JACOBS UNIVERSITY

Shape-based Machine Perception of Man-Made Objects on Underwater Sensor Data

by

Daniel Köhntopp

A thesis submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

Date of Defense: April 24, 2018

Computer Science & Electrical Engineering

Statutory Declaration

Family Name, Given/First Name	Köhntopp, Daniel
Matriculationnumber	20330854
What kind of thesis are you submitting: Bachelor-, Master- or PhD-Thesis	PhD-Thesis

English: Declaration of Authorship

I hereby declare that the thesis submitted was created and written solely by myself without any external support. Any sources, direct or indirect, are marked as such. I am aware of the fact that the contents of the thesis in digital form may be revised with regard to usage of unauthorized aid as well as whether the whole or parts of it may be identified as plagiarism. I do agree my work to be entered into a database for it to be compared with existing sources, where it will remain in order to enable further comparisons with future theses. This does not grant any rights of reproduction and usage, however.

The Thesis has been written independently and has not been submitted at any other university for the conferral of a PhD degree; neither has the thesis been previously published in full.

German: Erklärung der Autorenschaft (Urheberschaft)

Ich erkläre hiermit, dass die vorliegende Arbeit ohne fremde Hilfe ausschließlich von mir erstellt und geschrieben worden ist. Jedwede verwendeten Quellen, direkter oder indirekter Art, sind als solche kenntlich gemacht worden. Mir ist die Tatsache bewusst, dass der Inhalt der Thesis in digitaler Form geprüft werden kann im Hinblick darauf, ob es sich ganz oder in Teilen um ein Plagiat handelt. Ich bin damit einverstanden, dass meine Arbeit in einer Datenbank eingegeben werden kann, um mit bereits bestehenden Quellen verglichen zu werden und dort auch verbleibt, um mit zukünftigen Arbeiten verglichen werden zu können. Dies berechtigt jedoch nicht zur Verwendung oder Vervielfältigung.

Diese Arbeit wurde in der vorliegenden Form weder einer anderen Prüfungsbehörde vorgelegt noch wurde das Gesamtdokument bisher veröffentlicht.

……
Date, Signature

First of all, I would like to thank my PhD supervisors Prof. Dr.-Ing. Dieter Kraus and Prof. Dr. Andreas Birk for guiding me through my time as a PhD student and showing me the value of precise notation and expressing my scientific work in an engaging way.

I would also like to thank Prof. Dr. Francesco Maurelli for joining my dissertation committee and reviewing my thesis.

The next person I would like to thank is Dr.-Ing. Benjamin Lehmann for providing the data and support without which this thesis would not be possible.

My gratitude goes to all my current and past colleagues both at the Jacobs University Bremen, as well as at the Hochschule Bremen. Special thanks to my office fellows Narunas, Ziliang, and Arturo for the valuable discussions about work – but also life.

Of course, my thanks goes to my family for their continuous support and especially my sister Katrin for giving me from time to time an "outside-view" on my scientific work.

Last but not least, my biggest gratitude goes to my girlfriend Andrea for her patience and continuous support.

Abstract

The underwater domain poses numerous challenges for robots. One of them is the adversarial effect water has on the usual means of perceiving the environment. Light based imaging systems are negatively affected by the high light attenuation which severely limits their range and causes a colour shift. Substitute methods like sonar are still far from revealing the same level of details as optical cameras can deliver. At the same time, the noise level is significantly higher on acoustics based imaging systems. Nevertheless, the need for an accurate recognition of the surrounding environment is especially high for underwater robots. The water only allows a very limited wireless data connection if at all. Therefore, autonomous or semi-autonomous behaviour is of paramount importance to handle connection loss cases. Recognizing the environment is one of the building blocks for autonomous behaviour.

This thesis focuses on the common case where the objects of interest are man-made and known a-priori. The idea is that the information about the shape of the object of interest can efficiently guide object localization, segmentation and classification algorithms to the correct result while mitigating the effect of noise and occlusions.

In this thesis different implementations of the general idea of using shape a-priori knowledge are investigated. One implementation is the introduction of an efficient screening algorithm that finds potential target objects on synthetic aperture sonar images. An efficient algorithm that preselects regions of interests is needed due to the high amount of image data produced every second. In this thesis a fast integral image based template matching framework is described. New template types and feature types that take the shape of the objects of interest into account and that are tailored to the detection on synthetic aperture sonar images are introduced.

The general idea of using shape pre-knowledge also sparked the idea of using superellipses as a representation of the shapes of man-made objects. Two different approaches are presented in this thesis. First superellipse fitting onto already extracted object contours is investigated. To this end, a novel linearisation of the fitting error equation was proposed. Experiments showed that the linearisation decreased the computation time significantly while only slightly affecting the fitting accuracy. But the experiments also showed that the superellipse representation should not be used as an afterthought, i.e. as post-processing technique after the contour extraction, but rather be employed by the contour extraction algorithm. To this end the active contours without edges framework was reformulated to accommodate a superellipse shape constraint. With the new formulation it was also discovered that the implicitly assumed underlying Gaussian pixel intensity distribution can easily be substituted to a more fitting pixel distribution. This is especially beneficial when the imaging system is not a optical camera but for example a sonar. The experiments showed that using the proposed superellipse guided active contours method not only was able to properly extract the contour of objects on synthetic aperture sonar images but also that the superellipse parameters can directly used to classify the objects. The superellipse guided active contours algorithm can also be beneficial for other challenging noisy imaging systems besides sonar. Imaging systems for medical applications suffer from similar noise levels. The novel formulation makes it easy to accommodate suitable pixel intensity in a plug-and-play fashion.

The last method proposed in this thesis is still employing a-priori known shapes but in contrast to the aforementioned methods it does so in a top-down fashion. The assumption is that there was already a classification and pose estimation of the found object and the goal is to verify the plausibility of the result. In order to do so the class information and the pose information is used to generate a simulated image that should show the same content as the original image. Via a similarity measure the classification result can be verified or rejected. In this thesis, this classification verification approach was used on synthetic aperture sonar images. To this end a set-up is proposed to approximately simulate the sonar image generation via a ray tracing program for light. The approximate simulation yields images that are similar to real synthetic aperture images while having a much faster processing time than a physically correct simulation of the synthetic aperture sonar imaging process. The experiments show that the top-down approach is a valuable tool to identify false positives.

List of Publications

The following publications were made in the time leading to the PhD:

[33] D. Köhntopp, B. Lehmann, and D. Kraus, “Computational efficient object detection exploiting advanced templates”, in *Proceedings of the 1st international conference and exhibition on Underwater Acoustics (UA2013)*, 2013

[34] D. Köhntopp, B. Lehmann, and D. Kraus, “Efficient superellipse fitting based contour extraction for mine-like shape recognition”, in *Proceedings of the 2nd international conference and exhibition on Underwater Acoustics (UA2014)*, 2014

[36] D. Köhntopp, B. Lehmann, D. Kraus, and A. Birk, “Segmentation and classification using active contours based superellipse fitting on side scan sonar images for marine demining”, in *International Conference on Robotics and Automation (ICRA)*, IEEE Press, 2015

[37] D. Köhntopp, B. Lehmann, D. Kraus, and A. Birk, “Autonomous mine recognition using AUV and ATR”, in *UDT 2017*, 2017

[38] D. Köhntopp, B. Lehmann, D. Kraus, and A. Birk, “Seafloor classification for mine counter measure operations on synthetic aperture sonar images”, in *Oceans ’17 MTS/IEEE*, Aberdeen, 2017

[35] D. Köhntopp, B. Lehmann, D. Kraus, and A. Birk, “Classification and localization of naval mines with superellipse active contours (accepted)”, *IEEE Journal of Oceanic Engineering*, 2018

[55] C. A. Mueller, T. Fromm, A. G. Chavez, D. Köhntopp, and A. Birk, “Robust Continuous System Integration for Critical Deep-Sea Robot Operations Using Knowledge-Enabled Simulation in the Loop (submitted)”, in *International Conference on Intelligent Robots and Systems (IROS)*, IEEE Press, 2018

Contents

List of Figures

List of Tables

List of Symbols

$\boldsymbol{A}$	Matrix of the linearised formulation of the parametrised superellipse equation
a	Half-axis parameter of a superellipse in horizontal direction
a^*	Approximation of a via linearisation
$\mathfrak{a}$	A region of the image given by the sign(s) of the levelset function(s)
$a_\bullet$	Horizontal half-axis parameter for the I, II, III, or IV quadrant
α	Natural parameter of the Gamma distribution
$\mathcal{A}$	Example area of a certain shape in an image
A_α	Surface of a 2-dimensional image for scale α for fractal dimension calculation
B	Bandwidth of sonar ping
b	Half-axis parameter of a superellipse in vertical direction
b^*	Approximation of b via linearisation
$b_\bullet$	Vertical half-axis parameter for the I, II, III, or IV quadrant
β	Natural parameter of the Gamma distribution
B^{-i}	Indices of all bootstrap sets that do not contain the i-th image
C	Extracted contour
c	Speed of sound in water
C^s	Curve of a superellipse
D	Fractal dimension
$d_\bullet$	A distance approximation between a point and a superellipse
δ	Dirac function
δ_ξ	Regularized Dirac function

ε	Half-axis parameter of a superellipse in vertical direction
ε^*	Approximation of ε via linearisation
err	Error for a certain feature threshold combination used in the gentle AdaBoost algorithm
$\widehat{Err}^{(1)}$	Leave-one-out bootstrap error
$\widehat{Err}^{(.632+)}$	".632+" estimator of the prediction error
$\overline{err}$	Training error
η	Short version of $\hat{\eta}_{\mathfrak{a}}$
$\hat{\eta}_{\mathfrak{a}}$	Natural parameters of the probability density function of the pixel values of region $\mathfrak{a}$ calculated by maximum likelyhood estimation
F	Implicit superellipse function
f	The active contours functional
FP	False positive rate of the cascaded classifier
fp	Maximal false positive rate per stage of the cascaded classifier
Γ	Gamma function
γ	Natural parameter of the Rayleigh distribution
$\hat{\gamma}$	No information rate for the bootstrap approach
$g_{\mathfrak{a}}$	Weight function in the active contours functional for the area $\mathfrak{a}$
$\mathcal{H}$	Highlight area
h	Loss function
$\boldsymbol{H}$	Feature matrix with number of samples rows and number of features columns
II	Integral image for rectangles
I^{-i}	Indices of all sample sets in the bootstrap approach that do not contain the i-th sample
K	Number of resulting classifiers in the bootstrap approach
k	Unknown fractal dimension constant
L	aperture of the sonar antenna
L'	Synthetic aperture of the sonar antenna
l	Levelset function in the active contours functional
L^*	Aperture of one element in the sonar antenna
λ	wavelength of sonar frequency
l_c	Isoline

$lowErr$	Lowest error for a certain feature used in the gentle AdaBoost algorithm
M	Number of uniformly with replacement drawn samples for one classifier of the bootstrap approach
$\mu_{a,b}$	Mean value of a for inputs from b
N	Number of points of the extracted contour
n	Number of elements forming the uniform linear array antenna
N_ω	Number of pixels in an area ω
ν	Superellipse parameter vector
Ω	Pixel space
ω	A certain part of the image ($\omega \subseteq \Omega$)
$\boldsymbol{P}$	Auxiliary matrix
ϖ	Natural parameter of the Weibull distribution
Ψ	Logarithmic derivative of the gamma function, also called Digamma function
$\widehat{p}$	No information rate for the bootstrap approach
q_{rot}	Rotation transformation
q_{tap}	Tapering transformation
q_{trans}	Translation transformation
r	Range to object
R_a	Along track resolution
$\boldsymbol{\rho}$	The half axis parameters of a superellipse in a vector
$\boldsymbol{\rho}^*$	Approximation of ρ via linearisation
RII	Integral image for 45 degree rotated rectangles
R_r	Across track resolution
$\widehat{R}$	Weight in the ".632+" estimator of the prediction error
$\mathcal{S}$	Shadow area
s	Scale i.e. neighbourhood size for fractal dimension calculation
$\sigma^2_{a,b}$	Variance of the value of a for inputs from b
T	Translation of the superellipse
t	Number of stages of the cascaded classifier
τ	Template representing shadow highlight formation on SAS image
Θ	Heaviside function

θ	Rotation of the superellipse
ϑ	Parameter of the parametrised superellipse definition
Θ_ξ	Regularized Heaviside function
thr	Threshold value used in the gentle AdaBoost algorithm
TII_a	Triangular integral image with right angled peak in direction $a = NW, NE, SE, SW$
TP	True positive rate of the cascaded classifier
tp	Minimal true positive rate per stage of the cascaded classifier
u	Image. Based on the context often used as u=u(z)
u_i	Pixel value at position i
υ	Natural parameter of the Weibull distribution
w	Scaling factor for superellipse distance approximation
$\boldsymbol{w}$	Weights for the gentle AdaBoost classifier
$\widehat{w}$	Weight in the ".632+" estimator of the prediction error
$\boldsymbol{X}$	Right-hand side of the linearised formulation of the parametrised superellipse equation
x	Horizontal pixel position
x^*	Horizontal pixel position of z^*
ξ	Regularization parameter for Θ and δ
y	Vertical pixel position
y^*	Vertical pixel position of z^*
$\bar{y}$	Ground truth class label
y_-	Weak label used in the gentle AdaBoost algorithm
y_+	Weak label used in the gentle AdaBoost algorithm
$\boldsymbol{z}$	Pixel position $\boldsymbol{z} = (x, y)^T$
$\boldsymbol{z}^*$	Point on a superellipse curve that approximates z^s
$\boldsymbol{z}^s$	Nearest point in Euclidean distance on a superellipse curve to a given point z
ζ_c	Sample correlation coefficient
ζ_F	Test characteristic of the F-test

ζ_t	Test characteristic of the Student's t-test
ζ_w	Test characteristic of the Welch's t-test

List of Acronyms

3PP	three parameter problem
4x3PP	four superellipses independently
8PP	four coupled superellipses
ATR	Automatic Target Recognition
AUC	Area Under the ROC Curve
AUV	Autonomous Underwater Vehicle
CMRE	Centre for Maritime Research and Experimentation
CNN	Convolutional Neural Network
CY	cylindrical objects
MCC	Matthews Correlation Coefficient
MCM	Mine Counter Measure
MCMV	Mine Counter Measure Vehicle
MMO	Man-Made Object
NMO	Non-Mine Object
NURC	NATO Undersea Research Centre
PCA	Principal Component Analysis
pdf	probability density function
RMS	Root Mean Square
RO	mine-sized rocks
ROC	Receiver Operator Characteristic
ROI	Region Of Interest
ROV	Remotely Operated Vehicle
SACE	Superellipse guided Acitive Contours without Edges
SAS	Synthetic Aperture Sonar
SNR	Signal-to-Noise Ratio
SONAR	SOund NAvigation and Ranging
SVM	Support Vector Machine
TR	truncated cones

UUV	Unmanned Underwater Vehicle
WE	wedge shaped objects

CHAPTER 1

Introduction

The oceans are a vast resource and more and more scientific work is done to further explore this domain. Unmanned Underwater Vehicles (UUVs) can support these endeavours tremendously by carrying and using heavy equipment and sensors which human divers could not bring along or directly operate themselves. Sometimes UUVs are even the only option to succeed with the mission task. The location can be too deep and hazardous for human divers.

Using an UUV gives raise to a whole new set of robotic challenges unique to the underwater domain. Many of those challenges stem from the fact that the communication with the vehicle is severely limited due to the inability of radio waves to penetrate the water. There are alternative options to communicate underwater but they usually have a low data transmission rate and a short range. Many limitations can be mitigated by giving the UUV the ability to react to its environment. Therefore, it is essential to investigate how autonomous capabilities can be given to an UUV. The first crucial step towards autonomous behaviour is the recognition step. Not only recording the surrounding water volume through perceptual sensors but actually processing the information to recognize the environment and the objects in it can be a key enabler for the UUV to efficiently fulfil its task.

