Carl von Ossietzky

Universität Oldenburg

Studiengang Marine Sensorik

MASTERARBEIT

Optimization of image processing operators
for the AI-based discrimination of planktonic morphotypes

vorgelegt von: Claudia Thölen

claudia.thoelen@uni-oldenburg.de

Betreuender Gutachter: Dr. rer. nat. Jan Schulz

Zweiter Gutachter: Prof. Dr. Oliver Zielinski

Wilhelmshaven, 10. September 2020

I Acknowledgements

First, and most important of all, I would like thank Dr. Jan Schulz for his exceptional support, which cannot be taken for granted, throughout this thesis and even beyond, during the entire time of the master program. I admire his devotion towards the LOKI project and appreciate every talk and email we shared to push this thesis forward. Thank you also very much for providing such a large pre-classified dataset!

I would like to thank André El-Ama for being extremely motivated, even within his free time, sharing his enthusiasm for deep learning approaches, and offering a helping hand for frustrating error messages and new CNN approaches.

I would like to thank Daniela Meier, not only for her super-speed proof reading and great critics, but also for always supporting me on my road within the working group and being a great mud shoveling companion.

I would like to thank Jana Schmitz and Lukas Roß, not only for being great listeners and critics at the general rehearsal for the defense, but also for being my team members and friends throughout the master program and beyond. I believe we have all learned a fair share from each other and I would definitely dare to say, that I have personally and professionally improved because of you.

I would like to thank Dr. Peter Kampmann for taking his time and providing an overview and ideas within the world of artificial intelligence.

I would like to thank Prof. Dr. Oliver Zielinski for the bright advices and many chances he has provided me by believing in my competence and for the ongoing support since my bachelor thesis.

I would like to thank Dr. Thomas Badewien for being a great chief scientist on most of my research vessel trips so far and for being a very honest adviser and supervisor during the research project.

I would like to thank the entire working group Marine Sensors, for being remarkably helpful and friendly colleagues throughout my time as a student assistant, during the bachelor thesis and master program and thesis. I am looking forward to being a member of this amazing team in a few days.

Finally, I would like to thank Pablo, my friends, and especially my family for their love and support during the thesis and always.

II Table of Contents

III Abstract

Investigating the abundance, distribution and diversity of plankton organisms holds a key element in understanding complex, marine food web structures, nutrient cycling, climate change effects and anthropological influences on the world's largest habitat. Over the past years, plankton research has developed from discrete, integrating net samples to fine-scale, *in situ*, optical detection of organisms. Camera systems like the Lightframe On-Sight Keyspecies Investigation (LOKI) system allow a high resolution in capturing organisms below the size of 60 µm, while connecting the image data with environmental measurements.

In this thesis the processing of LOKI plankton images, recorded 2014 in the Sognefjord, Norway, is investigated and improved from edge detection over feature selection to the classification of morphological groups with multivariate and machine learning classifiers. For the classifications, a subset of calculated image features is determined individually for each method by evaluating a decreasing number of image feature ranked by their *Gini Index* significance. Transfer learning is used to implement a fine-tuned *AlexNet* convolutional neural network (CNN) as a first approach towards deep learning classification of LOKI images.

The improvement of the image processing and edge detection was successful, as the implementation of a Canny edge detection algorithm detects the edges much closer to the organism. With the selected features the highest classification accuracy of 88.72 % is achieved with a random Forest classification while the transfer learning CNN results achieve a classification accuracy of 87.75 %.

This thesis has laid out new approaches for LOKI plankton image classification, which will help to progress providing a complete processing chain from image capturing towards autonomous classification and presorting of major morphological groups.

IV Zusammenfassung

Die Untersuchung der Abundanz, Verteilung und Diversität von Planktonorganismen ist ein Schlüsselelement für das Verständnis komplexer, mariner Nahrungsnetzstrukturen, Nährstoffkreisläufe, Auswirkungen des Klimawandels und anthropologischer Einflüsse auf den größten Lebensraum der Welt. In den letzten Jahren hat sich die Planktonforschung von der diskreten, integrierenden Netzprobe bis hin zur feinskaligen, *in situ* optischen Detektion von Organismen entwickelt. Kamerasysteme wie das Lightframe On-Sight Keyspecies Investigation (LOKI)-System ermöglichen die hochauflösende Erfassung von Organismen unterhalb einer Größe von 60 µm und verbinden die Bilddaten mit gemessenen Umweltparametern.

In dieser Studie wird die Verarbeitung von LOKI-Planktonbildern, die 2014 im Sognefjord, Norwegen, aufgenommen wurden, untersucht und von der Kantendetektion über die Merkmalsauswahl bis zur Klassifizierung morphologischer Gruppen mit multivariaten und maschinell lernenden Klassifikatoren verbessert. Für die Klassifikationen wird eine Untermenge von berechneten Bildmerkmalen individuell für jede Methode bestimmt, indem eine abnehmende Anzahl von Bildmerkmalen ausgewertet wird, die nach ihrer *Gini-Index*-Signifikanz gereiht werden. Transfer-Lernen wird verwendet, um ein fein abgestimmtes *AlexNet* (faltendes neuronales Netz) als ersten Ansatz für eine „Deep-Learning“ Klassifikation von LOKI-Bildern zu implementieren.

Die Verbesserung der Bildverarbeitung und Kantenerkennung war erfolgreich, da die Implementierung eines Canny-Kantenerkennungsalgorithmus die Kanten viel näher am Organismus erkennt. Mit den ausgewählten Merkmalen wird bei einer „random Forest“ Klassifikation die höchste Klassifikationsgenauigkeit von 88,72 % erreicht, während die Ergebnisse des Transfer Lernens eine Klassifikationsgenauigkeit von 87,75 % erreichen.

In dieser Arbeit wurden neue Techniken für die LOKI-Planktonbildklassifikation vorgestellt, die dazu beitragen werden, eine vollständige Verarbeitungskette von der Bilderfassung bis zur autonomen Klassifikation und Vorsortierung der morphologischen Hauptgruppen voranzubringen.

V List of Figures

VI List of Tables

VII Abbreviations

a	Local edge orientation angle (Equation 6)
ANN	Artificial Neural Network
API	Application programming interface
bmp	Windows Bitmap
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
CTD	Conductivity Temperature Depth
CV	Cross validation
D	*Dilator* (Equation 2)
DOF	Depth of field
DS_c	Discriminant score (Equation 10)
DT	Output of double thresholding
FOV	Field of View
FSDA	Forward stepwise discriminant analysis
G	Gaussian filter kernel (Equation 7)
GPS	Global Positioning System
h	Hours
HE	*R/V Heincke*
I	Source Image
ID	Identification number
IMG	Image
IMGmask, IM	Image mask, final step of image processing
ISIIS	Ichthyoplankton imaging system
I_x, I_y	Horizontally and vertically *Sobel* filtered images (Equation 3)
k_c	Class constant (Equation 10)
K_x, K_y	Horizontal and vertical modified *Sobel* operator (Equation 1)
LD	Linear discriminant
LDA	Linear discriminant analysis
LOKI	Lightframe On-sight Keyspecies Investigation System
m	Meter
M	Gradient magnitude image, square root of M^2 (Equation 5)
M^2	Gradient magnitude image of summed squares of I_x *and* I_y (Equation 4)
M_b^2	Binarized images of M^2
min	Minutes
ml	Milliliter
mm	Millimeter
ms	Milliseconds
M_s	Sum of M_b^2 *and* M, input of Canny edge detection
μ	Mean feature values (Equation 9)
µm	Micrometer
N	Output of non-maximum suppression
NMS	Non-maximum suppression
png	Portable network graphics
φ	Local edge orientation angle
R^2	Linear correlation coefficient
ReLU	Rectified linear unit
ROI	Region of interest

ROV	Remotely operated vehicle
s	Seconds
s	Feature variance (Equation 9)
S, S_x, S_y	*Sobel* filter with horizontal and vertical components (Equation 8)
SE	Output of *FindConnectedWeakEdges()* function
SVM	Support vector machines
t, t_{high}, t_{low}	Thresholds for binarization and non-maximum suppression
TMD	Telemetry
VPR	Video plankton recorder
w_{cs}	Coefficients of linear discriminants (Equation 10)
x_s	Variables for LDA (Equation 10)

1 Introduction

1.1 Plankton Biology and Sampling

The marine environment is under constant change. It is influenced by geological processes, climatic impact or living organisms, especially human beings (Rockström et al., 2009). This impact is reflected by the biological response to the environmental influences. In marine environments, planktonic organisms are a key within bio-geochemical cycles (Mitra et al., 2014), food web structures (Steele, 1974), ecosystem health (Beaugrand et al., 2002), and the biological response to climate change effects (Roemmich & McGowan, 1995; Hays et al., 2005).

The term plankton originates in the Greek adjective πλαγκτός (*planktos*) and means “to wander”. It was introduced by Hensen in 1887 (Hensen, 1887) and is used to summarize aquatic organisms, which mostly float passively in the horizontal direction, being entrained in ocean currents (Tait & Dipper, 1998). Multiple plankton species can move vertically between water masses, for feeding purposes. The size of plankters ranges from microns to meters and includes representatives in all domains of life and trophic levels (Lombard et al., 2019).

Phytoplankton are autotrophic pro- and eukaryotic algae and responsible for most of the primary production in the aquatic environment. Zooplankton, the faunistic component, include protozoans and metazoans that primarily feed on phytoplankton and maintain the main food source for higher trophic taxa (Tait & Dipper, 1998).

Within aquatic food webs, climate-driven changes or fluctuations in environmental parameters can affect the plankton community and cause nutritional effects cascading through the food web up to the trophic levels of commercially exploited fish stocks (Frederiksen et al., 2006). The distribution of plankton in the water column depends on small scale hydrographical conditions (Banse, 1964; Yamazaki et al., 2002; Schulz et al., 2007 & 2012) and is sensitive to changes. Since plankton organisms have short life spans, and only a few species are commercially exploited, several observed long-term changes can be linked to both climate change and anthropogenic influences (Hays et al., 2005) and their derived successive consequences.

Understanding the distribution, biodiversity and abundance of plankton organisms will provide valuable information of the overall ecosystem health. Plankton is deeply involved in crucial marine processes ranging from primary production over nutrient cycling to large scale marine food web structures even influencing commercial fisheries. Plankton is a biological proxy for the effects of climate change and overall global environmental changes (Beaugrand et al., 2002; Lombard et al., 2019).

Historically, information about plankton distribution, diversity and abundance have been collected with net samples since the time of Hensen (1887). Sample processing needs large amounts of time as well as taxonomists manpower to be evaluated. Over the years a multitude of standard nets have been developed from simple WP2 or Bongo-Net to multi-opening/closing nets or pumping systems, to multi-net systems and finally to multi-sensor systems being deployed on towed devices or remotely operated vehicles (ROV). A good overview of the variety of net and other plankton sampling systems is given by Wiebe & Benfield (2003).

Net sampling systems integrate over the sampled depth. Contact with strong turbulences can be destructive to fragile plankton organisms. Thus, precise *in situ* information of plankton distribution and morphology is diminished. To tackle this shortcoming, new possibilities arose during the past decades as plankton research started using optical imaging technologies to capture and identify plankton images in a non-invasive approach within their natural environment. Imaging systems face many challenges like an even illumination, a fitting depth of field (DOF) - distance of sharp focus in front of a lens -, and the quality of the image (Schulz et al., 2010; Schulz, 2013; Pollio et al., 1979; Mustard et al., 2003). The organism of interest itself and its aquatic habitat provide even more challenges. Plankton specimens not only vary greatly in size, some taxa also undergo strong morphological changes during ontogenesis. Orientation within the photographed volume - abundance of also entrained marine snow, sediment particles, air bubbles or other organisms - further complicate identification (Benfield et al., 2007). Facing these challenges, multiple approaches in plankton imaging investigation where contrived, designed and implemented by different institutions worldwide (see reviews in Benfield et al., 2007 and Lombard et al., 2019). Nevertheless, the optical systems often operate at the borders given by the laws of physics (Schulz, 2013).

Plankton images are envisaged to replace the hands-on taxonomic classification by making it possible to classify the images on the level of morphological groups further down to the species level. The images can then be linked to simultaneously recorded fine-scale environmental data and deliver high resolution information about biodiversity, distribution, state, and behavior of planktonic organisms in relation to the ambient hydrography (Culverhouse et al., 2006).

Some developed imaging devices can be deployed from research vessels in hauled or towed systems and deliver *in situ* images (e.g. VPR (Davis et al., 2004), ISIIS (Cowen & Guigand, 2008), LOKI (Schulz et al., 2009, 2010), GUARD 1 (Corgnati et al., 2016), etc.). Other devices rely on a prior taken net or water sample and image the specimens in the laboratory (e.g. ZooScan (Grosjean et al., 2004)). The different approaches need to find tradeoffs in sampled volume and depth of field and therefore the size range of organisms, which can be detected. Further reviews and summaries about the existing visual plankton systems can be found in Benfield et al. (2007), Lombard et al. (2019) and Lumini & Nanni (2019).

1.2 *Lightframe On-sight Keyspecies Investigation* – LOKI

This thesis uses images captured by the *Lightframe On-sight Keyspecies Investigation* (LOKI) system developed by Schulz et al. (2009 and 2010) (Figure 1). LOKI consists of multiple modular parts and can be deployed in hauled systems as well as on moorings. LOKI was deployed on a COSYNA Underwater-Node System in the North Sea to deliver valuable information on the spatio-temporal zooplankton community assemblage (Baschek et al., 2017). LOKI can also serve as a benchtop device with a Flow-cell (FLOKI, Schulz et al., 2008). The benchtop device has the advantage that illumination conditions can be adjusted to capture brighter images, allowing a more accurate classification, but it lacks the additional environmental information an *in situ* device is able to record.

During deployment, a standard LOKI haul uses the up-cast to concentrate plankton with a tailored net to least harm fragile plankton specimens entrained in the water column and guides them into a flow-through imaging chamber. A specific optical design of the flow-through chamber a relatively high DOF at a small sample volume (1.6 - 3.5 ml) with high magnification is realized. The developed flash module consists of 96 XML_T6_6000K LEDs, which provide a white light with a spectral composition of 6000K (Hagemann, 2016). “The system is based on an illumination technique that either projects a light frame of high luminous flux into the water or constrains the volume physically with transparent boundaries. Particles within this area are illuminated. Only directly illuminated objects are visible for the camera, while those outside the focus range are nearly invisible” (Schulz et al., 2010). The used camera is an Allied Vision Technologies Prosilica with GigE interface and allows imaging of plankton and particles of sizes below 60 µm at a high resolution with shutter times below 50 µs. To adjust the optimal depth of field, the camera must be able to change the distance to the object. To achieve this inside the sealed LOKI housing the camera is mounted on a piezo-motorized linear stage to adjust the distance between camera and object (Hagemann, 2016) (Figure 2). Further technical details can be found in Schulz et al. (2009 and 2010).

The images used in this thesis have been taken by deploying LOKI as a hauled system from a research vessel. Figure 1 shows such a deployment. The device is lowered into the water by the research vessels winch. It is lowered (down-cast) to maximum depth and heaved (up-cast) again. During the up-cast the upstream plankton net collects organisms and directs them through the flow-through chamber were the camera images them. Figure 3 shows a collage of sample plankton images collected during different cruises with the LOKI system. Next to the LOKI system a CTD (device measuring Conductivity, Temperature and Depth) is attached to the frame, delivering essential information about the environmental factors of the plankton's habitat.

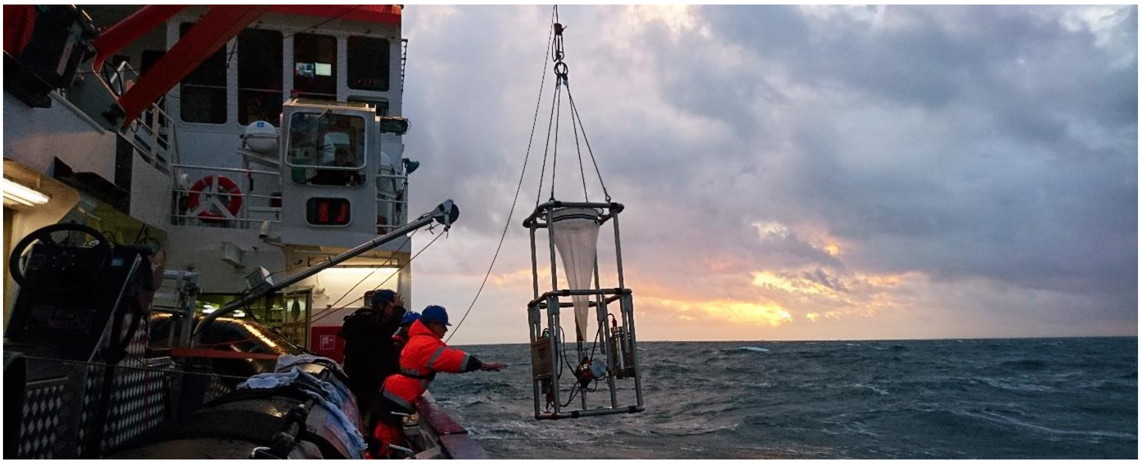


Figure 1: Lightframe On-sight Keyspecies Investigation (LOKI) system being deployed on board R/V Heincke in November of 2019. This image was taken on the R/V Heincke cruise HE545 and visualizes the plankton net with the camera underneath. Attached to the sides of the LOKI frame are the computing and the battery units and a CTD. Image captured by C. Thölen.

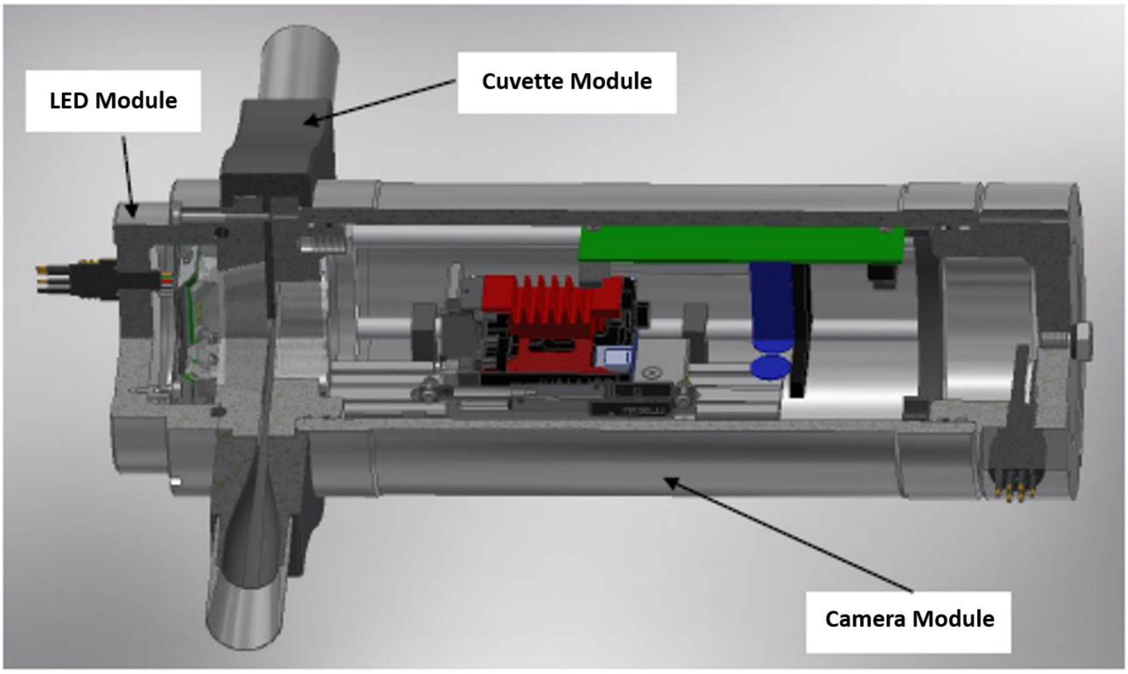


Figure 2: Cross section and overview of Lightframe On-Sight Keyspecies Investigation (LOKI) system. Attached to the cuvette module are the in- and outflow tubes. The LED and camera module are separated from the cuvette module by an optical window. The camera module is mounted on a piezo-motorized linear stage to adjust the distance between camera and object. The image was created with the CAD-Software by Hagemann (2016) *for a student report on the LOKI system.*

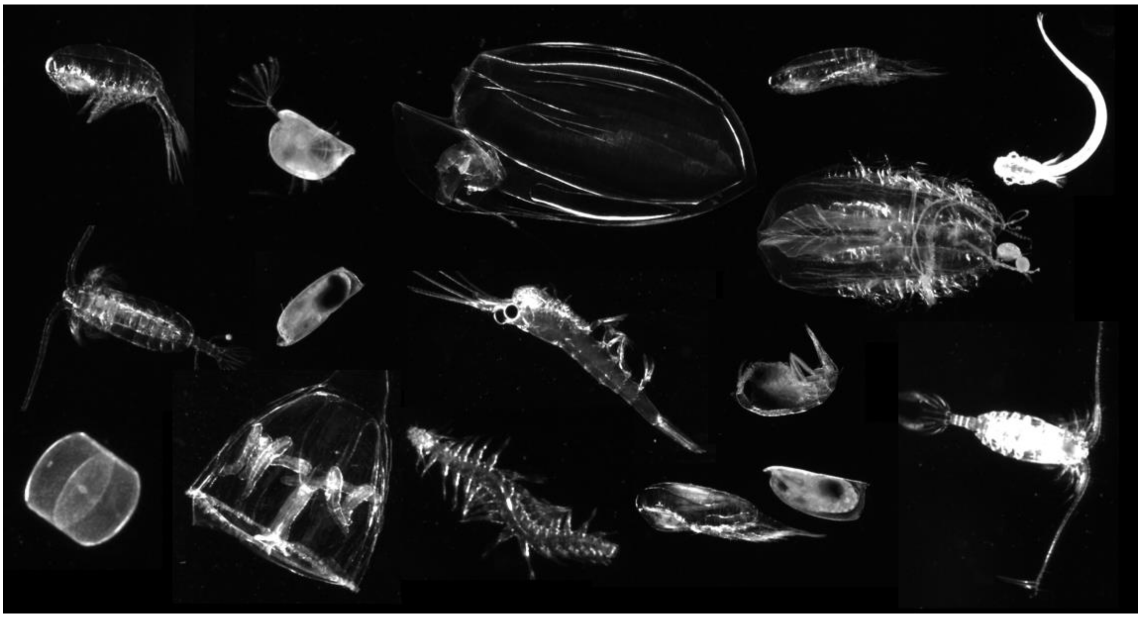


Figure 3: Images of multiple plankton specimen captured with the Lightframe On-sight Keyspecies Investigation (LOKI) system. This image was created as a collage out of multiple images; therefore, the imaged organisms are not to scale.

1.3 Image Processing, Features, and Classification Methods

The classification of plankton images is a requirement to precisely match species appearance in relation to environmental parameters and to gain a deeper understanding about biodiversity, spatio-temporal distribution, community structure, and behavior of plankton organisms.

Optical plankton detection systems are aiming towards an unsupervised, automated recognition and classification of plankton images (Benfield et al., 2007; Bi et al., 2015; Corgnati et al., 2016; Leow et al., 2015; Luo et al., 2018; Schröder et al., 2020; Sieracki et al., 1998; Sosik & Olson, 2007; Tian et al., 2019; Wang et al., 2016; Zheng et al., 2017). Different approaches exist for the classification process, ranging from multivariate statistics and machine learning algorithms (e.g., Hu & Davis, 2005; Schulz et al., 2016; Sosik & Olson, 2007) to deep learning algorithms (e.g., Leow et al., 2015; Lumini et al., 2019; Luo et al., 2018). In the following it is explained how an image with mere pixel intensities can be transformed into a classified result which can help to answer relevant biological questions.

1.3.1 Object Detection

Before an organism can be classified, it must be recognized within an image. This is achieved by detecting the edges of this organism. Edges are pixel intensity changes within an image distinctly along a particular orientation. If the intensity change is high, the evidence for an edge at that position is provided (Burger & Burge, 2016). This gradient in intensity can be detected and enhanced with edge detection operators, respectively filters or kernels. Some simple edge detection operators have been defined by Kirsch (1971), Prewitt (1970) and Sobel in 1968 (Davis, 1975; Sobel & Feldman, 2015). Edge detection is used in multiple fields of research. Kekre & Gharge (2010) state, that for mammographic images an extended *Sobel* filter (5x5 matrix) delivers the best results compared to extended Kirsch (1971) or Prewitt (1970) filters. Simple edge detection filters consider the first derivative of an image to find strong gradients, respectively edges. Other methods use the second derivative of an image, in which edges can be found at zero points or zero crossings and can be localized more precisely (Burger & Burge, 2016). The Canny-Operator (Canny, 1986) is considered *state of the art* in edge detection and uses the second derivative. It minimizes the number of false edge points (pixel, which do not represent the maximum gradient of an edge), while achieving good localization of edges. Within the Canny edge detection algorithm, the width of an edge is reduced to a single pixel (Burger & Burge, 2016). Bature et al. (2015) compare the simple edge detection filters (*Sobel*, *Prewitt*, etc.) to the Laplacian of Gaussian (Marr & Hildreth, 1980) and the Canny operator. In their results Gaussian methods like Canny deliver the most robust solution localizing the edge points even in noisy images.

1.3.2 Feature Extraction and Selection

The input for standard multivariate and machine learning classification algorithms are calculated images features. To extract such feature information, objects on the images need to be segmented and pixel - that belong to the respective object - need to be identified and localized, as it was explained in the previous section. For common geometric features of plankton images and a review of recent feature extraction methods used see Cheng et al. (2018). Features that have been extracted from an image manually will be referred to as “manual” features in this thesis to distinguish them from the deep features created by neural networks, which will be explained in the following chapter. For the LOKI plankton images recurrence features have been used in a previous study by Schulz et al. (2016), which, in the combination with standard manual features and a classification using the linear discriminant analysis (LDA), produced a discrimination success of 62.8 %.

Wang et al. (2016) used multi feature combinations and achieved good classification results combining different types of image features. For classification purposes, it is important to find useful feature combinations. Redundant or unimportant feature information might compromise the performance of the classifier (Sosik & Olson, 2007). Feature selection methods aim to find a subset of features that would either improve the classifiers performance or at least maintain the same level like with more features, while saving operational time (Dash & Liu, 1997). There are a multitude of approaches for feature selection based on the dataset type and size, the number of classes and for weather or not a dataset contains imbalanced class sizes. Leow et al. (2015) used forward stepwise discriminant analysis (FSDA) and Zheng et al. (2017) implemented a wrapper method by Kohavi & John (1997) for their feature selection. There are different strategies based on multivariate statistics allowing to exclude redundant features. Linear discriminant analyses (LDA) can be used on a pre-classified dataset to detect the variables or features that account for the most variance within the dataset (Schlittgen, 2009). With the random Forest method, developed by Breiman (2001), the *Gini Index* (Gini, 1912) can be calculated, providing information about the separating importance (inequality) of different features. The dataset used within this thesis contains continuous features, all representing the same number of classes. These characteristics make the datasets features eligible to using the *Gini Index* for the importance measures (Strobl et al., 2007).

1.3.3 Image Classification from Standard Multivariate Statistics over Machine Learning to Convolutional Neural Networks

The selected features of a dataset span a multidimensional room containing every data element. Most multivariate methods aim to reduce the dimensions of a dataset to describe the relationships between the given features (Zelterman, 2015). Multivariate statistics can be used in a variety of different technological, sociological, and scientific fields, allowing structuring, simplifying, and classifying the elements that belong to the same class in a dataset. Within chapter 2.3 and 2.4 the multivariate statistical linear discriminant analysis (LDA) and the random Forest method are described. Both are used for the feature selection and classification of the plankton images. Multivariate statistical methods work with a predefined set of rules, which distinguishes them to machine learning methods.

Machine learning means, that - instead of calling an algorithm with passed data of extracted features and predefined rules - the extracted features are passed along with the associated classes for the data and the algorithm produces the rules for classification (Chollet, 2018). A feedback mechanism is used to optimize the algorithm calculations to produce rules that are more suitable for the data to be sorted into the correct classes (Chollet, 2018). A “traditional” classification approach of machine learning are support vector machines (SVM). SVMs have the key idea of maximizing margins between data classes by finding applicable kernel functions

(Bennett & Campbell, 2000). They will briefly be explained in chapter 2. For more detailed information see Christianini & Shawe-Taylor (2000) or Vapnik (1995).

The learning process gets "deeper" and more versatile with the supply of multiple neuronal calculation layers between input and output, hence deep learning. A representation of deep learning models are convolutional neural networks (CNN), which are increasingly used for image classification or speech recognition (Gu et al., 2018). CNNs are able to process a raw image directly, without prior feature extraction, as the calculation of discriminating deep features is a built-in part within their multi-layered network (Luo et al., 2018).

The architecture of a CNN is to some extent similar to the human visual system as they use restricted receptive fields, and a hierarchy of layers, which progressively extract more and more abstracted features (Lumini et al., 2019). A CNN consists of a series of stages, where the first few stages contain convolutional and pooling layers. Figure 4 shows an example of a simple pre-trained CNN with its architectural layers. This particular CNN is called AlexNet and was developed by Krizhevsky et al. in 2012. In the following, the single layers of a CNN will be explained briefly.

