information to evaluate the rating criteria.

![Figure 28](image)

**Figure 28:** Event rating tool consisting of five windows containing relevant information to support the raters decisions.

1. The manually marked event cut out.
2. 3D surfacec plot of the manually marked event cut out.
3. The event summed at each time instance.
4. 2D FFT.
5. FFT of the summed signal.

**Event rating scale**

To better understand where along the 0-5 different events ended up, figure 29 shows a scale with examples of events positioned at their corresponding average rating scores. Note that only the 2D event image is shown and other plots used for basing the rating assessment on, such as the Fourier transform, are not.
3.2.1 Event rating results and discussion

The resulting ratings for all the events done by the four raters can be seen in figure 30.
The first thing to notice is the resulting ratings of all the rater have similar distributions of the rated events, even if some are stricter raters than others. Also interesting to notice is that there are more events with lower ratings than event with the higher ratings, indicating that there are lots of events that could be considered bad or not useful for later the classification stages.

By setting a rating threshold for when an event should be considered good or bad, the disagreement among the raters could be explored. Using a rating threshold of 2.5 and exploring the events which did not have full agreement regarding it being considered as a good or bad event, four different cases of disagreement was found. The ratings given by the four raters are presented in the format (rater 1, rater 2, rater 3, rater 4).

1. (2,2,3,3) Evenly distributed ratings that happen to end up on different sides of the threshold. This disagreement does not pose any problems and is a natural disagreement that just happen to end up on two sides of the chosen rating threshold.

2. (2,2,2,4) Consensus with one outlier rating disagreeing with the others. This could be explained by the tiering process of rating over 500 events and risk of loosing concentration with the consequence of some wrong ratings.

3. (0,0,4,4) There were some examples where it was unclear if to rate the event as a crossing or not. In this case, some participants rated it as a crossing and others based their rating on the normal rating criteria.

4. (2,3,4,5) For a few events there where complete disagreement. This was often the case for long and very weak events with a clear frequency. Then the difference in preference between clear frequency, length and SNR was more divided between the participants. However, when later discussed the consensus turned out to be approximately the mean of the ratings.
These disagreements justify using more than a single rater to acquire a more reliable classification of good and bad events. The final rating is then given by following the consensus of the raters rather than one rater that might be biased towards putting too much weight on some particular rating criteria. Taking the average rating of all the events resulted in the ratings seen in figure 31. The final rating threshold was set to 2.5, which is set low on purpose not to miss any good events, even if some higher rated events probably should not be considered to be good ones.

Figure 31: Average rating of the event images.
4 Suggested extraction method

In this section the suggested methods for doing detection, segmentation and merge classification are presented, followed by their respective test results and discussions about these.

4.1 Detection and segmentation

The suggested segmentation method combines edge and intensity thresholded data to find a first foreground and background segmentation with high confidence in the pixels being considered foreground. Parts of the old method then decides what belongs to each event and sets a maximum border around each event. For each of the cut out events, a dilation is done to fill out holes and parts of the event that might have been missed. The number of dilation iterations is decided by a stopping criteria based on background noise statistics. The full method is illustrated in figure 32 accompanied with a more thorough description of the method.
Figure 32: Suggested method for detection and segmentation.
1. **Pre-processing**

Pre-processing steps that create the big image used as the starting point in this thesis. This big image contains all the events that need to be extracted by the method.

2. **Edge detection**

To find all the edges of the events in the big image, the Canny edge detection algorithm described in section 2.3 is used. A simple adaptive upper and lower threshold based on the standard deviation of the background noise is suggested, as the background noise is constant across each big image, but varies depending on other environmental circumstances such as sunlight. The upper and lower thresholds are then given by,

\[ U_t = C_1 \sigma_{\text{back}} \]
\[ L_t = C_2 \sigma_{\text{back}} \]

Where:

- \( U_t \): Upper threshold value
- \( L_t \): Lower threshold value
- \( \sigma_{\text{back}} \): Standard deviation of background noise in the big image calculated from an area in the image known not to contain any events
- \( C_1 \): Constant upper threshold
- \( C_2 \): Constant lower threshold

3. **Intensity threshold**

The big image is thresholded based on the intensity of each pixel. The focus of this step is to include as much as possible that comfortably can be considered to be part of the insect event without also including any noise. As the intensity threshold of the old method explained in section 1.4 have been verified to finds all except very weak events, this is used as a part of the suggested method.