In this thesis the topic of underwater object recognition is investigated. Unfortunately, the underwater domain imposes severe quality limitations on the sensor data. On the bright side, the tasks of an UUV are usually very specific. Hence, the knowledge that in many applications the geometry of the occurring man-made objects of interest is already known can be exploited. Based on this general idea to use domain specific knowledge the following contributions will be made in this thesis

- The introduction of new integral image based templates in the framework of an efficient detection algorithm based on an cascaded AdaBoost,
- Linearisation of the loss function used for superellipse fitting
- Reformulation of the basic active contours without edges algorithm towards a more abstract version which makes it possible to
  - instantiate it with a superellipse shape prior and to
  - instantiate it with probability density functions that properly model the pixel intensity distributions of an given imaging system
- A new classification approach based on extracted superellipse parameters
- A setup to approximately simulate sound based images
- A new top-down approach to verify classification results via simulation

All these contributions are evaluated on *images* taken underwater. Most example images in this thesis are Synthetic Aperture Sonar (SAS) images acquired during naval mine hunting operations. The next chapter will go into more detail about sonar, SAS and mine hunting. Besides sonar data also visual data gathered with an optical underwater camera is used in some experiments. The application in mind for this data is in the field of deep sea operations. An introduction to the visual data will also be given in the next chapter.

1.1 Related work and outline

There are many applications that can benefit from shape-based machine perception of man-made objects on underwater sensor data. The particular applications that motivated this thesis are, on the one hand, naval demining operations with an Autonomous Underwater Vehicle (AUV) equipped with an SAS. And, on the other hand, inspection and maintenance tasks for underwater oil and gas production, where a ROV equipped with an optical underwater camera is used. In both tasks the recognition of man-made objects is crucial.

Demining with land robots has been intensively studied since well more than a decade and there is a significant amount of research on the related locomotion aspects (e.g., [32], [28]), the manipulation of the mines (e.g., [31], [54], [72]), the perception of the mines (e.g., [2], [40], [13], [41]), and the related mapping and planning problems (e.g., [86], [65], [42], [89]), which has led to a very advanced state of fieldable solutions (e.g., [29], [74]). There is in contrast much fewer work on demining with marine robots [59], [60], which is likely due to the fact that it is a very challenging perception problem in the underwater domain that fortunately can profit from recent advances in sensor technology.

Naval mines pose a huge threat to civil and military shipping alike. They are easy to deploy and highly efficient in relation to their inherent damage potential and cost of removal. In order to combat naval mines, sonar can be used to get high-resolution images of the seafloor. To minimise the risk for humans, an UUV equipped with a SAS can be used to survey the potential mine fields. Besides safety implications, this so called stand-off approach where a mine hunting ship stays safely out of the mine field has direct financial implications. Nowadays, so called Mine Counter Measure Vehicle (MCMV) have to be build very sturdy i.e., with a high shock resistance to keep the crew safe in case of exploding mines in close proximity. If the ship does not has to travel in the mine field, design restrictions can be relaxed and the overall costs can be decreased. These safety and financial advantages have let to a substantial interest in changing the classical approach of a MCMV that travels into a mine field towards unmanned approaches. Procuring plans of the next decades for several navys reflect that, e.g., Norway [57].

The first step in the perception workflow is usually a rough detection of the location of the object of interest in the image showing the full view. Especially SAS images can show several hundred square metres of seafloor. If the AUV is meant to react to the perceived environment, the recognition needs to be fast. Thus, it is unfeasible to employ the necessarily complex recognition algorithms at every point in the image. Therefore, a fast screening method has to be used to reduce the image to smaller meaningful regions of interest. There are several options to do so. The authors of [85] or [84] use unsupervised algorithms that essentially search via thresholds for local anomalies that are related to the special highlight shadow structure objects cast on SAS images. By using specific domain knowledge they filter out step by step candidate regions.

In contrast to this unsupervised approach but with the same general idea regarding the use of domain knowledge a supervised algorithm is investigated in this thesis. Looking into detection problems in other research fields an efficient cascaded template-based detection algorithm for face detection can be found [79], [48], [63]. The authors of [46] transferred this algorithm to be used on SAS images. In Chapter 3 their work will be the starting point to further adapt the algorithm to the conditions on SAS images and improve its detection capabilities. Parts of the results are also published in [33].

After the location of a potential object of interest is detected a classification follows. In the literature the object-shadow is considered as the most distinctive feature of an object on a sonar image. Therefore many classification methods use descriptors derived from the segmented shadow [7], [64] or comparisons between the shape of the shadow and model based simulations [67] to train classifiers.

For the object classification on SAS images the authors of [18] introduced the idea of approximating the extracted contours by a special class of geometric shapes called superellipses. The parameters of the superellipses can then be used for a successful classification between different object types. In Chapter 4 a linearisation of the loss function used for the fitting is presented to make the computation more efficient. But even with an efficient computation the main drawback of this approach is the dependence on an already extracted contour.

Inspired by the approach of [45] our method presented in the second part of Chapter 4 incorporates superellipses directly in a segmentation algorithm called active contours without edges. The authors of [45] do this by adding a penalty term. This results in a contour that is not strictly restricted to superellipse shapes but just *pushed* towards them, whereas the *force* is adjusted by a parameter. But this means the result will not include the superellipse parameters which [18] proved to be useful for the classification. Hence, the approach presented in second part of Chapter 4 will incorporate the superellipse shapes in the segmentation algorithm as a constraint. The experiments will show that this improves the classification results gained from the approach by [18]. Preliminary results leading to the results presented in Chapter 4 are published in [34] and [36].

One of the biggest challenges in the underwater domain is that the quality of the available image data is subpar to the image quality one is used to in in-air applications. Photography, i.e., an optical imaging system operating in the visible light spectrum, can be used underwater. But the resulting images are negatively affected by the high light attenuation and particles in the water. The colour information reaching the optical camera is shifted and objects further away than a few meters are hardly visible. The low range makes optical cameras unfeasible for demining tasks. One the one hand, great areas of seafloor have to be surveyed in short time. On the other hand, bringing the usually very expensive AUV near a potential explosive object risks the total loss of the vehicle. Alternatively, one can use sonar systems, i.e., sound based imaging systems. Sonar systems can have a far greater range than underwater photography. But the image quality with respect to the prospect of correctly identifying the objects is worse.

The subpar image quality makes classification especially challenging and can lead to many false classifications. There are several works on how the SAS image quality is linked to the detection and classification accuracy [84],[81], [47], [26] or how the information from the raw sonar data can be increased [80]. In general, false classifications due to the poor image quality have to be anticipated. Hence, the thesis concludes with classification verification.

In Chapter 5 simulation of SAS images for a given class and pose is integrated in a verification step of the classification result. A top-down approach is followed. An image is simulated according the classification result. This expectation image is compared to the original image. The similarity decides whether to trust the classification or not. This top-down classification approach is to the best of the authors knowledge new to the field of underwater image classification. Albeit there are authors who used simulation as a tool for classification [67],[20]. The findings of Chapter 5 were also submitted as journal article [35].

As outlined, the use of optical underwater cameras is unfeasible for the detection and classification during demining operations. However, in contrast to the just described demining task, optical underwater cameras are a valuable asset for ROV based inspection and maintenance tasks in, for example, underwater oil and gas production. The ROV is in near vicinity to the objects of interest because it usually has to interact with the objects, e.g., turn a valve. The high level of details an optical camera offers is needed for a precise interaction with objects or to spot anomalies, e.g., cracks in pipes.

Machine perception of man-made objects can be a valuable addition to aid the ROV operator. Besides the passive highlighting of objects of interest, it can elevate the raw ROV control towards a more abstract, task oriented control.

In Chapter 4 an algorithm for the recognition of handles and their poses on a representative mock-up panel is presented. Again, the shape of these man-made objects is known a-priori and this knowledge can be exploited for a robust recognition. Thus, the previously described superellipse guided active contours without edges algorithm can be applied again. Results were also submitted as conference paper [55].

To summarize the outline; the next chapter will give an introduction to the applications in which the image data used in the experiments was generated. Afterwards, in Chapter 3 an efficient detection algorithm on SAS images will be discussed. This is followed by Chapter 4 with the introduction of the assumption that the objects of interest can be approximated by superellipses. This leads in the first part of the chapter to a linearised formulation of the superellipse fitting loss function to make the computation more efficient. The second part introduces a superellipse guided active contours without edges method to overcome the shortcomings of the fitting approach.

In the final Chapter 5 a method to validate the classification result of an object on a SAS image is proposed. A method to *approximately* simulate SAS images of the objects of interest is presented alongside with the evaluation of different methods to measure the similarity between real and simulated SAS image. The thesis ends in Chapter 6 with a summary and an outlook on what could be done next.

CHAPTER 2

Underwater Sensor Data

In this chapter an introduction of the origin of the image data used in the different experiments is made. Since most of the experiments will be done on SAS images, it will begin with a motivation for the use of SAS images and one application. The section after this covers visual image data from an optical underwater camera.

2.1 Side scan sonar images for mine hunting

Sonar is an essential sensor for perception in the underwater domain. But sonar has quite some limitations with respect to, e.g., the presence of noise, its resolution, and its update rates. Core machine perception tasks like the detection, recognition, and localization of objects are hence not trivial when using sonar; especially when a high robustness is required. In this thesis, an approach to object recognition and localization with sonar is presented that is designed to tackle these challenges. As one concrete application scenario, the detection and classification of naval mines is presented.

Interestingly, using an UUV poses a new challenge. Due to the sophisticated sensors, the amount of generated data is huge. This is in theory good, since more information means higher confidence in the interpretation. But all the gathered data has a significant impact on the time needed for the post mission analysis afterwards. Since the mine removal task is usually time-critical, there might be not enough time for a full analysis. Therefore, it is advantageous if the vehicle not only presents the raw data after the mission, but also a list of potential mine candidates.

Following this line of thought, if the UUV is already capable of automatically recognizing potential mines, the next logical step is giving the UUV autonomous capabilities to get an Autonomous Underwater Vehicle (AUV) that is able to adjust the mission path dynamically. This could potentially lead to multi robot systems that efficiently gather the best possible data for high confidence decisions eventually done by a human operator [87]. Note that the appearance of objects on sonar images is highly aspect angle-dependent. Getting multiple views from different directions of the same position on the seafloor can hence substantially improve the recognition rate of any classification algorithm [82],[83].

An essential prerequisite for this multi image and aspect approach is – besides good global localisation – a reliable recognition. If the AUV finds a potential mine at every position on the seafloor, the benefits vanish; or worse, if a mine is missed, the consequences can be catastrophic. Therefore, to fulfil the need for a reliable recognition on sonar images, approaches to reach this goal are investigated in this thesis.

2.1.1 Synthetic aperture sonar images

In this section a very brief overview of the sonar principle is given to provide a basic understanding why the sonar images used in the experiments and the objects that

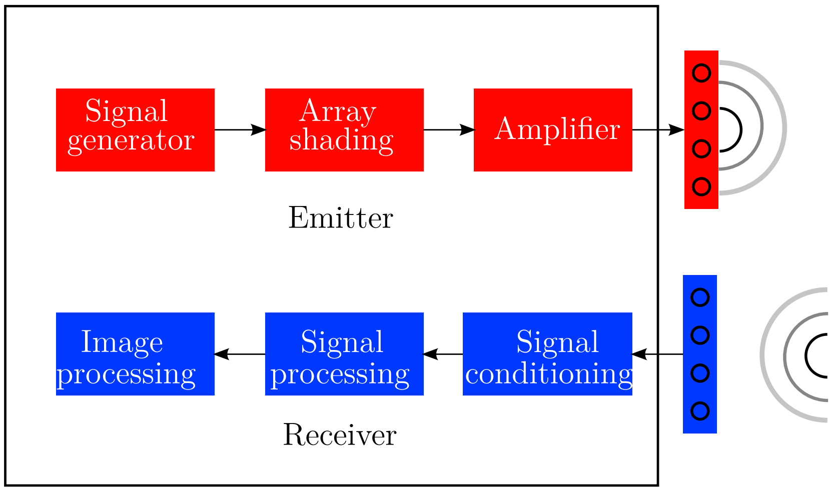


Figure 2.1 – Schematic representation of the basic components of a sonar system. The emitter and receiver arrays are usually within one housing.

appear on them look the way they do. For a more thoroughly introduction refer to [1], [6] and [76].

Without SOund NAvigation and Ranging systems, commonly referred to by its acronym *sonar* systems, imaging of most of the underwater domain would not be feasible. Usually, when talking about images, it is implicitly assumed that the imaging is done by an optical imaging system operating in the visible light spectrum. Unfortunately, the absorption of light in water is much higher than in air. Small floating particles in the water only add to the light attenuation by scattering the light so that the image appears blurred. Thus, objects farther away than a few meters can only be seen in very clear water and even then, the range is quite limited. Alternatively, sound propagates far better in water. Therefore, the basic idea is to interpret reflected sound instead of reflected light to generate an image. There are two different basic categories of sonar systems. If the sonar system generates its own sound waves to insonify the target area, it is called an active sonar, otherwise passive sonar. Since the underlying assumption of this work is that the objects of interest are man-made objects without self-noise, only active sonar systems are considered. Figure 2.1 illustrates the basic structure of an active sonar system.

Even for active sonars there are again different types. All example sonar images used for the experiments in this thesis originate from a side scan sonar mounted on an AUV. Side scan sonar send the sound pulses sideways normal to the vehicle motion. In order to direct the beam in a certain direction, an array of hydrophones and the so called beam forming is used. Figure 2.2 shows the resulting beam angles and how slice per slice the sea floor is insonified. Figure 2.3 shows one slice and the response in more detail. After the acoustical pulse is sent, everything in line of sight and with the same distance to the sonar system is hit at the same time.

The amount of reflected sound depends on the acoustic reflectivity of the object. The acoustic reflectivity is a complex composition of the aspect angle, the geometrical

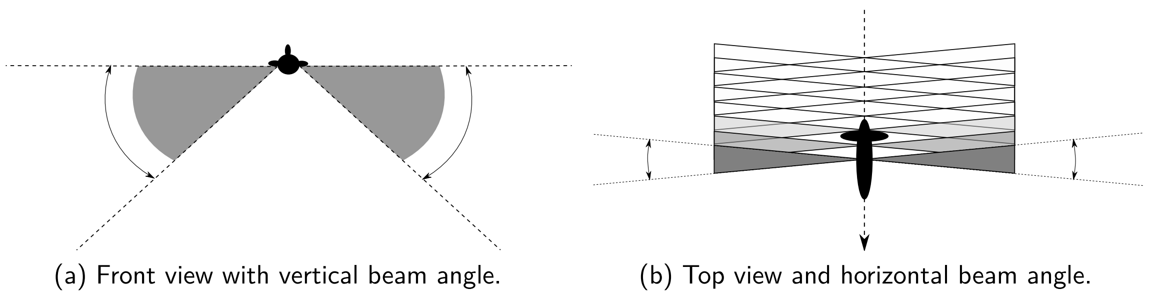

(a) Front view with vertical beam angle. (b) Top view and horizontal beam angle.

Figure 2.2 – Insonified area of a side scan sonar on an AUV.

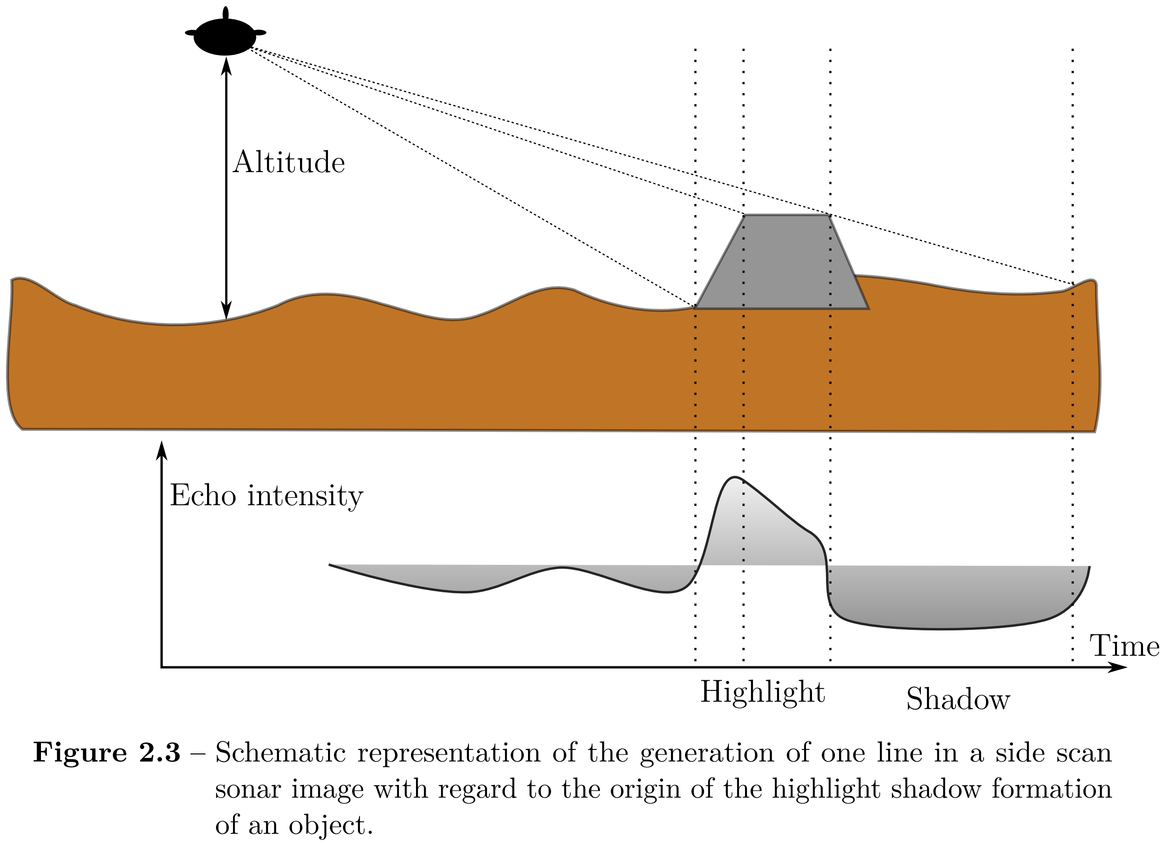


Figure 2.3 – Schematic representation of the generation of one line in a side scan sonar image with regard to the origin of the highlight shadow formation of an object.

shape of the object, the material, the internal structure etc. Usually, the acoustic reflectivity of an object is higher than of the sea floor. Hence, an object can be seen as high intensity values in the sonar image. This area is referred to as highlight of the object. Behind the highlight area, the so called shadow follows. In this area, the echo response is low in relation to the sea floor reverberation.