The function of the convolutional layer is the detection of local feature combinations from the input or previous pooling layer (Khan et al., 2020). The convolution layer contains a set of convolutional kernels which work with the same basic principle like the edge detection kernels convolution mentioned in chapter 1.3.1 and chapter 2.2.2. These kernels process the image by dividing it into smaller pieces helping for the extraction of image features and create feature maps (Khan et al., 2020). This deep feature extraction can be similar to the manual feature extraction mentioned in the previous chapter. But through the depth of a CNN convolutional layers become more and more abstract to the human eye. These layers have a specific set of weights adjusting their behavior when data is passed through them. The weights are influenced by the feedback mechanism of the CNN, called backpropagation (Aggarwal, 2018). For neural networks, learning means to adjust the values for the weights of all layers in a network by backpropagation, until the network assigns the right classes to testing samples (Chollet, 2018). This process is called training and is conducted with a dataset of labeled images. For a deep neural network millions of weights need to be trained, hence the training dataset needs to be large and training might take up a long time.

A layer not shown in Figure 4 is the activation layer, which is also an important part of CNNs. The output of a convolutional layer is assigned an activation function. One of the most popular ones is called ReLU (rectified linear unit). It adds non-linearity to the output and transforms it, taking away the negative values and setting them to zero (Khan et al., 2020).

Convolutional layers are followed by pooling layers. The role of the pooling layer is to merge semantically similar features into one by summing up the information of the input feature map and returning the dominant response for a local region (Khan et al., 2020; LeCun et al., 2015). If the pooling operation delivers the maximum of a certain input the approach is referred to as max-pooling (Aggarwal, 2018). Pooling reduces the spatial resolution and the number of deep features required for classification, which saves computational time.

The last few layers visible in Figure 4 are fully connected layers. Fully connected layers create a non-linear combination of selected deep features (Khan et al., 2020) and lead the information to a *softmax* layer which then creates possibilities rates for classification of the input data.

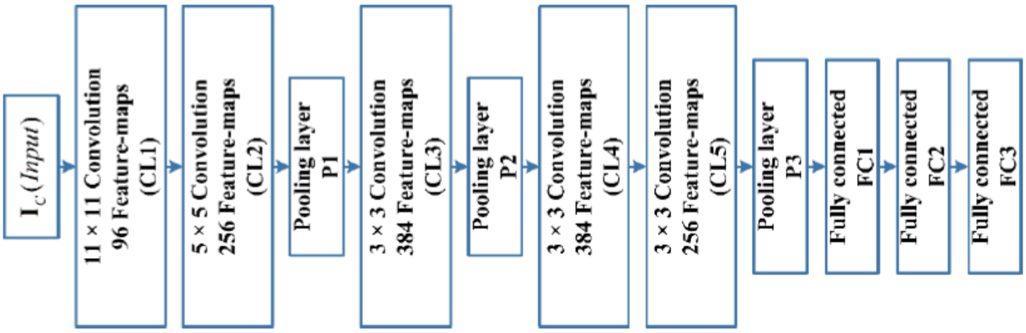

I_C (Input)
11 × 11 Convolution 96 Feature-maps (CL1)
5 × 5 Convolution 256 Feature-maps (CL2)
Pooling layer P1
3 × 3 Convolution 384 Feature-maps (CL3)
Pooling layer P2
3 × 3 Convolution 384 Feature-maps (CL4)
3 × 3 Convolution 256 Feature-maps (CL5)
Pooling layer P3
Fully connected FC1
Fully connected FC2
Fully connected FC3

Figure 4: Example of a simple pretrained convolutional neural net (CNN) architecture with 12 layers. This particular CNN is called AlexNet and was developed by Krizhevsky et al. in 2012. *Modified according to* Khan et al. (2020).

CNNs need minimal preprocessing of the images and require no prior knowledge in designing manual features for classification. This represents a significant advance compared to “traditional” machine learning methods such as artificial neural networks (ANN) and support vector machines (SVM) (LeCun et al., 2015; Luo et al., 2018). Furthermore, it was indicated by Lumini et al. (2019) that ensembles of more CNNs gain a higher performance than a single CNN. For further details and comprehensive reading about CNNs the textbook “Neural Networks and Deep Learning” by Aggarwal (2018) is recommended.

When a new CNN is created the weights of the layers are created randomly. It takes a large amount of training to adjust the weights of the CNN layers for an accurate classification result. Many CNNs for image classification have been developed over the past years and they are able to distinguish between thousands of categories. So instead of building a completely new CNN from scratch, transfer learning can be used to transfer the previously compiled knowledge of a pre-trained CNN like AlexNet to an adapted CNN for plankton classification (Pan & Yang, 2009). The adapted CNN has the right weight settings to classify images of everyday items like vacuum cleaners, cats, or cars. It can be fine-tuned by using a labeled training dataset of plankton images to allow classification of planktonic morphotypes.

Lumini et al. (2019) state that research on plankton image classification has started to replace “traditional” classifying methods, based on manual feature extraction, such as support vector machines or random Forests, in favor of deep learning approaches. In their publications, Bi et al. (2015) and Verikas et al. (2015) provide reputable summaries of recent approaches in plankton image classification, that are noticeably developing towards deep learning solutions. Transfer learning has also been used in the plankton classification implementations of González et al. (2019) who determined that deep features calculated by pre-trained networks achieve better classification results than manual features. For further information and a survey on transfer learning see Pan & Yang (2009).

1.4 Objectives

Some time has passed since the last innovations for the LOKI system were implemented. While the technological side still provides *state of the art* plankton images, the software and classification system need to catch up to modern standards of plankton recognition. Within this thesis, the edge detection within the provided *LOKI-Complete-data-primer* script will be reviewed and improved, and first attempts towards an automatic classification of LOKI plankton images and feature tables using CNN and SVM will be taken and compared to multivariate classifiers. Therefore, a selection of the most important manually extracted image features will be implemented. This results in the following objectives for this thesis:

Objective 1: Evaluation of the adapted edge detection operation.

Objective 2: Evaluation of the implemented feature selection.

Objective 3: Evaluation of the reduced feature table regarding the discriminability of labeled morphotype groups, based on numerical features using LDA, random Forest and SVM.

Objective 4: Approach of direct image discrimination with convolutional neural networks.

2 Material & Methods

2.1 Sampling and Data Availability

Data was collected in 2014 during *R/V Heincke* cruise HE434 (Badewien, 2014) in the North Sea and in the Sognefjord (Norway) by Dr. Jan Schulz and Nicole Hildebrandt (Figure 5). On 17 stations the LOKI system was deployed with one haul each, except for the first LOKI station, where two separate hauls were conducted. Every haul included one cast (down- and up-cast), except for the last LOKI station, where the device was deployed for three hauls down to 450 m and then left at a depth of about 175 m for 10 minutes.

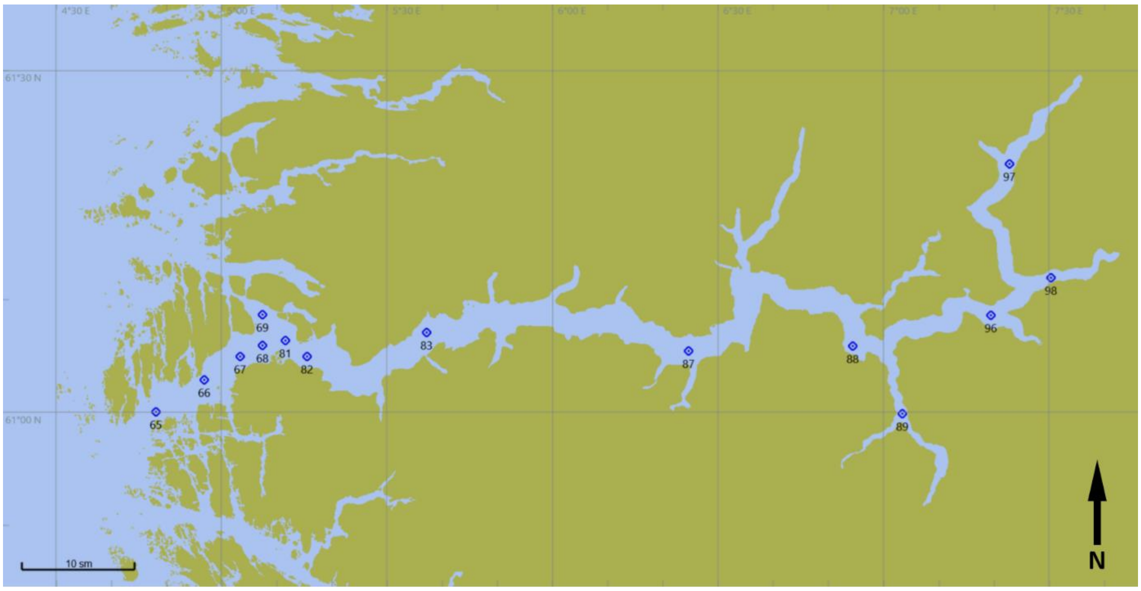


Figure 5: Sognefjord map of the Lightframe On-Sight Keyspecies Investigation (LOKI) system stations during R/V Heincke cruise HE434 in 2014.

The raw data used in this thesis includes 53.4 GB of images and telemetry data (481,455 files) containing both down- and up-cast of the LOKI system. The downcast data is not usable for further calculations since the camera only captures organisms and particles that find their way into the field of view through the narrow outflow at the cod-end of the LOKI. The captured organisms would come from a much smaller sample volume than intended for the recordings. At its deepest point, the LOKI system is stopped for the few seconds it takes to start hauling it back up to the vessel. During this short period of time, the organisms and particles that might be in front of the camera are captured multiple times which would falsify the results. In order to extract only valid images at a steady flow during the up-cast, it was necessary to discard data prior to a continuous heaving speed of 0.3 m s^{-1} during up-cast. This was achieved by using the telemetry data and first smoothing the depth information with a running mean of 5 s to suppress the influence of the ship's movement. The deepest point of the cast was identified and the velocity of the device during the cast was calculated. The timestamp was identified for the moment when the device reached a speed of 0.3 m·s^{-1}. Thereupon the beginning of the image data was cut to this timestamp. At the end of each cast the LOKI system was heaved through more turbid surface waters, which are influenced by the research vessel. This resulted in air bubbles in the field of view (FOV). Also, the heaving speed decreased as the crew was waiting for further instructions before heaving the device back on deck. As the LOKI systems velocity reaches values under 0.05 m·s^{-1} the timestamp is saved and used as the ending point of the up-cast. Figure 6 visualizes the cropping of the up-cast to the valid timestamps.

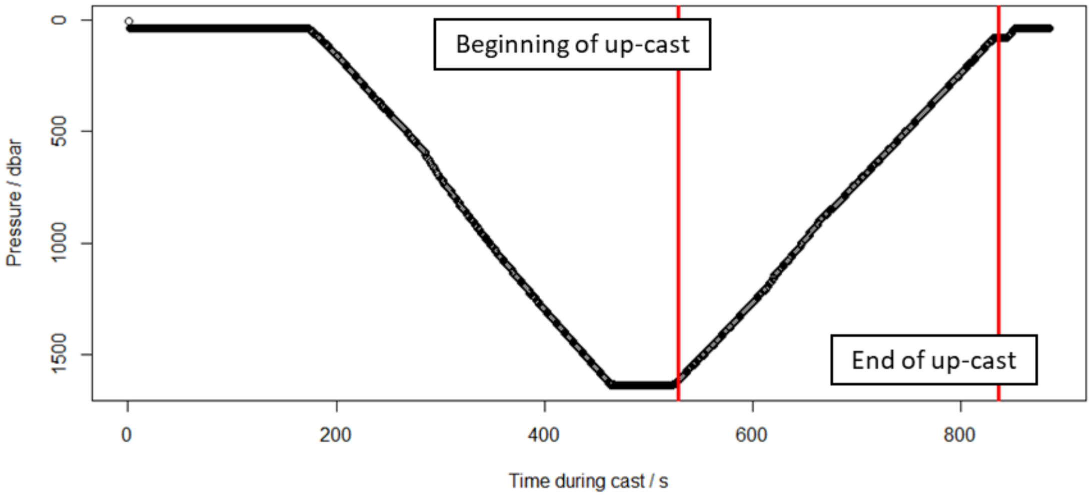


Figure 6: Recorded pressure in decibar versus the time in seconds during the cast of the Lightframe On-Sight Keyspecies Investigation (LOKI) system at station 67 on R/V Heincke cruise HE434 in 2014. The red lines indicate the moments of the device reaching a mean heaving speed of 0.3 m·s^{-1} and the end of the cast when the speed falls below a heaving speed of 0.05 m·s^{-1}.

Considering only the Sognefjord images during the cut-out timeframe, 113,551 images were left for further processing and classification (Appendix 1). Out of this large dataset a total of 37,057 images were manually classified by Dr. Jan Schulz into morphological groups of different taxonomical ranks from phylum down to species and genus level. This dataset (*Sognefjord classed*) was used for classification approaches during this thesis. The dataset was originally partitioned into 64 classes. For a first trail, the 64 classed were summarized into 26 morphological groups (*Sognefjord 26 groups*) most of them containing organisms taxonomically determined to a family level. For the following classification approaches the dataset was then summarized into the most dominant morphological groups in 14 classes (*Sognefjord 14 groups*). Figure 7 provides an overview over the plankton classes within the *Sognefjord 26 groups* dataset used for classification. The ten classes colored in gray are complemented by four classes for bubbles, detritus, eggs and faeces.

A third LOKI dataset with a total of 1023 images pre-classified into 21 morphological groups was used for feature selection and testing of the classifying algorithms. This dataset was collected off the coast of Peru in 2009 during Meteor expedition M77/4 by Hans-Jürgen Hirche and Kristina Barz (Schulz et al., 2016; Stramma, 2009) (Appendix 1).

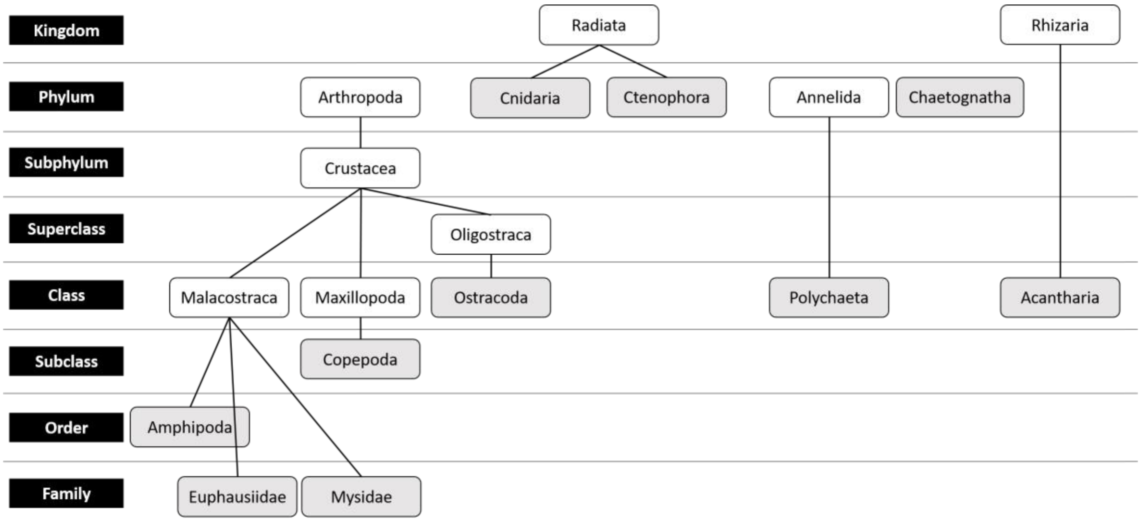


Figure 7: Overview over the taxonomic ranks of the morphological plankton groups (gray) included in the dataset used for plankton image classification. This graph does not show the complete taxonomic classification, just an extract of the most important plankton affiliations for this thesis.

2.2 Image Processing

After extracting the Sognefjord up-cast dataset, the processing of images takes place in multiple steps. The first step is preparing and priming the images using the *LOKI-Complete-data-primer script* in R (R-Core-Team, 2020). The script, in its original form, was internally provided by Dr. Jan Schulz and further changed and optimized as one of the main objectives of this thesis. The main scripts used within this thesis can be found in Appendix 7 and under https://gitlab.uni-oldenburg.de/gorm2097/loki-image-processing-and-classification/.

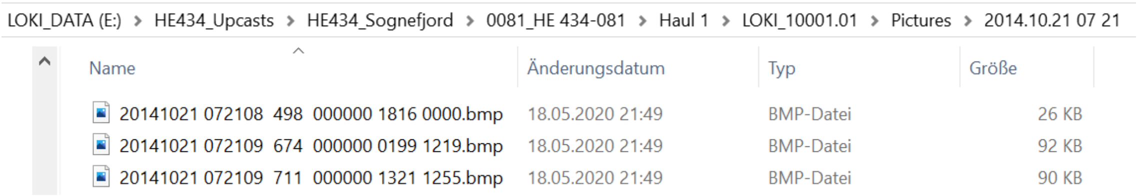
2.2.1 Documentation of the Original LOKI-Complete-data-primer Script

The *LOKI-Complete-data-primer* script compiles all available image and telemetry information collected during a LOKI cast and prepares and summarizes these information in two output tables linked to the images over the image timestamp. The output of this script can further be used for discrimination and classification purposes.

The script can be subdivided into three sections:

- (i) the set up and update section,
- (ii) the telemetry processing section
- (iii) the image processing section.

Changes and improvements within this thesis are mainly associated with the image processing section. The first two sections are explained briefly in their use for the script itself and the generated output.



LOKI_DATA (E:) > HE434_Upcasts > HE434_Sognefjord > 0081_HE 434-081 > Haul 1 > LOKI_10001.01 > Pictures > 2014.10.21 07 21

Name	Änderungsdatum	Typ	Größe
20141021 072108 498 000000 1816 0000.bmp	18.05.2020 21:49	BMP-Datei	26 KB
20141021 072109 674 000000 0199 1219.bmp	18.05.2020 21:49	BMP-Datei	92 KB
20141021 072109 711 000000 1321 1255.bmp	18.05.2020 21:49	BMP-Datei	90 KB

Figure 8: Example of the folder structure used by the LOKI-Complete-data-primer script , in which the Lightframe On-sight Keyspecies Investigation (LOKI) images are saved.

Set-up and update section (i). Within the set-up and update section the working directory and data paths for the in- and outputs are defined. The input path points to a folder that contains image and telemetry data in the lowest instance of its structure. The folder structure has a predefined order and needs to be accurate for a correct output. Therefore, the right option for the input path is the folder for the cruise number. The folder structure should be as following: Cruise ID > Station > Haul > Device Name (Figure 8). Under Device Name there are three folders: Log, Pictures, and Telemetry. The *Log* folder contains .log files with header information for the output. This file is created once per haul. The *Telemetry* folder contains secondly averaged telemetry data from the CTD (attached to the LOKI frame) and GPS information. Finally, the *Pictures* folder contains minutely separated folders with the image data in lossless bitmap (.bmp) or portable network graphics (.png) format. In this context an image is a part of the original photographed picture by the LOKI camera. A closed code algorithm (provided by Medea-AV-GmbH) is preselecting objects within the full camera frame that possess pixel intensities over a certain threshold and crops them into smaller images by surrounding the object of interest with a bounding box. The created images have a file name that contains the date, time, millisecond (Format string: yyyyMMdd HHmmss SSSS) number of the image cropped from one full camera frame within the respective millisecond range (one frame can hold multiple organisms) and the x- and y-coordinates of the upper left corner of the image within the original camera frame (Figure 8).

After the definition of the input and output path the state of several switches and sub switches can be adjusted. This means, that entire sections or single operations can be activated or deactivated by the user, depending on the kind of information needed. The entire processing sections of the telemetry and the image data can be switched on or off. The ability of the script to update itself and the necessary R toolboxes can also be switched on or off. The set-up section further defines several variables which are used within calculations throughout the script. An important variable defined at this point is the filter kernel for the edge detection within the image processing section. These filters or image processing operators are explained in detail in chapter 2.2.2.2. If required, some installations of tools and packages are also performed in this section. A fundamental package which must be installed and linked to the script here, is the *BiocManager* package as it provides many of the image processing functions. Other necessary packages are: *magick*, *readr*, *data.table*, *IM*, *Momocs* and *tools*. Appendix 6 contains an overview over the used packages and their references.

Telemetry processing section (ii). The telemetry data priming section is not a main object to modifications and was just changed minimally. The changes being implemented were for reasons of saving computation time and are further explained in chapter 2.2.2.4. This section's output is a table called *Data.primedTMDdata* which summarizes all the available telemetry data in 31 columns for each measured second of the haul (Appendix 5). The telemetry data is extracted from the TMD files in the same folder structure. A single TMD file contains the sensor identification numbers (ID) and corresponding measurements for one second of the haul. The TMD files are listed and an additional file containing the sensor IDs as well as the associated column headers is loaded into the script. The sensor IDs, the column headers and the iterative number of observations are associated in a data frame to be able to assign the correct sensor names to the TMD file information. Next, an empty matrix is created serving as the basis for the output table. The number of columns is set to the number of sensor information plus seven slots for the date and time information, which can be derived from the TMD file name, as well as information about the cruise, station and haul which can be derived from the file path. The predefined column names are then copied to the empty matrix. Then the result table is defined, and a temporary matrix is set up to be filled by iterating recursively through all the available TMD files. Each iteration reads a new TMD file from the TMD file list and extracts the data into the temporary matrix. The file path is split up and the information about cruise, station and haul are also included in the temporary matrix. If any of the superordinate folder structures is unavailable, it is marked as unavailable in the matrix. The file name is extracted and parsed for the time value which is also included in the temporary matrix. At the end of each iteration, the temporary matrix is concatenated to the result table. The last step in the telemetry priming section is saving the final table as *Data.primedTMDdata* in the result folder.

Image processing section (iii). The image processing section starts with preparing the final output table by defining the table headers for the 15 columns for metadata information and 77 image features (Appendix 3 and Appendix 4) plus 20 Taxonomical columns for future taxonomic information after the classification. Next, a list is created that contains all the .png or .bmp image file names that are found recursively within the defined data path. This operation is equivalent to the creation of the TMD file list in the telemetry processing section. In the following, a large for-loop processes each image individually and multiple computations are conducted. The calculated information will be added to the output table. Within this loop the data path and file information are read and checked weather superordinate folders are available. If so, their name information will be added to the table in the previously mentioned order (Cruise Number > Station > Haul > Device Name). To search at the right folder in the path the number of superordinate folders is previously counted and if the information is unavailable, it will also be marked as unavailable in the table. As the image file names contain all the necessary meta information regarding the date and time and the position of the image within the original camera frame, the file name itself is used to extract this information per image and later add it to the output table. The timestamp within the file name can be used to link the image processing information to the TMD data measured in the same second. The R package *magick* is used to get some of the internal file information such as the format, color space and size. All the extracted information concerning each image file is added as characters to the output table.

Thereafter starts the process of on extracting fundamental inherent image information. As a security precaution, and to avoid the script from crashing, a small 7x7 fake matrix, with 3x3 white pixel in the middle, is created to be processed in case the actual image fails to load. This fake matrix could easily be spotted and ruled out within the results. Two different loading commands are implemented for either .png or .bmp images. The source images are then converted to grayscale images using the *EBImage*::*channel()* function. The *EBImage::filter2()* function is used to apply the *modified Sobel* kernel filters (Equation 1), which were defined in the first section of the script, vertically and horizontally to the grayscale image for edge detection. The *EBImage::filter2()* function uses the setting *replicate* for how the filter behaves at image boundaries. With this setting, the pixel outside of the image boundaries are assumed to have the same value as the nearest border pixel value.

Modified Sobel operator

$$K_x = \begin{pmatrix} 2 & 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 & -2 \\ 3 & 2 & 0 & -2 & -3 \\ 2 & 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 & -2 \end{pmatrix} \quad K_y = \begin{pmatrix} 2 & 2 & 3 & 2 & 2 \\ 1 & 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -2 & -1 & -1 \\ -2 & -2 & -3 & -2 & -2 \end{pmatrix}$$

Equation 1

Dilator

$$D = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 2 & 1 & 1 \\ 1 & 2 & 4 & 2 & 1 \\ 1 & 1 & 2 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix} \times 28^{-1}$$

Equation 2

A horizontal and a vertical edge detection matrix is created. The individual results are squared to remove possible negative values and then added to each other. Next, a binarized image is produced by setting all values within the image over a predefined threshold to 1 and all underneath this threshold to 0. It is possible that certain image attributes belong together even if their connecting pixel had a low intensity on the image so that they were set to 0 by the binarization. To reconnect those attributes a 5x5 *dilator* (Equation 2) is used with the *EBImage*::*dilate()* function. The dilation reconnects prominent image attributes which are located close together. This step creates the image mask (*IMGmask*) which is then applied to the *EBImage*::*bwlabel()* function, which labels all connected pixel in the image above a value of 0, according to their pixel-cluster membership. Each cluster is assigned a unique increasing integer. The contour length of the clusters within the image is calculated with the *EBImage::ocontour()* function and the object with the longest contour is set to be the object of interest and therefore chosen for the final *IMGmask*. This means that all other clusters within the image are ignored and only the mask is retrieved for further calculations. The pixel within the *IMGmask* are set to 1 with the *EBImage::fillHull()* function. This completes the image processing step. Next, and if switched on, multiple calculations are being conducted on the *IMGmask* matrix for feature extraction to gain more information about the object of interest within the image.

The manual extracted image features contain size, distance, intensity, and shape information, Haralick parameters (texture features), contour line statistics, and recurrence analysis features. For the size values, the image height and width are retrieved and then used to calculate the mass center and elliptical fits. Then the distance values to the nearest border are calculated for each pixel of the *IMGmask* using the *Manhattan* metric. In the next step the image pixel statistics are calculated with the *EBImage*::*computeFeatures.basic()* function. These information include the pixel intensity mean, standard deviation and quantiles. With the *EBImage::computeFeatures.shape()* function shape information like the radius and area of the *IMGmask* are calculated. Haralick features are textural features based on gray tone spatial dependencies (Haralick et al., 1973) and can also be calculated with the *EBImage::computeFeatures()* function. The *MOMOCS* package is used to calculate contour line and shape statistics like area, fractionality and convexity of the shape and many more. Last, basic recurrences features are extracted from the contour outline. Recurrence features give information about the contour line's curve progression by using an embedding of contour line pixel distances from the mass center and can be used for discrimination purposes (Schulz et al., 2016).

The calculated features are combined in a data table, which is saved as *Image data.primedIMGdata* and accounts for the second and most important output of the *LOKI-Complete-data-primer* script.

The third output is an image sheet with four plots containing different aspects of each image processed. If the *CreatePlots* sub switch is activated, a folder is created on the same folder level as the TMD and IMG output tables to save all the result plots. Then, an empty result .png-image is created, named after the currently processed image file, and depicts four to six results plots for inspecting image and success of the processing steps.

The first plot displays the grayscale image together with the contour line of the *IMGmask*. The second plot shows an image intensity bar plot with lines for the mean, standard deviation, and median absolute deviation value. The third plot displays the recurrence plot and the fourth a border pixel distance map. After the creation of the result plots, the image processing for-loop is finished and restarts with the next image. After all images are processed a final report is printed containing the processing time and number of processed files.

2.2.2 Changes and Improvements for the Script

This chapter will explain the changes and improvements implemented to the original *LOKI-Complete-data-primer* script in detail. The changes that were implemented into the (iii) image processing and edge detection of the original script, are aiming at the optimization of the detection of the region of interest (ROI) and creating an edge contour as close to the real structure as possible. Additionally, it was an objective to increase general processing speed and to minimize possible errors from corrupted or overexposed images and image structures.

The resulting changes can be divided in four main topics. Change A, B and D are implemented into the final script with switches for de-/activation. Change C was implemented without a switch as it saves the script from crashing when processing inappropriate images. If the switches for these changes are deactivated the complementary parts of the original script will be used for the image processing parts instead. The changes are not named in an order within the script but after the implementation order during the improvement progress within the thesis.

2.2.2.1 Neutralize Image Artefacts (Change A)

The original script sometimes produces reoccurring misidentifications of the ROI. These misidentifications occur when there are multiple objects or organisms in the image, often associated with some overexposed areas in the background (Figure 9). Next to the occurrence of multiple organisms, bubbles or detritus, some images have artefacts of incoming light from the LOKIs LEDs at the margins of the camera frame. When an object is cropped out right at the margin of the original camera frame by the Medea-AV-GmbH code, the resulting image can have a bright artefact on its margins as well. This artifact can be recognized as an object since its pixel intensity values are high. It also often has a larger contour than the actual ROI containing the organism. Change A was implemented to prevent the script from choosing the light artefacts as the ROI.
As described in chapter 2.2.1 the detected objects on the original image are turned into integer clusters with the same integer for every pixel of one object (Figure 9). The contour length is than calculated from this cluster. To improve the cluster selection, the ROI should not only be determined by the longest contour line but also by the vertical and horizontal expansion of the object in the image. This will ignore the light artefacts, which are just a few pixel high and only stretch usually horizontally over the margin of the image (Figure 10). In addition, it will improve the detection of organisms over false image attributes since most organisms are more bulky objects.