4. **Find borders**

The elliptical borders of the old method is also used in the suggested method. However, in the suggested method these borders are not used to decide the final pixels that are considered to be foreground. Instead they are used as a first mask deciding what parts of the old image belongs to each event.

5. **Combining edge and intensity threshold data**

To include as much as possible that comfortably can be considered to be foreground, without also including any noise, the edge and intensity threshold data is combined with an OR operation.

The intensity thresholding is good at finding the peaks in the image. It is also less sensitive to the noise compared to the edge detection as finding edges are based on image gradients. Thus finding edges require the image to be more heavily smoothed which in turn require events to have higher
intensity if its edges is to be found. On the other hand, edges more precisely outline the borders of the events when they are found.

This means that a combined image of intensity threshold data and edge data gives a larger number of pixels that comfortably can be considered to be foreground, compared to using only one of them. The intensity and edge data also give complementary information about the events.

6. Masked edge and threshold image
The border mask from the old method, created in point 4, is combined with the foreground data through an AND operation. This results in event cut outs containing the edge data combined with the respective intensity thresholded data for all the events detected in the big image.

7. Connected regions
The image in point 6 contains a separation of foreground and background pixels. The foreground pixels consists of separated islands which represent the different insect events. A built in method in OpenCV[5] looking for connected components is used that divide the foreground pixels into separate regions based on if the pixels in the regions are connected or not. Thus all foreground pixels that are connected to at least one other foreground pixel in the same region is considered belonging to the same region. The result is a collection of all the separated event cut outs.

8. Finding tighter border step
From previous steps each event have been cut out and consist of a set of pixels which is considered to be known foreground pixels for that event. To fill out all the events and find the final tighter borders without missing any part of the event, each event is dilated with a 3x3 kernel for a number of iterations depending on a suggested statistical stopping criteria further explained in point 9.

9. Dilation stopping criteria
The known foreground from previous steps is dilated with a 3x3 dilation kernel. Due to high variation in the size and shape of the insect events, the number of dilation iterations needed too fill out the missing parts of the event varies greatly. Thus the stopping criteria

$$\frac{\lambda * \sigma_{btot}}{\sqrt{n_b}} - \mu_{event} \leq a = stopping\ threshold,$$

(12)

is used where,

- \(\lambda\): Constant deciding number of standard deviations included in the confidence interval.
- \(\sigma_{btot}\): The standard deviation of the background in the full image, calculated from behind the focus line where there are no events.
- \(n_b\): Number of sample background pixels in the cut out event image.
- \(\mu_{event}\): The mean calculated from the background pixels in the event.
The stopping criteria is based on the known characteristics of the background noise being additive zero mean Gaussian noise with a constant standard deviation across the same frame. Thus the calculated background mean for the current event should be zero when no foreground pixels are included in that selection. However, the calculation of the background mean for the current cut out of an event is based on a varying amount of background pixels depending on the event. To compensate for this a confidence interval for the Gaussian mean is used. An image series illustrating the dilation iteration, from the first foreground selection to the final foreground selection where the dilation stopping criteria is met, is shown in figure 33.

Figure 33: Dilation iteration stopping criteria example showing how the segmentation is grown until the stopping criteria is met.

10. **Stopping criteria insurance**

The final step is to mask the dilated events with the first image mask from the old method once more. This is done to make sure that even a failure of the stopping criteria will yield the same segmentation as the old method in the worst case. After this step, the final detection and segmentation is finished for all the events and they are ready to be sent to the next step in the process which is the merge classification.
4.1.1 Border segmentation test results and discussion

To compare how well the suggested segmentation method finds the segmentation borders of the insect events compared to the old method, the Jaccard index as explained in section 2.1 was used. The comparison was done on 35 insect events chosen based on being rated as good events by the event rating procedure explained in section 3.2. Each used insect event was manually cut out to create a golden standard, which was used with the segmented insect events for each of the tested method respectively, to calculate the Jaccard index for the two methods.