It is assumed that the object lies prominently on the sea floor or is only partly buried. Thus, the line of sight from the sonar system to the area directly behind the object is blocked by the object itself. Consequently, the echo response is low for this area, since the sound beam was already reflected beforehand by the object.

Extracting the shape of this shadow area in conjunction with the shape of the highlight is an important task to deduce the geometry of the object, which in turn means identifying it. Examples of real sonar images showing mock-up man-made objects can be seen in Figure 2.4. This already gives an impression on the challenges of image processing with sonar images. The transition between the different areas are fuzzy and several non-target echoes, generally referred to as clutter, can be seen.

Not shown here are difficulties posed by the seafloor. There can be other objects, e.g., rocks, that effectively *overshadow* the object of interest. Not unusual are also so called sand ripples where an object can hide in the shadow of a ripple. Furthermore, the echo from the seafloor can be so low that there is no contrast to the object shadow, so that the shadow vanishes.

There are, moreover, artefacts due to the signal processing itself. Those can for example lead to double images of objects.

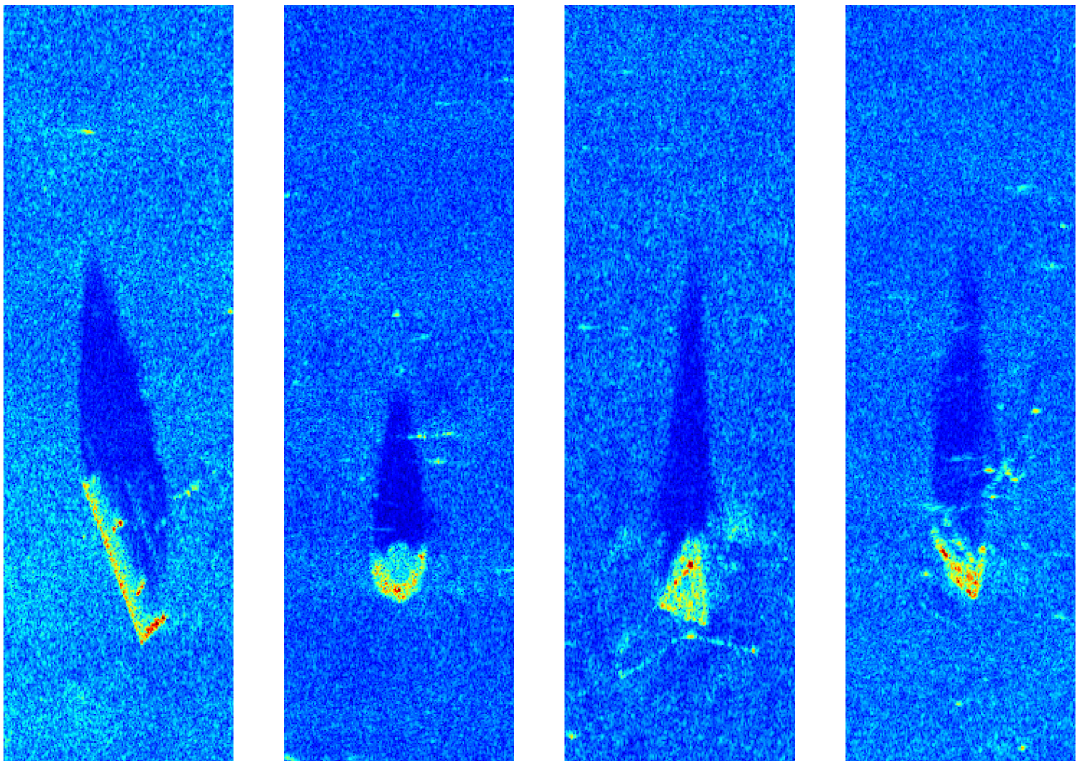
All the previously described effects make the object recognition on sonar images very challenging. In this thesis these effects are generally summarized as *noise*. The idea is to deduce robust algorithms that can perform successfully independent of the different sources of noise. Therefore, the different noise types are usually not differentiated in this thesis.

The image resolution is, independent from the presence of noise, a very important factor for the quality of an image and, consequently, for the potential for a successful object recognition. The resolution of a sonar image depends on the acoustic pulse, the frequency and the antenna. The radial resolution R_r is given by

$$R_r = \frac{c}{2B}, \tag{2.1}$$

where B is the bandwidth of the acoustic pulse and c is the speed of sound in water. It is usually between $c = 1405\,\mathrm{m\,s^{-1}}$ for cold water with low salinity and $c = 1550\,\mathrm{m\,s^{-1}}$ for warm water with high salinity. Obviously, the larger the bandwidth, the smaller and therefore better the resolution. However, it is important to note that with a higher frequency, the absorption of water rises and therefore the maximum range of the sonar decreases. Typically, resolutions in radial direction are between 15 mm and 50 mm. The azimuthal resolution depends on the used frequency with wavelength λ, the aperture of the antenna L and the sonar to object distance r.

$$R_a(r) = \frac{\lambda}{L} r \tag{2.2}$$



(a) Cylindric object (b) Truncated cone (c) Wedge shaped object (d) Stone

Figure 2.4 – Example sonar images of 526 pixel by 160 pixel showing 8 m by 4 m patches showing three mock-up man-made objects and a stone representing the four object categories investigated in this thesis. The same image dimensions apply throughout the thesis, if not stated otherwise. The AUV with the side scan sonar was located several meters downwards from these objects with respect to the image coordinates. The left-right-direction is usually referred as along track, the up-down-direction as across track.

As previously said, the potential for object recognition depends on the resolution. Therefore according to (2.2) the AUV has to be as close as possible to the object to get the highest resolution. At the same time, it is not feasible to let the AUV inspect everything from only a few metres. Fortunately, a technique called synthetic aperture sonar (SAS) eliminates this range dependence via a coherent addition of consecutive echoes. Looking again at Figure 2.2b, it can be seen that the current insonified area overlaps with previous ones. Hence, it is possible to not only use the information about the seafloor perpendicular to the sonar gained from the current echo, but also from previous ones. For this approach, a few requirements have to be fulfilled. First, consecutive pings can only be at most $0.5L$ apart. This has obviously consequences for the maximum speed of the sonar vehicle. Moreover, the position of the vehicle at each point has to be known quite accurately. If the navigation is off by more than $\lambda/16$ the SAS processing gives inconclusive results. If these presumptions are fulfilled, L can be substituted with the synthetic L' with

$$L'(r) = \frac{\lambda}{L^*} r, \tag{2.3}$$

where L^* is related to the aperture L by $L = nL^*$ and n is the number of elements forming the uniform linear array antenna. Applied to (2.2) and accounting for transmitter motion, this gives the range independent azimuthal resolution

$$R_a(r) = \frac{L^*}{2}. \tag{2.4}$$

With this range independent sonar image generation objects for example 100 m from the AUV are surveyed with the same resolution as objects at minimum range. For the object recognition, this has the advantage that the appearance of an object is less range dependent. Because of this advantage, only SAS images are considered.

2.1.2 Automatic target recognition

Previously, a brief introduction was given in how SAS images are generated. The focus of this work, however, is the image processing after the image generation. After receiving a complex valued SAS image, one or several preprocessing steps are done on the image. Figure 2.5 shows the schematic structure of an Automatic Target Recognition (ATR) system. For this work, we only consider real valued images, therefore, the preprocessing step consists at least of transforming the complex valued pixel intensities to real valued pixel intensities by calculating the absolute value. After the preprocessing of the image, typically, the automated recognition of objects on SAS images is realised in two phases. Like in almost all object recognition approaches, the first phase is a coarse detection process during which the raw data is searched for potential objects of interest. The idea is that empty image regions that only show seafloor can be discarded with little effort. The purpose of this phase is to reduce the vast amount of raw data to substantially less points of interest in a fast manner while not missing any actual targets [71], [85], [33]. The surveyed area can typically encompass several square kilometres. Hence, a low false alarm rate is crucial. In Chapter 3 an efficient template-based approach is presented.

The classification of whether it is a man-made object of interest or not, as well as the recognition, i.e., the classification which type of object it is, subsequently happens in a second phase on the images with the object candidates that passed the detection. This is done by extracting features that describe the image and using machine learning algorithms to classify them accordingly. Chapter 4 will address the feature extraction and subsequent classification. The additional information about the class label can help with the decision on how to handle the object. Furthermore, knowing the type plus a pose estimation allows to find the best aspect angle for an additional sonar image that can confirm or reject the detection and classification.

In this thesis a third step is introduced in addition to the two standard steps of detection and classification. After the classification, the class label is checked for plausibility in a top-down approach. This will be covered in Chapter 5.

2.2 Optical underwater cameras

In land or air based robotic applications it is common to use the visual data from a video camera to identify objects of interest. Unfortunately, being underwater poses certain difficulties for vision based systems. First of all, the image is distorted due to the refraction. The light is bend when passing from water to glass and from glass to air. A calibration has to be done first to determine the camera and housing specific parameters to correct for this effect [51]. Moreover, the attenuation rate of light is much higher in water than in air. In the immediate vicinity this leads to a colour shift and a haze effect for which correction algorithms are investigated [50]. But there is not only a colour shift, the much more severe effect of the high attenuation is the very limited visual range. Particles in the water column can even amplify this range deterioration. The scattering of the light due to those particles is the reason why laser or structured light based methods to generate 3D point clouds are usually not feasible.

But albeit its drawbacks, it is sometimes necessary or even beneficial to use a video camera underwater. Besides the lower costs, camera or video images reveal much clearer and easier to interpret details of the surrounding environment. The image quality for a further processing are superior to other imaging methods, as long as the object is in the very near field with reasonably clear water. Example applications where these details are needed are for example species counting of fish or other aquatic life forms [10],[73], and inspection and monitoring tasks [56].

Since underwater cameras have also their place in underwater robotics, some of the presented methods in this thesis are also evaluated on images from a video camera mounted on an ROV. This is done whenever the method is not specifically tailored to characteristics of SAS images. The data for those experiments was acquired with the ROV shown in Figure 2.6 during the EU-funded project *Dexterous ROV: effective dexterous ROV operations in presence of communication latencies* or short DexROV

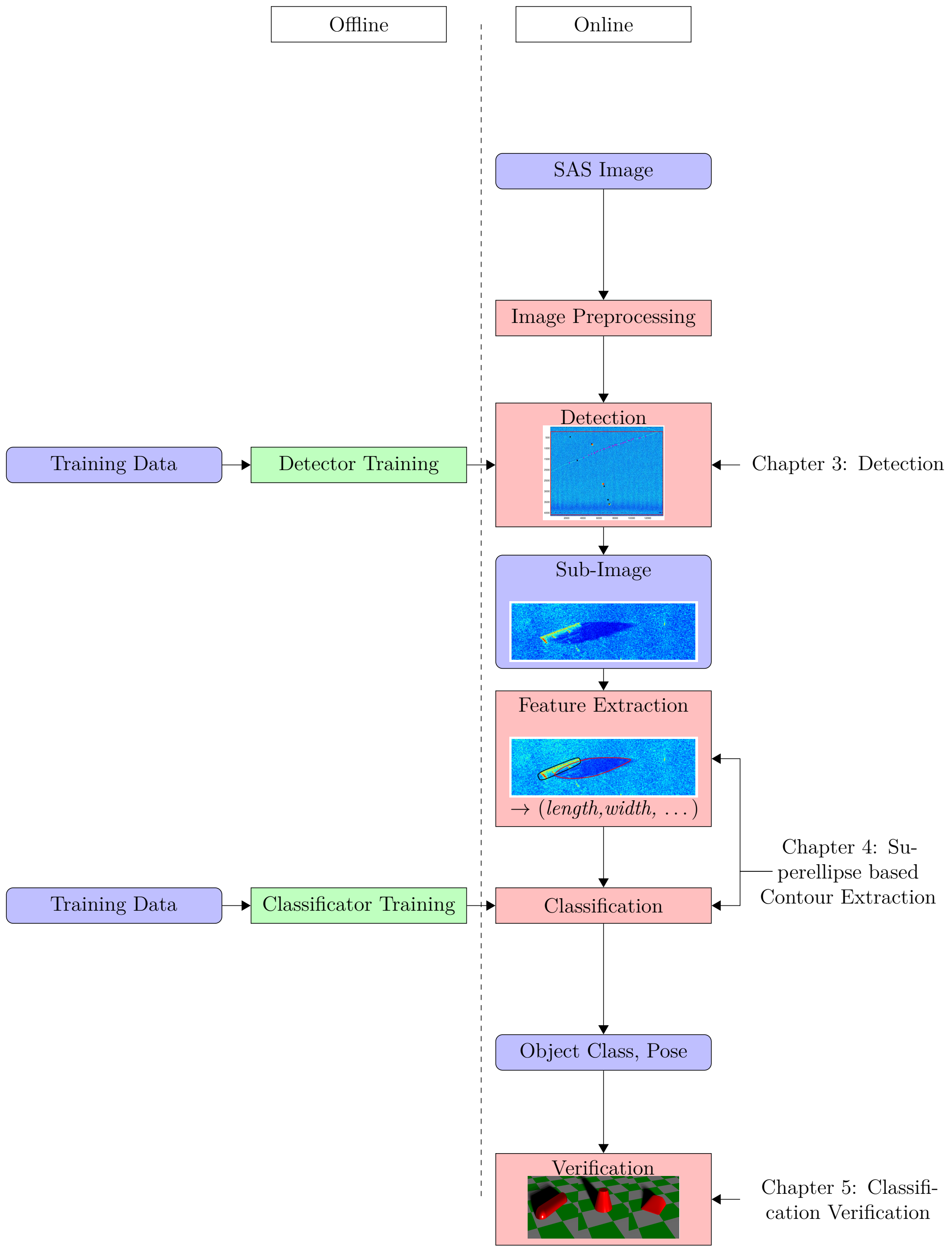


Figure 2.5 – Flowchart of the ATR process. An SAS image is given as input. After an image preprocessing stage an efficient screening method is applied to reduce the image data to Regions Of Interest (ROIs). This detector used in this stage is trained offline before the mission. The resulting sub-images show objects of interest. Descriptive features of the object are extracted and used for classification. Again, the classifier is trained offline. Finally, the recognized object class and pose are verified via a top-down plausibility check based on simulation.

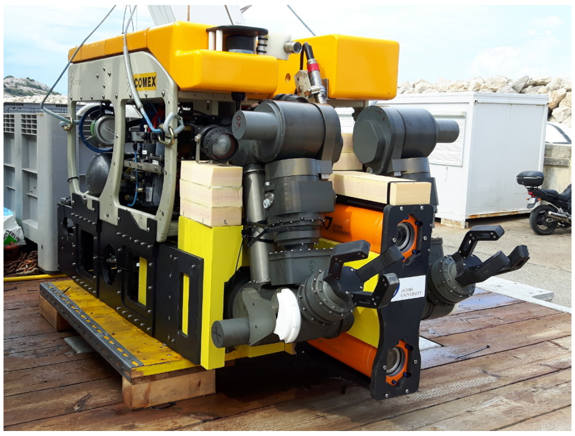


Figure 2.6 – The visual data presented in this thesis was recorded with the cameras that can be seen on the depicted ROV in the orange housing.