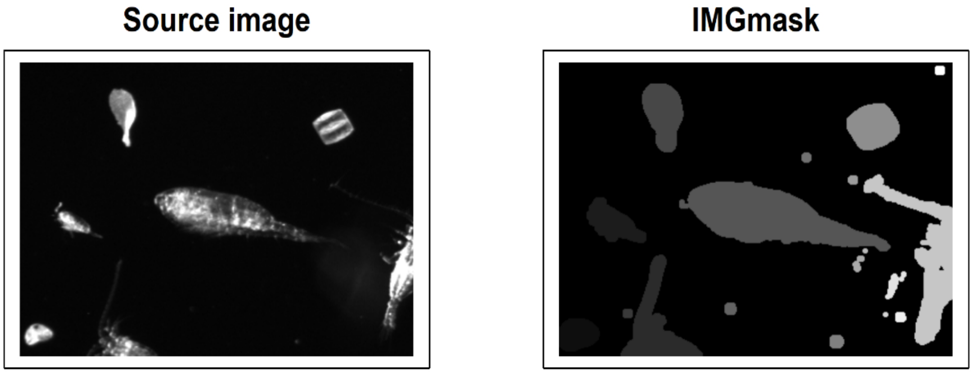


Figure 9: Source image with multiple organisms (left) and an example of the integer clusters of detected, unconnected objects within the image on the image mask (right). The different shades of gray imply different clusters, which each have an identifying integer number.

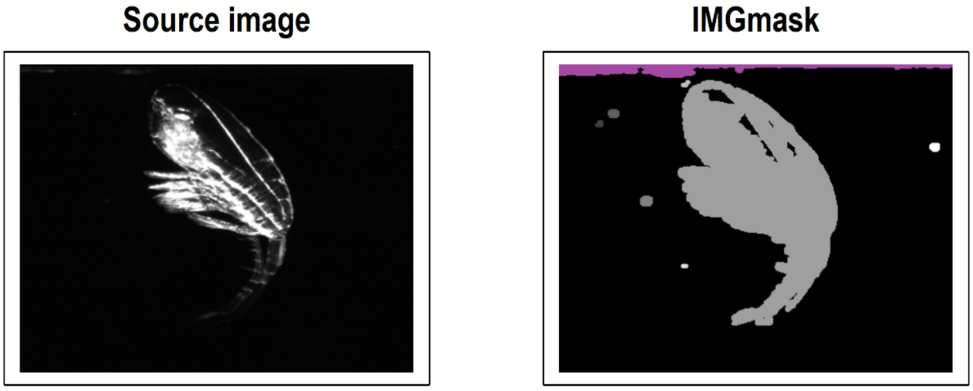


Figure 10: Source image (left) and calculated image mask (right) showing the detected region of interest (ROI), which is a light attribute of the LOKIs LEDs at the upper margin of the image (purple).

This change is implemented at the point in the script after the first *IMGmask* is created through dilating the image with the detected edges. The previous method of determining the ROI is expanded in the following way:

If there are multiple objects recognizable on the image, the three objects with the longest contour lines are extracted and their integers are saved in a new vector. In addition, a matrix called *SizeOfCluster* is created to serve as a summery for the following calculated lengths. In the next step, a loop iterates through this vector and for every integer cluster it searches for the first and last row and column of its appearance. Beforehand, the first appearance variable is set to the maximum number of rows or columns, the last appearance variable to 1. Two separate loops then iterate through the rows and columns of the *IMGmask.* The code searches for the first match of the selected integer in the row or column and if it appears, as well as, if it is smaller than the predefined number, it is set as the first or last row or column of the cluster. Through these parameters the length of vertical and horizontal expansion is calculated. The expansions are weighted and then summed. As the horizonal expansion is supposed to have less influence it is weighted with 0.1, while the vertical expansion is weighted with 1.

Within the *SizeOfCluster* matrix, the cluster with the largest weighted sum of expansions is picked to identify as the ROI. In the following the *IMGmask* is reduced to the pixel that are equal to the chosen cluster. This *IMGmask* will now serve as the final mask for the feature calculations.

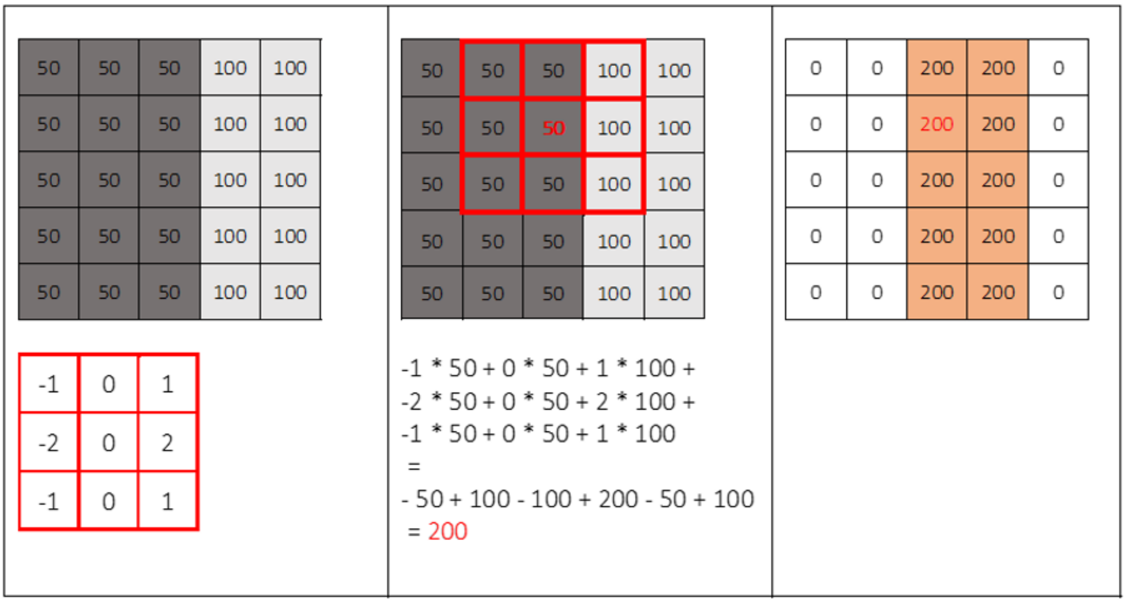
To evaluate how well Change A improved the detection of the actual ROI, a random subfolder of the LOKI dataset is chosen which contains a total of 1264 images. These images are processed with the *LOKI-Complete-data-primer* script in two cases, once with the Change A switch on and once with the Change A switch off. In the results, the source image is displayed together with a contour line around the ROI. Within the 1264 result images it is manually counted how many times the light attributes were chosen as the ROI for each case.

2.2.2.2 Operator and Edge Detection Improvement (Change B)

Change B aims to improve the edge detection itself. To calculate exact features from the *IMGmask*, it is important, that the edges are detected as close to the ROI as possible and allow the *IMGmask* to represent the organism as good as possible. Edges in an image can be detected by using filter kernels in the form of *ZxZ*-matrices. Filtering is carried out by convolving the original image I with the appropriate filter kernel *h*, producing the filtered image I' (Equation 3, (Burger & Burge, 2016)). Figure 11 shows an example of a linear convolution with a *Sobel* kernel on a matrix that clearly shows an intensity gradient and therefore an edge. In the resulting matrix on the right, the area of the edge is detected and enhanced, while regions of no gradient or intensity changes are 0.

$$I'[a,b] = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} h[i,j] * I[a-i, b-j]$$

Equation 3



50	50	50	100	100
50	50	50	100	100
50	50	50	100	100
50	50	50	100	100
50	50	50	100	100

-1	0	1
-2	0	2
-1	0	1

50	50	50	100	100
50	50	50	100	100
50	50	50	100	100
50	50	50	100	100
50	50	50	100	100

-1 * 50 + 0 * 50 + 1 * 100 +
-2 * 50 + 0 * 50 + 2 * 100 +
-1 * 50 + 0 * 50 + 1 * 100
=
- 50 + 100 - 100 + 200 - 50 + 100
= 200

0	0	200	200	0
0	0	200	200	0
0	0	200	200	0
0	0	200	200	0
0	0	200	200	0

Figure 11: Example of a linear convolution operation with a vertical 3x3 Sobel filter kernel (in red on the left side) on a matrix with a clear intensity change (5x5 matrix on the left). The middle shows the calculation steps for one cell (red number 50). The matrix on the right is the resulting matrix if the kernel was applied to all cells. In this case, in order to calculate the cells on the margin the matrix had to be replicated. The result shows a clear enhancement of the edge pixel cells while the surrounding pixel are 0.

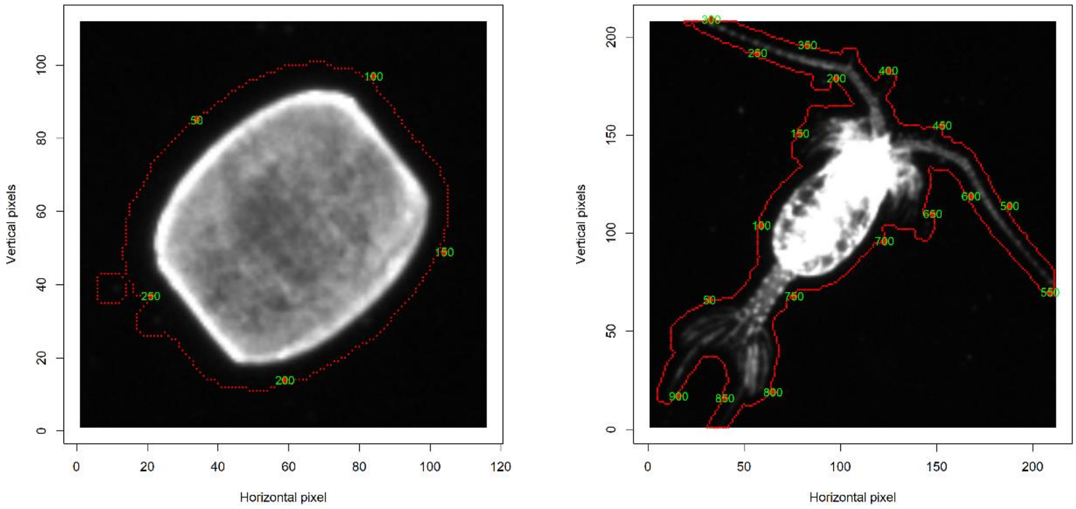


Figure 12: Two examples of source images together with the contour line (red dashed) using the edge detection operation of the original LOKI-Complete-data-primer script. The green numbers count the contour points. A gap between the contour and the organism is visible.

The original script uses a modified *Sobel* operator (Equation 1) on the LOKI images. It detects the edges of the ROI by emphasizing the region of the edge gradients (Figure 11). This leads to the edge contour line appearing to have some distance to the actual ROI in some images, as can be seen in Figure 12. During manual inspection of the resulting images, it was seen that the modified

Sobel operator is generally working well for finding the ROI. Therefore, the improvement of the kernel itself is not a concern for narrowing the gap between the edge contour and the ROI. During the work on an improvement of the narrow contouring of the ROI different methods were tested. One of the first implementations was the Canny edge detection (Canny, 1986) as it was implemented for MATLAB by Liang in 2017 and published on Github (Liang, 2017). The motivation to try this method for this thesis was found in Kekre & Gharge (2010) and Bature et al. (2015). Using this method alone helped narrowing the gap between the contour line and the ROI but also provided new problems as significant features of the organisms like the antennas of copepods were cut off in the edge detection process. Different methods were tried for a better overall image enhancement. But even if the LOKI has a relatively large depth of field (DOF) the imaged organisms may be unevenly illuminated. Rotation of body and uneven illumination inhibit enhancement strategies, e.g. contrast limited adaptive histogram equalization (CLAHE). Single body features like antennas are not as enhanced as the main body. The following, final chosen method turned out to be the most successful in narrowing the contour line while detecting the organism with all its body-features. A successful edge detection was achieved by applying a combination of different methods on the source image. Figure 14 shows a function graph, which contains graphical elements that were inspired by figures in Burger & Burge, 2016. This graph is divided into three sections (i-iii) showing every image processing step:

First section (i): A gaussian 5x5 filter *G* (Equation 7) is used to slightly blur the source image I in order to remove unwanted noise in the background. Next, a 5x5 *Sobel* filter *S* (Equation 8), which is very similar to the modified *Sobel* filter *K* (Equation 1), is used to detect the edges on the blurred image. This filter has a slightly higher value in the middle which emphasizes this layer. Both, the *Sobel,* and the modified *Sobel* filter kernel have a horizontal S_x and a vertical S_y component. The horizontal is equal to the vertical, turned by 90° and vice versa. The horizontal I_x and vertical I_y components of the filtered image are further processed in three different procedures. The first equation uses the original edge detection implementation with the sum of the squared components (Equation 4). The resulting image M^2 does not have strong visible edges and will be further processed in the next section. The second equation calculates the gradients magnitude (Equation 5, Burger & Burge, 2016) with the two components. This equation pronounces the body edges well, but also does not show very strong edges at the antennas. The third calculation on the horizontal and vertical components produces a matrix with information about the local edge orientation angle φ (Equation 6, Burger & Burge (2016)), which will later be used in the following image processing steps.

Second section (ii): During manual revision of the individual image processing results, it was clear, that neither of these first two edge detections (Equation 4, Equation 5) alone delivers good results for the overall edge detection. Keeping in mind the goal for Change B of finding edge contours that are close to the organism and not to cut off important features like antennas on copepods. To achieve this goal the detected edges had to be further enhanced. *M* and M^2 are therefore summed in the next step. At first, a threshold method is used on M^2 to produce a binarized image with thick edges M_b^2. This image will be used to further enhance the edges detected in *M*. M_b^2 is multiplied with 3 to enhance its effect on the sum. With *M* enhancing the finer edge structures and M_b^2 giving more weight to the broader contours of the ROI the resulting summed image M_s can be used successfully in the following Canny edge detection.

$$M^2 = {I_x}^2 + {I_y}^2$$ *Equation 4*

$$M = \sqrt{{I_x}^2 + {I_y}^2}$$ *Equation 5*

$$a = \tan^{-1}\left(\frac{{I_y}^2}{{I_x}^2}\right)$$ *Equation 6*

Third section (iii): The Canny edge detection implementation from Liang (2017) was translated to R and slightly adapted do deliver the best results. The first steps of this edge detection implementation have already been carried out with the grayscale conversion and the Gaussian blur *G* (first two sections of the function graph, Figure 14). Equation 4 and Equation 5 determine the magnitudes of the edges. Here the implementation was adapted by summing the calculated magnitudes to solve the previously mentioned problem of the detection of thinner body features like antennas. Next, the local edge orientation angle φ of the detected edges was determined with Equation 6. The detected edges of the resulting summed-up image M_s are still thick. Therefore, a non-maximum suppression (NMS) is used on the image M_s to find the pixel with the maximum value along the edge gradient. The NMS implementation works by going through each pixel of the image and if there is an edge, by checking the local edge orientation angle φ, and then performing an interpolation between the pixel according to the angle φ. For example, all angles with a value between 0° and 45° or between -135° and -180° are treated equally. Angle categories and corresponding pixel are shown in Figure 13. The chosen neighboring pixel correspond to the direction of the edge gradient. The interpolation between the pixel is used to improve the accuracy of the NMS. If the intensity value of the image is greater than the interpolated values the intensity value is kept for the output. If the neighboring interpolated value is higher, the regarded intensity pixel is set to 0. Only the pixel with the maximum will be left to build the output *N* of the NMS. After the NMS, the output is normalized and taken to the next processing step.

Double thresholding is performed on the image *N* to differ between strong and weak edges. The threshold values (t_{high}, t_{low}) are calculated for every image depending on their maximum pixel intensity, by multiplying a threshold ratio to the maximum value of the image *N*. If the image *N* pixel values lie over the high threshold t_{high} they are categorized as strong edges and set to 1. If they lie under the low threshold t_{low}, they are set to 0. Between these thresholds, pixel are interpreted as weak edges and not changed in their values. While the edges are evaluated, four vectors are filled with the indices of the row and column of the strong or weak edges. They are used in the following step to find connected weak edges.

A loop iterates through the number of strong edge pixel and the *Find-connected-weak-edges()* function scans through a +/-5 pixel neighborhood for weak edges. If a weak edge pixel is found, it is set to 1 and the function is recalled for that pixel to scan its +/-5 pixel neighborhood for more connected weak edges. Here a small adaptation was made to the code as this recursive function would recall itself so many times it would crash the script. Therefore, a maximum number of iterations was included into the function call. The last step of the Canny edge detection implementation is the deleting of remaining weak edges. This step was given a sub switch in the set-up section of the *LOKI-Complete-data-primer* script, as it turns out that deleting all the remaining weak edges often deletes the antenna parts of the copepods or other thin or smaller plankton body features. Therefore, deleting of weak edges was again disabled within the final script and is not part of the function graph (Figure 14).

Since the Canny edge detection algorithm produces many intermediate result images a new output was defined for the script where all image processing steps are displayed. In the final implantation of the script only every nth-image has an output for the processing steps since it takes up a high calculation time to produce the image plots. An example for the image processing steps can be seen in the Results chapter 3.2 or in the function graph (Figure 14).

To evaluate Change A, 1264 images were processed with the *LOKI-Complete-data-primer* script for two cases with Change A switched on for the first case and Change A switched off for the second. For both cases Change B was switched off. To evaluate Change B, the script was run again with Change B switched on. 100 out of the 1264 images were then randomly chosen and their contour line features were extracted from the *Image data.primedIMGdata* output table. The contour line length and area results of the case of Change B switched on were compared to the results of Change B being switched off (Change A on). It was made sure, that those 100 images did not contain cropped body features, that both results contain the entire organism and that the contour line is not connected to light attributes or other organisms.

For a second evaluation of the improvement through Change B the resulting feature tables were classified using linear discriminant analysis (LDA) on the *Peru* dataset, which was also primed twice (Change B switched on, Change B switched off). Since the *Peru* dataset contains a relatively small number of images (1023) the splitting and the testing were conducted fifty times to calculate the mean classification accuracy and its standard deviation.

Gaussian blur filter

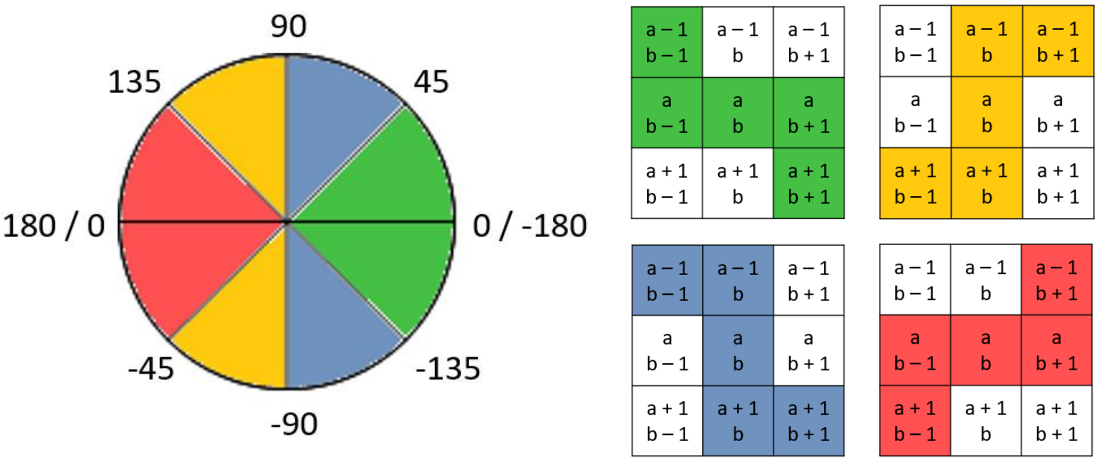
$$G = \begin{pmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{pmatrix} \times 273^{-1}$$

Equation 7

Sobel operator

$$S_x = \begin{pmatrix} 2 & 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 & -2 \\ 4 & 2 & 0 & -2 & -4 \\ 2 & 1 & 0 & -1 & -2 \\ 2 & 1 & 0 & -1 & -2 \end{pmatrix} \qquad S_y = \begin{pmatrix} 2 & 2 & 4 & 2 & 2 \\ 1 & 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -2 & -1 & -1 \\ -2 & -2 & -4 & -2 & -2 \end{pmatrix}$$

Equation 8



🟩 a − 1 b − 1	a − 1 b	a − 1 b + 1
🟩 a b − 1	🟩 a b	🟩 a b + 1
a + 1 b − 1	a + 1 b	🟩 a + 1 b + 1

a − 1 b − 1	🟨 a − 1 b	🟨 a − 1 b + 1
a b − 1	🟨 a b	a b + 1
🟨 a + 1 b − 1	🟨 a + 1 b	a + 1 b + 1

🟦 a − 1 b − 1	🟦 a − 1 b	a − 1 b + 1
a b − 1	🟦 a b	a b + 1
a + 1 b − 1	🟦 a + 1 b	🟦 a + 1 b + 1

a − 1 b − 1	a − 1 b	🟥 a − 1 b + 1
🟥 a b − 1	🟥 a b	🟥 a b + 1
🟥 a + 1 b − 1	a + 1 b	a + 1 b + 1

Figure 13: Within the Canny edge detection algorithm, a process called non-maximum suppression selects only the one pixel of an edge gradient that has the highest value. Within the applied non-maximum suppression, the spaces between the pixel were interpolated for a more accurate decision. To interpolate the correct pixel for each edge-point the direction of the edges angle - as calculated in Equation 6 - is used to define which neighboring pixel are to be utilized for the interpolation. All angles between 0 to 45 and -135 to -180 are treated in the same way. Here the green color within the angle circle (left, first matrix) corresponds with the green colored matrix of neighboring pixel (right). The same applies for different angles as can be followed by the color code.

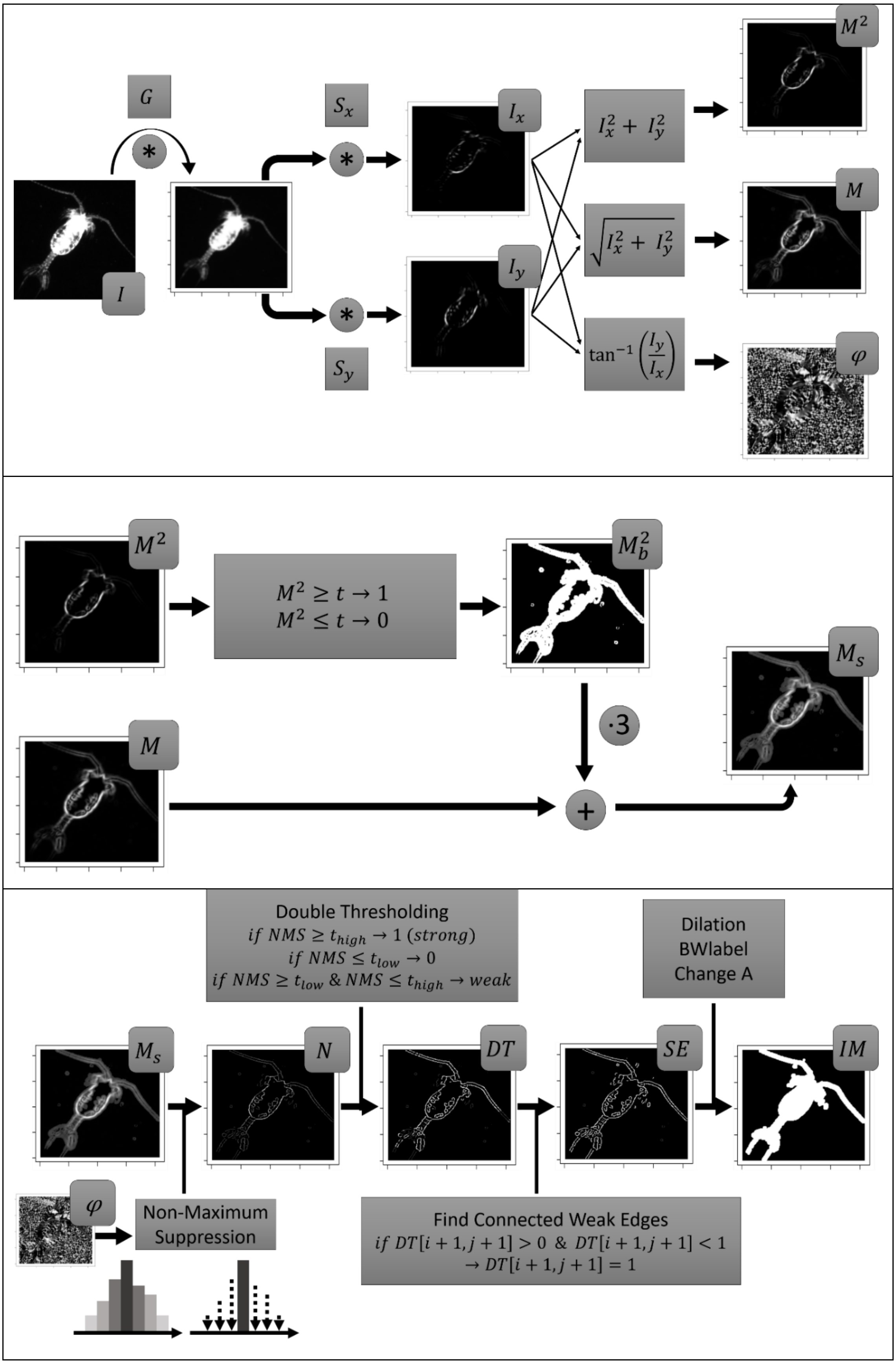

G
$*$
I
S_x
$*$
I_x
S_y
$*$
I_y
$I_x^2 + I_y^2$
$\sqrt{I_x^2 + I_y^2}$
$\tan^{-1}\left(\frac{I_y}{I_x}\right)$
M^2
M
φ
M^2
$M^2 \geq t \rightarrow 1$
$M^2 \leq t \rightarrow 0$
M_b^2
$\cdot 3$
M
$+$
M_s
Double Thresholding
$if\ NMS \geq t_{high} \rightarrow 1\ (strong)$
$if\ NMS \leq t_{low} \rightarrow 0$
$if\ NMS \geq t_{low}\ \&\ NMS \leq t_{high} \rightarrow weak$
Dilation
BWlabel
Change A
M_s
N
DT
SE
IM
φ
Non-Maximum Suppression
Find Connected Weak Edges
$if\ DT[i+1, j+1] > 0\ \&\ DT[i+1, j+1] < 1$
$\rightarrow DT[i+1, j+1] = 1$

Figure 14: Function Graph in three sections, some of the graphic elements were inspired by Burger & Burge (2016)*. The first and second section correspond to the image processing steps before the Canny edge detection and the last section displays the Canny edge detection algorithm. First section: The raw plankton image* $\mathbf{I}$ *is filtered with a Gaussian blur filter G and then further filtered*

with the horizontal S_x and vertical S_y Sobel filter. The horizontal and vertical component are then used in three different equations (Equation 4 and Equation 5 to combine the detected edges M^2 and M and Equation 6 for the angle of the direction of the edge φ). Second section: M^2 is used in a thresholding procedure, where the edges of the region of interest are enhanced. This result M_b^2 is then tripled and added to M. Third section: The Canny edge detection algorithm consists of the non-maximum suppression NMS. A double thresholding DT and the finding of connected weak edges SE. Finally the discovered edges are dilated again to connect disconnected features that belong together and then a binary image mask is created which is further processed in Change A (3.1).

2.2.2.3 Detection of Large or Corrupt Images (Change C)

Some of the images that were provided by the Medea-AV-GmbH's algorithm are corrupted. They lead to errors in the original script, which is why Change C was implemented with the *tryCatch()* function to find the corrupted images before the image processing procedure starts. If a corrupted image is found image processing is skipped and instead of image features the entry for the image data table is "Image is corrupted".

When the LOKI is heaved with less speed during the last meters of the up-cast, it may occur that large masses of plankton are accumulated close together and the ROI detection algorithm of the Medea-AV-GmbH creates large images with multiple organisms and/or air bubbles (Figure 15). When these images are processed in the image processing chain in the original script, as well as in the so far improved script, they cause errors within the feature detection. To prevent these images from getting processed two measurements are performed.

First, the up-casts were generally cut at the second mark when the telemetry data showed a speed of less than 0.05 $m \cdot s^{-1}$ (chapter 2.1). Second, a security step was taken within the script to exclude all images with a horizonal or vertical pixel length of over 1,500 pixel or a contour length of over 20,000 pixel. For these cases, the feature extraction is also skipped and the entry in the image data table is either "IMG too large" or "Contour too long". When the image has the right size but the organism is so large, that it has a very long contour line (over 20,000 pixel) the calculation is slowed down and eventually the feature extraction will produce errors, and these images are therefore not processed.