![Figure 34: Jaccard index of 35 good events calculated for the old and the new segmentation method.](image)
The results of the test is shown in figure 34 with the calculated Jaccard index for each of the tested insect events. The first important thing to note is that the new method always gives a higher or equal Jaccard index compared to the old method. Event 7 in the figure shows an example where both methods result in the same Jaccard index. This is by design and is thus expected, as the old method creates a border mask which the dilation process of the new method are not allowed to exceed, as explained in section 4.1. This works as an insurance against changes in noise levels or other unexpected effects in future optic setups, which could cause an increase in spurious edges risking very big over segmentations. Secondly and perhaps most important, is that the new method gives an a Jaccard index that is almost 40 percent higher on average, clearly showing an improvement in the segmentation for the 35 tested good events.

Figure 35: An event where the new method showed a substantial improvement.
To further illustrate the segmentation performance improvement achieved by the suggested method, the segmentation of the insect event corresponding to event 3 in figure 34, is separately displayed in figure 35. The lower images show the pixels that are considered to be the insect event and are marked in yellow for the manual cut/golden standard, old method and new method respectively. The upper images show the actual event cut out for each method. As this example is the most extreme case tested, showing the biggest difference in the Jaccard index between the old and new method, it clearly illustrates a case where the new method gives a more precise segmentation while the old method clearly over-segments severely.

However, there are still cases where the new segmentation method does not perform as well as in the example. Figure 36 shows one type of event where the idea of using a dilation stopping criteria has some difficulties. The right part of the figure shows the insect event cut out and the left part the 3D surface of the insect event cut out. Observing the figure one can see an event with asymmetric sides. That means that it has high intensity peaks, which slowly gets weaker on one of the sides and faster on the other. This asymmetry leads to extra noise being added on the side with a rapid intensity change. The reason for this is the dilation process, which does not stop its iteration until the broader and weaker side is also included in the segmentation. Unfortunately this leads to extra noise being added on the side with a strong edge due to the strong side also getting dilated even if it does not need to be dilated.

Figure 36: Illustration of the result when the dilation process can not handle the asymmetry of the event such that unnecessary noise is included on one side of the event to be able to completely fill out the other.
Problems with a stopping criteria based on background pixels
The stopping criteria is based on statistics of the background pixels in an event image, as the background noise distribution is well known and easily measurable. However, this means that the dilation process only stops when it is confident enough that there are no foreground pixels labeled as background pixels, but not the other way around. Consequently, it does not actually stop the dilation based on the amount of noise/background pixels that are labeled as foreground pixels, even if the goal is to minimize the noise being wrongly included in the foreground. An example illustrating the consequences of this have already been shown in the one sided oversegmentation in figure 36.

Need for edge data and better gradient filters
During the thesis non linear bilateral filters was tested instead of the linear filters that is originally used in the Canny edge detector, due to its ability to smooth images while preserving edges [25]. However, bilateral filters are harder to optimized compared to the linear filters even if there are several attempts of doing it such as [26]. An implementation of bilateral filters was readily available in OpenCV[5] and was therefore tested to improve the edge detection. Unfortunately, the initial testing showed the execution time increased by a factor of 30 compared to the full original Canny edge detection, and was therefore considered too slow compared to the minor improvements that was visually indicated.