[25]. The goal of this project was as follows.

When a ROV is employed the handling and operation of the robotic platform has to be done by specially trained personnel. Assuming it is an off-shore operation the operator or operators have to be brought along on the ship. The intense working period once the research vessel is at its destination is preceded and followed by long stretches of idle time.

The DexROV idea is to instead of having the operators on the research ship where usually the number of persons is very restricted, to have a delocalized onshore operations centre. From this onshore operations centre the ROV shall be operated via satellite link, i.e., the onshore operations centre and the research vessel are connected via satellite link, and research vessel and ROV are connected via cable. This approach has the promising benefit of being a more cost and time effective solution. The operators can work in more reasonable shifts without idle time and the free slot on the research vessel can go to an additional scientist, mechanic or similar.

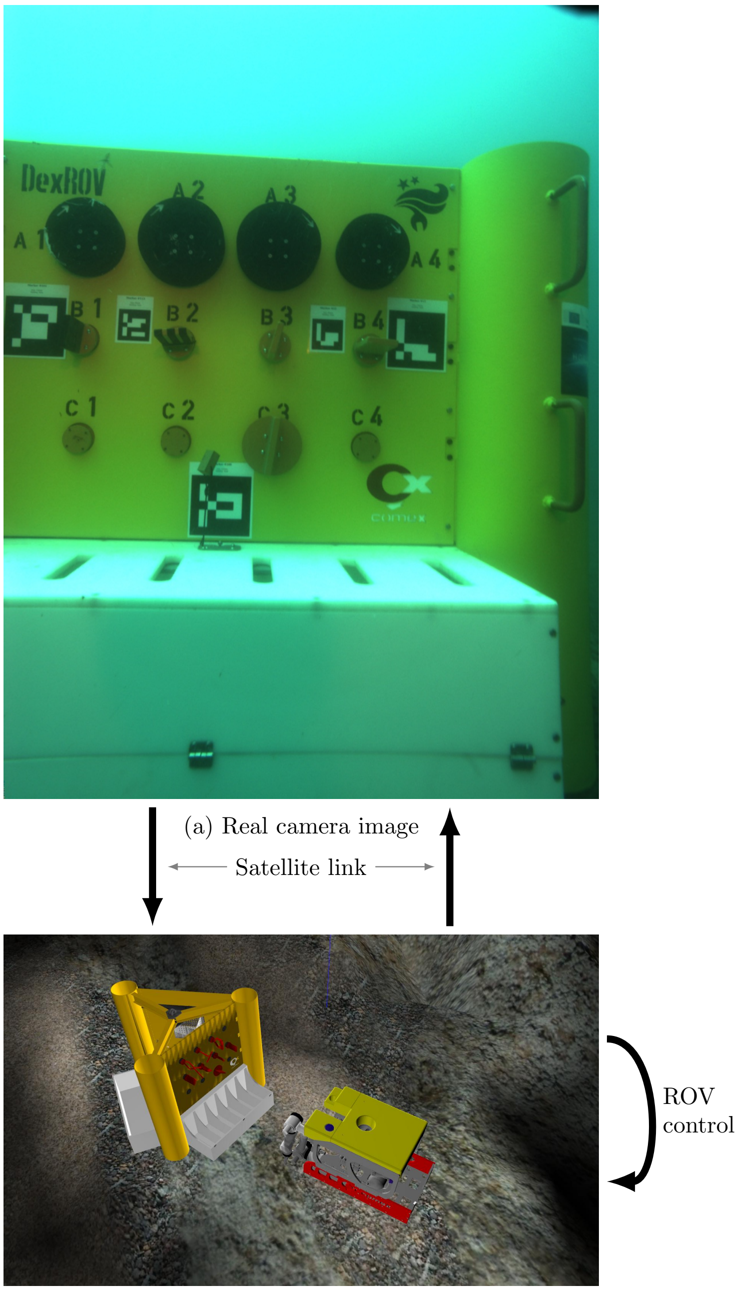
Besides its advantages, this proposal also poses numerous challenges. One of them is the input lag between command, action and (visual) confirmation of said action due to the satellite link. It is for example intuitive to press *move forward* until one sees via the onboard cameras that the ROV has reached its destination. But since the data stream would show a delayed state the ROV maybe already reached its destination while still getting the *move forward* command. Building up a model of the environment around the ROV and controlling the local simulation could mitigate the perceived input lag on the receiving site in the control centre. The real ROV would then mimic the behaviour of the simulated ROV. If the operator wants to move for example the robot arm forward, he or she should see the effect immediately even if it is just a simulated prediction of what will happen a split second later in reality. But in order to accurately simulate the vicinity of the robot, the objects in it and their states they have to be recognized first.

An important element within the DexROV scenario is the perception of a panel and the state of its handles. The panel pose and the handle states are needed to represent the environment in the simulation as outlined before. Moreover, the ROV needs this information to properly grasp the handles. Figure 2.7 illustrates the workflow. Figure 2.7(a) shows a panel with different handles in different states, i.e., rotations. To properly represent the panel in the simulation shown in Figure 2.7(b) these states have to be inferred from the perceptional data. The panel itself is a mock-up that can be seen as a generic representation of typical ROV tasks in underwater industry applications. For example, in the offshore oil and gas industry panels on the seafloor similar to the one shown in Figure 2.7(a) are used to control and maintain the oil and gas flow.

Standard vision algorithms used to extract objects in air are likely negatively affected by the effects present on underwater images described above, i.e., high light attenuation, colour shift, and distortion due to refraction. A common approach to mitigate these effects is preprocessing the image so that it appears as if it were made in air. Afterwards standard vision-based methods can be applied. For example, the authors of [21] use colour correction to enhance the image of an underwater pipeline. Afterwards a neural network is used to detect edges and eventually the position of the pipeline itself.

Again, colour correction is the first step used in the work of [43]. The presented problem is that of underwater tracking of objects. Thanks to the correction the use of a colour based tracking algorithm was possible.

The methods presented in this thesis will instead focus on employing the fact that the shapes of the man-made objects of interest are usually known a-priori. With this assumption it is possible to mitigate some of the adversarial underwater effects. Extracting the contour of an object becomes much more robust if only a subset of possible contours is explored. In the experiment section of Chapter 4 a method to do this is shown. The visual data used in the experiments was collected during a sea trial campaign for the DexROV project in June 2017. Every day the ship depicted in Figure 2.8 would be used to ferry the ROV and the mock-up panel to different trial sites in the Mediterranean Sea near Marseille, France. The data in this thesis is from a trial site at 23 m below the surface.



(a) Real camera image

(b) Physics simulation

Figure 2.7 – Schematic workflow of the DexROV EU-project. The camera image recorded on the ROV is, besides others, used to recognize the panel and handle poses. Only this information is send via satellite link to the delocalized control centre to minimize bandwidth usage. The information is used to mimic the real environment in a physics simulation. The operator controls the ROV in the simulation. The control command are send back to the real ROV via the satellite link. Consequently, the real ROV mimics the simulated ROV.

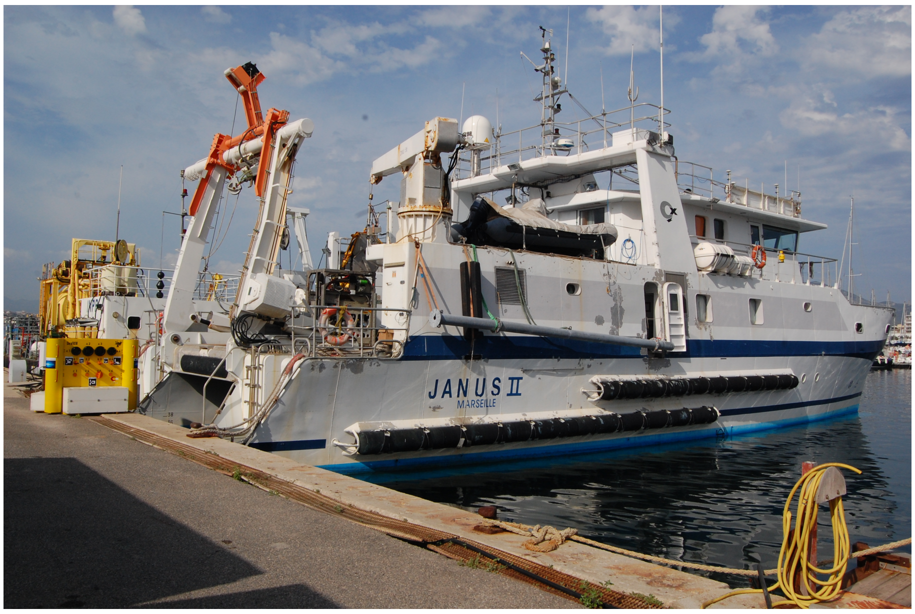


Figure 2.8 – The ship used for deploying and recovering the ROV and mock-up panel during the June 2017 DexROV sea trial. Also visible in the lower left corner is the mock-up panel.

CHAPTER 3

Detection on SAS Images

The overall goal is automatically identifying different objects on underwater images. Since this is no trivial task, complex algorithms are needed, which in general equals time-consuming algorithms. But long processing times pose a problem in real time applications. As stated in the previous chapter for the case of SAS images, the amount of data the AUV gathers during a survey mission can be huge. If we assume the AUV has a speed of $2\,\mathrm{m\,s^{-1}}$ and the SAS covers approximately 150 m to both sides, then a typical coverage rate is $\geq 600\,\mathrm{m^2\,s^{-1}}$. This translates to roughly 1.2 million pixel $\mathrm{s^{-1}}$, assuming a resolution of 2.5 cm along the track of the AUV and 2 cm across the track of the AUV. Since the goal is online object recognition, i.e., during the mission, the whole ATR processing chain from image generation to the final classification has to be efficient enough to process the incoming SAS data before new data arrives. Applying sophisticated and, therefore, usually time consuming classification algorithms to all pixels in the image is not very efficient knowing the image shows predominantly only empty seafloor. Therefore, a screening algorithm that extracts ROIs becomes mandatory. A fast and reliable localization of ROIs allows the reduction of the vast amount of data provided to the segmentation algorithm.

The authors of [85] or [84] use unsupervised algorithms to solve the detection problem. The unsupervised algorithms essentially search via thresholds for local anomalies that are related to the special highlight shadow structure objects cast on SAS images. By successively using specific domain knowledge they filter out step by step candidate regions. First, the mean local echo strength is compared to the mean local echo strength of the surrounding neighbourhood. The ROI is reduced to every position where the local mean pixel intensity is sufficiently smaller than the mean pixel intensity of the local neighbourhood assuming this indicates the presence of a shadow. Afterwards the size of the individual detected regions is used to discard shadows that are too small. The whole algorithm can be implemented efficiently by using the so called integral image representation. A technique also used in this chapter and explained in detail in Section 3.1.

In contrast to the unsupervised approach but with the same general idea regarding the use of domain knowledge a supervised algorithm is presented in this chapter. The proposed algorithm follows the idea of Viola and Jones [79] who have used template matching in combination with the integral image representation and boosted classifiers. This approach was employed for face-detection and has attracted a lot of attention due to its performance. To tailor the method more towards the application at hand, two new template classes are introduced that on the one hand are more suitable for representing the highlight shadow structure found on SAS images and on the other hand fulfil the requirement of being computable via integral images.

Additionally, the single feature per template approach is expanded by integrating three statistical test values in the framework. Afterwards, it is outlined how these features are used to train a cascade of boosted classifiers. Experiments on real SAS data show that the new templates and features are a valuable addition in the context of object detection on sonar images.

This chapter is organized as follows. In Section 3.1 the theory for the proposed

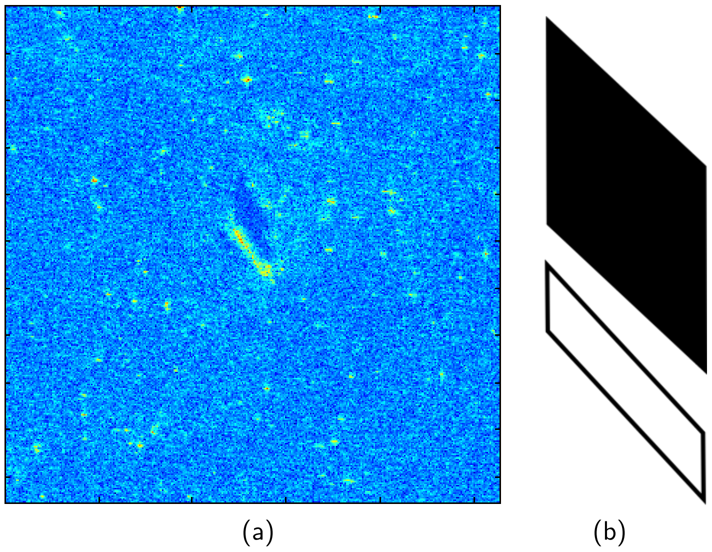


Figure 3.1 – Template matching: (a) Example image for highlight shadow pattern an object causes on a SAS image. The image shows an area 9.5 m high and 12.5 m wide. (b) Example template to detect the object in (a). The black part represents -1's, the framed white part $+1$'s. Every other point has the value 0.

detection method is presented. Beginning with a brief summary of how integral images can be employed to efficiently obtain the correlation between a template and a sonar image. This is followed by the introduction of the two new template classes specifically tailored to the detection of objects on SAS images, the test statistics and an introduction to the cascaded AdaBoost. In Section 3.2 the experiments are made. The chapter concludes with a summary.

3.1 Features and integral images

The foundation of the following proposed detection algorithm is basic template matching. The idea is to have a simple mask or template that looks similar to the object of interest. In the case of SAS, objects cause a characteristic highlight shadow pattern. Figure 3.1(a) shows an example highlight shadow pattern of an cylindric object. To detect this pattern with template matching, a template is needed where the shape of the highlight is approximated with $+1$'s, the shape of the shadow with -1's. Figure 3.1(b) shows an example template one could use after proper scaling. If this template is slid over every pixel position and the bivariate correlation with the image is calculated, a peak should appear wherever the object can be found on the image.

This approach, unfortunately, has some drawbacks. The template has to be quite similar to the object of interest to cause clearly distinguishable peaks in the correlation result. Since the objects on sonar images cast a high variation of highlight and shadow shapes even within one object class, many different templates have to be used. This causes two disadvantages. On the one hand, the matching itself is not computationally efficient for arbitrary shapes, as it is depended on the number of non-zero points of the template. On the other hand, it is not feasible to manually compose the best but small set of templates to get optimal results. Both challenges will be addressed in the following. First, computationally efficient matching will be discussed, then the problem of choosing the correct templates is explained.

3.1.1 Integral images

To combat the computationally inefficiency of naive template matching, integral images can be used. The advantage of the integral image representation of an image is the fact that the correlation feature can be calculated in constant time, independently of the template's size or position. Only four array references and basic arithmetics are needed.

The classical integral image $II(\boldsymbol{z})$, $\boldsymbol{z} = (x, y)^T$ for rectangular features represents in a certain pixel the cumulative sum of the pixel values above and to the left of this pixel in the original image u.

$$II(\boldsymbol{z}) = II(x, y) = \sum_{i=1}^{x} \sum_{j=1}^{y} u(i, j) \tag{3.1}$$

With this representation of the image, the cumulative pixel values within a rectangular area $\mathcal{A}$ can be calculated in constant time as

$$\sum \mathcal{A} = II(\boldsymbol{z}_1) - II(\boldsymbol{z}_2) - II(\boldsymbol{z}_3) + II(\boldsymbol{z}_4). \tag{3.2}$$

This is also illustrated in Figure 3.2. To see why the efficient summation over certain areas is beneficial for the calculation of the correlation between the template and the image, one has to have a closer look at the sample correlation coefficient formula given by

$$\begin{aligned} \zeta_c &= \frac{\sum\limits_{z \in \omega} (u(\boldsymbol{z}) - \mu_{u,\omega})(\tau(\boldsymbol{z}) - \mu_{\tau,\omega})}{\sqrt{\sum\limits_{z \in \omega} (u(\boldsymbol{z}) - \mu_{u,\omega})^2} \sqrt{\sum\limits_{z \in \omega} (\tau(\boldsymbol{z}) - \mu_{\tau,\omega})^2}} \qquad (3.3) \\ &= \frac{\sum\limits_{z \in \omega} u(\boldsymbol{z})\tau(\boldsymbol{z}) - \mu_{u,\omega} \sum\limits_{z \in \omega} \tau(\boldsymbol{z}) - \mu_{\tau,\omega} \sum\limits_{z \in \omega} (u(\boldsymbol{z}) - \mu_{u,\omega})}{\sqrt{\sum\limits_{z \in \omega} (u(\boldsymbol{z})^2 - 2\mu_{u,\omega} u(\boldsymbol{z}) + \mu_{u,\omega}^2)} \sqrt{\sum\limits_{z \in \omega} (\tau(\boldsymbol{z})^2 - 2\tau(\boldsymbol{z})\mu_{\tau,\omega} + \mu_{\tau,\omega}^2)}}, \qquad (3.4) \end{aligned}$$

where $\omega \subseteq \Omega$ is all pixel positions for which the template τ is non-zero, u is the image, $\mu_{\tau,\omega}$ is the average non-zero template value and $\mu_{u,\omega}$ is the average image value for all pixels below a non-zero pixel of the template. Please note that this is the correlation for one specific template position $\tilde{\boldsymbol{z}}$. So ζ_c is, in fact, $\zeta_c(\tilde{\boldsymbol{z}})$, $\tau(\boldsymbol{z}) = \tau_{\tilde{\boldsymbol{z}}}(\boldsymbol{z})$, and $\omega = \omega_{\tilde{\boldsymbol{z}}}$.