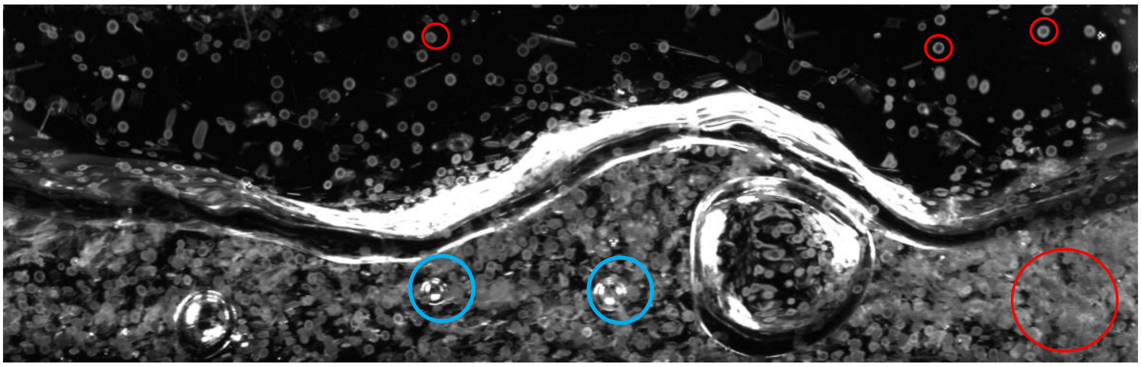


Figure 15: LOKI image of large accumulation of plankton, mostly diatoms (filling some parts of the picture, examples in red circles) and air bubbles (examples in blue circles). This image has a height of 696 pixel and a width of 2336 pixel and is therefore discarded from the image processing since with all its detail it would slow down the image processing dramatically.

2.2.2.4 Parallelization (Change D)

By default, R uses only one of the central processing unit (CPU) cores of the computer to run an R script. When a loop needs to be repeated several times to process an image, each time, each of these iterations is carried out successively. There are packages in R that allow the script to be parallelized, to use multiple cores and to speed up computation time. For a Windows operating system the required packages are *parallel* and *doSNOW*, which provides a parallel backend for the *%dopar%* function, and *foreach()*, which provides a foreach looping construct. This looping construct is similar to the for-loop but is able to occupy multiple CPUs at the same time. The input for a foreach-loop is an iteration variable and several different options plus the required expression. For this part it was necessary to concatenate the output rows via *rbind*, include the image processing packages and get an *in-order* output. Additionally, a progress information is printed into the console and errors that come up within the expression are passed to keep the entire script running. To achieve a parallel process, the entire telemetry and image priming sections of the script were captured within a function for each process which were then called in the foreach-loops. These loops are implemented at the end of each section. For them to run and use multiple cores, first the number of available cores needs to be detected and are clustered. This cluster is then given to a function to register the *SNOW parallel* backend with the *foreach* package, which provides the mechanism to run the loop in parallel. After the foreach-loop is done the cluster needs to be stopped again. As an evaluation measure the script was implemented with and without switching on Change D on the same dataset and the processing time was determined and compared for both runs.

2.3 Identification of Discriminating Features - Feature Selection

In the following an image feature is considered as a numerical value representing a specific characteristic extracted from the image. The performance of a classifier can be compromised if redundant or irrelevant features are included in the classification process (Sosik & Olson, 2007). Thus, feature selection is an important task to reduce the dimensions of a dataset before classification methods are applied. For the selection of image features, which - when combined - make it possible to discriminate and classify plankton images, it is necessary to work with a pre-classified dataset, in order to confirm the separating power of the selected features.

For this task, the *Peru* dataset is used. It contains 1,023 images in 21 classes (Appendix 1). The *LOKI-Complete-data-primer* script produces an output vector with 92 parameters for each image. Seventeen of these parameters represent meta-data information and 75 are the manually calculated image features (Appendix 4). To find final features of high importance for the discrimination of the plankton images, redundant information is excluded from the feature pool. At first, the 75 features are examined for their linear dependencies. A *pairs* plot displays the linear correlation coefficients of all features tested against each other. The R-script ‚R-Basic‘ from Schulz (2017) for the lecture ‚Multivariate statistics‘ in the study course Marine Sensorik, University Oldenburg, produces a *pairs* plot (Figure 16) showing the linear correlation coefficient on the upper triangular sub-matrix and data elements plotted against each other on the lower side. Figure 16 shows an example of a *pairs* plot with a few image features. The *pairs* plot is used to gain an overview over the feature dependencies. To sort out features with high correlation coefficients the *cor()* function is used in R to access the linear correlation coefficients. All features which have linear correlation coefficients of >= 0.98 are listed and excluded from the feature pool. During the exclusion process a linear discriminant analysis (LDA) is computed. If there are strongly collinear features still present within the dataset, the *lda()* function from the R package *MASS* will display a warning message. After excluding features with correlation coefficients >= 0.98 this warning was not displayed anymore.

Subsequently a random Forest model is computed to provide the importance of the given features in the form of the *Gini Index*. Since random Forests provide results based on a randomized selection of features, a sufficiently large number of trees needs to be occupied to provide a stable *Gini Index*. Strobl et al. (2007) imply that if the feature values in the dataset are continuous and only features representing the same number of classes are considered in the sample, feature selection with random Forest feature importance measures is applicable.

A random Forest is a classifier consisting of a collection of single tree-structured classifiers. Each classifier is an independent decision tree, which casts a vote for a target class from a randomly selected vector out of a dataset (Breiman, 2001). Due to a large number of trees, single misclassifications do not weight heavily into the overall classification result. A decision tree consists of a root, containing all data and all features, nodes, where decisions about separating the data based on the given features are made, and branches, representing the classified data after each node. For the creation of a decision tree it is also necessary to use a pre-classified dataset, which was previously divided into a training and a testing dataset. At each node, a predefined number of features is randomly drawn from the training dataset. If multiple features are drawn, the feature that best separates the data is chosen to work as the decision rule within this node. The data is then separated by this rule into different branches each leading to a new node. If a branch with separated data only contains data elements from one class, it is turned into a leaf and not further processed, to save computational time. If the testing dataset is now processed with this tree the data elements are led through to branches of the tree into their predicted class. Within a random Forest, which contains dozens to thousands of decision trees, each tree casts a vote for the class of each data element. In a final majority vote the class for each element is determined. The importance measure of the features is the previously mentioned *Gini Index*.

The *Peru* dataset is used to run a random Forest model and to obtain the *Gini Index* information on the features. Although each random Forest model is independent and randomly created, a pooled analysis was performed to identify the importance of the individual features in the analysis. A random Forest model with 1000 trees was performed 100 times, the *Gini Index* was recorded for each iteration and the mean value of all iterations as well as the standard deviation were calculated.

At this point it is not known how many features are needed for the best classification. Therefore, the mean *Gini indices* were sorted decreasingly and turned into an exclusion vector within a for-loop that excludes the features stepwise; based on their *Gini Index*, unimportant ones first. Within this loop the accuracies of a LDA, a SVM and a random Forest classification are calculated. It is noted that each classification method has their accuracy maximum at a different number of features within the feature subset. The accuracy was calculated from the sum of the diagonal proportion of the confusion matrix of each method. These accuracies were then plotted against the number of features still included in the model to define the optimal feature subset size.

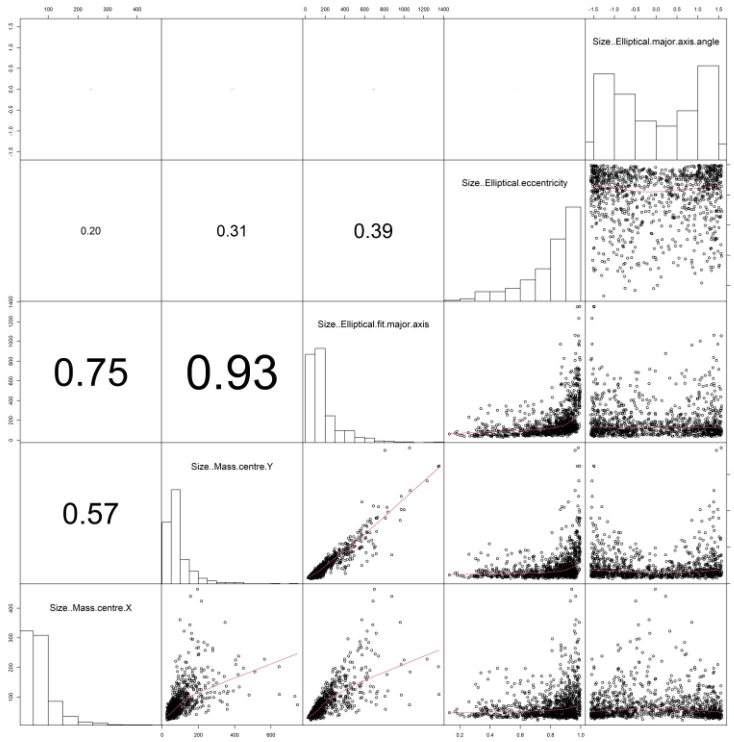


Figure 16: Example of a pairs plot of 5 image features. The diagonal cells show the names of the features and a histogram of point distribution of all the cells underneath each diagonal cell. On the upper triangular matrix, the linear correlation coefficient can be found, printed in a font size corresponding to the value. On the lower triangular matrix, the elements of each feature are plotted against the elements of the other features.

2.4 Classification Approaches with Multivariate Statistics, Machine Learning & Convolutional Neural Networks

The classification of plankton images can be attempted with a multitude of possible methods. The methods chosen within this thesis try to cover different types of classification tools. The “traditional” classification uses multivariate statistics and machine learning algorithms. These methods require a table of image features, which were extracted from the images as invariant values such as contour line metrics, grayscale distribution, pattern characteristics and others. The computation of a respective table for each classification method was introduced in section 2.3.

A more recent approach to plankton image classification is deep learning, respectively convolutional neural networks (CNN). Their basic structure and architecture have been described in chapter 1.3.3. CNNs need a large training dataset to produce reasonable image classification. In contrast to machine learning methods, they do not require a previous manually extracted feature table since they operate directly on the images, calculating all necessary deep features during their own computation. For this thesis, the use of CNNs was an outlook to further improvements to the LOKI software. Therefore, classifying the LOKI dataset was tested by using transfer learning with the *AlexNet* (Krizhevsky et al., 2012) in MATLAB (The MathWorks, 2019), and within the Object Detection application programming interface (API) (Roboflow, 2020) from TensorFlow 2 (TensorFlow).

As a first approach the *Sognefjord 26 groups* (Appendix 1) dataset was classified with a random Forest analysis. The respective results are displayed in 3.5. It was then decided to summarize a few classes to their next higher rank in order to achieve better classification accuracies. The motivation for this step will be discussed in chapter 4.3.

All following classification methods use the pre-classified dataset *Sognefjord 14 groups* or *Sognefjord augmented* (Appendix 1) in order to train, validate and test their models. This dataset contains 14 morphological classes (Figure 7 and Appendix 2). CNNs work best when a large training set is available, therefore, the classes that contain less than 250 images are supplemented with augmented images. For the image augmentation the function *Augmentation()* within the R package *OpenImageR* was used. The images were shifted, rotated, or flipped to produce a larger dataset for the smaller classes. For the classification approaches with the LDA, random Forest and SVM the manually calculated feature tables are needed. The feature tables were created on the original dataset without the augmented images due to a high sensitivity of the image processing to edges that are created during the augmentation process. As previously described, the result from the *LOKI-Complete-data-primer* script comes in form of a table with 17 metadata parameters and 75 image features. For the pre-classified datasets, the result tables are supplemented with a column for the folder name, respectively the class name. The R script *20200909_CreateTrainAndTestDataset_FromClassedTable.R* (Appendix 7) is used to split the result table into two separate tables, one for training and one for testing. The size percentages for each subset are 80 % for training and 20 % for testing. These split datasets are used for the "traditional" classification methods. In the CNN approach the training and testing datasets are produced within the MATLAB implementation. All classification approaches will be visualized with a confusion matrix. Confusion matrices are used to quantify the accuracy of a classifier (Hu & Davis, 2005; Luo et al., 2018). To create a confusion matrix, the actual class affiliation or each data element needs to be known next to a class prediction for each element by a classifying method. In a matrix or table, the number of true positive elements, the number of true negatives, the number of false positives and the number of false negatives for each class are derived from the actual classes and the predictions (Lumini & Nanni, 2019).

2.4.1 Multivariate Statistics - Linear Discriminant Analysis & random Forest

The linear discriminant analysis (LDA) searches for linear combinations of features allowing the best possible discrimination of the given classes (Schlittgen, 2009). The LDA model is set up by using a dataset with previous known classes. Building the model works by searching for the largest distance between the mean feature values μ of the predefined classes with the smallest feature variance *s* within the class (Equation 9). The LDA attempts to reduce the dimensions of a dataset by condensing many explanatory features to a few derived gradients with the least possible loss of information (Leyer & Wesche, 2007). A linear plane - called the discriminant - is searched, that best separates between the classes. A discriminant can always just separate two classes, so when multiple classes are present linear discriminants are calculated for every combination of classes. A goal of the LDA is to determine which features of the dataset are most important in the discrimination of the different classes (Schlittgen, 2009). The *lda()* function delivers the coefficients of linear discriminants which weigh the features in terms of the ability to distinguish between the classes. Higher values indicate a higher separating power of the features (Leyer & Wesche, 2007).

The coefficients of linear discriminants w_{cs} can be used to calculate the discriminant score DS_c (Equation 10; Schulz, 2017). The discriminant score shows the probability of an element with properties for the given features to belong to a predefined class. The class constant k_c is added to the sum of the product of the discriminant coefficients w_{cs} and the variables x_s for the number of features *n* in a dataset. The results of the LDA can be summarized and displayed in a classification matrix, where the number of elements that have been sorted into a class is displayed in comparison to the predefined classes (Schulz, 2017). When the LDA model is set up with a training dataset, a testing dataset can be used to verify the ability of the model to discriminate between the different classes. The *lda()* function in R (*MASS* package) can perform a leave-one-out-cross-validation. This means that subsets of the original data are left out during the creation of the model and the model is later validated by those subsets.

Next to using a LDA to classify the images with the feature table, a random Forest classification is performed as well. The methods of a random Forest analysis have previously been described in chapter 2.3.

$$\frac{(\mu_1 - \mu_2)^2}{s_1^2 + s_2^2} \qquad \textit{Equation 9}$$

$$DS_c = k_c + \sum_{i=1}^{n} w_{cs} * x_s$$ *Equation 10*

2.4.2 Support Vector Machines (SVM)

When searching for the best separation of data element clusters in a two-dimensional dataset with two classes, the best separator would be a line that lies in the middle of the outermost data elements of each class (Figure 17, left). This line is called the support vector classifier. The space on both sides of the separating line up to the first data elements of each cluster is called the margin. These elements are called support vectors (circled in green in Figure 17, left) (Bennett & Campbell, 2000). The example in Figure 17 (left) is simple since the data is linearly separable in a two-dimensional space. If this does not apply, like for a one-dimensional dataset with two classes, which has one class lying in-between the elements of the other one (Figure 17, right - empty circles), the dataset can be squared (Figure 17, right - filled circles), to separate the classes by a line (the support vector classifier).

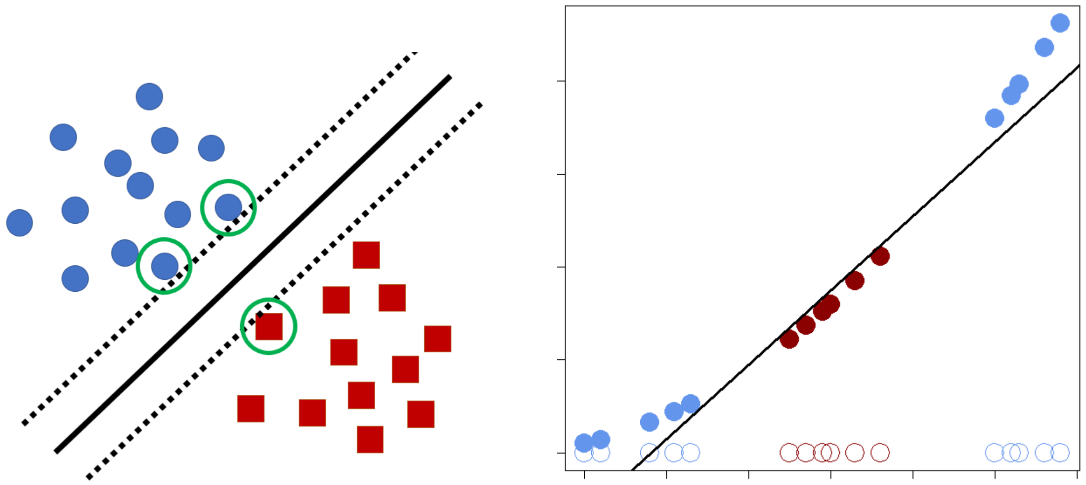


Figure 17: Left: The figure was adapted from Bennett & Campbell (2000). *Two datasets (blue circles and red squares) are visualized in a two-dimensional plane. The solid line is a good separator between the data clusters since it has maximally large margins (space until the dashed line) towards the datapoints. The green circled data points represent the support vectors. Right: A one dimensional dataset with two classes has one class lying in between values of the other and cannot be separated by a line (empty circles). If the dataset is squared (filled circles), the classes can be separated by a line. This is an example of a support vector machine.*

This method is called support vector machine (SVM). Basically, the method starts with data in a relatively low dimension and moves it to a higher dimension to find a support vector classifier that separates the higher dimensional data. The SVM converts a dataset from a lower to a higher dimensional space, by applying a fitting kernel function to systematically find support vector classifiers in the higher dimension. The kernel function calculates the relationships between the data elements, which are then used to find the support vector classifier. This calculation is conducted by applying the "kernel trick", which allows the calculation of the relationships between the data elements to take place in the same dimension using the dot product, instead of converting every element to the higher dimension making the calculation more complex.

Generally, the margin of the support vector classifier is sought to be maximized while the error is minimized (Bennett & Campbell, 2000). This sometimes means that there will be a tradeoff between the bias of a classifier to sort a point into the right class and the variance a single class may have.

To implement a SVM model in R, the R package *e1071* is used. Within the SVM-script *20200909_SVM_14classes_37057_26features.R* (Appendix 7) the pre-classified training table - with the selected features for the SVM - is called to create a fitting SVM model. After the first SVM model has been set up, it is tuned with the *tune()* function to find the best values for the required input variables *gamma* and *cost*. The *gamma* variable defines how much influence every training element has towards finding the support vector classifier. When *gamma* is low even the elements further away from the support vector classifier influence its layout and when *gamma* is high, just the elements close to the support vector classifier impact its layout. The cost variable defines a tradeoff between the training accuracy and the margin space for the prediction function. This means that high cost values create a complex support vector classifier which misclassifies the least possible amount of training data, while a lower cost value will lead to a simpler prediction function (Krizhevsky et al., 2012). The *tune()* function returns the best values for these variables, which are included into the improved SVM model, which is then used to predict the classes for the test set.

2.4.3 Convolutional Neural Networks

The general architecture of a convolutional neural network (CNN) has been outlined in the introduction (see chapter 1.3.3). Within this chapter the learning process of a deep neural network is further explained and first approaches with transfer learning using a 12-layer *AlexNet* (Krizhevsky et al., 2012) as well as training a larger CNN for object detection with *TensorFlow2* (TensorFlow) will be introduced.

Figure 18, was adapted from (Chollet, 2018) and shows the learning structure of a deep neural network. The input is processed by convolutional, pooling, and fully connected layers until resulting in a prediction of a class affiliation. These results are compared with the true target class and the loss function computes a score of how well the prediction fit to the true target. This loss score is used as a feedback signal, which is send to an optimizer adjusting the layer weights in the deep neural networks' architecture. This feedback mechanism is called backpropagation and is the central algorithm in deep learning (Chollet, 2018). If a new CNN is created, the layer weights are first set randomly and are then adjusted during the training process, with transfer learning the weights have predefined values from previous learning processes (Pan & Yang, 2009). The weights are further adjusted and tuned with the new dataset during the transfer learning training process.

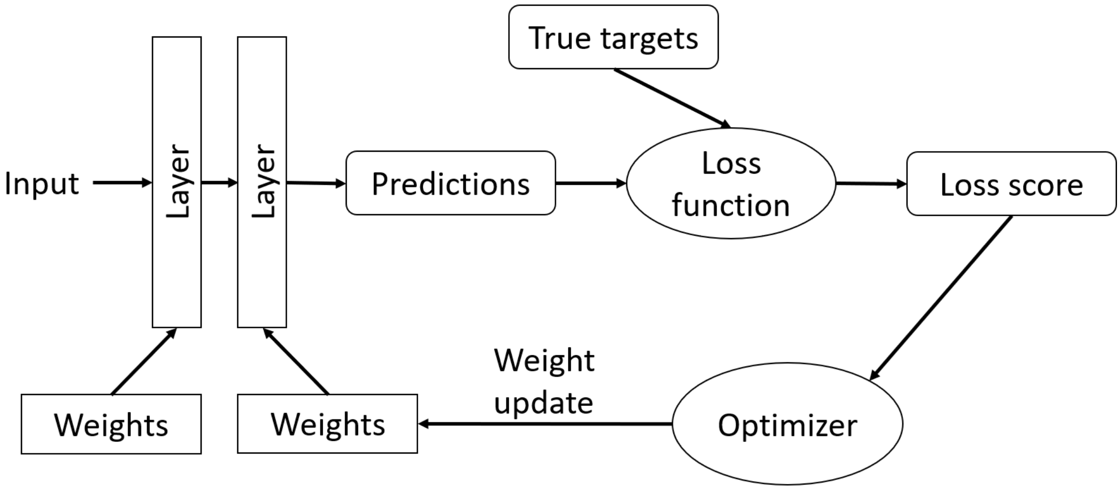


Figure 18: Scheme of the learning process within deep neural networks architecture. The prediction result of the layers is compared to the true targets creating a loss score. This loss score is used to optimize the weights on the layers as a feedback and learning mechanism over a destinated number of iterations. The figure was adapted from Chollet (2018)

To approach the classification of LOKI plankton images using CNNs transfer learning was used with the *Deep Learning Toolbox Model for AlexNet Network* (Appendix 6) in MATLAB. Additionally to this toolbox, the *Deep Learning Toolbox* (Appendix 6) was also installed to provide the needed framework. *AlexNet* was trained on 1.2 million images for 1,000 classes by Krizhevsky et al. in 2012. The pre-trained architecture has learned how to classify images and can be fine-tuned with the LOKI plankton images to serve as a classifier for this custom cause. In MATLAB *AlexNet* can be loaded into the workspace as a network structure. Its layers can be alternated and supplemented, and the network options can be modified. The two layers of a CNN which are modified for a transfer learning implementation are the fully connected layer and the output layer. The required numbers of classes to be detected within the dataset are provided to these layers.

Within the MATLAB implementation, the LOKI images are stored in an image datastore with the *imageDatastore()* function. This function allows access to the images without having to load them into the MATLAB workspace. Furthermore, they can be stored using their folder names as a label.

This is useful when the images are split into training, validation, and testing dataset in the next step. The validation dataset has not been mentioned for the "traditional" methods. It is used within the training of a CNN to validate the classifying accuracy right away and provide feedback information within the training process before using the created CNN on the testing dataset. With the function *splitEachLabel()* all three data subsets can be created randomized in the required dataset sizes. The size of the training dataset should be equal for all classes to avoid an overfitting of the network for the classes with the largest amount of data. An established partition of the dataset is to use 70 % of the images for training, 20 % for validation and 10% for testing. If the dataset is imbalanced a fixed number of images can be set for each set. *AlexNet* requires the images to have a format dimension of 227 x 227 x 3 pixel. Therefrom, 227 pixel are for width and height and 3 pixel in the third dimension for color images (red, green and blue channels). The LOKI images have various sizes and are all grayscale images, thus having only two dimensions. The image datastore can be augmented to fit the LOKI images to the required format.

The most important CNN options - that can be modified and experimented with - are the learning rate and the momentum. The learning rate represents the step size approaching the optimal weights for the CNN in the learning process. After each iteration, the feedback mechanism changes the step direction to get closer to the best weights. The momentum is the proportion of the angle of the step taken for the next iteration. If the momentum were set to 1, the next step would turn 100 % into the new direction, calculated by the adjusted weights. If the momentum were set to values < 1, only this proportion of the angle would be used for the next step (Chollet, 2018).

After the neuronal network is fine-tuned with the LOKI training and validation images, it is used on the test dataset. To acquire an overview over the CNNs results, a confusion matrix is created, and the percentage of the overall accurate classification is calculated. The important variables to consider for judging the overall success of the CNN classification are the accuracy and the loss score.

A second approach uses a *TensorFlow2* environment. *TensorFlow* is an end-to-end open source platform for machine learning (TensorFlow). For custom object detection, it provides an application programming interface (API) in the form of a *Google colab* (Roboflow, 2020) notebook with an established deep learning algorithm. This API uses the YOLOv4 (Bochkovskiy et al., 2020) CNN architecture.

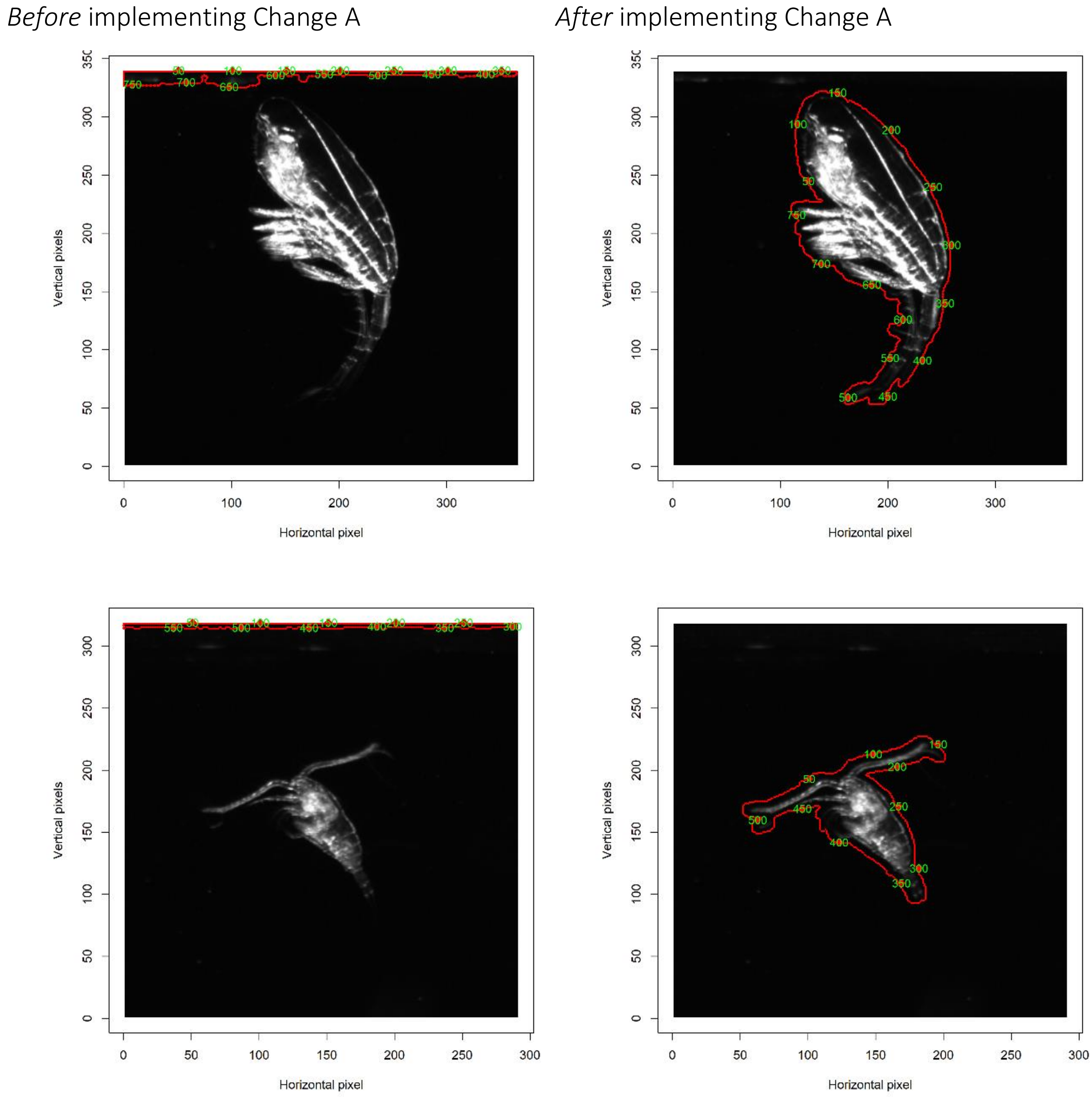
To implement custom data into their setup, a tutorial (Solawetz, 2020) is provided. For a faster computation speed an online connection to a GPU is provided. It was possible to run this algorithm with the provided example images. In a next step the LOKI images are provided to the network. To use the LOKI images for this cause, they need to be annotated by a labeling tool beforehand. Roboflow is a website and online tool (Roboflow Inc., 2020) for data augmentation, labeling and creating datasets split into training, validation, and texting sets. It was used to annotate a small dataset containing 180 images in four classes (*Peru small*, Appendix 1) which could be implemented within the TensorFlow environment afterwards. Training this implementation of a CNN takes up a long time and requires a lot of computational power. Due to technical limitations the approach for this implementation was left at this state and no results are available. When a GPU is locally available, it will be tested again in the future.

3 Results

The results chapter is organized into the Changes A-D to improve the original script and into the outcome of the feature selection, and image classification approaches.