A more fundamental question to ask, is how much the use of edge data actually contribute to the segmentation performance of the new method. During the development, the first prototype method that showed good potential, only consisted of the edge data and the dilation stopping criteria process. At that point the edge data was needed. However, this prototype method was later combined with parts of the old method enabling it to draw closer borders without resulting in an increase in the number of splits. At that point the usefulness of the edge data decreased as the intensity threshold added more information to the foreground and background segmentation. Visually it was still clear that the edge data contributed with complementary information to intensity threshold data and was therefore still considered useful. However, to be able to guarantee that there were no spurious edges in data with higher background noise, the lower and upper thresholds in the Canny edge detector was set at a level where a lot of edge information in normal events were lost, consequently reducing the usefulness of the edge data. If the thresholds are set too low, there is a risk of noise being considered edges leading to spurious edges and in the worst case the suggested segmentation method would result in the same segmentation borders given by the old method. Additionally, the edge data mainly contribute with information about the outer part of the event. Thus the dilation process might even be able to fill these parts of the event by itself, but without the risk of spurious edges.
Need for more scalable segmentation measure

Using the Jaccard index together with focusing on the good events gave a clear result in favor for the suggested segmentation method as was seen in figure 34. Restricting the performance measure to the good events mean that the measure is focused on the methods ability to segment the truly important events. However, these events are likely easier to segment due to higher intensities and clearer borders. The risk is that the performance is a lot worse on the less good events. Although, that would likely be the case for both the old and new method and the importance of being able to segment a bad event is not as high. It is also likely that there would have been more error in the manual segmentation if done on bad events as their boarders are less clear, consequently leading to more uncertainty in the Jaccard index.

The big drawback of using the Jaccard index is the need to manually create the golden standard segmentations. This is very time consuming even with the help of the developed tools. It consists of finding the events, rating them, marking their exact boarders manually in an image editing program such as the one used for this thesis called Pixelmator[32] and sometimes even reiterate parts of the process e.g. when realizing that the manual segmentation was not good enough when looking at its 3D surface plot. This limits the scalability of the measure and made it unfeasible to test more than a limited amount of events within the time frame of this thesis. Thus the lack of scalability meant that the segmentation performance test was unable to include more possible variation between the frames in the results.

To improve the scalability, it was considered to create a mathematical model describing the events precise enough, such that it could be used instead of a manual golden standard. Unfortunately, due to the large variation in shape and size of the events, this proved to be very difficult to model precisely enough such that it would be useful.

Another explored option was to base the performance measure on data from 1000 events with validated WBF. The measure would then be based on how close to the real WBF the results came. However, this would include some intermediate steps used when finding the WBF which would effect the result and might even be changed later. Also this would only be looking at the WBF which is only one of many needed features later needed for insect species classification. A good result with this measure would likely result in tight segmentation boarders, such that only the insect event peaks are considered foreground as this would make it easier to find a clear WBF from the insect profile. Consequently, other aspects like the insect body might then have been completely discarded.
4.2 Merge classification

In this subsection a machine learning based method is suggested to do merge classification. The suggested method consists of a convolutional neural network (CNN) structure that is presented, followed by classification test results and related discussions.

4.2.1 Dataset creation and CNN architecture

The suggested segmentation method described in section 4.1 resulted in an improved segmentation performance. However, the problem with splits and merges still remained the same as for the old method. To better handle this, a dataset of merges and non merges was created and used to train a convolutional neural network (CNN) that learned to do merge classification. By knowing if the event is a merge or not the system can discard the event such that it is not sent to the insect species classifier. Alternatively, the detected merge can be analyzed manually or sent to a method that can potentially split the event into its respective parts.

Merge data set creation

To enable the use of a training based classification method to classify the insect events as merges or not, a dataset consisting of 13550 non merges and 832 merges was manually created from events segmented by the suggested segmentation method described in section 4.1. This means that approximately 6 percent of the classified events were manually classified as merges. Even if 6 percent merges might not be considered all that bad, it is still high enough that it needs to be handled in some way. Furthermore, it should be noted that the data set was created using method parameters leading to the same amount of splits and merges as the old method. This means that the company may later decide to increase the ellipse border threshold to avoid more of the splits with the drawback of an increase in the number of merges. This uncertainty further motivates the need for a merge classification system, such that merges can be completely discarded or sent to a method with the ability to split them up into their corresponding parts.