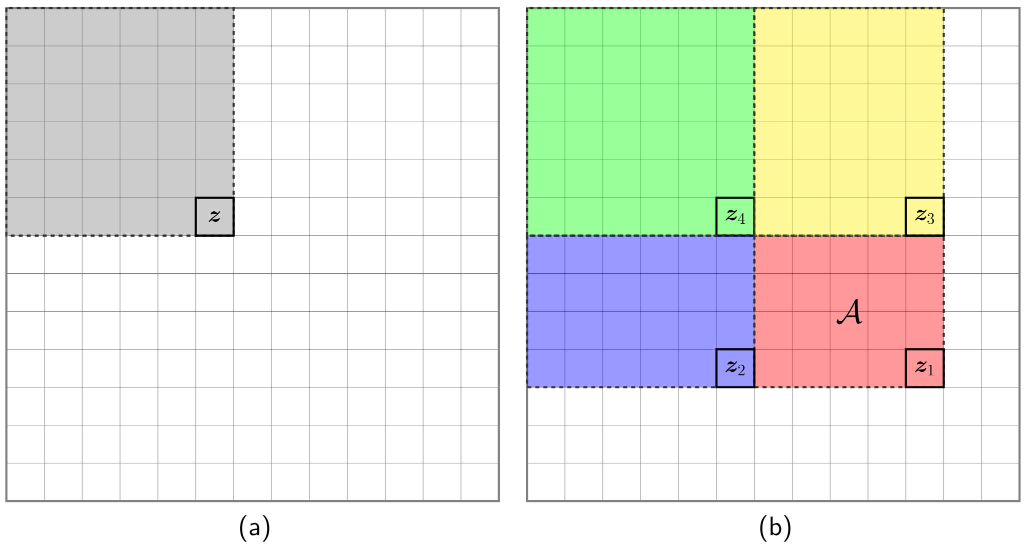


Figure 3.2 – Illustration of integral images to efficiently calculate the sum of all pixel values in a rectangular area. (a) The integral image is computed once as the cumulative sum of all pixel values in the original image to the left and above for each position $\boldsymbol{z}$ in the image including the pixel value at position $\boldsymbol{z}$. (b) The sum of all pixel values in an arbitrary rectangle $\mathcal{A}$ can be efficiently calculated over the four edge positions, the integral image and (3.2).

To take advantage of integral images (3.4) can be reformulated. Since the template is defined as

$$\tau(\boldsymbol{z}) = \begin{cases} +1 & \text{for the highlight part,} \\ -1 & \text{for the shadow part,} \end{cases} \tag{3.5}$$

and assuming there are $N_{\mathcal{H}}$ highlight pixel and $N_{\mathcal{S}}$ shadow pixel on the template with $N_{\mathcal{H}} + N_{\mathcal{S}} = N_\omega$ the formulation in (3.4) can be simplified to

$$\zeta_c = \frac{1}{2} \frac{\sum \mathcal{H} - \sum \mathcal{S} - (N_{\mathcal{H}} - N_{\mathcal{S}})\mu_{u,\omega}}{\sqrt{\mu_{u^2,\omega} - \mu_{u,\omega}^2}\sqrt{N_{\mathcal{H}} N_{\mathcal{S}}}} \tag{3.6}$$

where $\sum \mathcal{H}$ and $\sum \mathcal{S}$ are the sum over all pixel values within the highlight and the shadow, respectively. Please note that $\mu_{u^2,\omega}$ is the average value when each pixel value is squared first. Alternatively, one can write

$$\zeta_c = \frac{1}{2} \frac{\sum \mathcal{H} - \sum \mathcal{S} - \frac{N_{\mathcal{H}} - N_{\mathcal{S}}}{N_{\mathcal{H}} + N_{\mathcal{S}}} \left(\sum \mathcal{H} + \sum \mathcal{S}\right)}{\sqrt{\sum \mathcal{H}^2 + \sum \mathcal{S}^2 - \frac{1}{N_{\mathcal{H}} + N_{\mathcal{S}}} \left(\sum \mathcal{H} + \sum \mathcal{S}\right)^2} \sqrt{\frac{N_{\mathcal{H}} N_{\mathcal{S}}}{N_{\mathcal{H}} + N_{\mathcal{S}}}}}, \tag{3.7}$$

where $\sum \mathcal{H}^2$ and $\sum \mathcal{S}^2$ are the sum over all squared pixel values within the highlight and the shadow, respectively. In this alternative formulation it becomes apparent

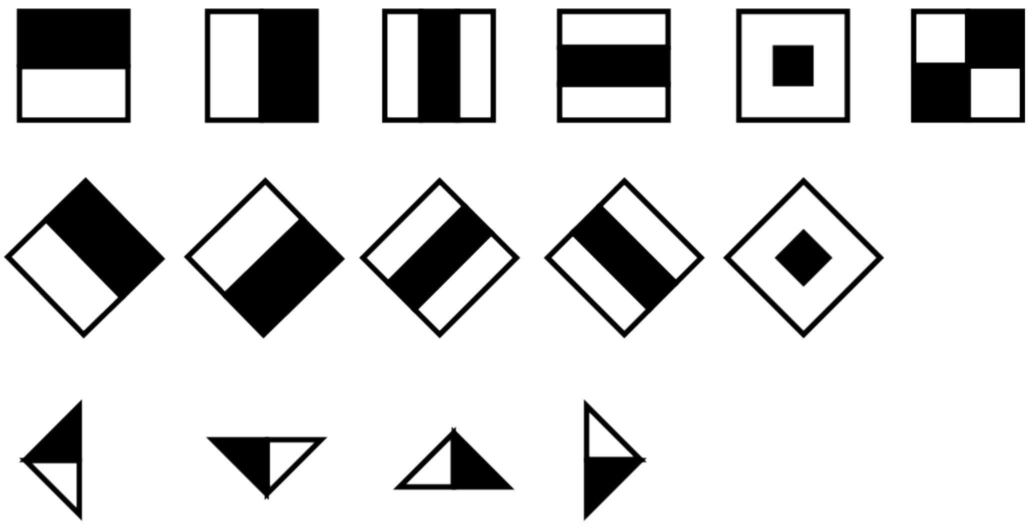


Figure 3.3 – Templates used for face detection. Within the white and the black area every point has the value +1 and −1, respectively.

why the efficient computation of sums of pixel values in certain areas is an important benefit. The individual values at a certain position are not needed as long as the sum over the respective area is known.

Interestingly, at no point an assumption was made of what shape those certain areas have to be to go from (3.3) to (3.6). Up to now, it is only limited to rectangular shapes because only for those shapes integral images were introduced. But, besides the integral image representation for upright-rectangles, there are image representations for rotated-rectangles and four kinds of triangles that can be computed. The rapid summation over the pixels within the additional basic shapes that can be constructed out of those enables the calculation of correlations for templates not only consisting of upright rectangles.

Figure 3.3 shows the different templates that were used in the field of face detection [79], [48], [63]. Recalling that the template should look as similar as possible to the object of interest and comparing the templates in Figure 3.3 with real SAS images, as for example Figure 2.4, one can see that the templates do not fit perfectly. One use case of integral images in the context of detection on SAS images can be found in [46]. In contrast to the aforementioned [79], [48] and [63], a gap between the highlight and shadow is introduced and different types of basic shapes are mixed to derive features that are more suitable for object detection on SAS images. Figure 3.4 shows these templates. Comparing them again with the SAS image in Figure 2.4, one can observe that some templates shown in Figure 3.4 are better suited to roughly approximate the highlight shadow formation of Man-Made Objects (MMOs). The question yet to be clarified is, whether there are any templates that would fit even better and can still be calculated as efficiently as the templates introduced before.

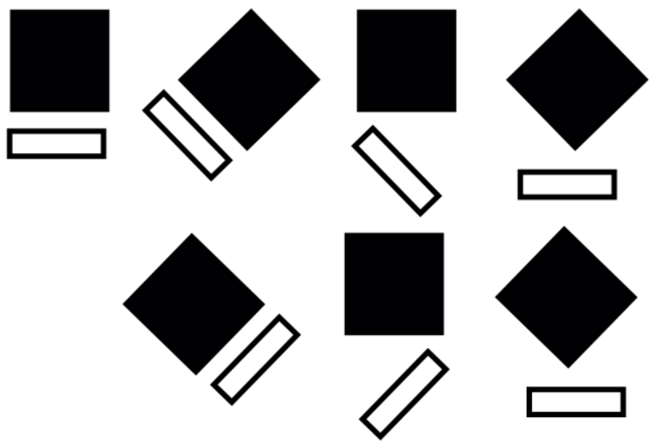


Figure 3.4 – Seven templates used in [46] for object detection on SAS images. Within the white area every point has the value +1, within the black area every point has the value −1.

3.1.2 Parallelogram and upright-triangle templates

Just as in the example SAS image in Figure 2.4(a), it can not be assumed that the orientation of a cylindrical object is parallel to the along track direction. Therefore, instead of having rectangular shapes, the highlight shadow formation looks more like a parallelogram. Hence, parallelogram templates might be a valuable addition to the existing templates for object detection on SAS images shown in Figure 3.4.

Figure 3.5 shows illustrates the need for a parallelogram template with an example. Different instances of the rectangular and parallelogram templates are generated. The instances differ in width, length of the highlight part, length of the shadow part, and gap between highlight and shadow. Each instance is correlated with the example image at every pixel position. Afterwards, the rectangular and parallelogram instances are ordered separately with respect to the maximum achieved correlation.

Figure 3.5(a) and Figure 3.5(b) show the best rectangular and parallelogram instances, respectively. It is notable that especially the shadow part of the rectangular template seems to be completely wrong. But recalling that the correlation is done with a monochrome shadow part with no variance, a high correlation is only reached when the respective part of the image is also approximately homogeneous. One can easily see that there is no position were the shadow rectangle covers only the shadow. This trade-off between matching pixel values and low variance can lead to less obvious solutions as in this case with the shadow. The parallelogram template, however, approximates the shape of the object much better. Therefore, it is more likely to visually match the shadow and highlight.

Even more importantly, Figure 3.5(c) and Figure 3.5(d) show the correlation values for the two templates in Figure 3.5(a) and Figure 3.5(b). It can be observed that the maximum correlation value is higher for the parallelogram, i.e., it is better distinguishable from the background. Furthermore, the peak is also much better localized. This means the position of the object of interest can be estimated more precisely. Both are obvious advantages regarding the automated detection.

The important part about rectangular templates is the capability to calculate the sum within rectangles very efficiently due to integral images. In order to make this new parallelogram template actually useable, a similar sum calculation method has to be derived.

There are already six integral image representations in total: the classical rectangular integral image II, the integral image for rotated rectangles RII and the four integral images for triangles TII_{NW}, TII_{SW}, TII_{NE} and TII_{SE}. The indices of the triangular integral images indicate the direction in which the right-angled peak points as Figure 3.6(a) illustrates for TII_{NE}. Using the latter, a parallelogram area as shown as the red area $\mathcal{A}$ in Figure 3.6(b) can be calculated rapidly.

$$\sum \mathcal{A} = TII_{NE}(\boldsymbol{z}_1) - TII_{NE}(\boldsymbol{z}_2) - TII_{NE}(\boldsymbol{z}_3) + TII_{NE}(\boldsymbol{z}_4). \tag{3.8}$$

For the mirrored version of the parallelogram, the sum is analogue except that this time TII_{NW} is required instead of TII_{NE}.

The second shape is an upright triangle. Figure 3.6(c) shows how this shape can be efficiently computed. This time several different integral images are needed. At the pixel denoted with $\boldsymbol{z}_1$ RII has to be used. The value $RII(\boldsymbol{z}_1)$ is the sum of all pixels in the triangle above the position $\boldsymbol{z}_1$ with its peak in $\boldsymbol{z}_1$. In $\boldsymbol{z}_2$ and $\boldsymbol{z}_4$ once again the triangular integral images are used, for $\boldsymbol{z}_2$ TII_{NW} and for $\boldsymbol{z}_4$ TII_{SW}. For the point $\boldsymbol{z}_3$ the classical rectangular integral image representation II is exploited. Overall the sum of all pixel values is given by

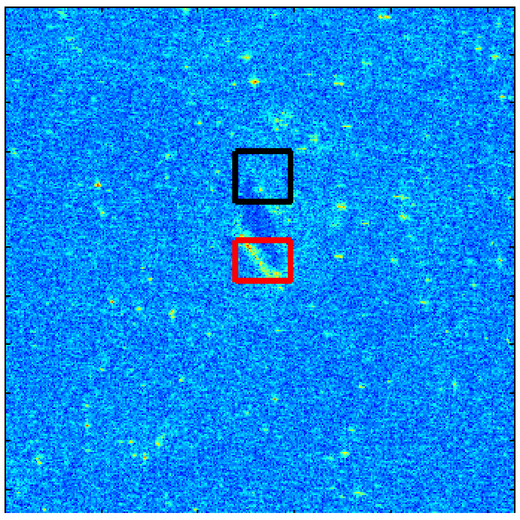
$$\sum \mathcal{A} = RII(\boldsymbol{z}_1) - TII_{NW}(\boldsymbol{z}_2) - II(\boldsymbol{z}_3) + TII_{SW}(\boldsymbol{z}_4). \tag{3.9}$$

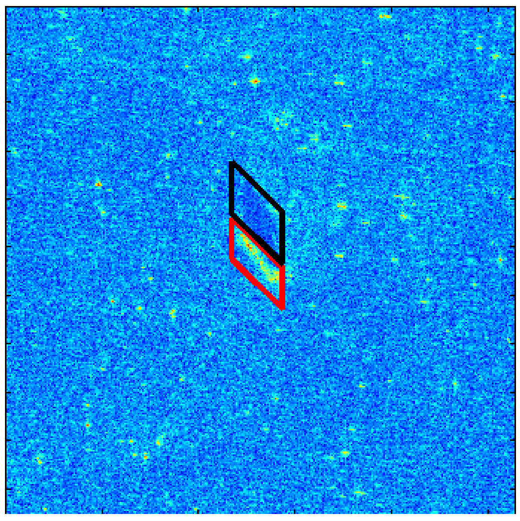
Figure 3.6(d) illustrates the new templates that can be formed using these two new shapes.

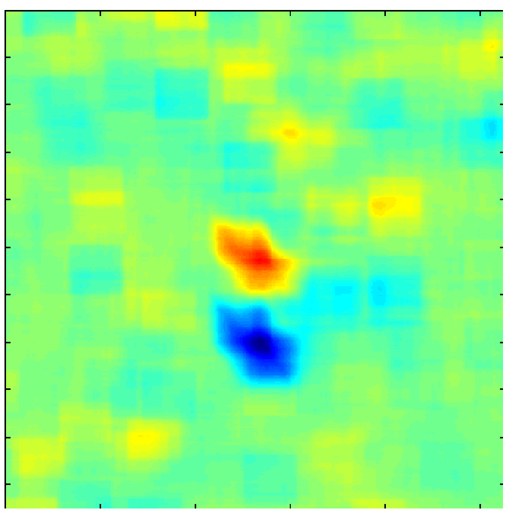
Combining them with the former templates which can be seen in Figure 3.4 means ten template classes can be used for the template matching. By varying the geometrical extend of the individual parts of the template, a vast feature pool can be created. Up to now, only the correlation between these templates and a sonar image is used as a feature for the detection of MMOs. In the next section, additional alternatives to the correlation will be introduced.