3.1 Change A

Change A aimed to improve the selection of the ROI within the image when light attributes were present, which have longer contour lines than the actual organism. Therefore, the vertical and horizontal expansions of the object clusters were considered. Figure 19 displays the *before* and *after* results for two sample organisms for Change A. On the left side, *before* Change A, the resulting contour line encloses an object on the top of the image (Figure 19, left). All images with these light attributes in the top margin have the y-coordinate 0 within the original camera frame. This means the images are taken from the outer margin of the frame and might be influenced by the LEDs illumination. On the right side, after Change A, the contours are now enclosing the ROI instead of the light attribute (Figure 19, right). To evaluate Change A a set of 1264 images was primed with the *LOKI-Complete-data-primer* script twice. Once with Change A switched off and once with Change A switched on. Within both result folders the number of images with a contour line surrounding the light attribute instead of the organism was manually counted. With Change A switched off 46 out of 1264 falsely identified images were counted, while with Change A switched on no images had a light attribute identified as the ROI. Change A improves finding the ROI in cases like the light attribute influence (Figure 19). But it has a weakness if there is an background object visible within the image that has a longer contour and also greater vertical and horizontal expansion (Figure 20). For these cases, no further improvements are implemented yet, but ideas are mentioned in chapter 5.



Before implementing Change A

After implementing Change A

Figure 19: Two example organisms in their source image together with the edge detection contour result from before (left) and after (right) implementing Change A. Before Change A the light attribute on the upper image margin was detected for both source images and after implementing Change A the organism was detected correctly.

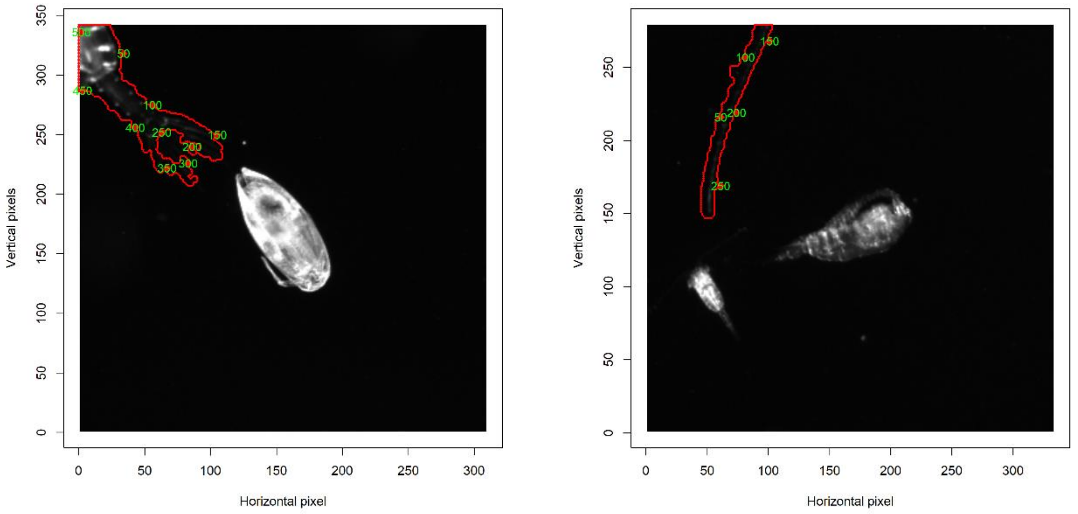


Figure 20: Two example plankton source images with the contour lines of the edge detection in misidentified regions of interest (ROI). The images show the actual ROI in the center, but each image also contains another organism which appears cropped at the margins of the image. In these cases, the cropped organism has a longer contour and a larger vertical expansion, which is an important part of the decision process in Change A to choose the correct ROI. Therefore, these ROI are misidentified before and after Change A. If the contour line of the cropped organism in the right image was to be shorter than the contour of the center organism, this would be an example of Change A worsen the output, just because the cropped organism has a large vertical expansion.

3.2 Change B

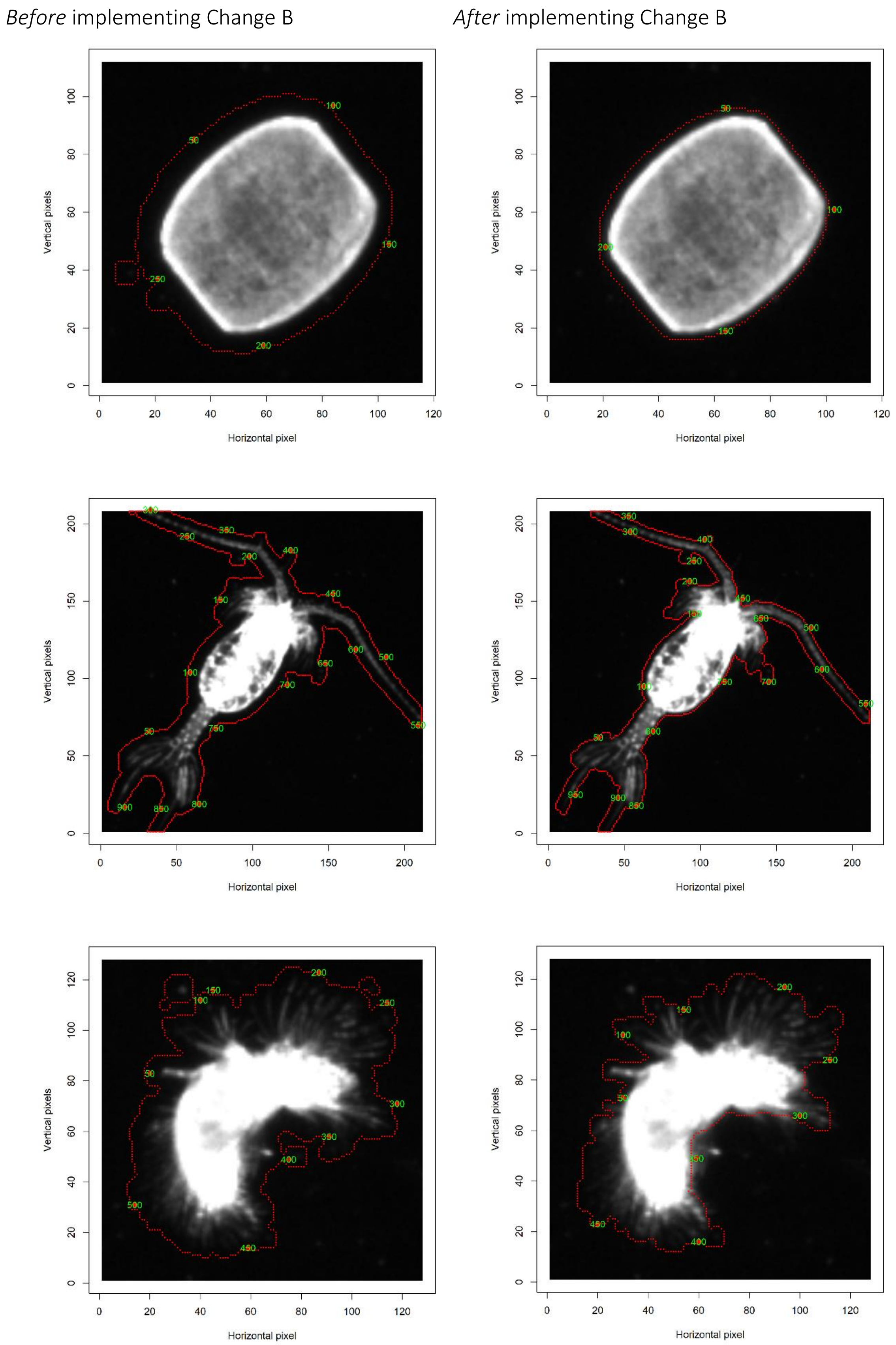
To goal of implementing Change B, was to improve the edge detection through narrowing down the contour line and finding the entire expanse of an organism without cropping body features like antennas. Figure 21 shows the *before* and *after* images for Change B. For these example images it is clearly visible that the contour line lies much closer to the organism after implementing Change B. The last pair of images (Figure 21, third row) shows a polychaeta organism with lots of fine bristles, their structure is harder to detect than the clearer lines of the copepod (Figure 21, second row) or the Bacillariophyceae (Figure 21, first row).

To evaluate the improvement through Change B a test was conducted using 100 random images out of a set of 1264 which were primed with the *LOKI-Complete-data-primer* script twice, once with Change B switched off and once with Change B switched on. For all selected images it was manually revised, that the contour line contains only one organism, without connections to other organisms or light attributes and without cropped body features. Two of the contour line image feature that were calculated within the feature extraction process are used here to compare the success of the contour line in each test to enclose the organism the tightest. The first feature is the *Contour Line Area*, basically the area of the *IMGmask*. This value is expected to decline as the contour moves closer to the organism and reduces the occupied space. Still it is possible that after Change B some finer organism structures are enclosed within the contour, which might increase the area slightly. The second feature, which is compared, is the *Center to Contour Points Median*. This feature describes the distance of each contour pixel to the center int of the *IMGmask*. As the contour closes is this feature is also expected to decrease in its value. Table 1 shows the results of this test. For the difference value of the features the a*fter* result was subtracted from the *before* result. After implementing Change B, the *Contour Line Area*, as well as the *Center to Contour Points Median* are smaller in their values. This result suggests an improvement success for the edge detection implementation in Change B.

Table 1: Evaluation of the improvement in edge detection through Change B. Two contour image features, which are expected to decrease in their value with a closer contour line are compared for 100 random images primed with (after Change B) and without (before Change B) switching on Change B. The difference columns show the subtraction of before-after.

	Contour Line Area (mean)	Contour Line Area (difference)	Center to Contour Points Median (mean)	Center to Contour Points Median (difference)
Before Change B	12123.11	2126.79	73.90	4.17
After Change B	9996.32		69.73	

Figure 22 displays one of the new outputs of the image processing, showing every processing step especially during the Canny edge detection process. On this particular example image it can be seen, that the image enhancement through adding the results of the binarized $S_{mag}{}^{2}$ (*IMGedges_SumSqrt_T* on Figure 22) to S_{mag} (*IMGsobelMag* on Figure 22) facilitates the detection of the antennas which would have otherwise had to small intensities to be detected. The *IMGdoubleThresh* and *IMGstrongEdge* show that through the function *FindConnectedWeakEdges()* the weak edges of the antenna are reconnected with the strong edge of the body of the copepod.



Before implementing Change B

After implementing Change B

Figure 21: Three examples LOKI plankton images of the edge detection from before (left) and after (right) implementing Change B. The produced contour line lays much closer at the organism itself when Change B is implemented.

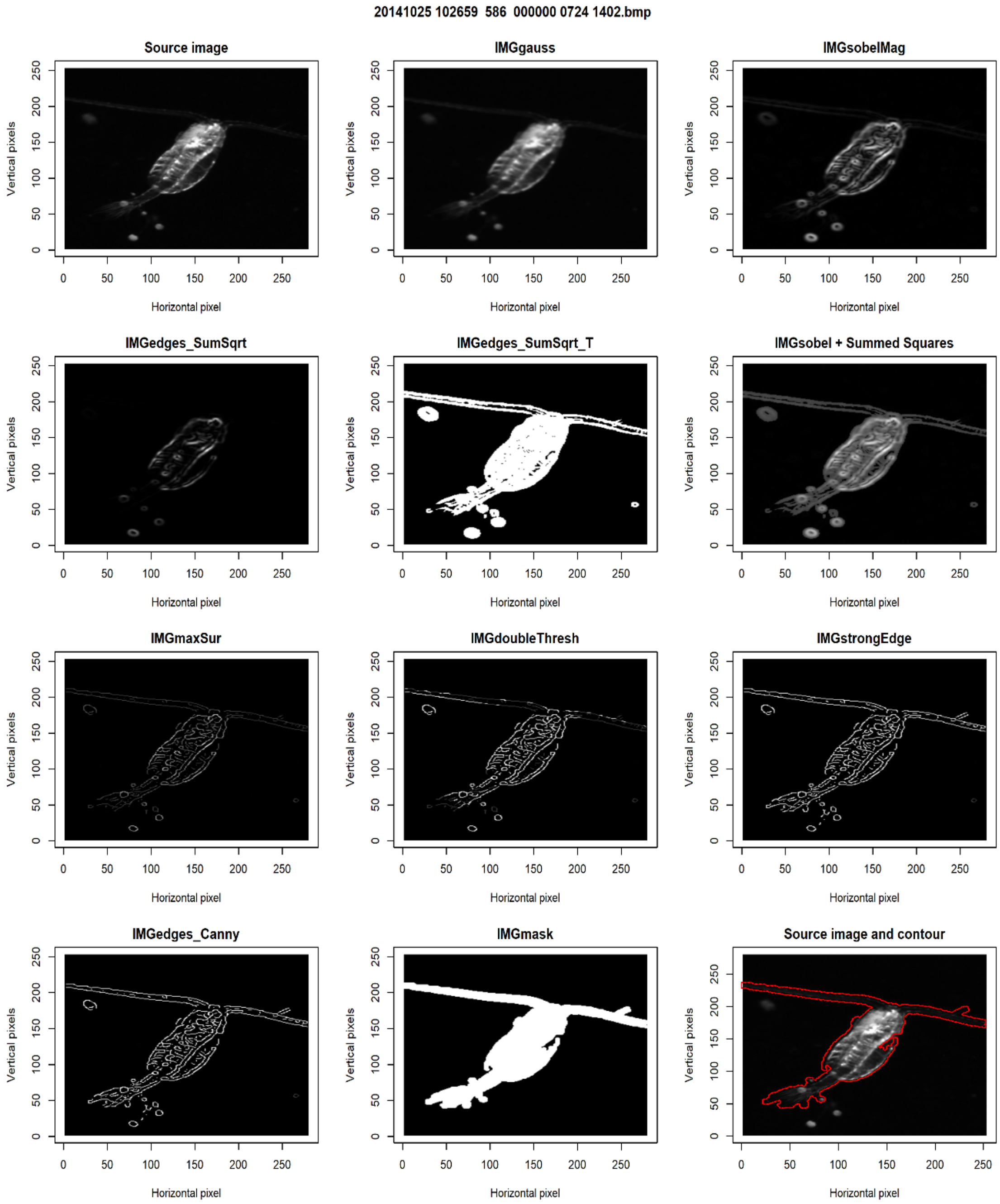


Figure 22: Example of the new output produced by the LOKI-Complete-data-primer script visualizing the single image processing steps. The order of the steps is left to right and top to bottom and will be numbered from 1-12 for better comprehension. The image processing starts with the source image (1) which is then blurred to cancel out unwanted noise (2). The edge detection images, produced through Equation 4 (3) and Equation 5 (4) and the threshold image (5) are displayed. (6) consist of the sum of (3) and (5) and visualizes the enhanced detected edges that will be further processed with the Canny edge detection algorithm. Non-maximum suppression (7), double thresholding (8) and the finding of connected weak edges (9) is conducted. For this image processing the deletion of the weak edges is disabled, to produce more accurate results with the current state of the LOKI-Complete-data-primer script, therefore, (9) and (10) are the same result. Finally, the binarized image mask is produced (11) and the contour can be applied to the source image (12) for visual confirmation of the edge detection process.

As a second evaluation of the image processing improvement through Change B, a LDA was used to classify the *Peru* dataset, which was pre-classified and primed twice, once with Change B switched off and once with Change B switched on. Both results tables were split randomly into training sets with 80 % of the image data, and testing sets with 20 % of the image data. Since the *Peru* dataset contains a relatively small number of images (1023) the splitting and the testing were conducted a hundred times to calculate the mean success value and standard deviation. Every iteration uses the same indices of randomly selected testing and training data for *before* and *after* Change B. Table 2 shows the results of this test. Before implementing Change B the classification accuracy is 69.59 with a standard deviation of 0.028 and after Change B was implemented the classification accuracy is 68.74 with a standard deviation of 0.029 .

Table 2: Mean accuracy and standard deviation of support vector machine classification of Peru data set before and after Change B was implemented in order to evaluate the impact of the improved edge detection on the classification accuracy.

	Before Change B	After Change B
Mean accuracy of SVM / %	69.59	68.74
Standard deviation	0.028	0.029

3.3 Change C and D

After implementing Change C and therefore skipping images that are either corrupted, too large or have a too long contour line, possible errors have been prevented and the script runs smoothly.

In a test, the *LOKI-Complete-data-primer* script was executed with Change D switched on and the processing time of the TMD and IMG processing sections were compared to a run of the script when Change D was switched off. Both tests were performed on the same computer. The parallelization used 11 instead of 1 CPU cores. The results can be seen in Table 3. Without parallelizing the process, the script takes 153 min to finish. When it is parallelized, it only takes a total of 25 min. Implementing Change D saved a lot of processing time (Table 3). For this test and amount of data, the parallel version runs more than six times faster than the normal version.

Table 3: Processing times of the LOKI-Complete-data-primer script on 1524 telemetry and 1936 image files before and after implementing the parallelization in Change D.

	Time Before Change D/min	Time After Change D/min
TMD processing time (1524 TMD files)	1.28	0.23
IMG processing time (1936 IMG files)	152.16	24.82
Total processing time	153.44	25.05

3.4 Feature Selection

The image processing of the LOKI plankton images results in a table with 92 features calculated for each image. The feature selection process goes through two separate steps to find features, which allow the best classification of plankton images. The first step is an exclusion of features having redundant information and linear correlation coefficients of R^2 >= 0.98. Therefore, a *pairs* plot and the *cor()* function are used in R. Within the *pairs* plot four features are visible which are constant for every image. Those are excluded before further analysis. Data points are also excluded for the feature selection when they show large outliers for many of the features. Outliers make it harder to detect visible dependencies in the *pairs* plot. After the exclusion of all features with a linear correlation coefficient of R^2 >= 0.98, the dataset was shortened to 49 features. The 49 features left are then ranked in the order of their *Gini Index* in the second step of the feature selection process. Table 4 lists the features in order of their mean *Gini indices* together with their standard deviation. The *Gini Index* was calculated by running a random Forest model with a thousand trees for a hundred times and then calculating the mean *Gini Index* and its standard deviation for each feature out of the hundred executions.

To find the best subset of features for the classification of the LOKI plankton images, it was noted that each method has their accuracy maximum at different numbers of features. Therefore, each method as assigned an own feature subset to create the highest classification accuracies. The results of the feature exclusion can be found in the corresponding classifier result chapter in 3.5. The optimal number of features for each method will include all features ranked by the *Gini Index* up to the optimal number.

Table 4: Image features sorted after the value of the mean Gini Index together with the Gini Index standard deviation (SD) after a hundred execution of a random Forest model with a thousand trees and the standard deviation of the Gini index.

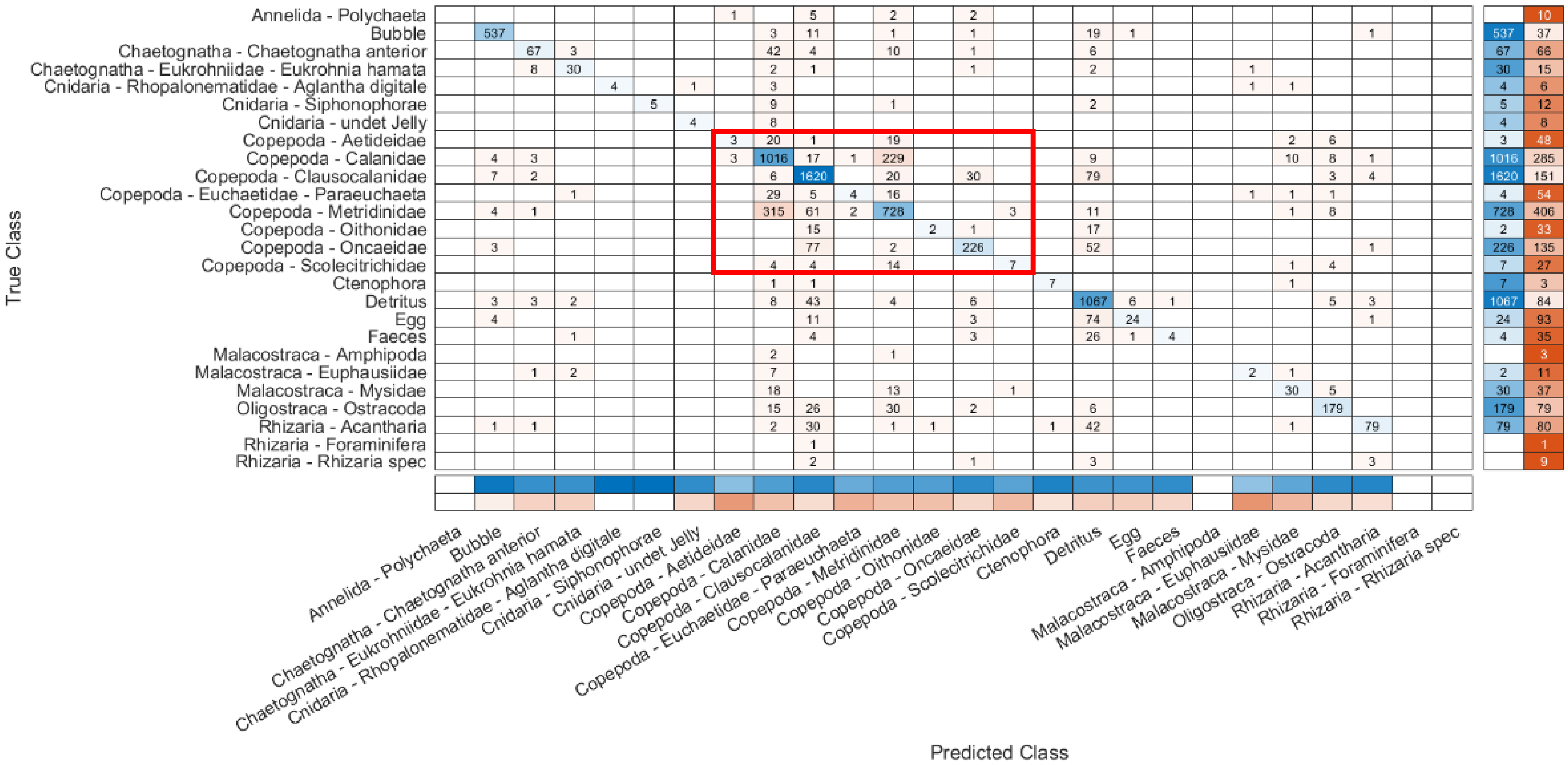
	Image Feature	Mean *Gini Index*	SD *Gini Index*		Image Feature	Mean *Gini Index*	SD *Gini Index*
1	COO: Solidity	40.63	0.55	26	Size: Mass center Y	18.12	0.40
2	COO: Centre to contour points variance	34.61	0.65	27	Intensity: Quantile 0.80	17.63	0.42
3	Pixel border distances: SD	31.71	0.48	28	Haralick: Homogeneity [ASM]	17.59	0.35
4	COO: Haralicks circularity	31.48	0.50	29	Pixel-border distances: median	17.57	0.38
5	Size: Elliptical fit major axis	30.41	0.61	30	COO: Convexity	16.82	0.29
6	Haralick: Image correlation [COR]	29.77	0.53	31	Haralick: Image information measure correlation [f13]	16.74	0.29
7	Pixel-border distances: sum	28.34	0.55	32	Recurrence: Laminarity [LAM]	16.61	0.32
8	Size: Elliptical eccentricity	26.79	0.49	33	Intensity: Quantile 0.40	16.35	0.37
9	Haralick: Sum of squares [VAR]	24.40	0.49	34	Recurrence: Ratio [DET/RR]	16.06	0.30
10	COO: Convex hull points	24.27	0.37	35	Haralick: Image difference entropy [DEN]	14.80	0.30
11	COO: Centre to contour points sum	24.22	0.60	36	Haralick: Image information measure correlation [f12]	14.70	0.26
12	COO: Eccentricity	24.21	0.48	37	Intensity: Quantile 0.30	14.52	0.32
13	Intensity: Quantile 0.60	23.89	0.47	38	Size: Mass center X	14.44	0.30
14	COO: Elongation	23.84	0.42	39	Haralick: Image entropy [ENT]	14.23	0.30
15	Intensity: Quantile 0.70	23.29	0.48	40	Intensity: Abs. deviation	14.09	0.28
16	Intensity: SD	23.14	0.48	41	Intensity: Quantile 0.90	13.61	0.37
17	Recurrence: Total recurrence	22.33	0.50	42	Haralick: Image sum of entropy [SEN]	13.35	0.27
18	Shape: Radius minimum	22.07	0.41	43	Intensity: Quantile 0.20	12.91	0.27
19	Recurrence: Trapping Time [TT]	21.05	0.44	44	Intensity: Quantile 0.95	12.66	0.32
20	Intensity: Quantile 0.50	20.21	0.38	45	Intensity: Quantile 0.10	9.62	0.21
21	Recurrence: Determinism [DET]	19.64	0.42	46	Size: Elliptical major axis angle	9.58	0.16
22	COO: Contour line height	19.41	0.40	47	Intensity: Quantile 0.05	8.03	0.19
23	Recurrence: Recurrence rate [RR]	19.32	0.40	48	Intensity: Quantile 0.01	6.79	0.15
24	Haralick: Inverse difference moment [IDM]	18.73	0.35	49	Intensity: Quantile 0.99	6.27	0.24
25	Haralick: Image summed variance [SVA]	18.60	0.35				

3.5 Classification using Multivariate Statistics, Support Vector Machines and Convolutional Neural Networks

To classify the LOKI plankton images four different methods were applied ranging from multivariate statistics with a LDA and a random Forest over "traditional" machine learning with SVM to a modern approach using CNNs. The results of the methods are addressed in the following chapters. Every chapter also contains the feature selection result of the respective method. The rows of the confusion matrices (Figure 23, Figure 25, Figure 27, Figure 29, Figure 30) displayed in this chapter contain the actual class affiliation while the columns contain the predicted class. The two extra rows on the bottom display the percentage of the prediction success per column, while the two extra columns on the right display the absolute number of data elements in a class which were predicted correctly, respectively incorrectly. To calculate the accuracy of a classification the sum of the diagonal proportion of the confusion matrix is calculated. The *prop.table()* function shows the proportion of the data sorted into each class, the diagonal contains all entries that where sorted into the correct class. Therefore, their sum accounts for the overall classifying accuracy. The classification accuracy of all methods on a testing dataset with will be discussed subsequently in chapter 4.4.

In a first approach the random Forest classification was used on the dataset *Sognefjord 26 groups*. The number of images features used was 40, which will be explained in detail in the respective random Forest chapter 3.5.2. The resulting confusion matrix is displayed in Figure 23. A red box was drawn around the Copepoda classes where a higher misclassification of images can be observed. This was taken as a reason to further summarize these classes for a better classification. The classification result of this random Forest analysis was 76.56 %.

The datasets used for the following classification are the *Sognefjord augmented* dataset for the CNNs and the *Sognefjord 14 groups* dataset for all other methods. During image processing with the *LOKI-Complete-data-primer* script 182 images of the *Sognefjord 14 groups* dataset were marked as too large or as having a too long contour line around the objects on the image. Therefore, they were excluded from the dataset processed with the classifying methods. Appendix 1 and Appendix 2 give an overview over the available datasets and their number of image files.



True Class

Predicted Class

	Annelida - Polychaeta	Bubble	Chaetognatha - Chaetognatha anterior	Chaetognatha - Eukrohniidae - Eukrohnia hamata	Cnidaria - Rhopalonematidae - Aglantha digitale	Cnidaria - Siphonophorae	Cnidaria - undet Jelly	Copepoda - Aetideidae	Copepoda - Calanidae	Copepoda - Clausocalanidae	Copepoda - Euchaetidae - Paraeuchaeta	Copepoda - Metridinidae	Copepoda - Oithonidae	Copepoda - Oncaeidae	Copepoda - Scolecitrichidae	Ctenophora	Detritus	Egg	Faeces	Malacostraca - Amphipoda	Malacostraca - Euphausiidae	Malacostraca - Mysidae	Oligostraca - Ostracoda	Rhizaria - Acantharia	Rhizaria - Foraminifera	Rhizaria - Rhizaria spec		
Annelida - Polychaeta								1		5		2		2														10
Bubble		537							3	11		1		1			19	1						1			537	37
Chaetognatha - Chaetognatha anterior			67	3					42	4		10		1			6										67	66
Chaetognatha - Eukrohniidae - Eukrohnia hamata			8	30					2	1				1			2				1						30	15
Cnidaria - Rhopalonematidae - Aglantha digitale					4		1		3												1	1					4	6
Cnidaria - Siphonophorae						5			9			1					2										5	12
Cnidaria - undet Jelly							4		8																		4	8
Copepoda - Aetideidae								3	20	1		19										2	6				3	48
Copepoda - Calanidae		4	3					3	1016	17	1	229					9					10	8	1			1016	285
Copepoda - Clausocalanidae		7	2						6	1620		20		30			79						3	4			1620	151
Copepoda - Euchaetidae - Paraeuchaeta				1					29	5	4	16									1	1	1				4	54
Copepoda - Metridinidae		4	1						315	61	2	728			3		11					1	8				728	406
Copepoda - Oithonidae										15			2	1			17										2	33
Copepoda - Oncaeidae		3								77		2		226			52							1			226	135
Copepoda - Scolecitrichidae									4	4		14			7							1	4				7	27
Ctenophora									1	1						7						1					7	3
Detritus		3	3	2					8	43		4		6			1067	6	1				5	3			1067	84
Egg		4								11				3			74	24						1			24	93
Faeces				1						4				3			26	1	4								4	35
Malacostraca - Amphipoda									2			1																3
Malacostraca - Euphausiidae			1	2					7												2	1					2	11
Malacostraca - Mysidae									18			13			1							30	5				30	37
Oligostraca - Ostracoda									15	26		30		2			6						179				179	79
Rhizaria - Acantharia		1	1						2	30		1	1			1	42					1		79			79	80
Rhizaria - Foraminifera										1																		1
Rhizaria - Rhizaria spec										2				1			3							3				9

Figure 23: Confusion matrix of the random Forest classification of the Sognefjord 26 groups dataset in 26 classes. The true classes are displayed in the rows and the predicted classes in the columns. On the right two extra columns are displayed showing the absolute number of images that have been correctly or incorrectly classified. The sum per row is the number of images per class. The two extra rows on the bottom show the percentage of correct and incorrect predictions per class. The red box outlines the higher misclassification within all Copepoda classes.