As an attempt to avoid over-fitting to data from one particular field test and capture time, data from three different field tests was used with images sampled at different times in the day. However, due to a large variation of the number of events in each big image, there might still be a risk that events captured at a particular time of day and field test setup dominates the data. This could also mean a risk of certain insect species dominating the data which might cause the CNN to over-fit to those types of insects.

Another potential problem with the created data set could be events that have been wrongly classified during the manual classification. Even if the majority of the insect events could clearly be distinguished as being a merge or not, there are examples where it is easier to make a mistake. One example of this is insect events following each other in the same direction, posing difficulties to know if they are two separate events or just have some part of its insect body being weak. A second example is when an event is slightly duplicated due to optical
effects such that one might think it is two events merged.

Event image input format
To work with a CNN, the insect event images had to be resized and reshaped to the same size. This means that depending on the original size and shape of the extracted event image, the reshape and resize might impact the events differently due to a large variation in the need of compression. Another side effect is that the size of the input image also effects the number of weight in a given CNN structure, as can be seen by the input output size described in section 2.5. For these two reasons the CNN was trained on both 28 times 28 and 64 times 64 event images to observe if any substantial change could be observed. Also testing on larger image sizes was considered, but discarded due to the increase in training time.

Suggested network architecture
The suggested network structure was based on similar architectures such as LeNet-5[30] that performs well on classifying handwritten digits from the popular MNIST dataset[18] whilst still being a relatively simple structure that is fairly easy and fast to train. Using a fairly simple structure with less weights also limits the risk of over-fitting caused by a limited amount of training data. The idea was that using a network that could perform well on handwritten digits by being able to learn their shapes, could also be able to learn if an event was connected in a merge or not.

![Figure 37: Suggested CNN architecture.](image)

The proposed convolutional neural network architecture is shown in figure 37 and was implemented in the open source machine learning framework TensorFlow[33]. The network is composed of two convolutional layers combined with two max-pooling layers and ReLu activations. This part is able to learn features and consider spatial information from the training images. This is followed by a fully connected network with the task of doing the final classification of events, classifying them as merges or non merges. It uses ReLu activations on the hidden layers and a sigmoid function as activation on the output layer. The sigmoid activation function produce outputs that correspond to the probability of the event being a merge or not. This is needed to enable it to be used to create ROC curves by changing the probability threshold of when to classify it as a merge where each threshold value corresponds to one point in the ROC graph as explained in section 2.1.
4.2.2 Merge classification test results and discussion

To do the final evaluation of the classification performance, the CNN was re-trained many times with different combinations of the training and test data to create ROC graphs as described in section 2.1.

**Training and creation of ROC graphs**

The created merge dataset was randomly divided into a test set containing 30% of the data and a training set consisting of 70% each time a new training session was run. By doing this one could train the network several of times and make sure that the network yielded similar results independent of what part of the data that was used for training and testing.

As previously mentioned in section 4.2.1, approximately 6 percent of the entire dataset was labeled as merges. This means that the dataset is imbalanced with an uneven amount of training data from the different classes. This can cause the network to over-fit to the majority class during the training phase[9]. There are several of ways to try to overcome this class imbalance problem, where a couple of them are described in "A systematic study of the class imbalance problem in convolutional neural networks"[9]. The two most common ways described is random minority oversampling, where the minority class is oversampled, and random majority under sampling, where the majority class is under sampled. Out of these two the the random minority oversampling is tested in this thesis. Also a more data specific approach is suggested, where artificial merges are created by randomly merging two non merged events into a single merged event. These artificial merges are then added to the existing training data to make it more balanced.