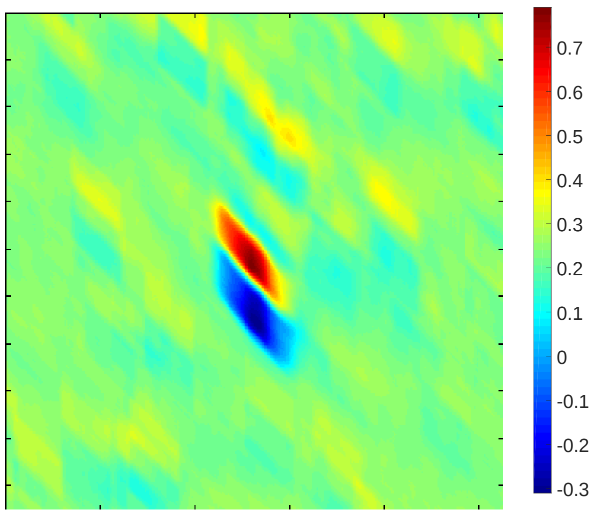
3.1.3 Test statistics as additional features

Besides the correlation between a template and an image, which effectively calculates how similar the two are, other statistically motivated features are conceivable. One can imagine that if the template is above just background, then the mean values below the highlight part and the shadow part, respectively, should be similar. If, however, the template covers the highlight shadow structure of an object, then the mean values will probably be quite different. The same might be true for the variance.


(a) Rectangular template

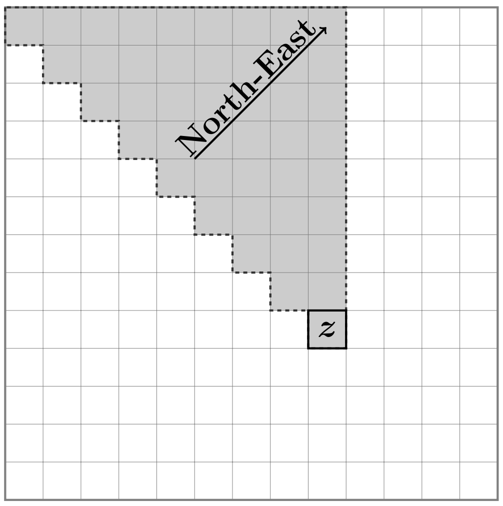

(b) Paralelogram template


(c) Rectangular template correlation

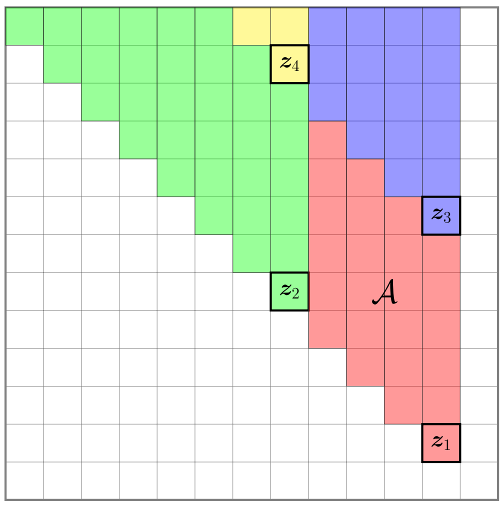


(d) Paralelogram template correlation

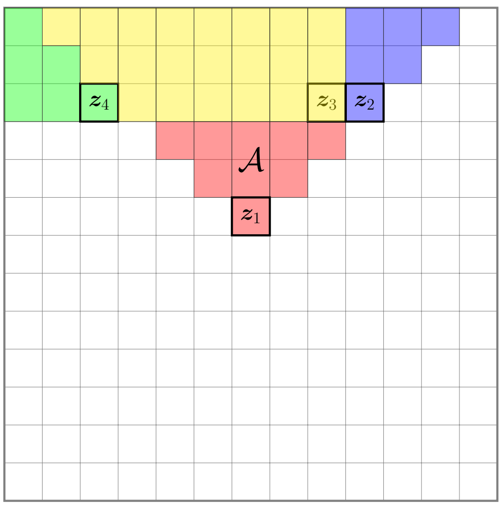
Figure 3.5 – Template instances with the highest correlation per type.



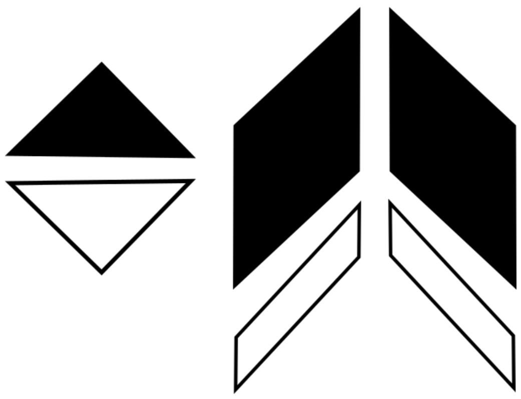
(a) North-East Integral Image



(b) Parallelogram



(c) Triangle



(d) New templates using triangles and parallelograms

Figure 3.6 – Example for a triangular integral image, the efficient calculation of the sum of all pixel values within a parallelogram and a triangle, and the new templates.

In statistics, if one samples data from two sources and suspects a certain statistical relationship, then a statistical hypothesis test is used. There are for example the F-Test, the Students's t-Test and the Welch's t-Test [8][3]. Usually, the F-Test is used as a hypothesis test to identify differences in the variance of two normally distributed random variables, whereas the Students's t-Test and the Welch's t-Test indicate differences in the mean value. For all three a test statistic is calculated and then thresholded to accept or discard the hypothesis.

Starting point for the calculation is the corrected sample variance for a given area $\mathcal{X} = \{\mathcal{H}, \mathcal{S}\}$, i.e., the highlight or shadow. It can be computed as

$$\sigma^2_{u,\mathcal{X}} = \frac{N_{\mathcal{X}}}{N_{\mathcal{X}} - 1} \left(\mu_{u^2,\mathcal{X}} - \mu^2_{u,\mathcal{X}} \right), \tag{3.10}$$

where $\mu_{u^2,\mathcal{X}}$ denotes the mean of the squared pixel values in $\mathcal{X}$, $\mu^2_{u,\mathcal{X}}$ the squared mean and $N_{\mathcal{X}}$ is the number of points within the area $\mathcal{X}$. The mean values for u and u^2 within $\mathcal{X}$ can be computed efficiently with the help of the integral image representation. Hence, (3.10) can be efficiently calculated.

Now the earlier mentioned test statistics can be employed. They are given by

$$\zeta_F = \frac{\sigma^2_{u,\mathcal{H}}}{\sigma^2_{u,\mathcal{S}}}, \tag{3.11}$$

$$\zeta_t = \frac{\mu_{u,\mathcal{H}} - \mu_{u,\mathcal{S}}}{\sqrt{\frac{(N_{\mathcal{H}} - 1)\,\sigma^2_{u,\mathcal{H}} + (N_{\mathcal{S}} - 1)\sigma^2_{u,\mathcal{S}}}{N_{\mathcal{H}} + N_{\mathcal{S}} + 2} \left(\frac{1}{N_{\mathcal{H}}} + \frac{1}{N_{\mathcal{S}}} \right)}}, \tag{3.12}$$

$$\zeta_w = \frac{\mu_{u,\mathcal{H}} - \mu_{u,\mathcal{S}}}{\sqrt{\frac{\sigma^2_{u,\mathcal{H}}}{N_{\mathcal{H}}} + \frac{\sigma^2_{u,\mathcal{S}}}{N_{\mathcal{S}}}}}. \tag{3.13}$$

Each of these values can subsequently be used via threshold for a hypothesis test. Those hypothesis test give the binary decision whether for instance the variance in the highlight and shadow part is different with a certain significance. However, this is not exactly the classification question in this detection context. The interest is rather in deciding whether the image shows a man-made object. It is unlikely that the established thresholds for the hypothesis tests are suitable for this decision. Therefore, the calculated continuous values itself are interpreted as yet another feature value that can be used to train a classifier.

Instead of one feature, i.e., the correlation, four features are calculated for every geometrical shape and used for the detection, i.e., the correlation, the F-Test test statistic, the Students's t-Test test statistic, and the Welch's t-Test test statistic. Figure 3.7 shows the pixel-wise values of the three statistical tests if the parallelogram template is applied to the same image as before.

With this addition, there are four features per template and nine template classes. As previously stated, the geometric extend of each template can be modified to

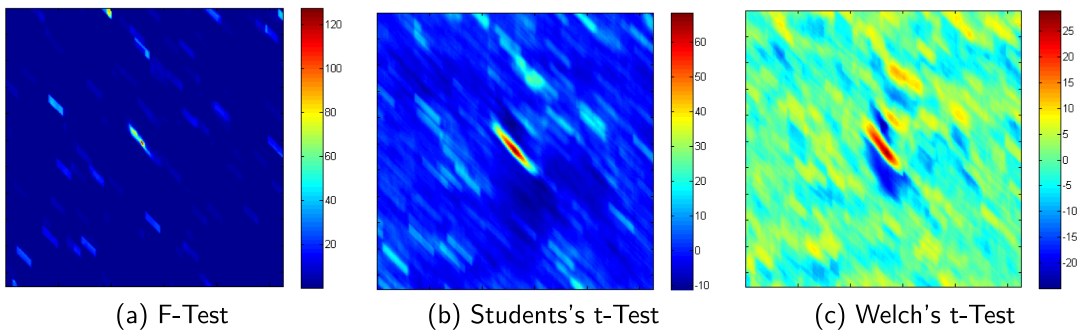


Figure 3.7 – Pixel-wise return values for the statistical tests for the same image and template as in Figure 3.5(b).

generate an arbitrary amount of templates per template class. Even though every single feature can be calculated very fast, the amount of values is too much to be computed every time; a classification is required that identifies and only uses the suitable template features.

3.1.4 Pixel-wise to image-wise feature values

To train the detector one usually divides the whole image in many small for example 5 m by 5 m sub-images. The question now is, whether there is an object of interest in this sub-image or not. The template matching described previously generates pixel-wise feature values. Albeit, there is only one label for the whole sub-image. Even if the whole image has the label *object of interest*, there are pixels in the sub-image that definitely are not object pixel. Therefore, it is necessary to instead of having feature values for every pixel rather having one feature value representing the whole sub-image.

The first idea could be to just use the pixel in the middle of the object. But finding the *exact* pixel that marks the object is in most cases not possible. At the same time, choosing the wrong one could be detrimental for training set quality and, thus, the detector training. In fact, if one calculates all the features and draws their maxima for a sub-image with an object, then those points are only clustered around the middle of the object but not exactly in one place.

To nevertheless solve this problem, one first can notice that even the training data does not reveal the exact position of the object. The label only specifies whether the sub-image contains an object of interest at all. Therefore, only the same level of accuracy can be expected from the detection algorithm trained on the data. Based on this insight a technique to subsample features can be used, which is known as maxpooling in convolutional neural networks. In convolutional neural networks is also a need to reduce the number of pixel-wise features. At the same time it is not necessary to exactly localize the object. The idea, therefore, is to reduce pixel neighbourhoods to single representatives.

Intuitively, using the average value comes to mind. But actually, it is more important whether a feature has a spike in the according neighbourhood rather than having a high value for all pixels on average. Therefore, the maximum is used.

Transferring this technique to the task at hand means defining the sub-image as neighbourhood and the maximum of the pixel-wise features is the representative of the whole sub-image.

3.1.5 AdaBoost

As mentioned in the previous section, every single feature can be calculated rapidly. Nevertheless, it is too time-consuming to compute all features, i.e. using all imaginable values for shadow height, shadow width etc., for a classification. Therefore, the task is now to find a small subset of all features. This subset equipped with suitable thresholds shall be combined to form a classifier.

For this task, the AdaBoost algorithm can be employed [48],[22]. Specifically, the gentle AdaBoost variant is used. It is empirically shown that it outperforms other versions like the Real AdaBoost or LogitBoost with respect to the accuracy [23]. The gentle AdaBoost is a greedy algorithm that takes a set of so called weak classifiers and successively forms a so called strong classifier. At every iteration step, the best-performing weak classifier is weighted according to its performance and added to the final strong classifier. The advantage of this approach is that every individual weak classifier is only required to be better than chance. By combining these tendencies smartly, the resulting strong classifier is able to outperform most monolithic classifiers.

Regarding the classification problem at hand, each feature in conjunction with every possible threshold can be considered as a weak classifier. Due to finite training data, this is simplified by only considering one single threshold between two distinct training examples. So in case the training set consists of N examples, each feature yields $N + 1$ weak classifiers. At the beginning of the AdaBoost algorithm, each training example is weighted equally. Afterwards, at each iteration step the algorithm seeks the best classifier with respect to the weights. This best weak classifier is added to the strong classifier. Subsequently, the weights for the training examples are updated, where the weights for correct classified examples are decreased and for the wrong classified examples are increased. Due to this update, those weak classifiers are favoured in the next iteration step that are able to classify the examples correctly which, in turn, the last best weak classifier could not.

The iteration ends when preassigned values for the true postive rate or the false postive rate are achieved or a certain number of weak classifiers is used. Most times, this number is in any case significantly smaller then the number of all features.

Algorithm 1 shows the pseudo code for the gentle AdaBoost algorithm. It can be distinguished from other AdaBoost variants by the computation of the labels for the weak classifier. The labels y_- and y_+ are not simply ± 1 but rather represent the confidence of the weak classifier. Assuming all weights are equal and the weak classifier performs just slightly better than chance, it is obvious that the numerator

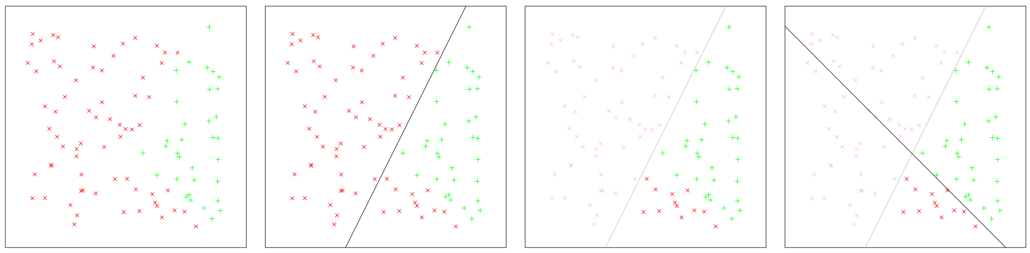


Figure 3.8 – Illustration of the training of a cascaded classifier with a simple linear classification algorithm.

in (3.14) and (3.15) will be near 0. Thus, y_- and y_+ will be near 0 representing the low confidence in the result. Consequently, even correctly classified examples have an impact on the error calculation if the confidence is small.

Another difference to other AdaBoost variants is that the output of the weak classifiers are just added up to form the output of the strong classifier. In other variants the weak output is usually first multiplied with a step factor based on the calculated error of the weak classifier. But exactly this step factor can be a source for instabilities for extremely good or bad weak classifiers whereas with the gentle AdaBoost the update will always be in the range of $[-1, +1]$ [23].

3.1.6 Cascade

The challenge for a successful classifier is to be able to detect nearly every object of interest, while no non-object shall be classified as an object. In general, these goals are contradictory, especially in the case that simple feature-threshold-classifiers are used. Therefore, in most cases only either a high true postive rate or a low false postive rate can be achieved. At this point, a cascade can help.

A cascade is a degenerated decision tree where at each stage the classifier is trained to detect almost all positive training examples, while a certain amount of negative training examples is rejected. The idea is that at each point, a very low false negative rate can be achieved, if the constraints on the false postive rate are relaxed.

Figure 3.8 shows a very simplistic illustration. Assuming the only available classification algorithm is a linear one, it would be futile to expect a good classification performance on the example in Figure 3.8(a). There is no single line that even approximately divides the two classes. But using the cascade approach, this is not necessary. Figure 3.8(b) shows a separation line that has all positive green examples $+$ on one side and at least sorts out a big part of the negative red class $\times$. If now all those negative examples are discarded as is illustrated in Figure 3.8(c) the simple linear classifier can be applied again on the remaining points (see Figure 3.8(d)). Every new sample will be consecutively tested whether it is *on the right side of the current line* and only then it will be checked whether it passes the next stage or not.