3.5.1 Linear Discriminant Analysis Classification

The linear discriminant analysis (LDA) uses 43 out of the 49 image features to reach its highest classification accuracy. Figure 24 shows the process of excluding the features stepwise in the order of their increasing *Gini Index* - unimportant features first - for the LDA. The accuracy declines slowly with a decreasing number of features. When only about 10 features are left in the dataset the accuracy declines a lot faster.

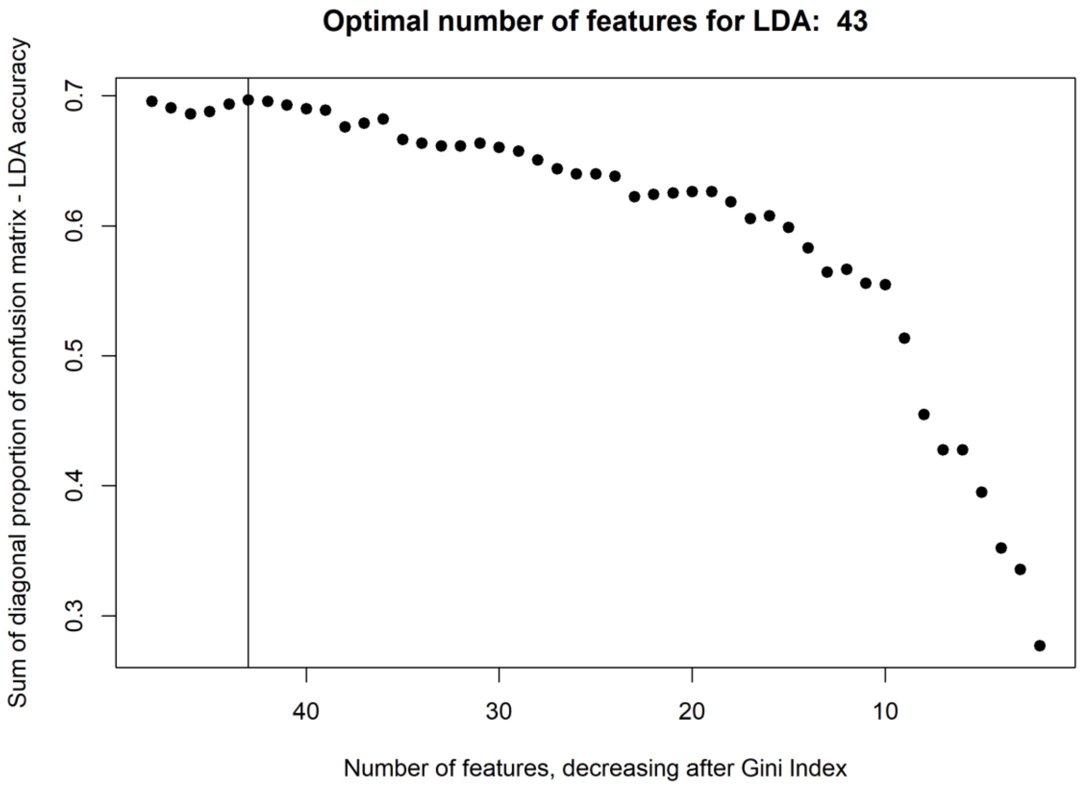
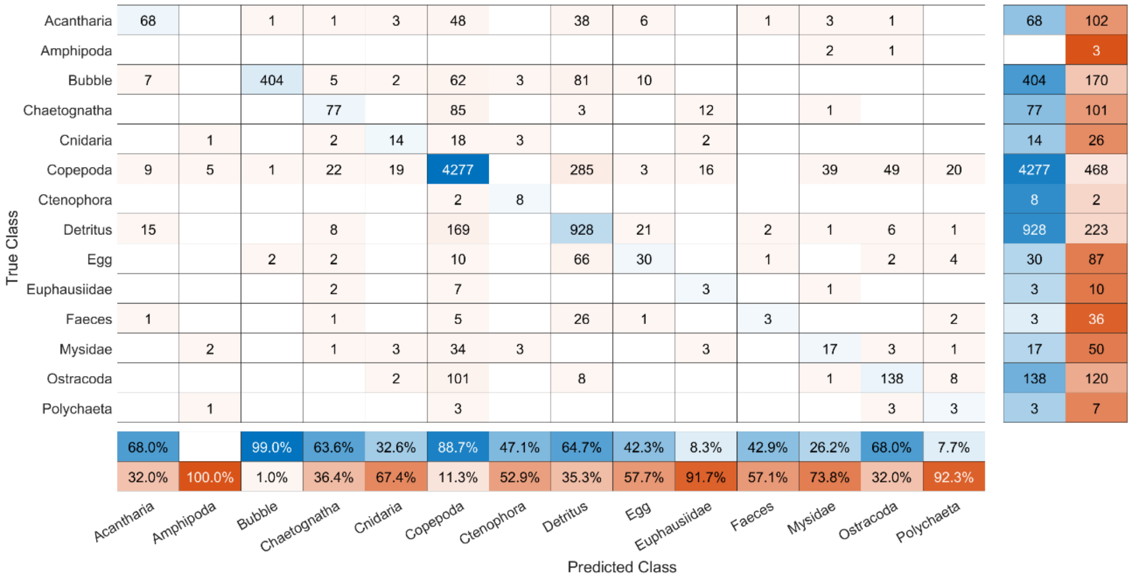


Figure 24: After the exclusion of highly correlating image features, a random Forest model was used to calculate the importance (Gini Index) of the remaining features. This graph shows the classification accuracy of a linear discriminant analysis (LDA) over a decreasing number of features ranked after the Gini Index. The LDA reaches its highest accuracy using 43 image features (vertical line).

The LDA classification was computed with the *lda()* function in the R package *MASS*. Two LDA models were created with the training dataset. For the first model the cross validation (CV) option within the *lda()* function was activated. If the CV is activated the model's return contains a list with the calculated components classes as a self-prediction. The classification accuracy of the model training data is 81.11 %. In the next step the LDA model without CV is used to predict the classes for the testing dataset. The classification accuracy of the prediction is 80.94 %.

The confusion matrix for the LDA classification is displayed in Figure 25. When it is described in this section the single classified elements will be addressed as images, it just needs to be noted, that all classifications, except the CNN approach, work with the image feature table for the classifications.

The true classes are displayed in the rows, while the predicted classes are displayed in the columns (Figure 25). The last two columns show the absolute true (blue) and false (red) classifications of each class. Their sum is the number of images that went into the testing dataset. On Figure 25 it is visible, that the dataset is strongly imbalanced. After using 80 % of the images for the training dataset the class Amphipoda only had three images left for the testing dataset. On the bottom, the two extra rows display the percentage of images that were predicted into the correct class within this column (blue) and that were predicted wrongfully (red). The Copepoda class contains most of the images and is also the class that has been predicted to belong to most of the other classes, while still having one of the highest accurate classification results. The highest accurate classification can be found within the bubble images and the lowest for the Amphipoda and Polychaeta images which have been classified to belong to the Copepoda class.



	Acantharia	Amphipoda	Bubble	Chaetognatha	Cnidaria	Copepoda	Ctenophora	Detritus	Egg	Euphausiidae	Faeces	Mysidae	Ostracoda	Polychaeta		
Acantharia	68		1	1	3	48		38	6		1	3	1		68	102
Amphipoda												2	1			3
Bubble	7		404	5	2	62	3	81	10						404	170
Chaetognatha				77		85		3		12		1			77	101
Cnidaria		1		2	14	18	3			2					14	26
Copepoda	9	5	1	22	19	4277		285	3	16		39	49	20	4277	468
Ctenophora						2	8								8	2
Detritus	15			8		169		928	21		2	1	6	1	928	223
Egg			2	2		10		66	30		1		2	4	30	87
Euphausiidae				2		7				3		1			3	10
Faeces	1			1		5		26	1		3			2	3	36
Mysidae		2		1	3	34	3			3		17	3	1	17	50
Ostracoda					2	101		8				1	138	8	138	120
Polychaeta		1				3							3	3	3	7
	68.0%		99.0%	63.6%	32.6%	88.7%	47.1%	64.7%	42.3%	8.3%	42.9%	26.2%	68.0%	7.7%		
	32.0%	100.0%	1.0%	36.4%	67.4%	11.3%	52.9%	35.3%	57.7%	91.7%	57.1%	73.8%	32.0%	92.3%		

Figure 25: Confusion matrix of linear discriminant analysis of the Sognefjord 14 groups dataset in 14 classes. The true classes are displayed in the rows and the predicted classes in the columns. On the right two extra columns are displayed showing the absolute number of images that have been correctly or incorrectly classified. The sum per row is the number of images per class. The two extra rows on the bottom show the percentage of correct and incorrect predictions per class.

3.5.2 Random Forest Classification

To classify LOKI plankton images with a random Forest the optimal number of features for the image feature subset is 40. This can be derived from Figure 26, identically to the feature selection for the LDA displayed in Figure 24, a random Forest model was calculated for every subset of features for a decreasing number of features. The features were ranked after their *Gini Index*. The accuracy of the random Forest models is quite stable in between 49 to about 15 features. After containing less than 15 features the accuracy declines rapidly. Since the random Forest are based on randomized computations, the exact number of features to deliver to highest classification accuracy might vary slightly if this test would be repeated multiple times.

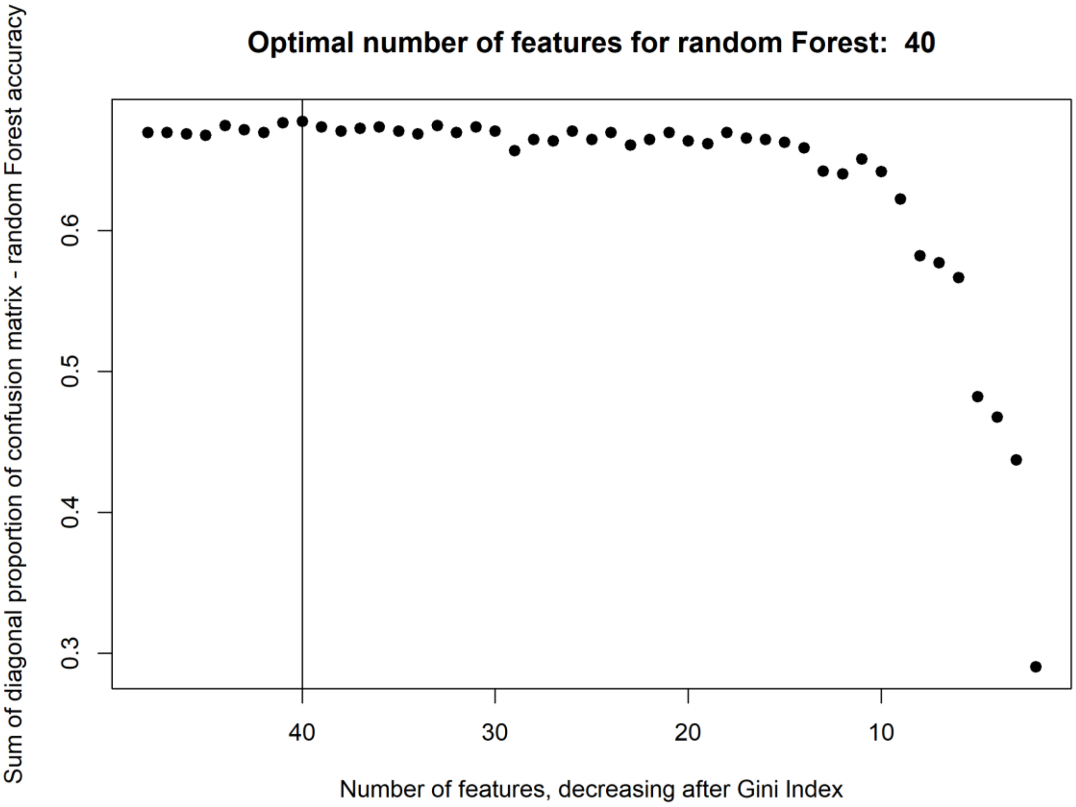
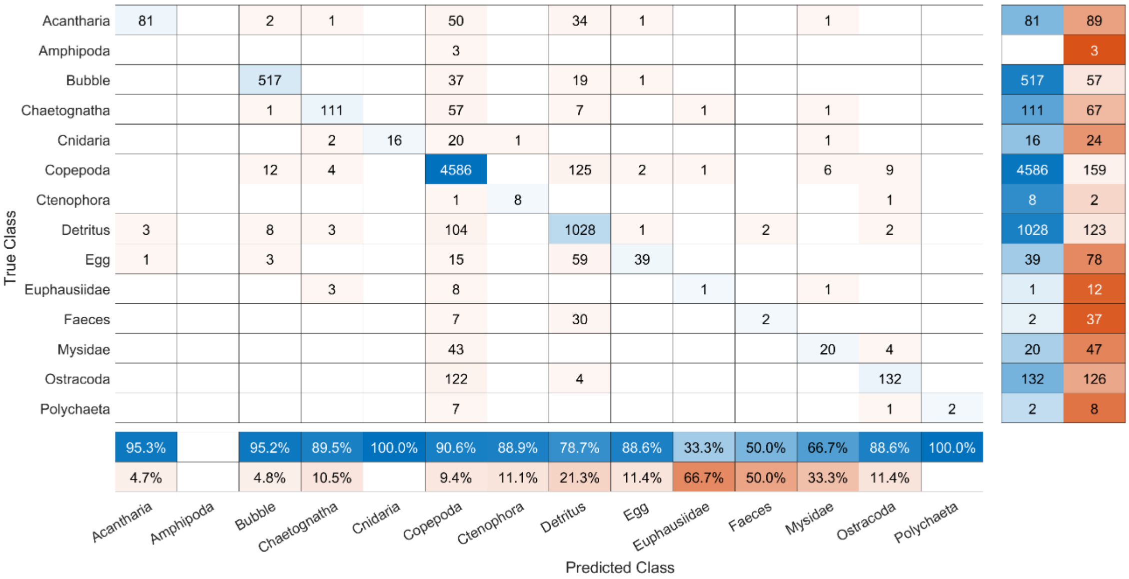


Figure 26: After the exclusion of highly correlating image features a random Forest model was used to calculate the importance (Gini Index) of the remaining features. This graph shows the classification accuracy of a random Forest model over a decreasing number of features ranked after the Gini Index. The random Forest reaches its highest accuracy using 40 image features (vertical line).

To calculate the classifiers accuracy on a training and testing dataset a 1,000 tree random Forest model was built with the *randomForest()* function in the equally named R package. One of the return values of a random Forest model is the confusion matrix. If the model is trained on the entire *Sognefjord 14 groups* dataset, the computation time takes 6.87 min and has a classification accuracy of 88.91 %. When a random Forest model is being trained on a training dataset, containing 80 % of randomly picked image data from each class and then tested on the remaining 20 % the computation time takes up 4.85 min and has a classification accuracy of 88.72 %.

Figure 27 displays the random Forest classifications confusion matrix, which is equivalent in its structure to the LDA confusion matrix. The random Forest classification has an overall higher classification accuracy, which is mirrored in the confusion matrix. Similar like in the LDA confusion matrix, it can be seen, that classes with a smaller number of images generally classify worse than classes with a high number of training and testing images.



	Acantharia	Amphipoda	Bubble	Chaetognatha	Cnidaria	Copepoda	Ctenophora	Detritus	Egg	Euphausiidae	Faeces	Mysidae	Ostracoda	Polychaeta		
Acantharia	81		2	1		50		34	1			1			81	89
Amphipoda						3										3
Bubble			517			37		19	1						517	57
Chaetognatha			1	111		57		7		1		1			111	67
Cnidaria				2	16	20	1					1			16	24
Copepoda			12	4		4586		125	2	1		6	9		4586	159
Ctenophora						1	8						1		8	2
Detritus	3		8	3		104		1028	1		2		2		1028	123
Egg	1		3			15		59	39						39	78
Euphausiidae				3		8				1		1			1	12
Faeces						7		30			2				2	37
Mysidae						43						20	4		20	47
Ostracoda						122		4					132		132	126
Polychaeta						7							1	2	2	8
	95.3%		95.2%	89.5%	100.0%	90.6%	88.9%	78.7%	88.6%	33.3%	50.0%	66.7%	88.6%	100.0%		
	4.7%		4.8%	10.5%		9.4%	11.1%	21.3%	11.4%	66.7%	50.0%	33.3%	11.4%			

Figure 27: Confusion matrix of the random Forest classification of the Sognefjord 14 groups dataset in 14 classes. The true classes are displayed in the rows and the predicted classes in the columns. On the right two extra columns are displayed showing the absolute number of images that have been correctly or incorrectly classified. The sum per row is the number of images per class. The two extra rows on the bottom show the percentage of correct and incorrect predictions per class.

3.5.3 Support Vector Machine Classification

Same like for LDA and random Forest the feature selection for SVM classification was conducted using the decreasing number of *Gini Index* ranked image features and a classifier loop testing its performance for every feature subset. Within the first feature selection tests it was noted that the SVM accuracy over a deceasing number of features had a clear maximum when it was used with constant *gamma* and *cost* variables for each iteration. Therefore, the final SVM feature selection loop also included a SVM tuning for each iteration. This loop was computational expensive and took over 10 h to finish. But by including the tuning, the variables can be set optimal for each new subset of features. The feature selection graph (Figure 28) shows an overall stable accuracy, which is slightly meandering until the number of features drops under 15. Next to the accuracy maximum at 26 features, a SVM with 16 features also show a similar high classification accuracy.

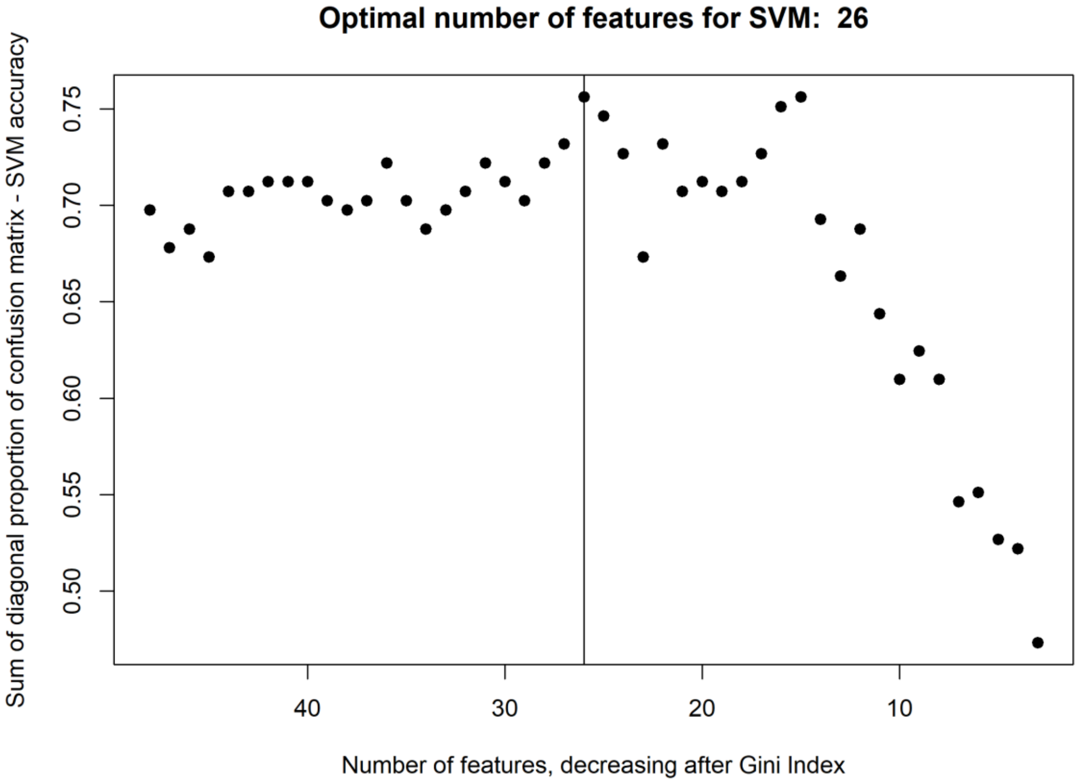
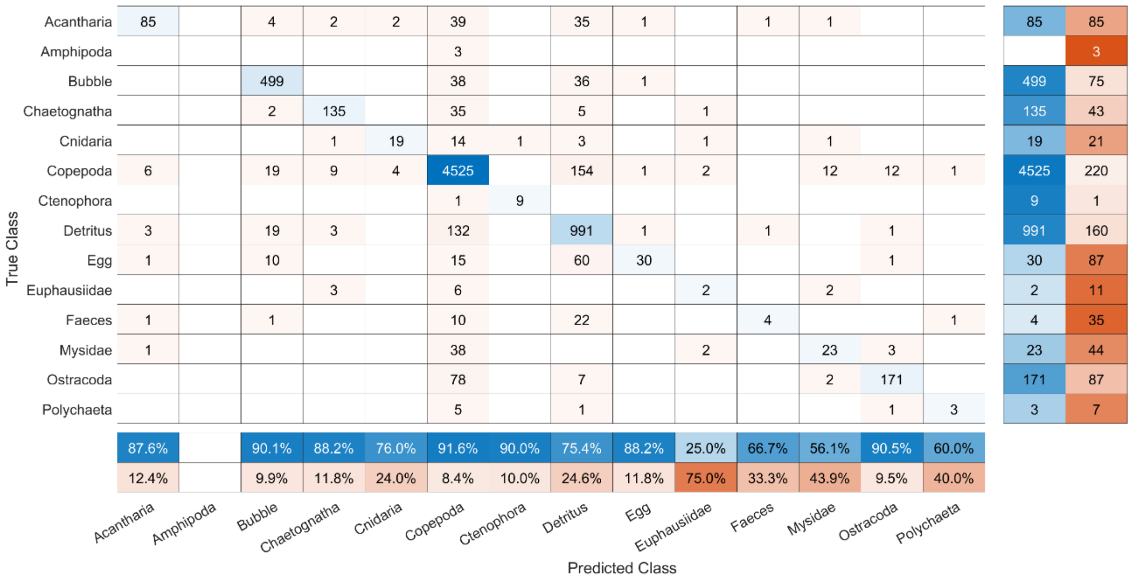


Figure 28: After the exclusion of highly correlating image features a random Forest model was used to calculate the importance (Gini Index) of the remaining features. This graph shows the classification accuracy of a support vector machine (SVM) over a decreasing number of features ranked after the Gini Index. The SVM reaches its maximum accuracy using 26 image features (vertical line).

The R package *e1071* is used to create a SVM model with a training dataset containing 80 % of the *Sognefjord 14 groups* dataset. The SVM model can be improved by using the *tune()* function. The tuned model is then used to predict the classes for a testing dataset containing 20 % of the *Sognefjord 14 groups* dataset. The tuning variables chosen by the *tune()* function are 0.05 for gamma and 30 for the cost variable. The SVM classification accuracy is 92.22 % for the training dataset and 88.08 % for the testing dataset. The elapsed time for tuning and training was 10.6 h and it took 18 s for the prediction of the testing dataset.

Figure 29 displays the confusion matrix for the SVM. The structure is again equivalent to the confusion matrices seen before. When the three confusion matrices of the “traditional” methods are viewed side by side, the higher classification accuracy of the random Forest and SVM methods are visible. In between these two, random Forest has less accurate classifications for 9 out of 14 classes but higher classification accuracies in the larger classes.



	Acantharia	Amphipoda	Bubble	Chaetognatha	Cnidaria	Copepoda	Ctenophora	Detritus	Egg	Euphausiidae	Faeces	Mysidae	Ostracoda	Polychaeta		
Acantharia	85		4	2	2	39		35	1		1	1			85	85
Amphipoda						3										3
Bubble			499			38		36	1						499	75
Chaetognatha			2	135		35		5		1					135	43
Cnidaria				1	19	14	1	3		1		1			19	21
Copepoda	6		19	9	4	4525		154	1	2		12	12	1	4525	220
Ctenophora						1	9								9	1
Detritus	3		19	3		132		991	1		1		1		991	160
Egg	1		10			15		60	30				1		30	87
Euphausiidae				3		6				2		2			2	11
Faeces	1		1			10		22			4			1	4	35
Mysidae	1					38				2		23	3		23	44
Ostracoda						78		7				2	171		171	87
Polychaeta						5		1					1	3	3	7
	87.6%		90.1%	88.2%	76.0%	91.6%	90.0%	75.4%	88.2%	25.0%	66.7%	56.1%	90.5%	60.0%		
	12.4%		9.9%	11.8%	24.0%	8.4%	10.0%	24.6%	11.8%	75.0%	33.3%	43.9%	9.5%	40.0%		

Figure 29: Confusion matrix of the support vector machines on the Sognefjord 14 groups dataset in 14 classes. The true classes are displayed in the rows and the predicted classes in the columns. On the right two extra columns are displayed showing the absolute number of images that have been correctly or incorrectly classified. The sum per row is the number of images per class. The two extra rows on the bottom show the percentage of correct and incorrect predictions per class.

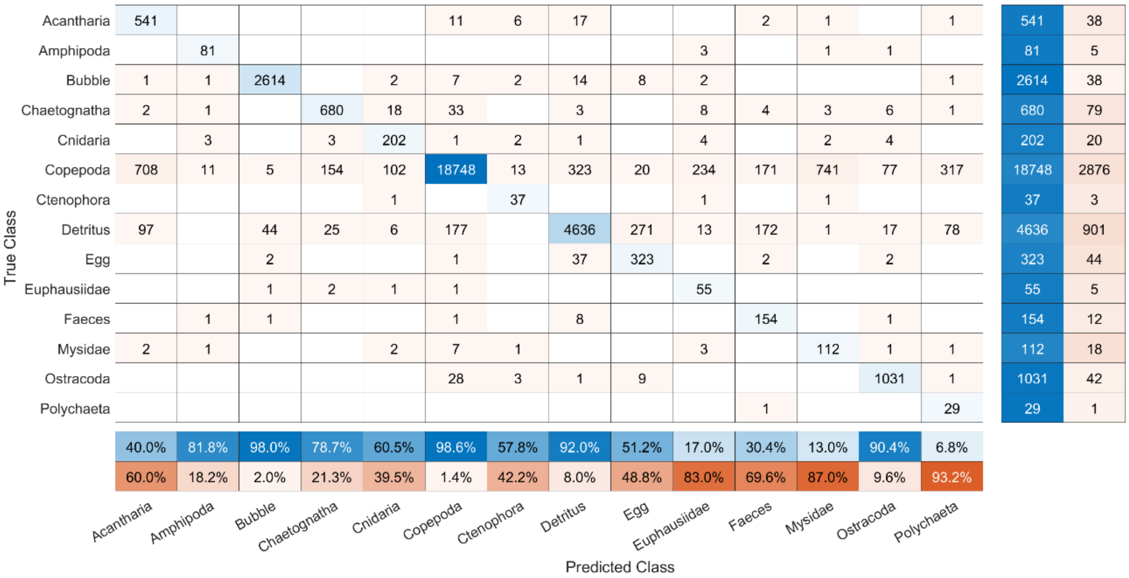
3.5.4 Convolutional Neural Network Classification

The MATLAB *AlexNet* transfer learning implementation was used to train an adaptation of the *AlexNet* neural network for LOKI plankton image classification. Since the dataset *Sognefjord 14 groups* is strongly imbalanced, the images in the smaller classes were augmented to create a larger basis for the training and testing dataset. The smallest number of images in one class after augmentation was 250. 190 random images per class were used for the training dataset, 30 for the validation dataset and all images left were used for the testing dataset. For the larger classes, this leads to having a testing dataset that is a lot larger than the training dataset, which is not the optimal setting. *AlexNet* was adapted to the number of classes available and then trained with the plankton images. The adapted network was named *LOKInet* and will be referred by this name. Within the experimental range (Table 5) the best learning rate for the *LOKInet* was 0.001 and the best momentum was 0.9. Each training and testing of the *LOKInet* takes about 3,5 h on one of the computers CPUs. Table 5 shows that the classification accuracy does not necessarily improve with a higher validation frequency (comparing trial 1, 11 & 13). When the momentum is smaller the learning rate needs to be higher to get a better accuracy (comparing trial 3 through 6). When keeping the learning rate constant but rising the momentum, the accuracy shows that the momentum has its optimal value at 0.9 (comparing trail 10 through 12). Trails 7, 8 and 11 indicate an optimal value for the learning rate of 0.001. With these values *LOKInet* achieves a classification accuracy of 87.75 %.