Imbalanced data does not only effect the training of the CNN but also the choice of how the evaluation metric should be chosen. Assuming that the test data consists of 6% merges and 94% non merges as for in the case of this thesis. In this case the non merges will effect the final test accuracy to a greater extent than the merges. This can be exemplified by assuming a classifier that simply classifies all events as non merges. A test accuracy of 94% would then be achieved, as all none merges would be classified correctly while all merges would be wrongly classified. This seems like a rather high score. However, it is extremely misleading as the classifier actually has not done anything useful. Exactly the same events would have been sent to the insect species classifier as if no classification had been done. For this reason and to better see the trade off between finding merges and loosing non merges by classifying them as merges, ROC graphs was concluded to be a better approach to evaluate the classifier. However, an imbalanced test data set also effects the ROC graphs. The true positive- and false positive rates are given by a percentage of the tested events(merges) and the other class(non merges) respectively. In the case of this thesis this means that a value on the true positive rate axis compared to the same value on the false positive axis, corresponds to a smaller amount of events as there is less test data available for the merges. On the other hand this is not necessarily considered a problem. However, it should be noted when reading the ROC graphs.
To evaluate the effect of the size compression, random minority oversampling and the use of artificially created merges, the four ROC graphs displayed in figure 39 was created from running the trained networks on the test data each run. These were created by running a training scheme consisting of 200 epochs, which was enough to get an accuracy on the training set of approximately 100% every time. This resulted in a slight over-fitting as can be seen by the difference of approximately 2% between the training and test classification error in figure 38 at the end of the training. However, as the difference was considered fairly small, this was not considered further in this thesis.

Figure 38: Slight over-fitting during training as can be observed from the difference in the training and testing error at the end of the training.
The network was retrained 60 times for each configuration with a different split of the test (30%) and training data (70%) each time. Thus each ROC curve in figure 39 have a blue center curve representing the average of all 60 training sessions, a gray and a yellow interval band representing one, respectively two standard deviations from the average curve in the true positive rate. In the graph the true positive rate axis gives the probability of a merge being correctly classified as such and the false positive rate the probability of a none merge being classified as a merge. In general all the ROC curves in figure 39 show surprisingly good results considering the large variation in appearance of different events and multitude of possible ways to combine them into merges. Note that all statements of how good results that are achieved in the ROC graph, are based on looking in the range 0-0.1 in the false positive rate as higher false positive rates would likely never be allowed by the company.

**Does larger image size improve the classification results**

Comparing the ROC graph created using 28x28 event images in figure 39a with the graph created using 64x64 event images in figure 39b, one can observe that the 28x28 images resulted in a higher true positive rate in the range 0-0.1 of the false positive rate. In addition to this, the curve for the 28x28 images showed less variation between the training runs. This is contradicting the expectation.
that less compression of the image size, would remove less information in the image and thus give a better, or at least equally good classification. However, this could perhaps be explained by the increase in number of weights in the CNN for the larger image size in combination with allowing the training phase to over-fit the training data. This means that there are more parameters that might adapt closer to the data in the training set, but not generalize as well during the testing. E.g. with enough weights each image in the training set could potentially have its own set of weights recognizing only that particular event giving a high test accuracy. The result of this is a network that has difficulties with events it has never seen before, such as the ones in the test set. Another explanation could be that more features, such as wing beats, are more likely to be visible in an image with less resolution compression. The network might then use some of its weights to learn these features that might be less relevant to classify an event as a merge or not. With the same type of reasoning, the images that are more compressed might not have enough resolution to show more than the general shape of the events, without any details such as wing beats, allowing the network to focus better on the more important features for doing merge classification.

**Improvement using oversampling**

Comparing figure 39a and figure 39c one can observe lower true positive rate values, for in the interesting range of 0-0.1 of false positive rate, when using oversampling. One possible explanation could be that as the same merge images are being used more than once during the training phase, the network is less likely to over-fit towards the dominating non merges. This could potentially lead to an increased number of non merges being classified as merges, while simultaneously keeping the classification of the merges the same, due to not actually adding any new data to help it improve. This would mean a lower true positive rate for the same false positive rate as more non merges would be classified as merges.