Algorithm 1: Training of a gentle AdaBoost classifier

```
**Data:** Training set with $N$ samples and $M$ features as $\boldsymbol{H} \in \mathbb{R}^{N \times M}$
Class labels $\bar{y} \in \{-1, +1\}^N$
Minimum true postive rate per stage $tp$
Maximum false postive rate per stage $fp$
**Result:** Trained gentle AdaBoost classifier

Initialization of weights $\boldsymbol{w} \in \mathbb{R}^N$;
Weights for positive samples to one over twice the number of positive samples;
Weights for negative samples to one over twice the number of negative
samples;

Set current false postive rate $fp_{strong} = 1$;
Set strong classification $y_{strong} = 0$;

while $fp_{strong} > fp$ do
    Set $lowErr = \infty$;
    foreach Feature, i.e., $\boldsymbol{H}(\cdot, j)$ or loop over $j$ do
        foreach Feature value, i.e., $\boldsymbol{H}(i, j)$ or loop over $i$ do
            Set the threshold $thr$ to the current feature value $\boldsymbol{H}(i, j)$;
            Calculate new class labels;

                $$y_- = \frac{\sum_{k:\boldsymbol{H}(k,j)<thr} \boldsymbol{w}(k)y(k)}{\sum_{k:\boldsymbol{H}(k,j)<thr} \boldsymbol{w}(k)} \quad (3.14)$$
                $$y_+ = \frac{\sum_{k:\boldsymbol{H}(k,j)\geq thr} \boldsymbol{w}(k)y(k)}{\sum_{k:\boldsymbol{H}(k,j)\geq thr} \boldsymbol{w}(k)} \quad (3.15)$$

            All samples smaller than the current threshold get the label $y_-$.
            All other samples get the label $y_+$;

                $$\text{For } k = 1, \ldots, N: \; \hat{y}(k) = \begin{cases} y_- & \boldsymbol{H}(k,j) < thr = \boldsymbol{H}(i,j) \\ y_+ & else \end{cases}$$

            Calculate the error as $err = \sum_{k=1}^{N} \boldsymbol{w}(k) \left(\bar{y}(k) - \hat{y}(k)\right)^2$;
            if $err < lowErr$ then
                Remember currently best feature threshold combination;

                    $$\text{For } k = 1, \ldots, N: \; \hat{y}_{best}(k) = \hat{y}(k)$$
            end
        end
    end
    Update weights with $\boldsymbol{w}(k) = \boldsymbol{w}(k) \exp(-\hat{y}_{best}(k)\bar{y}(k)$;
    Normalize weights to 1;
    Update strong classification $y_{strong} = y_{strong} + \hat{y}_{best}$;
    Calculate strong threshold $thr_{strong}$ so that $y_{strong} \geq thr_{strong}$ has a true
    positive rate of at least $tp$;
    Calculate $fp_{strong}$ of $y_{strong} \geq thr_{strong}$;
end
```

Only a sample that passes all stages is classified as positive. This framework allows that overall a false postive rate $FP = (fp)^t$ and a true postive rate $TP = (tp)^t$ can be expected, where fp, tp and t denote the maximal false postive rate and the minimal true postive rate at each stage, as well as, the number of stages, respectively.

In this chapter, in contrast to the illustration example, the training of the classifier at every stage is done by the gentle AdaBoost algorithm explained in Section 3.1.5. After every training stage the true negative examples are removed from the training set and the remaining training subset is used to train the gentle AdaBoost classifier for the next cascade stage. Algorithm 2 outlines the process.

Algorithm 2: Cascaded classifier training

```
Data: Training set with $N$ samples and $M$ features as $\boldsymbol{H} \in \mathbb{R}^{N \times M}$
Class labels $y \in \{-1, +1\}^N$
Minimum true postive rate per stage $tp$
Maximum false postive rate per stage $fp$
Final number of stages $t$
Result: Cascaded gentle AdaBoost classifier
Set current number of stages $t_{cur} = 0$;
Divide training data set in positive and negative samples;
while $t_{cur} < t$ do
    $t_{cur} = t_{cur} + 1$;
    Train gentle AdaBoost classifier for the $t_{cur}$-th stage with positive and
    negative training samples, $tp$ and $fp$;
    Use newly trained classifier to classify negative samples;
    Discard all negative samples that correctly were classified as negative;
end
```

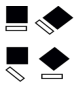
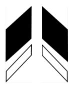
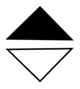
3.2 Experiments

In a first experiment, the question is whether the addition of the parallelogram and triangle templates on the one hand, and the F-test, T-test and V-test as new features on the other hand, is advantageous. The challenge lies in the manner of evaluation.

The detection and classification rate are input parameters for the cascade. This means the resulting cascaded classifier has, at least on the training data, either the required true postive rate and false postive rate, or the algorithm will not converge. Hence, they are ill-fitted performance measures. Instead, the AdaBoost algorithm itself is used.

As described in Section 3.1.5, AdaBoost is a greedy algorithm that incorporates the (locally) best fitting weak classifier at every iteration step. With this in mind, the algorithm is applied to determine whether the new shapes and features are useful for the detection of objects on sonar images or not. Whenever a new shape and/or

Table 3.1 – Average proportion (%) of the template-feature-combinations. The used templates are depicted for the columns. For the features, ζ_c denotes the correlation, ζ_F the F-test statistic, ζ_t the Student's t-test statistic and ζ_w the Welch's t-test statistic from (3.11).

				Σ
ζ_c	17.3	1.8	0.6	19.7
ζ_F	18.5	8.8	2.6	29.9
ζ_t	19.3	2.8	-	22.1
ζ_w	20.7	6.7	0.9	28.3
Σ	75.8	20.1	4.1	100

feature is chosen, it is at that iteration step more useful for the detection than any other feature derived of the shapes proposed in [46] with correlation alone.

For training and testing, a dataset of real sonar images with a given classification has been used. Overall the dataset consists of hand-labelled examples of about 2000 MMOs and 13000 images without any object of interest. For this dataset the features derived by the new shapes as well as the features derived by the shapes from [46] are computed and combined to train the classifier. Within every stage the desired true postive rate was 0.995 and the maximum false postive rate was 0.5. The final false postive rate was set to $(\frac{1}{2})^{8} \approx 0.004$, so a maximum of eight stages is needed. This results in a final true postive rate of $(0.995)^{8} \approx 0.96$. The total number of used features after training, i.e., all stages combined, was about 400 compared to about 500 when only the standard templates and correlation was used.

The average distribution over all used features in percent by the Adaboost algorithm can be found in Table 3.1. For the different shapes, the proportions are plotted row-wise, while the different features for the same shape via test statistics are plotted column-wise. The overall distribution can be found in the rightmost column or in the bottom line. The table illustrates, that all features independent of template were used about evenly. Interestingly, the correlation was the least used one. For the templates, the parallelogram based shapes have a significant impact on the final classifier with about 20% of the weak classifiers were derived from the parallelogram shape with one of the features. In contrast to this, the triangle templates were not significantly utilized. Taking into account that calculating the integral images contributes a big part to the total processing time and four out of the six integral images are just calculated for the triangular templates, using them seems not reasonable. Therefore, we excluded them from further experiments.

In a second experiment, we used the real data from two different areas at sea gathered in Italy. In the first case, the environment can be regarded as manageable regarding the detection phase. The seafloor consists mostly of flat bottom with just a few line-like distortions caused probably by fishing. In the second case, the environment is very demanding. The seafloor type alternates between rocks and

Table 3.2 – Confusion matrix of the Man-Made Object (MMO) detector trained on the first area second data set and applied to the first area first data set. The results are broken down to the individual stages in the cascade.

(a) Stage 1: Precision: 1, Recall: 0.20

		Detected class	
		MMO	non-MMO
Class	MMO	20	0
	non-MMO	78	-

(b) Stage 2: Precision: 1, Recall: 0.31

		Detected class	
		MMO	non-MMO
Class	MMO	20	0
	non-MMO	45	-

(c) Stage 3: Precision: 0.95, Recall: 0.79

		Detected class	
		MMO	non-MMO
Class	MMO	19	1
	non-MMO	5	-

sand ripples. Both areas, simple seafloor and complex seafloor, were each surveyed twice in a lawnmower pattern, where the second mission path was perpendicular to the first. To evaluate the performance of the algorithm, the detector was trained for each sea area on the manually classified first mission and then applied to the perpendicular, second mission and vice versa. Moreover, the cascade was limited to only three steps. For the application in mind, it is crucial to not miss any actual MMO. Therefore, a slightly higher false postive rate is accepted.

For the simpler first area Table 3.2 and Table 3.3 illustrate the results. In order to have more insight in the cascade, a confusion matrix after every stage is displayed. Even though there are only very few positive training examples, the results look promising. As can be seen already after the first stage, the ROI is reduced significantly. After the second stage again a big part of the remaining false detections are rejected. Unfortunately, after the third and final stage also an actual MMO gets rejected.

Figure 3.9 shows the detection on a real SAS image. The colour of an target shows the maximum number of stages it could pass. Purple for one stage, black for two stages, and red for three stages. In order to also analyse the border pixels with the template matching, a mirroring boundary strategy was employed.

The figure also shows an example of an object that did not pass all three stages. The problem here is that the object is almost within the darker band that can be seen over the complete lower part of the image. This band is due to an artifact of the SAS processing, i.e., signal strength normalization. The strength of signal returning to the SAS antenna is not only dependent on the reflectivity of the object it was reflected of, but also on the distance travelled. Hence, without accounting for this in the SAS processing the image would be very bright near the vehicle position and very dark at a far across track distance. In the image shown the SAS processing has accounted for this effect but overcompensated in the very near ranges. Unfortunately for the

Table 3.3 – Confusion matrix of the Man-Made Object (MMO) detector trained on the first area first data set and applied to the first area second data set. The results are broken down to the individual stages in the cascade.

(a) Stage 1: Precision: 0.95, Recall: 0.44

		Detected class	
		MMO	non-MMO
Class	MMO	21	1
	non-MMO	27	-

(b) Stage 2: Precision: 0.91, Recall: 0.95

		Detected class	
		MMO	non-MMO
Class	MMO	20	2
	non-MMO	1	-

(c) Stage 3: Precision: 0.82, Recall: 0.95

		Detected class	
		MMO	non-MMO
Class	MMO	18	4
	non-MMO	1	-

detection, this causes the shadows of objects to disappear. In fact, the only object not even detected in the first stage (see Table 3.3) was directly inside this band. Without this processing artefact the detection would have most likely succeeded.

For the second sea area, the results were unfortunately not convincing. Figure 3.10 shows one example. The sand ripples cause a lot of false detections that pass the whole cascade, while two of the contacts get rejected in the second stage. One way to cope with this challenging environment is seafloor type classification [38].

3.3 Summary of the results on the efficient detection on SAS images

Based on the results of [79] for the task of face detection, an algorithm for identifying ROIs in sonar images is presented in this chapter. For the algorithm, a cascade of boosted classifiers is used. The classifiers themselves consist of extracted features with suitable thresholds given by the gentle Adaboost algorithm. For a rapid feature extraction, integral images are employed. Besides the classical template matching with different shapes, statistical tests are used to derive multiple features per shape. Towards this concept two new shapes are proposed, a parallelogram based shape and a triangle based shape that are both especially tailored to the context of object detection on SAS images. Experiments suggest that the parallelogram shape makes a positive contribution to the task of identifying ROIs. The triangle template, however, were only barely used in the cascaded AdaBoost classifier which is why it was not used in further experiments. Besides the two new template shapes, three new features per template match were introduced in addition to the correlation. The analysis of the make-up of a trained cascaded AdaBoost classifier revealed that the new features were actually used more often then the correlation. Gauging the impact on the

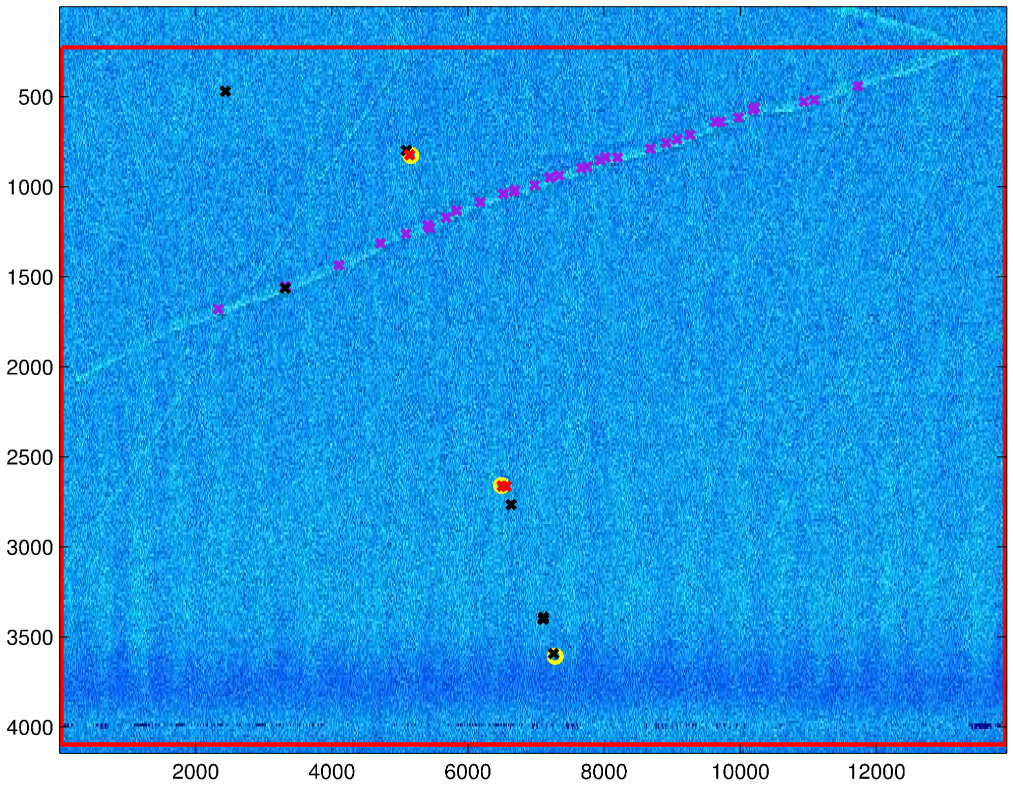


Figure 3.9 – Example detection on flat sea bottom. Actual targets are marked with yellow circles. The number of passed stages is colour coded. Purple targets have passed the first stage, black targets the second stage, and red targets passed the final stage. The red box indicates the analysed area and represents 325 m along track, i.e., in horizontal direction, and 65 m across track, i.e., in vertical direction.

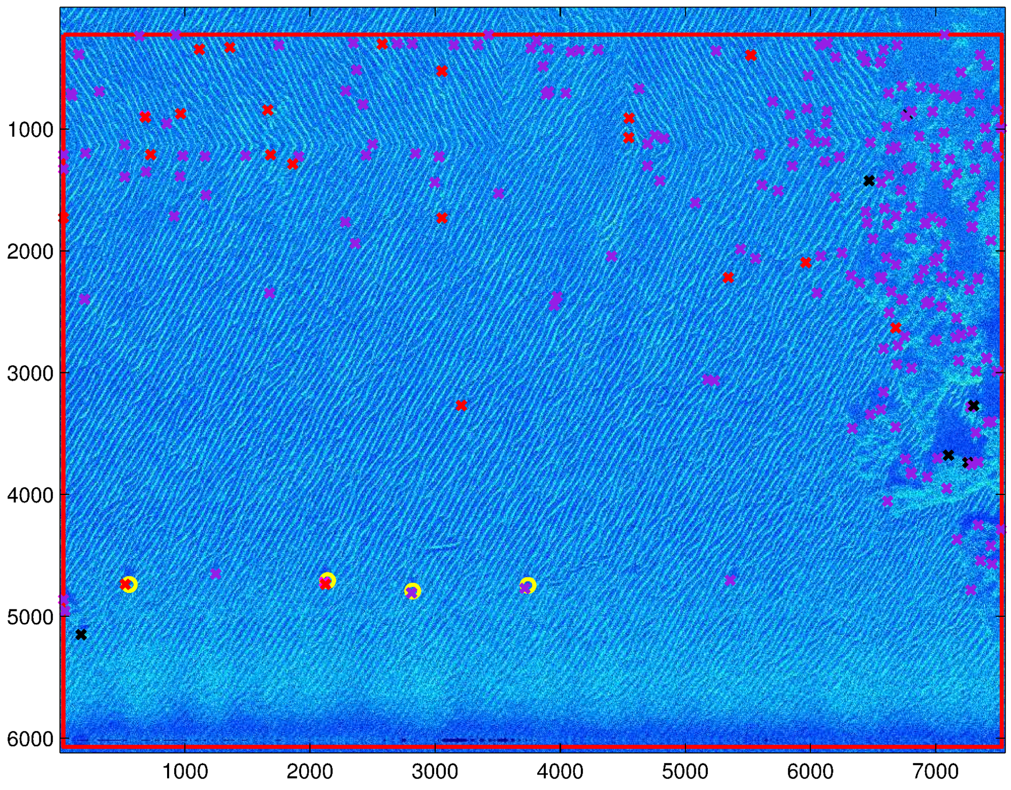


Figure 3.10 – Example detection on rocky sea bottom with sand ripples. Actual targets are marked with yellow circles. The number of passed stages is colour coded. Purple targets have passed the first stage, black targets the second stage, and red targets passed the final stage. The red box indicates the analysed area and represents 190 m along track, i.e., in horizontal direction, and 100 m across track, i.e., in vertical direction.

performance measures like true postive rate or false postive rate is not trivial in this case, since those are design parameters of the classification algorithm. Still, having better fitting templates had an effect on the evaluation time of new samples. The number of used features in total was about 20 % less when the additional templates and features were used. This meant in turn less template matching that had to be performed.