Table 5: Variation of learning rate, momentum, and validation frequency to improve the classification accuracy of the LOKInet approach.

No. of trail	Learning rate	Momentum	Validation frequency (every nth iteration)	Classification accuracy / %
1	0.0010	0.90	3	84.00
2	0.0050	0.95	3	77.89
3	0.0010	0.60	3	84.20
4	0.0005	0.60	3	80.31
5	0.0010	0.60	2	83.26
6	0.0100	0.60	2	83.44
7	0.0100	0.90	2	78.61
8	0.0005	0.90	2	81.94
9	0.0005	0.95	2	83.66
10	0.0010	0.80	2	82.35
11	**0.0010**	**0.90**	**2**	**87.75**
12	0.0010	0.95	2	77.47
13	0.0010	0.90	1	79.38

Figure 30 displays the *LOKInet* confusion matrix. It differs to the previously seen confusion matrices by having a much larger testing dataset for the classes since the images were augmented for this dataset. The highest misclassification appears within the Copepoda class, for images being mistaken as Mysidae or Acantharia. Since the Copepoda class has so many images, the majority is still classified correctly. Classes that have been filled up with augmented images are for example Amphipoda or Polychaeta. They have been classified correctly almost entirely by *LOKInet*.



	Acantharia	Amphipoda	Bubble	Chaetognatha	Cnidaria	Copepoda	Ctenophora	Detritus	Egg	Euphausiidae	Faeces	Mysidae	Ostracoda	Polychaeta		
Acantharia	541					11	6	17			2	1		1	541	38
Amphipoda		81								3		1	1		81	5
Bubble	1	1	2614		2	7	2	14	8	2				1	2614	38
Chaetognatha	2	1		680	18	33		3		8	4	3	6	1	680	79
Cnidaria		3		3	202	1	2	1		4		2	4		202	20
Copepoda	708	11	5	154	102	18748	13	323	20	234	171	741	77	317	18748	2876
Ctenophora					1		37			1		1			37	3
Detritus	97		44	25	6	177		4636	271	13	172	1	17	78	4636	901
Egg			2			1		37	323		2		2		323	44
Euphausiidae			1	2	1	1				55					55	5
Faeces		1	1			1		8			154		1		154	12
Mysidae	2	1			2	7	1			3		112	1	1	112	18
Ostracoda						28	3	1	9				1031	1	1031	42
Polychaeta											1			29	29	1
	40.0%	81.8%	98.0%	78.7%	60.5%	98.6%	57.8%	92.0%	51.2%	17.0%	30.4%	13.0%	90.4%	6.8%		
	60.0%	18.2%	2.0%	21.3%	39.5%	1.4%	42.2%	8.0%	48.8%	83.0%	69.6%	87.0%	9.6%	93.2%		

Figure 30: Confusion matrix of the convolutional neural net classification of the Sognefjord 14 groups dataset in 14 classes. The true classes are displayed in the rows and the predicted classes in the columns. On the right two extra columns are displayed showing the absolute number of images that have been correctly or incorrectly classified. The sum per row is the number of images per class. The two extra rows on the bottom show the percentage of correct and incorrect predictions per class.

4 Discussion

4.1 Objective 1: Evaluation of the adapted edge detection operation

Improving the edge detection was the first objective of this thesis. The “traditional” image classification is based on the manual image features, which can be derived from the image mask or contour line, which again is extracted from detected edges. As the edge detection is improved the derived image features and classification should be more accurate. Within this thesis four major changes were implemented into the original *LOKI-Complete-data-primer* script. Two of them aimed at receiving a more accurate image mask. Change A helped in the detection of the actual region of interest (ROI) in images with multiple organisms or objects. Despite being successful for the reviewed set of images, the object selection, based heavily on the mere expansion of the object within the image, has some remaining flaws. Organisms with a slim appearance might be discarded when being compared to imaging artefacts within the image having a larger horizontal expansion. Usually, the object of interest is the most illuminated and centered object within the image. In the Conclusion & Recommendation chapter 5 a suggestion is made for how to improve the selection of the object of interest in images with multiple detected objects in further studies.

Change B includes established edge detection methods and combines them in a new way to enhance the ROI and deliver tightly cut edges that include even fine-scale body-features and appendages of the organisms or object.

Viewing the result images of the original script, the edge detection was working, but it still had some room for improvement. Therefore, the original approach using a *Sobel* operator was not banned from the process. Within the pool of simple edge detectors, the extended (5x5) *Sobel* operator works fairly well (Kekre & Gharge, 2010). It was also used by Corgnati et al. (2016) on their imaging system for monitoring gelatinous zooplankton. The Canny edge detection is a well-established method (Bature et al., 2015; Oskoei & Hu, 2010) and is successfully used for different fields of visual research, e.g. mammography (Rampun et al., 2017). The Canny edge detection implementation by Liang (2017) worked well in R and produced reasonable results.

The simple process of adding two different multiplication results of the vertical and horizontal *Sobel* edge detection delivered satisfying image masks. This was evaluated by classifying the images by their feature table with a LDA *before* Change B was applied to the image processing chain. The result was then compared to the classification results from the same feature table *after* Change B was applied. The classification accuracy was similar for both cases with a slightly higher accuracy of the image feature table produced *before* Change B was implemented. This result slightly contradicts the expected outcome of the *improved* edge detection. But it also shows that important discrimination features are not necessarily derived from a more accurate contour line. As Change B also still has objectives for improvement, the development of this classification accuracy should be observed in further experiments.

Other developers of plankton imaging systems report difficulties applying the Canny edge detector. Sosik & Olson (2007) indicate that the Canny algorithm could not be optimized to avoid artifacts from noise and illumination variations while reliably detecting challenging cell features. Their images display the organism with higher pixel intensities on a light gray background which varies in brightness. LOKI's advantage for this issue is, that it provides images with a high signal to noise ratio and high contrast to the background.

It needs to be noted that the Canny edge detection requires a lot of computational time. Akiba & Kakui (1998) even state that edge detection in general is a computational expensive operation. Nevertheless, it is necessary to manually produce image features for "traditional" classification methods.

In their publication Dai et al. (2017) used different approaches to extract information from their plankton images. Their overall approach using a combination of calculating texture features and a CNN classification is discussed in detail in chapter 4.4. One of their image processing steps leading to the feature extraction includes operations similar to the ones in this thesis as they also use the *Sobel* operator and Canny edge detection. Their overall classification accuracy was improved by adding information of global and local feature extraction - which were produced by prior edge detection - to their model. This shows that even if there are more modern approaches to image classification, the manual feature extraction and a previous well working edge detection are still crucial to producing highly accurate results.

Change C was successful in discarding errors from too large or corrupt images and will further be employed in the script. Change D helped to save computational time but can further be improve by measures explained in the Recommendations 5.

4.2 Objective 2: Evaluation of the implemented feature selection.

Each plankton image contains information about the size, shape, roundness, etc. of the captured organism. To extract these information, the image is processed with the *LOKI-Complete-data-*

primer script, detecting the edges of the organism, and producing a final image mask outlining the exact position of the organism within the image. From the image mask a variety of features describing the organism can be obtained. Cheng et al. (2018) review feature extraction techniques for plankton images and their publication includes a few features that have not yet been calculated for the LOKI images. For example, the *Fourier Boundary Descriptor* is a common geometric feature used by Sosik & Olson (2007), Tang et al. (1998) and Verikas et al. (2015). Wang et al. (2016) propose a method where local binary patterns, inner-distance shape context and geometric and grayscale features are extracted separately and used in different classifiers before the results are merged in a final classification. They state that they achieve better performance by combining these different types of features processed in separate classifiers. The *LOKI-Complete-data-primer* script also uses different types of features but separated classification was not yet attempted within this thesis. The different types of features calculated with the *LOKI-Complete-data-primer* deliver similar information about certain image characteristics. If processed separately, the information could be kept and would not be sorted out during feature selection. This should be a point considered for further research with the LOKI plankton images.

The *LOKI-Complete-data-primer* script produces an output with 75 manual image features from which some include similar information. Classification methods work best with a set of features without redundant information and collinearity (Dash & Liu, 1997). The feature selection process implemented in this thesis investigates different phases to find the best subset of features for each method. Highly (>= 0.98) correlating features are sorted out manually by using a *pairs* plot and the *cor()* function. The *Gini Index* is computed for the remaining features. Using the corresponding classification method, its classification accuracy is computed for each feature subset in a loop with step-wise excluded features, based on their *Gini Index* significance. The usage of the *Gini Index* was validated with the conclusion from Strobl et al. (2007), who imply, that if a dataset has continuous values for its features and if the features do not account for different numbers of classes - as it is common in gene expression studies - the feature selection with random Forest importance measures is not affected by their findings. Their findings imply that the *Gini Index* method would be very biased and not reliable if applied to datasets that do not meet the mentioned criteria. The Sognefjord dataset meets the criteria as the features have continuous values and account for the same number of plankton classes for each image.

The process implemented within this thesis can be assigned to the wrapper feature selection methods. Feature selection methods can generally be subdivided into filter and wrapper methods (Dash & Liu, 1997). While filter methods are not dependent on a classification algorithm, wrapper methods use such an algorithm for the evaluation of the selected features. This makes wrapper methods more accurate but also very time intensive.

Within the publications for plankton classification only a few make it transparent which method was used for their feature selection. Verikas et al. (2015) also conducted a separate feature selection for their different classifiers, being a random Forest and a SVM. They have used a classification-accuracy-based floating search for both methods where the classification accuracy of the method is determined within a loop similar to the used method within this thesis. The difference is that their method starts with a best pair of features, then decides which one presents the most decrease in classification accuracy when excluded, discard this one, and adds two more features to repeat this process. This process leaves more room for variation within the feature selection than the applied method. For future implementations, method like classification-accuracy-based floating search can be accessed within R, over the packages *FSinR* and *caret*. They could be included in further experimentation with the LOKI image features.

4.3 Objective 3: Evaluation of the reduced feature table regarding the discriminability of labeled morphotype groups, based on numerical parameters using LDA, random Forest and SVM.

Since early versions of the LOKI system and software the authors are aiming at providing a complete processing chain from image capturing towards autonomous classification and presorting of major morphological groups. First adaptations of SVMs have been implemented on morphological features in 2009 (Barz et al., 2009). Latest approaches further combined time series analyses methods, like recurrence features (Schulz et al., 2016) with a LDA approach to classify morphological groups achieving a classification accuracy of 62.8 %.

Within this thesis further classification possibilities are applied to the reduced LOKI image features table. A LDA , a random Forest analysis and a SVM were conducted on a dataset with 37,057 labeled LOKI images. Table 6 summarizes the classification results of the different multivariate and machine learning methods. For the *Sognefjord 14 groups* dataset the LDA was the fastest but least accurate classifier with 80.94 % accuracy. The SVM took a long time for tuning the function parameters but was then trained a little faster than the random Forest. Both methods have similar high classification results of 88.08 % (SVM) and 88.7 % (random Forest). The random Forest reached 76.56 % classification accuracy on the *Sognefjord 26 groups* dataset. Each method had a distinct set of features, chosen to achieve the best possible classification accuracy.

Table 6: Summary of classification accuracies of different classifying methods. The dataset name, number of classes, number of features, training and testing time and the accuracy are listed.

Classification method	random Forest	LDA	random Forest	SVM	CNN
Dataset	*Sognefjord 26 groups*	*Sognefjord 14 groups*	*Sognefjord 14 groups*	*Sognefjord 14 group*	*Sognefjord augmented*
Classes	26	14	14	14	14
Features	40	43	40	26	-
Training time / min	4.39	0.05	4.97	Tuning: 631.63 Training: 4.65	192.00
Prediction time / min	0.04	0.01	0.10	0.29	13.33
Classification accuracy of a testing set / %	76.56	80.94	88.72	88.08	87.75

The intragroup specific misclassification of Copepoda organisms (Figure 23) show that the basic morphotype recognition of Copepoda-Like organisms has worked well and has led to the reduction of the dataset by summarizing the Copepoda organisms to achieve higher classification accuracy. Further and more detailed approaches should try to optimize the classification accuracy of the *Sognefjord 26 groups* dataset to reach lower taxonomic levels that can be discriminated by image feature extraction and classification.

For the *Sognefjord 14 groups* dataset the highest classification accuracy of the random Forest can be compared to the results of similar approaches in research on plankton image recognition from the past. Verikas et al. (2015) list the accuracy of some different techniques for automated plankton classification of phytoplankton species from 1989 until 2007 which lie between the values of 70 % and 89 %. Bi et al. (2015) classify their images for *Jelly-Like*, *Arrow-Like* and *Copepod-Like* morphotypes which can also be found within the *Sognefjord 14 groups* dataset. They can also account for classification accuracies of > 80 %. It needs to be noted that most of the different plankton classification approaches are working on different sets of images captured by differently working devices. While LOKI or ZooScan (Grosjean et al., 2004) images have a very high signal to noise ratio, since the background noise is prevented by having a narrow flow-through chamber or a plankton scanner, other devices like the GUARD1 (Corgnati et al., 2016) take images in the open water column and also account for classification accuracies of ~ 85 %.

LOKI images have a very high resolution and therefore a high potential to achieve even higher classification rates.

Gorsky et al. (2010) compare six different classification algorithms including random Forest and SVM for the question which one performs best on zooplankton classification. Their results also find the highest classification accuracy with the random Forest application. The SVM classification performs almost equally well and is a classifying tool used by several researchers of the plankton image classification community (Bi et al., 2015; Corgnati et al., 2016; Hu & Davis, 2005; Sosik & Olson, 2007; Wang et al., 2016; Zheng et al., 2017) and beyond (Guyon et al., 2002). In R tuning a SVM is a timely procedure but delivers better overall classification.

All confusion matrices (Figure 23, Figure 25, Figure 27, Figure 29 and Figure 30) show the highest misclassification among the Copepoda class, mostly for other classes being falsely classified as Copepoda. For LDA, random Forest and SVM the dataset was split 80/20 % for training and testing. Since the Copepoda class contains a much larger number of images compared to the other classes, the classification rates are likely biased while eventually disregarding the information the smaller classes provide. It is believed that the classification accuracy for equally large datasets would be higher. Many images were also classified as Detritus which is also a very large class compared to others. There is an approach for addressing imbalanced datasets by Lee et al. (2016) for CNN based classification where they constructed class-normalized data by data thresholding for large-sized classes. Another approach which also applies for the “traditional” methods is data augmentation. In order to produce larger training and testing datasets the original images are flipped, rotated, shifted, distorted, etc.

The R package *OpenImageR* provides an *Augmentation()* function to apply such changes to the images. The issue that came up while trying to augment the LOKI plankton image data was, that the newly created background by rotating or shifting an image creates an intensity gradient to the original image background which is then detected by the *LOKI-Complete-data-primer* script. Therefore, different methods need to be applied in order to augment the dataset for classifications on the feature tables. Of course, manual labeling and sorting of new datasets is also a possibility. Especially after the classification with random Forests has reached a fairly high accuracy which can help to classify new datasets and create larger training datasets for future classification approaches.

Figure 31 shows example organisms of all 14 classes used for the classification. For the human eye most of them probably seem easy to differentiate. Except for maybe image k (Mysidae) and l (Amphipoda) which to the untrained eye appear fairly similar. The few Amphipoda images have been classified as Mysidae - and the other way around - only with the LDA classification.

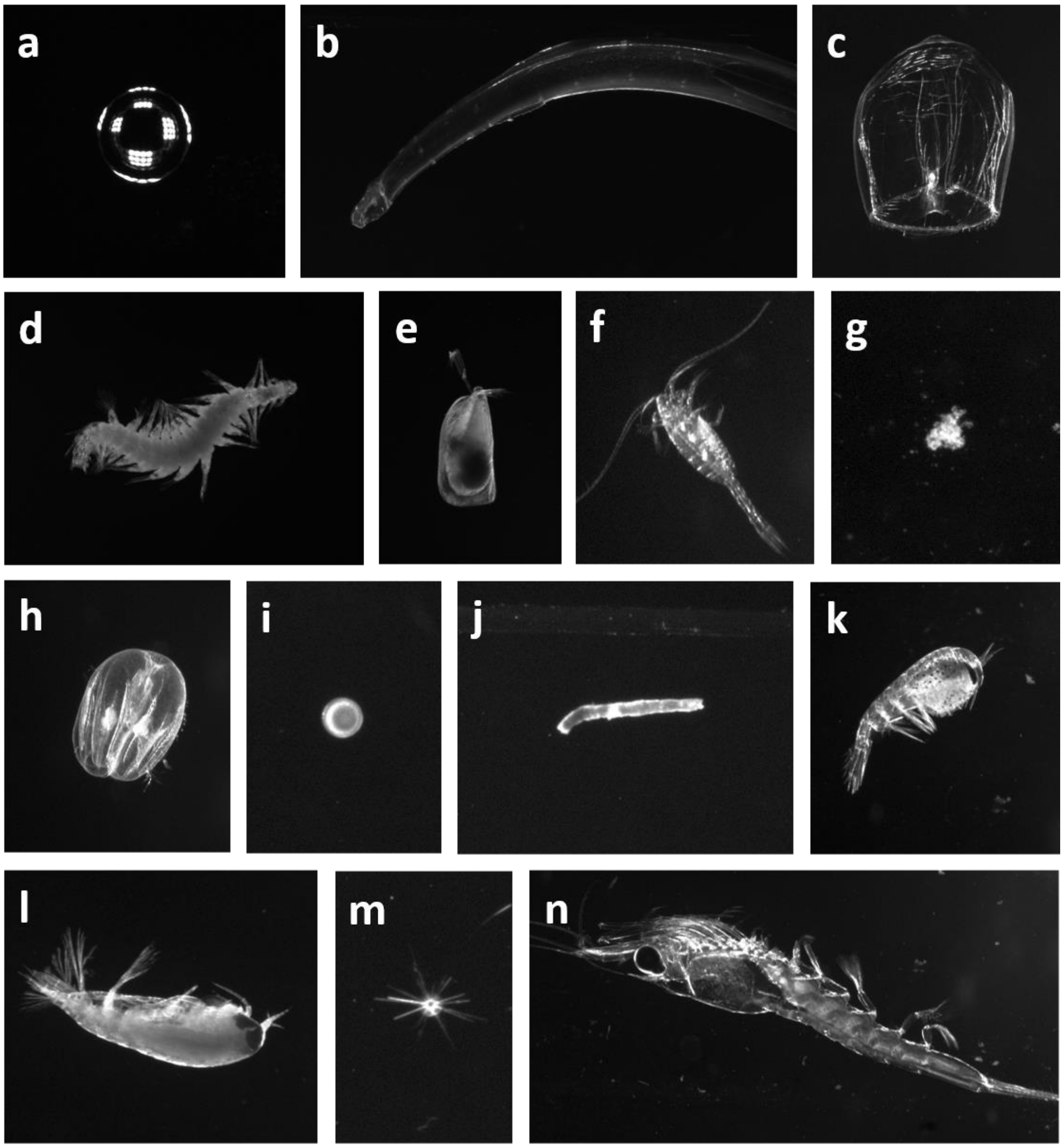


Figure 31: Example organisms of all 14 classes from the Sognefjord 14 groups or Sognefjord augmented dataset captured with the Lightframe On-Sight Keyspecies Investigation (LOKI) system. a – Bubble, b – Chaetognatha, c – Cnidaria, d – Polychaeta, e – Ostracoda, f – Copepoda, g – Detritus, h – Ctenophora, i – Egg, j – Faeces, k – Mysidae, l – Amphipoda, m – Acantharia, n – Euphausiidae

4.4 Objective 4: Approach of direct image discrimination with convolutional neural networks.

The last objective of this thesis was the approach to classify LOKI plankton images with a CNN. The multivariate and machine learning methods applied in this thesis achieved a good image feature classification of up to 88.72 %. Nevertheless, Lumini et al. (2019) state that deeper learning approaches are increasingly replacing the manual feature extraction and classification by “traditional” methods. CNNs have become an important tool for visual and speech recognition and natural language processing over the past years (Gu et al., 2018). There are many variants of CNN architectures, but their basic components are similar, consisting of convolutional, pooling and fully connected layers (Gu et al., 2018). To successively train a CNN for image recognition, large amounts of training data need to be available and even then, it might take weeks, even on high-performing systems, to adjust all weights for the CNN layers (Aggarwal, 2018). There have been multiple constructions of shallow and deep CNNs for image recognition. An online contest for image recognition (*ImageNet*, 2016) has been running over the past years challenging researchers to constantly build better CNNs classifying the large ImageNet database. For example, *AlexNet* was built by Krizhevsky et al. (2012), winning the ImageNet challenge of 2010.

Instead of constructing a whole new CNN for the LOKI image recognition, the architecture, and trained weights of *AlexNet* can be used to classify LOKI images. This process is called transfer learning (Pan & Yang, 2009; Shao et al., 2015). Transfer learning is particularly useful when the existent resources, like labeled dataset size or computational power, are not meeting the requirements to build an individual CNN. MATLAB offers a toolbox for deep and transfer learning applications with *AlexNet* (Appendix 6).

The *Sognefjord augmented* dataset was used to fine-tune the *AlexNet* network. The architecture was slightly adapted to fit the number of classes in the *Sognefjord augmented* dataset. The adapted network was trained on the plankton images and called *LOKInet*. *LOKInet* was then used to classify the remaining testing dataset and achieved a classification accuracy of 87.75 %.

AlexNet and other pre-trained CNNs were also used for multiple plankton classification publications (Dai et al., 2016; Dunker et al., 2018; Lumini & Nanni, 2019). Dai et al. (2016) use a dataset with 9460 images in 13 classes and achieve a classification accuracy of 93.6 %. Lumini & Nanni (2019) worked with three different datasets and 17 different pre-trained CNN architectures and achieved classification accuracies reaching from 80.4 % – 95.1 %. Dunker et al. (2018) use a hybrid method of CNNs, which will be explained in the next section, with 46,797 phytoplankton images in 9 classes in 3 three different life cycle stages and achieve a classification accuracy of 97 %.

Deep learning approaches for plankton classification have repeatedly been superior to multivariate and machine learning classification of manual features (González et al., 2019; Lumini et al., 2019; Moniruzzaman et al., 2017). Dunker et al. (2018) provide a summarizing table of the accuracies for phytoplankton classification. They also include a few references with SVM, decision trees and LDA classification. It is obvious, that CNN classification reaches much higher accuracies.

After comparing the classification results of the "traditional" and multivariate methods to the deep learning approaches of other publications, one might ask why even spend more time on the development of the manual features for "traditional" plankton classification. The reason may be seen in classification approaches like the one from Dai et al. (2017) who use a combination of CNNs and "traditional" manual feature extraction to classify plankton images and achieve a better classification rate using their hybrid method than only using CNNs. Lumini et al. (2019) use an ensemble of CNNs, which are each fine-tuned differently to classify multiple plankton and coral image datasets and show that they achieve higher classification accuracies with their ensemble than with single CNNs. Dunker et al. (2018) work exclusively with phytoplankton images and present a similar approach to Dai et al. (2017), by using multiple CNN classifications on separate but complementary image channels assessing characteristic information. Dunker et al. (2018) even go a step further and classify the life cycle stage of the respective phytoplankton organisms. Reviewing the Sognefjord datasets it is noted that the majority of classes are zooplankton organisms instead of phytoplankton. A lot of the recent plankton classification research has been conducted on phytoplankton images (Dunker et al., 2018; Sosik & Olson, 2007; Verikas et al., 2015). Morphologically, the major difference of phyto- and zooplankton lies within their cell structure (Tait & Dipper, 1998). While plants have solid cell walls and usually make for a more static appearance of the organism from different viewing angles, animal cells do not have cell walls, which leads to a higher variability of the organism's morphology. By taking this into consideration, it was expected that classifying methods might have a higher accuracy for phytoplankton images than for zooplankton images. This can be ruled out since Dai et al. (2016, 2017) have made advances in specifically training their *ZooplanktonNet* on zooplankton. Their hybrid approach was previously mentioned and achieved a high classification rate of up to 96.3 %. Since the LOKI dataset contains large groups of zooplankton Dai et al.'s (2017) approach could also be a future perspective for the LOKI image classification.

Another interesting software that has been developed for the classification of plankton images is *MorphoCluster* by Schröder et al. (2020). Their clustering method is based on assigning annotations to clusters of similar images. A user can then decide if those clusters are annotated correctly, which leads to a growing of the original cluster to enclose more similar images.

Deep features are the features calculated by a neural network. They can also be extracted from the network and used with "traditional" classifying methods. González et al. (2019) also employ CNNs to obtain deep features, rather than use the CNN as a classifier. They extract deep features from plankton images and feed them in their machine learning based quantification algorithms.

As previously mentioned in chapter 2.4.3 the TensorFlow approach was not pursued further for this thesis but it is noted as a potential approach for further studies.

5 Conclusion & Recommendations

Plankton image classification has come a long way in the past years and has advanced from time consuming manual classification to automated classification approaches with deep neural networks, and fine-scale information retrieval on planktonic organisms. On this road, image processing has improved for edge detection and feature extraction to feed multivariate and machine learning algorithms as a predecessor in classifying. This thesis has specifically worked with labeled LOKI plankton images from in *R/V Heincke* cruise HE434 in the Sognefjord, Norway on their path from edge detection through feature selection and classification. New approaches have been developed in all parts of the thesis helping to progress the entire LOKI software in the future.

The LOKI image edge detection algorithm has been progressed using the Canny edge detection for more narrow contour lines leading to more exact image masks. LOKI image features have been selected using the *Gini Index* and individual stepwise feature selection for the "traditional" classifiers LDA, random Forest and SVM. The classification accuracies of these methods has reach up to 88.72 %, which can be considered a successful but still improvable classification for the 37,057 images in 14 classes.

Self-implemented neutral networks need large amounts of labeled training data and computational power and time. Therefore, transfer learning is an accessible and reasonable tool for plankton image classification. Studies have shown that the use and combination of traditional image classification by manual image features can successfully be combined with newer approaches of deep learning to further improve the classification accuracy. Pre-trained networks can be adapted and fine-tuned in order to serve as plankton image classifiers.

Therefore, the development of the LOKI plankton classification system should further consider working with deep features, deep learning classifier and maybe even hybrid models.

For further work and development on the edge detection algorithm within the *LOKI-Complete-data-primer* script a few objectives will be suggested for additional improvements. For choosing the object of interest within an image with multiple organisms, particles, or light attributes a further manual experimentation is needed on whether the object is always the one in the center of the image. If so, there could be a higher importance set on the center pixel. This could be implemented in a form of an image partition, like a chessboard, and if the object of interest covers more of the center cells it is more likely to be chosen for the final image mask. Another idea could be to investigate the illumination and intensity of the cluster of pixel of multiple organisms since the organism in the center of the image tend to be illuminated better.

The computational time of the *LOKI-Complete-data-primer* script should also be object of further research and improvement as the parallelized approach has some vulnerabilities. Since a large (e.g. 113,551 x 92) result matrix is created by the parallel *foreach()* operation and the concatenation of new result rows takes up much computing time, an empty matrix of that size should be created prior to the image processing and feature extraction.

Within the most recent version of the *LOKI-Complete-data-primer* script there are some functions within the implementation of Change B that can be further improved, e.g., the deleting of weak edges within the Canny edge detection algorithm. As of now, this function is ignored within the script since it worsens the results. However, with the conducting of further experiments with the double thresholding and the retrieval of connected weak edges this function might help to further detail the edge detection.

For the feature extraction there are additional image features, used by other authors (as discussed in chapter 4.2), that could also be implemented into the script. It should also be examined if certain feature combinations describe the differences of certain classes better than others. Following the lead of Wang et al. (2016) or Dai et al. (2017), there is a large scope for experiments for feature extraction, selection and classification.

As data augmentation was also critical for LOKI plankton images due to newly created edges by rotation or shifting, an approach for this issue might be the following idea: An image primer similar to the *LOKI-Complete-data-primer* script could be applied, using the image mask as a cutout to create a new source images with all pixel intensities of the non-image mask set to 0. This would create a completely black background which would leave all kind of possibilities for image augmentation.

There is more room for experimentation and further improvement in all aspects of the LOKI plankton image classification, starting at some hardware traits up to fine tuning of various classification parameters. Edge detection as well as feature extraction, selection and classification should be reconsidered on the basis of the results of this thesis, which has revealed new approaches and further possibilities towards *state of the art* plankton recognition.

References

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning*. Springer International Publishing. https://doi.org/10.1201/b22400-15

Akiba, T., & Kakui, Y. (1998). Design and testing of an underwater microscopy for the study of zooplankton distribution. *UT 1998 - Proceedings of the 1998 International Symposium on Underwater Technology*, *25*(1), 17–20. https://doi.org/10.1109/UT.1998.670051

Badewien, T. (2014). *Cruise Report HE434*.

Banse, K. (1964). On the vertical distribution of Zooplankton in the sea. *Progress in Oceanography*, *2*(C). https://doi.org/10.1016/0079-6611(64)90003-5

Barz, K., Schulz, J., & Hirche, H.-J. (2009). LOKI, a new optical system for high-resolution plankton investigations. *ICES ASC I:17*.