**Improvement using artificial merges**

Comparing figure 39a and figure 39d one can observe higher true positive rate values, in the interesting range of 0-0.1 of false positive rate, when artificially created merges have been added to the training set. This is reasonable for two reasons. Firstly, unlike using oversampling, the artificial merges actually add new data to the training set consisting of merge combinations of events otherwise not seen by the CNN. Secondly, the created merges are very similar and could not easily be distinguished from the real merges. However, that is only true when referring to none overlapping merges and simple merged events that are not combined in some all too strange way. Comparing figure 40a with a real merge and figure 40b with an artificial merge, one could not distinguish which one is real and which is created artificially. Thus the improvement in the ROC graph is likely and improvement in the classification of these simple type of merges, while the persisting error consists of other type of merges such as overlapping or other more difficult merges.
Figure 40: Comparison of a real non crossing merge and artificially created merge.

Miss-classified examples
To get a better understanding of when the CNN has difficulties in detecting merges, six examples of event images that were wrongly classified as none merges, even if they were manually classified as merges, are shown in figure 41. The chosen examples are chosen such that they represent some typical categories of when the CNN has difficulties.
(a) Optical merge  
(b) Events following each other merge

(c) Merged side by side  
(d) Merge long and short

(e) Merge head and bottom connected  
(f) Merge strong and weak

Figure 41: Merges miss-classified as non merges.

(a) This image shows an example of an event image being classified as a none merge, which would actually be the correct classification. However, this event image was wrongly classified as a merge during the manual data set creation. The image looks very much like a merge and it is thus understandable that it was manually miss-classified as such. Especially considering that creating the data set manually was a quite tiering process
going through thousands of events. Consequently, there is a risk of more events being miss-classified in the data set. These images are then event images that looks very similar to a merge, even if it is really an optical effect caused by a not perfectly calibrated instrumental setup. These can often be spotted by noting that the two events that have merged look almost identical, as the merge is caused by the same insect being duplicated by the optics. Although important to note, this type of miss classification errors in the created data set is estimated to be quite rare and should have no substantial effect on the results.

(b) This event image shows two events following one another and going in a very similar direction. This case proved to be difficult, as it is very similar to when there is one insect event which have a missing or very weak body, such that it consists of non connected peaks following each other. One way to attempt to see the difference when manually creating the data set, is to look at the shape of the wing beat peaks. These typically look slightly different when the merge consist of two different events, while they often look very similar, or even only consist of one peak, when they are considered belonging to the same event. Thus there exist a relatively well working strategy when doing it manually even if it is difficult for the CNN. Using a larger image size than the current 28x28, could potentially help by including more of the peak details, helping the CNN to learn a similar strategy to the one used when doing it manually. However, as previously discussed, the ROC graphs created using a larger image size did not indicate better classification performance.

(c) This event image shows two events going side by side in a similar direction with a slight overlap causing them to merge. Speculating about a reason why the CNN wrongly classified this event as a non merge is more difficult than the two previous examples. A qualified guess is that it has some resemblance of the image in figure 41a, which unlike the one mentioned here normally are classified as a none merge during the data set creation, thus fooling the CNN to believe that is also the case here.

(d) This event image shows one very long event being merged with one that is a lot shorter but placed very close to the longer event. One interesting reflection on this combination of events is that the way that the smaller event is connected to the longer, makes it look similar to the appearance of wing beat peaks on some types of insect events. This resemblance could be the reason why the CNN wrongly classified this event image as a none merge.

(e) This event image shows two events merged by the head of one event being connected with the bottom of the other. Furthermore, the two events have a similar directions. This likely contributes to it being classified as a none merge by the CNN, as this makes it difficult for the CNN to detect the difference between this event image and long insect events with a slight curve.

(f) This event image shows two events merged where one has a substantially stronger intensity than the other. The domination of the stronger event
might lead to the CNN missing the weaker one as it might only be con-
sidered to be noise, which in turn leads to miss classifying it as a none
merge. In later steps when extracting the WBF, the weaker event might
not have a great impact on the result as the image would be dominated
by the stronger event. On the other hand the weaker event would then
be missed entirely compared to if it was not merged with the stronger one
effecting the insect count negatively.
5 Summary and Conclusions

This thesis explored ways to detect, segment and classify merges of insect events captured in large 2D time-range maps by a Scheimpflug based LIDAR system. This required the development of tools for visualizing and rating the insect event to help understand the data and better and guide the development and method testing towards the parts with most potential for improvement. This led to the development of a detection and segmentation method as well as a method for classifying merges. The detection of insect events is still done by the old segmentation method as it was able to find all events except useless weak ones. The remaining problem was and to some extent still is how to minimize the number of splits and merges. This led to the need to be able detect merges by classify events as merges or non merges. With the suggested CNN structure, surprisingly good classification performance have been shown in the ROC graphs. Further using artificial merges to handle the class imbalance problem also showed improvement, while oversampling and less image size compression did not indicate any improvement.

The segmentation method was divided into two separate steps, one finding the events and deciding what belonged to an event and one step finding closer boarders of the event combining intensity threshold data, event edge data and using dilation with a background statistically based stopping criteria. This resulted in a segmentation performance improvement based on the Jaccard index.

Not to be forgotten are the developed helping tools which successfully guided the focus of the thesis, enabled rating of events directing the choice of segmentation test set, was used to create all important data sets, helped in detecting splits and merges, sped up the method evaluation process and visualized the data in a useful way. Without them this thesis would not have been possible. On the downside they took some time to create, leaving less time to use the benefits they brought to create larger test sets to include more variation in the tested data and less time to develop the extraction method.
6 Future improvements

During the course of the thesis many possibilities for improvements were discussed. However, some improvements and investigations had to be set aside to allow for time to focus on more important issues. This section gives a short overview of potential improvements or areas that would benefit from further exploration.

Detection

With indications of good enough detection results early in the thesis, the old method was kept even in the new suggested method to do the detection and it was thus not further investigated. Thus, exploring methods that could find the same amount of events while minimizing the number of merges and splits would be beneficial to the event extraction system.

Segmentation

Using the Jaccard index as the segmentation performance measure limited the segmentation evaluation to a small subset of the data. Therefore it would be beneficial to have a segmentation measure which does not require manual segmentation to create a golden standard. This could possibly be based on some mathematical model that models a true insect event signature. However, it would likely be very difficult to make a model precise enough to be useful due to the large variation between insect events. On the other hand, if a good solution was found it would allow for a better scalability of the segmentation performance tests.

As showed in section 4.1.1, the suggested segmentation method had difficulties with one sided over segmentation in asymmetric events, due to the dilation stopping criterias inability to restrict the dilation to a limited part of the event image. Thus exploration of a method that could restrict or focus the dilation to a limited part of the event image would have potential to further improve segmentation performance. One idea could be to detect the asymmetry by analyzing the event profile or possibly use the data about the strength of the edges and adapt the dilation process to more heavily dilate some part of the event image based on this information.

Classification

The first step in improving the merge classification would naturally be to combat the slight visible over-fitting not yet taken care of. There are several techniques that could be used to try to do that, such as drop out[34] or early stopping[35]. Additionally, it would be beneficial to create an even larger data set or at least explore how the amount of training data effects the classification performance. It could also be beneficial to increase the complexity of the way artificial merges are created to achieve an even higher variation and resemblance of real merged events.

After the surprisingly good results in merge classification, an interesting continuation of the idea could be to create a dataset and corresponding classifier that can classify the event images into good and bad events, or according to
the rating scale in section 3.2. This type of classification could give additional information about the event images as they are passed into the insect species classifier, enabling to early discard or give less confidence to bad events.

Other improvements
As this thesis constrained itself to not put any strict requirements on execution speed of the algorithms, there is room for investigating how the used algorithms and their implementations could be optimized to run even faster.
References


[16] Ran SHI, King Ngi NGAN, Fellow, IEEE, and Songnan LI. *Jaccard index compensation for object segmentation evaluation*, 2014


[23] Slobodan Ilic.(2016) *Tracking and Detection in Computer Vision Filtering and Edge detection* [Lecture note handout], Technischen Universität München, 2016


58