Baschek, B., Schroeder, F., Brix, H., Riethmüller, R., Badewien, T. H., Breitbach, G., Brügge, B., Colijn, F., Doerffer, R., Eschenbach, C., Friedrich, J., Fischer, P., Garthe, S., Horstmann, J., Krasemann, H., Metfies, K., Merckelbach, L., Ohle, N., Petersen, W., … Ziemer, F. (2017). The Coastal Observing System for Northern and Arctic Seas (COSYNA). *Ocean Science*, *13*(3), 379–410. https://doi.org/10.5194/os-13-379-2017

Bature, U. I., Murtala, M. B., & Nasir, A. Y. (2015). Evaluation Of Image Detection Techniques. *Journal of Multidisciplinary Engineering Science and Technology (JMEST)*, *2*(12).

Beaugrand, G., Reid, P. C., Ibanez, F., Lindley, J. A., & Edwards, M. (2002). Reorganization of North Atlantic marine copepod biodiversity and climate. *Science*, *292*(1–3), 1692–1694. https://doi.org/10.1029/134GM10

Benfield, M. C., Grosjean, P., Culverhouse, P. F., Irigoien, X., Sieracki, M. E., Lopez-Urrutia, A., Dam, H. G., Hu, Q., Davis, C. S., Hanson, A., Pilskaln, C. H., Riseman, E. M., Schultz, H., Utgoff, P. E., & Gorsky, G. (2007). RAPID: Research on Automated Plankton Identification. *Oceanography*, *20*(SPL.ISS. 2), 172–218. https://doi.org/10.5670/oceanog.2007.63

Bennett, K. P., & Campbell, C. (2000). Support vector machines: hype or hallelujah? *SIGKDD Explor. Newsl.*, *2*(2), 1–13. https://doi.org/10.1145/380995.380999

Bi, H., Guo, Z., Benfield, M. C., Fan, C., Ford, M., Shahrestani, S., & Sieracki, J. M. (2015). A semi-automated image analysis procedure for in Situ plankton imaging systems. *PLoS ONE*, *10*(5), 1–17. https://doi.org/10.1371/journal.pone.0127121

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. http://arxiv.org/abs/2004.10934

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*, 5–32.

Burger, W., & Burge, M. J. (2016). Working with Discrete Signals. In *Digital Image Processing*. https://doi.org/10.1080/03043799408928319

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PAMI-8*(6), 679–698.

Cheng, K., Cheng, X., & Hao, Q. (2018). A review of feature extraction technologies for plankton images. *ACM International Conference Proceeding Series*, 48–56. https://doi.org/10.1145/3292425.3293462

Chollet, F. (2018). Deep Learning with Phyton. In *Manning*. http://faculty.neu.edu.cn/yury/AAI/Textbook/Deep Learning with Python.pdf

Christianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Corgnati, L., Marini, S., Mazzei, L., Ottaviani, E., Aliani, S., Conversi, A., & Griffa, A. (2016). Looking inside the ocean: Toward an autonomous imaging system for monitoring gelatinous zooplankton. *Sensors (Switzerland)*, *16*(12), 1–28. https://doi.org/10.3390/s16122124

Cowen, R. K., & Guigand, C. M. (2008). In situ ichthyoplankton imaging system (ISIIS): System design and preliminary results. *Limnology and Oceanography: Methods*, *6*(2), 126–132. https://doi.org/10.4319/lom.2008.6.126

Culverhouse, P. F., Williams, R., Benfield, M., Flood, P. R., Sell, A. F., Mazzocchi, M. G., Buttino, I., & Sieracki, M. (2006). Automatic image analysis of plankton: Future perspectives. *Marine Ecology Progress Series*, *312*(April), 297–309. https://doi.org/10.3354/meps312297

Dai, J., Wang, R., Zheng, H., Ji, G., & Qiao, X. (2016). ZooplanktoNet: Deep convolutional network for zooplankton classification. *OCEANS 2016 - Shanghai*. https://doi.org/10.1109/OCEANSAP.2016.7485680

Dai, J., Yu, Z., Zheng, H., Zheng, B., & Wang, N. (2017). A Hybrid Convolutional Neural Network for Plankton Classification. *Computer Vision – ACCV 2016 Workshops. ACCV 2016. Lecture Notes in Computer Science*, *10118*, 102–114. https://doi.org/https://doi.org/10.1007/978-3-319-54526-4_8

Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis*, *1*(3), 131–156. https://doi.org/10.3233/IDA-1997-1302

Davis, C. S., Hu, Q., Gallager, S. M., Tang, X., & Ashjian, C. J. (2004). Real-time observation of taxa-specific plankton distributions: An optical sampling method. *Marine Ecology Progress Series*, *284*, 77–96. https://doi.org/10.3354/meps284077

Davis, L. S. (1975). A survey of edge detection techniques. *Computer Graphics and Image Processing*, *4*(3), 248–270. https://doi.org/10.1016/0146-664x(75)90012-x

Dunker, S., Boho, D., Wäldchen, J., & Mäder, P. (2018). Combining high-throughput imaging flow cytometry and deep learning for efficient species and life-cycle stage identification of phytoplankton. *BMC Ecology*, *18*(1), 1–15. https://doi.org/10.1186/s12898-018-0209-5

Frederiksen, M., Edwards, M., Richardson, A. J., Halliday, N. C., & Wanless, S. (2006). From plankton to top predators: Bottom-up control of a marine food web across four trophic levels. *Journal of Animal Ecology*, *75*(6), 1259–1268. https://doi.org/10.1111/j.1365-2656.2006.01148.x

Gini, C. (1912). Variabilità e mutabilità. In *Reprinted in Memorie di metodologica statistica (Ed. Pizetti E)*. https://ui.adsabs.harvard.edu/abs/1912vamu.book.....G

González, P., Castaño, A., Peacock, E. E., Díez, J., del Coz, J. J., & Sosik, H. M. (2019). Automatic plankton quantification using deep features. *Journal of Plankton Research*, *41*(4), 449–463. https://doi.org/10.1093/plankt/fbz023

Gorsky, G., Ohman, M. D., Picheral, M., Gasparini, S., Stemmann, L., Romagnan, J. B., Cawood, A., Pesant, S., García-Comas, C., & Prejger, F. (2010). Digital zooplankton image analysis using the ZooScan integrated system. *Journal of Plankton Research*, *32*(3), 285–303. https://doi.org/10.1093/plankt/fbp124

Grosjean, P., Picheral, M., Warembourg, C., & Gorsky, G. (2004). Enumeration, measurement, and identification of net zooplankton samples using the ZOOSCAN digital imaging system. *ICES Journal of Marine Science*, *61*(4), 518–525. https://doi.org/10.1016/j.icesjms.2004.03.012

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuaib, B., Liu, T., Wang, X., Wang, L., Wang, G., Caic, J., & Chenc, T. (2018). Recent Advances in Convolutional Neural Networks. *Elsevier Pattern Recognition*, *77*, 354–377. https://doi.org/10.1016/j.patcog.2017.10.013

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. N. (2002). Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning*, *46*, 389–422. https://doi.org/10.1007/978-3-540-88192-6-8

Hagemann, J. (2016). *Umweltwissenschaftliches Forschungsprojekt - LOKI*.

Haralick, R. M., Dinstein, I., & Shanmugam, K. (1973). Textural Features for Image Classification. *IEEE Transactions on Systems, Man and Cybernetics*, *SMC-3*(6), 610–621. https://doi.org/10.1109/TSMC.1973.4309314

Hays, G. C., Richardson, A. J., & Robinson, C. (2005). Climate change and marine plankton. *Trends in Ecology and Evolution*, *20*(6 SPEC. ISS.), 337–344. https://doi.org/10.1016/j.tree.2005.03.004

Hensen, V. (1887). *Ueber die Bestimmung des Planktons oder des im Meere treibenden Materials an Pflanzen und Thieren* (Jahrgang 1). V. Bericht der Commission zur Wissenschaftlichen Untersuchung der Deutschen Meere.

Hu, Q., & Davis, C. (2005). Automatic plankton image recognition with co-occurrence matrices and Support Vector Machine. *Marine Ecology Progress Series*, *295*, 21–31. https://doi.org/10.3354/meps295021

ImageNet. (2016). Stanford Vision Lab, Stanford University, Princeton University. http://www.image-net.org/

Kekre, H. B., & Gharge, S. (2010). Image segmentation using extended edge operator for mammographic images. *International Journal on Computer Science and Engineering*, *June 2014*.

Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 1–70. https://doi.org/10.1007/s10462-020-09825-6

Kirsch, R. A. (1971). Computer Determination Biological of the Constituent Images. *Computers and Biomedical Research 4*, *328*, 315–328.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, *97*(1–2), 273–324. https://doi.org/10.1016/s0004-3702(97)00043-x

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, *25*(2). https://doi.org/10.1145/3065386

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lee, H., Park, M., & Kim, J. (2016). Plankton Classification On Imbalanced Large Scale Database Via Convolutional Neural Networks With Transfer Learning. *IEEE International Conference on Image Processing (ICIP)*, 3713–3717. https://doi.org/10.1109/ICIP.2016.7533053

Leow, L. K., Chew, L. L., Chong, V. C., & Dhillon, S. K. (2015). Automated identification of copepods using digital image processing and artificial neural network. *BMC Bioinformatics*, *16*(18), 1–12. https://doi.org/10.1186/1471-2105-16-S18-S4

Leyer, I., & Wesche, K. (2007). *Multivariate Statistik in der Ökologie*. Springer Lehrbuch. https://doi.org/10.1007/b137219

Liang, J. (2017). *CannyEdgeDetector.m*. http://justin-liang.com/tutorials/canny/

Lombard, F., Boss, E., Waite, A. M., Uitz, J., Stemmann, L., Sosik, H. M., Schulz, J., Romagnan, J. B., Picheral, M., Pearlman, J., Ohman, M. D., Niehoff, B., Möller, K. O., Miloslavich, P., Lara-Lopez, A., Kudela, R. M., Lopes, R. M., Karp-Boss, L., Kiko, R., … Appeltans, W. (2019). Globally consistent quantitative observations of planktonic ecosystems. *Frontiers in Marine Science*, *6*(MAR). https://doi.org/10.3389/fmars.2019.00196

Lumini, A., & Nanni, L. (2019). Ocean Ecosystems Plankton Classification. In M. Hassaballah & K. M. Hosny (Eds.), *Recent Advances in Computer Vision, Studies in Computational Intelligence*. Springer Nature Switzerland. https://doi.org/10.1007/978-3-030-03000-1_11

Lumini, A., Nanni, L., & Maguolo, G. (2019). Deep learning for plankton and coral classification. *Applied Computing and Informatics*, 1–15. https://doi.org/10.1016/j.aci.2019.11.004

Luo, J. Y., Irisson, J. O., Graham, B., Guigand, C., Sarafraz, A., Mader, C., & Cowen, R. K. (2018). Automated plankton image analysis using convolutional neural networks. *Limnology and Oceanography: Methods*, *16*(12), 814–827. https://doi.org/10.1002/lom3.10285

Marr, D., & Hildreth, E. (1980). Theory of edge detection. *Proceedings of the Royal Society of London - Biological Sciences*, *207*(1167), 187–217. https://doi.org/10.1098/rspb.1980.0020

MATLAB (No. 2019b). (2019). The MathWorks, Inc. https://de.mathworks.com/

Medea-AV-GmbH. (n.d.). *medeaLAB Imaging System*. Retrieved September 9, 2020, from http://www.medealab.de/

Mitra, A., Castellani, C., Gentleman, W. C., Jónasdóttir, S. H., Flynn, K. J., Bode, A., Halsband, C., Kuhn, P., Licandro, P., Agersted, M. D., Calbet, A., Lindeque, P. K., Koppelmann, R., Møller, E. F., Gislason, A., Nielsen, T. G., & st. John, M. (2014). Bridging the gap between marine biogeochemical and fisheries sciences; configuring the zooplankton link. *Progress in Oceanography*, *129*(PB), 176–199. https://doi.org/10.1016/j.pocean.2014.04.025

Moniruzzaman, M., Islam, S. M. S., Bennamoun, M., & Lavery, P. (2017). Deep Learning on Underwater Marine Object Detection: A Survey. *Advanced Concepts for Intelligent Vision Systems. ACIVS 2017. Lecture Notes in Computer Science*, *10617*. https://doi.org/https://doi.org/10.1007/978-3-319-70353-4_13

Mustard, A. T., Conquer, M. D., & Allen, J. T. (2003). Laboratory evaluation of a high specification digital CMOS camera for imaging zooplankton at high towing speeds. *Southampton Oceanography Centre, Internal Document*, *89*(17).

Oskoei, M. A., & Hu, H. (2010). A Survey on Edge Detection Methods. *Evaluation*, *February*, 1–36. http://dces.essex.ac.uk/staff/hhu/Papers/CES-506.pdf

Pan, S. J., & Yang, Q. (2009). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359. https://doi.org/10.1109/TKDE.2009.191

Pollio, J., Meyer, R., & Sivak, L. D. (1979). Model analysis of underwater photographic and visibility systems from observed data. *Proceedings of the Society Photo-Optical Instrumental Engineering: Ocean Optics*, *208*(VI), 239 p.

Prewitt, J. M. S. (1970). Object enhancement and extraction. In B. Lipkin & A. Rosenfeld (Eds.), *Picture Processing and Psychopictorics* (pp. 415–431). Academic Press.

Rampun, A., Morrow, P. J., Scotney, B. W., & Winder, J. (2017). Fully automated breast boundary and pectoral muscle segmentation in mammograms. *Artificial Intelligence in Medicine*, *79*, 28–41. https://doi.org/10.1016/j.artmed.2017.06.001

R-Core-Team. (2020). *R* (4.0.2). R: A language and environment for statistical computing. R Foundation for Statistical Computing. https://www.r-project.org/

Roboflow. (2020). *Roboflow-TensorFlow2-Object-Detection.ipynb*. https://colab.research.google.com/drive/1sLqFKVV94wm-lglFq_0kGo2ciM0kecWD#scrollTo=fF8ysCfYKgTP&uniqifier=1

Roboflow Inc. (2020). *roboflow*. https://roboflow.com/

Rockström, J., W. Steffen, K. Noone, Å. Persson, F. S. Chapin, E. F. Lambin, T. M. Lenton, M. Scheffer, C. Folke, H. J. Schellnhuber, B. Nykvist, C. A. de Wit, T. Hughes, S. van der Leeuw, H. Rodhe, S. Sörlin, P. K. Snyder, R. Costanza, U. Svedin, … J. A. Foley. (2009). A safe operation space for humanity. *Nature*, *461*(September), 472–475.

Roemmich, D., & McGowan, J. (1995). Climatic warming and the decline of zooplankton in the California current. *Science*, *267*(5202), 1324–1326. https://doi.org/10.1126/science.267.5202.1324

Schlittgen, R. (2009). *Multivariate Statistik*. Gruyter, Walter de GmbH.

Schröder, S. M., Kiko, R., & Koch, R. (2020). Morphocluster: Efficient annotation of Plankton images by clustering. *Sensors (Switzerland)*, *20*(11), 1–29. https://doi.org/10.3390/s20113060

Schulz, J. (2013). Geometric optics and strategies for subsea imaging. In *Subsea Optics and Imaging*. Woodhead Publishing Limited. https://doi.org/10.1533/9780857093523.3.243

Schulz, Jan. (2017). *R - Basics & statistics*.

Schulz, Jan, Barz, K., Ayon, P., Lüdtke, A., Zielinski, O., Mengedoht, D., & Hirche, H. J. (2010). Imaging of plankton specimens with the lightframe on-sight keyspecies investigation (LOKI) system. *Journal of the European Optical Society*, *5*(April). https://doi.org/10.2971/jeos.2010.10017s

Schulz, Jan, Barz, K., Mengedoht, D., Hanken, T., Lilienthal, H., Rieper, N., Hoops, J., Vogel, K., & Hirche, H. J. (2009). Lightframe on-sight key species investigation (LOKI): The art of imaging minute plankton species on-the-fly. *OCEANS '09 IEEE Bremen: Balancing Technology with Future Needs*, 0–4. https://doi.org/10.1109/OCEANSE.2009.5278252

Schulz, Jan, Mengedoht, D., Barz, K., Basilico, A., Henrich, M., & Hirche, H.-J. (2008). Remote sensing of zooplankton. *Observing the Coastal Sea - an Atlas of Advanced Monitoring Techniques*, *LOICZ Repo*, 10–13.

Schulz, Jan, Mentges, A., & Zielinski, O. (2016). Deriving image features for autonomous classification from time-series recurrence plots. *Journal of the European Optical Society*, *12*(1). https://doi.org/10.1186/s41476-016-0003-y

Schulz, Jan, Möllmann, C., & Hirche, H. J. (2007). Vertical zonation of the zooplankton community in the Central Baltic Sea in relation to hydrographic stratification as revealed by multivariate discriminant function and canonical analysis. *Journal of Marine Systems*, *67*(1–2), 47–58. https://doi.org/10.1016/j.jmarsys.2006.09.004

Schulz, Jan, Peck, M. A., Barz, K., Schmidt, J. O., Hansen, F. C., Peters, J., Renz, J., Dickmann, M., Mohrholz, V., Dutz, J., & Hirche, H. J. (2012). Spatial and temporal habitat partitioning by zooplankton in the Bornholm Basin (central Baltic Sea). *Progress in Oceanography*, *107*(June 2018), 3–30. https://doi.org/10.1016/j.pocean.2012.07.002

Shao, L., Zhu, F., & Li, X. (2015). Transfer learning for visual categorization: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(5), 1019–1034. https://doi.org/10.1109/TNNLS.2014.2330900

Sieracki, C. K., Sieracki, M. E., & Yentsch, C. S. (1998). An imaging-in-flow system for automated analysis of marine microplankton. *Marine Ecology Progress Series*, *168*, 285–296. https://doi.org/10.3354/meps168285

Sobel, I., & Feldman, G. (2015). An Isotropic 3x3 Image Gradient Operator. *Stanford Artificial Intelligence Project (SAIL)*, *June*, 271–272.

Solawetz, J. (2020). *How to Train a TensorFlow 2 Object Detection Model*. https://towardsdatascience.com/how-to-train-a-tensorflow-2-object-detection-model-25d4da64b817

Sosik, H. M., & Olson, R. J. (2007). Automated taxonomic classification of phytoplankton sampled with imaging-in-flow cytometry. *Limnology and Oceanography: Methods*, *5*(6), 204–216. https://doi.org/10.4319/lom.2007.5.204

Steele, J. H. (1974). *Marine food webs and overfishing* (President and Fellows of Harvard College, Ed.; 3rd ed.). Harvard University Press.

Stramma. (2009). *Short cruise report METEOR cruise leg M77 / 4 Climate-biogeochemistry interactions in the tropical ocean of the SE-American oxygen minimum zone*.

Strobl, C., Boulesteix, A. L., Zeileis, A., & Hothorn, T. (2007). Bias in variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, *8*. https://doi.org/10.1186/1471-2105-8-25

Tait, R., & Dipper, F. (1998). *Elements of marine ecology*. Oxford: Butterworth Heinemann.

Tang, X., Stewart, W. K., Vincent, L., Huang, H., Marra, M., Gallager, S. M., & Davis, C. S. (1998). Automatic Plankton Image Recognition. *Artificial Intelligence Review*, *12*, 177–199.

TensorFlow. (n.d.). Retrieved September 9, 2020, from https://www.tensorflow.org/

Tian, Y., Wu, Q., Engineering, I., & Han, Z. (2019). *Automatic Recognition Method of Zooplankton Image in Dark Field*. *XXIX*, 1894–1903.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer Science + Business Media, LLC.

Verikas, A., Gelzinis, A., Bacauskiene, M., Olenina, I., & Vaiciukynas, E. (2015). An Integrated Approach to Analysis of Phytoplankton Images. *IEEE Journal of Oceanic Engineering*, *40*(2), 315–326. https://doi.org/10.1109/JOE.2014.2317955

Wang, R., Dai, J., Zheng, H., Ji, G., & Qiao, X. (2016). Multi features combination for automated zooplankton classification. *OCEANS 2016 - Shanghai*, 8–12. https://doi.org/10.1109/OCEANSAP.2016.7485675

Wiebe, P. H., & Benfield, M. C. (2003). From the Hensen net toward four-dimensional biological oceanography. *Progress in Oceanography*, *56*(1), 7–136. https://doi.org/10.1016/S0079-6611(02)00140-4

Yamazaki, H., Mackas, D., & Denman, K. (2002). Coupling small-scale physical processes with biology. In *The sea* (Vol. 12).

Zelterman, D. (2015). Applied Multivariate Statistics with R. In *Applied Multivariate Statistics with R*. https://doi.org/10.1007/978-3-319-14093-3

Zheng, H., Wang, R., Yu, Z., Wang, N., Gu, Z., & Zheng, B. (2017). Automatic plankton image classification combining multiple view features via multiple kernel learning. *BMC Bioinformatics*, *18*(238), 1–18. https://doi.org/10.1186/s12859-017-1954-8

Appendix

Appendix 1: Datasets used within the thesis with their number of images and classes, if they were sorted into classes.

Dataset	Images	Number of classes (if classified)
Sognefjord total	113,551	-
Sognefjord classed	37,057	64
Sognefjord 14 groups	37,057	14
Sognefjord 26 groups	37,108	26
Sognefjord augmented	36,406	14
Peru	1,023	21
Peru small	180	4

Appendix 2: The 14 classes of the Sognefjord 14 groups and augmented dataset. Their original number of image files and the number of files after image processing and augmentation are listed.

Class	Files original	Files after image processing	Files after image augmentation (CNN)
Acantharia	850	848	799
Amphipoda	17	14	306
Bubble	2,872	2,872	2,872
Chaetognatha	979	890	979
Cnidaria	221	199	442
Copepoda	23,766	23,727	21,844
Ctenophora	52	51	260
Detritus	5,757	5,757	5,757
Egg	587	587	587
Euphausiidae	70	63	280
Faeces	193	193	386
Mysidae	350	335	350
Ostracoda	1,293	1,289	1,293
Polychaeta	50	50	250

Appendix 3: Metadata information extracted from the file path of each plankton image.

Metainformation	Format	Metainformation	Format
File: GMT	Date-Time	File: Bounding box Y-offset	Pixel
File: Cruise	String	File: Base directory	String
File: Station	String	File: Relative path	String
File: Haul	String	File: Name	String
File: LOKI Device ID	String	File: Image format	String
File: Millisecond	Frac-Second	File: Colourspace	Integer
File: Millisec index	Integer	File: Filesize (Bytes)	Integer
File: Bounding box X-offset	Pixel		

Appendix 4: Image feature information extracted from each plankton image.

Feature	Format	Feature	Format
Size: Image width	Pixel	Haralick: Image summed variance [SVA]	Scalar
Size: Image height	Pixel	Haralick: Image sum of entropy [SEN]	Scalar
Size: Mass centre X	Pixel	Haralick: Image entropy [ENT]	Scalar
Size: Mass centre Y	Pixel	Haralick: Image difference variance [DVA]	Scalar
Size: Elliptical fit major axis	Pixel	Haralick: Image difference entropy [DEN]	Scalar
Size: Elliptical eccentricity	Scalar	Haralick: Image information measure correlation [f12]	Scalar
Size: Elliptical major axis angle	Radians	Haralick: Image information measure correlation [f13]	Scalar
Pixel-border distances: sum	Scalar	COO: Contour line points	Scalar
Pixel-border distances: median	Scalar	COO: Contour line length	Scalar
Pixel-border distances: mean	Scalar	COO: Contour line height	Scalar
Pixel-border distances: SD	Scalar	COO: Contour line area	Scalar
Intensity: Mean	Scalar	COO: Calliper length [Ferret's diameter	Scalar
Intensity: SD	Scalar	COO: Centroid size	Scalar
Intensity: Abs. deviation	Scalar	COO: Centre to contour points sum	Scalar
Intensity: Quantile 0.01	Scalar	COO: Centre to contour points mean	Scalar
Intensity: Quantile 0.05	Scalar	COO: Centre to contour points median	Scalar
Intensity: Quantile 0.10	Scalar	COO: Centre to contour points variance	Scalar
Intensity: Quantile 0.20	Scalar	COO: Centre to contour points SD	Scalar
Intensity: Quantile 0.30	Scalar	COO: Convex hull points	Scalar

Intensity: Quantile 0.40	Scalar	COO: Circularity	Scalar
Intensity: Quantile 0.50	Scalar	COO: Circularity normalized	Scalar
Intensity: Quantile 0.60	Scalar	COO: Haralick's circularity	Scalar
Intensity: Quantile 0.70	Scalar	COO: Convexity	Scalar
Intensity: Quantile 0.80	Scalar	COO: Eccentricity	Scalar
Intensity: Quantile 0.90	Scalar	COO: Elongation	Scalar
Intensity: Quantile 0.95	Scalar	COO: Perimeter	Scalar
Intensity: Quantile 0.99	Scalar	COO: Solidity	Scalar
Shape: Area	Pixel	Recurrence: Embedding dimension	Scalar
Shape: Perimeter	Pixel	Recurrence: Embedding point distance (tau)	Scalar
Shape: Radius mean	Pixel	Recurrence: Threshold	Scalar
Shape: Radius SD	Scalar	Recurrence: Minkowski order	Scalar
Shape: Radius minimum	Scalar	Recurrence: Total recurrence	Scalar
Shape: Radius maximum	Scalar	Recurrence: Recurrence rate [RR]	Scalar
Haralick: Homogeneity [ASM]	Scalar	Recurrence: Determinism [DET]	Scalar
Haralick: Contrast	Scalar	Recurrence: Laminarity [LAM]	Scalar
Haralick: Image correlation [COR]	Scalar	Recurrence: Ratio [DET/RR]	Scalar
Haralick: Sum of squares [VAR]	Scalar	Recurrence: Averaged diagonal length [L]	Scalar
Haralick: Inverse difference moment [IDM]	Scalar	Recurrence: Trapping Time [TT]	Scalar
Haralick: Image summed average [SAV]	Scalar		

Appendix 5: Elements of telemetry data measured by the CTD and on bord GPS to complement the Lightframe On-Sight Keyspecies Investigation image data.

Device	Oxy Saturation	Cond Speed	Loki Frame
GPS Longitude	Oxy Temperature	Flour 1	Cam Stat
GPS Latitude	Cond Conductivity	Roll	House Stat
Press	Cond Temperature	Pitch	House T1
Temp	Cond Salinity	Loki Record	House T2
Oxy Conductivity	Cond Density	Loki Picture	House Voltage

Appendix 6: Links and refences for used R packages and MATLAB toolboxes

Package / Toolbox Name	Software	Reference / Link
BiocManager	R	https://cran.r-project.org/web/packages/BiocManager/index.html
EBImage	R	https://bioconductor.org/packages/release/bioc/html/EBImage.html
magick,	R	https://cran.r-project.org/web/packages/magick/magick.pdf
readr,	R	https://cran.r-project.org/web/packages/readr/readr.pdf
data.table	R	https://cran.r-project.org/web/packages/data.table/data.table.pdf
IM	R	https://cran.r-project.org/web/packages/IM/index.html
Momocs	R	https://cran.r-project.org/web/packages/Momocs/Momocs.pdf
tools	R	https://www.rdocumentation.org/packages/tools/versions/3.6.2
png	R	https://cran.r-project.org/web/packages/png/png.pdf
bmp	R	https://cran.r-project.org/web/packages/bmp/bmp.pdf
parallel	R	https://stat.ethz.ch/R-manual/R-devel/library/parallel/doc/parallel.pdf
doSNOW	R	https://cran.r-project.org/web/packages/doSNOW/doSNOW.pdf
foreach	R	https://cran.r-project.org/web/packages/foreach/foreach.pdf
MASS	R	https://cran.r-project.org/web/packages/MASS/MASS.pdf
OpenImageR	R	https://cran.r-project.org/web/packages/OpenImageR/OpenImageR.pdf
e1071	R	https://cran.r-project.org/web/packages/e1071/e1071.pdf
randomForest	R	https://cran.r-project.org/web/packages/randomForest/randomForest.pdf
FSinR	R	https://cran.r-project.org/web/packages/FSinR/FSinR.pdf
caret	R	https://cran.r-project.org/web/packages/caret/caret.pdf
Deep Learning Toolbox Model for AlexNet Network	MATLAB	https://de.mathworks.com/matlabcentral/fileexchange/59133-deep-learning-toolbox-model-for-alexnet-network
Deep Learning Toolbox	MATLAB	https://de.mathworks.com/products/deep-learning.html?requestedDomain=

Appendix 7: R and MATLAB code for image priming, data preparation, feature selection and classification the code can be accessed at https://gitlab.uni-oldenburg.de/gorm2097/loki-image-processing-and-classification/.

Software	Script
Image priming	
R	2020-09-09-LOKI-Complete data primer.R
Dataset preparation	
R	20200909_AddClassColumn2DataTable.R
R	20200909_CreateTrainAndTestDataset_FromClassedTable.R
R	20200909_ImageAugmentation.R
Feature selection	
R	20200909_ExcludeHighCorrelation.R
R	20200909_GiniRanked_AccuracyTest_LDA_rF.R
Classification	
R	20200909_rF_14classes_37057_40features.R
R	20200909_LDA_14classes_37057_43features.R
R	20200909_SVM_14classes_37057_26features.R
MATLAB	LOKInet_20200905.m

Eidesstattliche Erklärung

Hiermit versichere ich Eides statt, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.