The proposed algorithm was also tested with respect to its detection performance on real world data. On non-complex seafloor it was able to discard most of the areas with no objects of interest, without missing actual targets. However, tests on a very challenging seafloor also revealed that still some work has to be done to lower the false postive rate.

After the algorithm is applied, the result are several sub-images showing potential objects of interest. Due to the reduction of the input SAS image to only a small fraction, it is now possible to apply more sophisticated algorithm to determine the exact class of the object. The topic of the next chapter is deriving such an classification algorithm.

CHAPTER 4

Superellipse based Contour Extraction

4.1 Introduction

After the previous chapter in which the topic was the image preprocessing, the focus of this chapter is shape based recognition of the image's content. In this classification phase, more complex algorithms will be applied, which is feasible thanks to the significant reduction of the ROI done in the previous chapter. The underlying assumption is that the methods shown in this chapter will eventually be applied on SAS images. However, the theory of the algorithm is imaging system independent. In fact, one of the strengths of the proposed methods is that they are blue prints that can easily instantiated for the characteristics of different imaging systems. Therefore, the experiment section will not only cover SAS images but also investigating how the proposed methods can be applied to visual underwater data.

Even though state-of-the-art image classification algorithms are dominated by methods using deep neural networks like Convolutional Neural Networks (CNNs) in different forms [39], there is still room for other methods. Despite their impressive performances, deep neural networks have one crucial downside. Usually, a huge amount of training data is needed to get those superior classifiers. Popular public evaluation datasets like ImageNet consists of more than 14.000.000 images with between 56.000 and 2.8 million images per class to train on [14]. The huge amount of data is needed since training a neural networks not only consist of training a classifier but also the automatic feature extraction is trained. In contrast to this consists for example the SAS image dataset used throughout this thesis of only 213 images covering 5 classes. This is not enough data to train a classifier *and* automatic feature extraction. To solve this problem, some authors use pre-trained CNN models [16], [66], [88] to initialise their networks and/or they only re-train the last layers. The idea is that in the first layers domain unspecific low level features are extracted that can also apply to other classification tasks. This approach is surprisingly successful, albeit there are limits. The transferability of features decreases as the difference between in the classification tasks increases. This means a CNN pre-trained on in-air photographies of everyday objects is probably not suitable to classify objects on SAS images.

An alternative is the *classic* approach of applying domain specific knowledge to design suitable features. This can be done by first segmenting the image in object and background. The result is then used to calculate features based on the shape of the extracted object or its contour. To have a robust and reliable classifier that is based on those shape features, a good shape extraction via segmentation is crucial. Unfortunately, it is usually not trivial to segment the image in object and background due to noise. On the one hand, there is internal noise due to the used imaging system and processing of the recorded data. For example, this noise could be caused by the limitations due to the quality of the used camera. But also by the method itself like in the case of sonar. Consequently, sometimes a clear distinction between object and background is very challenging, due to the absence of a clear edge between the two regions.

On the other hand, there is external noise like occlusions from other objects. If only part of the object is visible, the extracted shape might not match with the expected shape, which in turn could inhibit a correct classification.

In this chapter approaches of how to mitigate these noise effects on the segmentation are investigated. The main idea is that often the automatic classification is not used to just recognize an arbitrary object, but instead the interest is in finding a *specific* object. This specific object or objects have specific shapes that are known a-priori. Exploiting this knowledge by restricting the set of possible segmentation shapes can make the segmentation itself and, thus, the classification more robust and accurate. The idea is to imagine how the object of interest would look like without the interfering noise. Many man-made objects can be represented by only a small set of plausible shapes. However, due to noise, arbitrary variations of the underlying object contour are extracted. The idea is that the noise can be suppressed by finding the closest representative within the subset of valid shapes. This chapter is dedicated to this idea.

In Section 4.2 the superellipse is introduced. Superellipses are a class of shapes that can represent rectangles, ellipses, diamonds and in-between shapes. The idea is that most man-made objects fall under this category. Therefore, superellipses are an ideal candidate for the above described small set of plausible shapes. As preparation for the following chapters also different point to superellipse distance approximations are presented.

The first implementation of the *only-accept-superellipse-contours-after-segmentation*-approach is shown in Section 4.3. The basic idea of having a superellipse after the segmentation can be implemented in two ways. In Section 4.3 the approach is to use superellipse fitting as post-processing step after a freely chosen segmentation algorithm extracted the object contour. The authors of [18] applied this principle by post-processing via fitting a superellipse onto the result of a markov-random-field based segmentation [53] of a side-scan sonar image. Section 4.3 will follow this idea and focus on efficiency and accuracy. To this end, a linearisation of the fitting cost function is introduced, as well as different fitting configurations are investigated. The proposed modifications to the standard superellipse fitting are evaluated on real and simulated contours.

In contrast to this post-processing approach, Section 4.4 presents an integration of the superellipse assumption directly into an active contours without edges segmentation algorithm [9]. The basic idea is already followed in [45]. The superellipse assumption is employed through a penalty term that measures the difference of the extracted contour to a superellipse. Consequently, the contour is not strictly restricted to superellipse shapes but just *pushed* towards them, where the *force* is adjusted by a parameter. Section 4.4 will improve this by the introduction of an abstract reformulation of the active contours without edges framework. A probabilistic view on the pixel intensities is introduced which allows to account for different pixel intensity distributions within the image. Ultimately, this means that the proposed active contours algorithm can easily be tailored to any imaging system by specifying suitable pixel intensity distributions. The second contribution in Section 4.4 is the integration of the superellipse assumption within the active contours framework without the use of a penalty term. The algorithm is evaluated with real SAS images and it is shown that the extracted superellipse parameters can even directly be used as shape features for classification. Moreover, the algorithm is also evaluated on visual underwater data to detect the location of valves on a deep sea panel.

4.2 Superellipses

Superellipses are characterised by the two half-axes a and b and the squareness parameter $\varepsilon > 0$ according to the implicit definition

$$F(\boldsymbol{z}|a,b,\varepsilon) = F(\boldsymbol{z}) = \left(\frac{x}{a}\right)^{\frac{2}{\varepsilon}} + \left(\frac{y}{b}\right)^{\frac{2}{\varepsilon}} \overset{!}{=} 1 \tag{4.1}$$

or the parametrised definition

$$x = a\cos(\vartheta)^{\varepsilon},\ y = b\sin(\vartheta)^{\varepsilon}. \tag{4.2}$$

with $\boldsymbol{z} = (x,y)^T$. In order to be well-defined, x and y need to be greater than 0 and $0 \leq \vartheta < 0.5\pi$. So strictly speaking, the superellipse equation is only valid for the first quadrant. But the other three quadrants of the superellipse can be generated by employing the symmetry, i.e., mirroring the first quadrant. In the implicit version, the exponent can also be separated into first squaring and then taking the ε-th root, thus avoiding the problem of negative roots and the need to mirror. Figure 4.1 shows the effect of varying the squareness parameter. If ε goes to 0 the superellipse becomes a rectangle, for 1 it is an ellipse and the in-between shapes are generated by squarenesses between 0 and 1. Moreover, for squareness values above 1 the superellipse transforms into a diamond and starting with squareness values over 2 into star-like shapes.

Since the object will most often be not centred at the origin without any rotation, the superellipse also needs to be able to be translated by a vector T and be rotated by an angle θ. Furthermore, a common transformation in the context of superellipses is the so-called tapering. It is given by

$$q_{tap}(\tau,\boldsymbol{z}) = \begin{pmatrix} \left(\left(\frac{\tau y}{b}\right)+1\right)x \\ y \end{pmatrix} \tag{4.3}$$

and inflates the contour above the x-axis and deflates it below for positive τ. This can be used to generate tear-like shapes.

With this additional transformations, the implicit definition, changes to

$$\tilde{F}(\boldsymbol{z}|\boldsymbol{\nu}) = F(a,b,\varepsilon) \circ q_{tap}^{-1}(\tau) \circ q_{trans}^{-1}(T) \circ q_{rot}^{-1}(\theta)(\boldsymbol{z}) \overset{!}{=} 1 \tag{4.4}$$

with

$$\boldsymbol{\nu} = \{a,b,\varepsilon,\tau,T,\theta\} \tag{4.5}$$

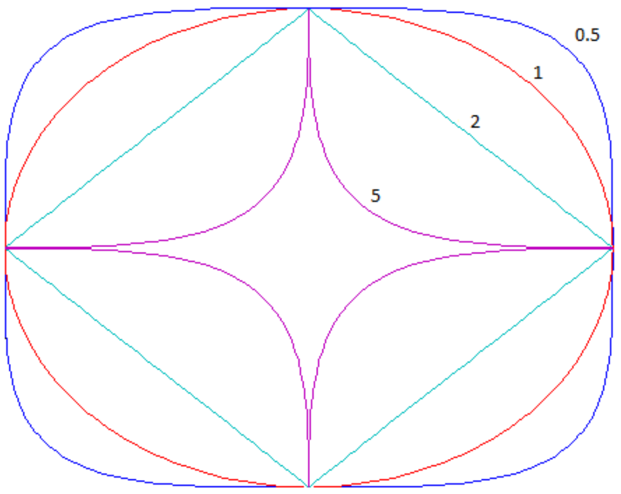


Figure 4.1 – Superellipses for different squareness parameters ε and constant half-axes a and b

and where the q denotes the transformations tapering, translation and rotation. Please note that the transformations are inverted.

Another deformation in the context of superellipses that is not considered is the so called bending. It bends the shape so that for example an ellipse can be deformed into a banana-like, concave shape. In general, bending can be a useful addition. However, the shapes occurring in the experiments will be predominantly convex. Preliminary tests on the SAS test images agree with this notion by using only negligible small bending to represent the contours. Thus, this deformation was not considered.

To come back to the original idea, with (4.4) it is now possible to restrict the segmenting contours to superellipses by ultimately only accepting those that fit the equation.

4.2.1 Distance approximations to a given superellipse

As described in the introduction two distinct approaches will be followed to fulfil the assumption that the contour of a man-made object of interest on a given image should be a superellipse. On the one hand, this can be achieved by fitting as post-processing of an already extracted contour. On the other hand, this can be done by incorporating the superellipse assumption directly in the segmentation algorithm. In both cases it will be necessary to calculate the distance between given points and the superellipse. The challenge lies in finding a suitable distance measure d between a point $\boldsymbol{z}$ and the superellipse C^s. Intuitively, one would define it as the Euclidean distances d between the contour point $\boldsymbol{z}$ and the nearest points on the contour $\boldsymbol{z}^s \in C^s$.

$$d(z, C^s) = \min_{\boldsymbol{z}^s \in C^S} d(\boldsymbol{z}, \boldsymbol{z}^s) \tag{4.6}$$

Unfortunately, there is no closed-form solution for the nearest point $\boldsymbol{z}^s$ on a given superellipse to a given point $\boldsymbol{z}$ and an iterative approach is computationally expensive. Various distance approximations were introduced as alternatives. The authors of [69] compared nine of them, but the result was inconclusive. There is no obvious best error measure and the performance depended on the level of noise. Anticipating the worst case for the quality of the test images it was decided to use error measures, which worked reasonable well under noise. With this in mind, three distance approximations qualify for a closer inspection. Generally speaking, a good distance measure approximation needs to fulfil two properties. Explaining them requires the introduction of isolines. A isoline l_c with respect to a certain distance approximation $d_{approx.}$ is defined as all the points that have the same constant distance in the distance approximation.

$$l_c = \{\boldsymbol{z} | d_{approx}(\boldsymbol{z}, C^s) = c = \text{const.}\}\,. \tag{4.7}$$

The first property is that the points on each isoline in the approximation should actually also be isolines in Euclidean distance, i.e., have constant distance to the superellipse in Euclidean distance. If the distance approximation would over- or underestimate the distance to the superellipse for certain directions, these points would have implicitly a higher or lower influence on the algorithm used later in this chapter, e.g., the fitting error.

The second property is that the distance between consecutive isolines has to stay constant for a fixed normal direction from the superellipse. The impact of violating this property should be less severe, however. The distance will be either squared, which breaks the linearity, or only distances in a narrow band will be evaluated, where the effect of the distortion should be small. The three distance approximations from [69] will be evaluated with respect to those two properties with focus on the first property.

The first out of the three distance approximation functions is defined as

$$d_\alpha(\boldsymbol{z}, C^s) = \left| \left(\frac{x}{a}\right)^{\frac{2}{\varepsilon}} + \left(\frac{y}{b}\right)^{\frac{2}{\varepsilon}} - 1 \right|. \tag{4.8}$$

This is the straight forward approach that will be called standard. The approximation is not very accurate, though. Especially within the superellipse d_α can only reach values between 0 and 1, regardless of the size of the superellipse. Moreover, the distance between isolines is not equal, neither considering consecutive isolines in the same normal direction, nor the same two isolines in different directions. Figure 4.2(a) illustrates this for an example.

The second method is to find a scaled version of the superellipse that passes through the point. Solving

$$\left(\frac{x}{wa}\right)^{\frac{2}{\varepsilon}} + \left(\frac{y}{wb}\right)^{\frac{2}{\varepsilon}} = 1 \tag{4.9}$$

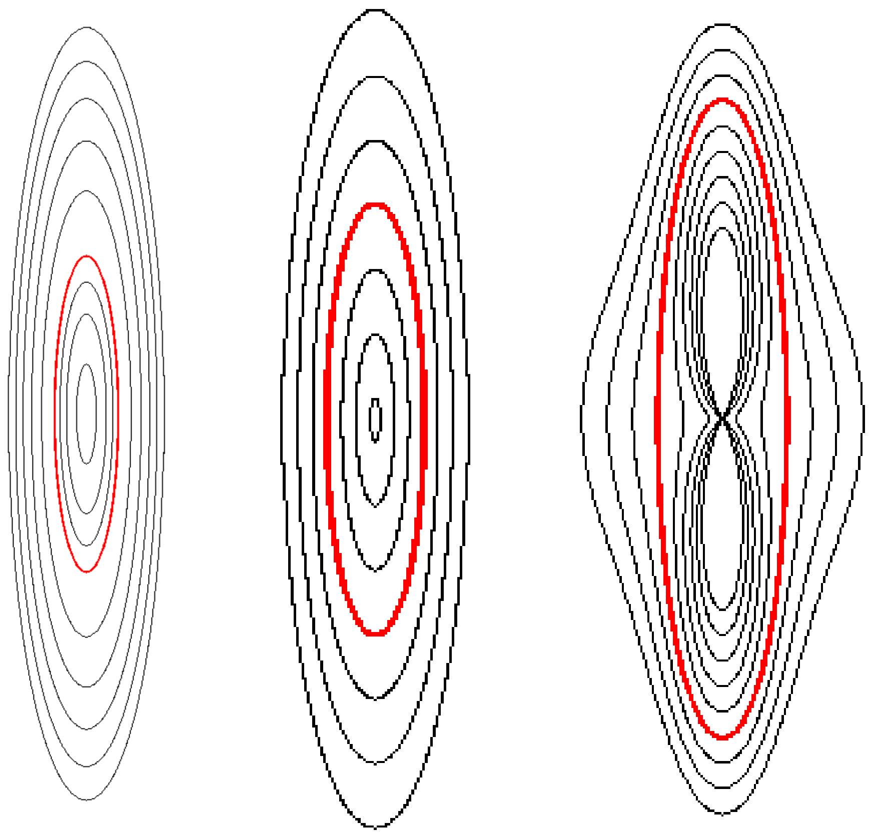


Figure 4.2 – Isolines of the approximated distances to the red superellipse. (a) The isolines illustrate that d_α overestimates the distance in the direction of the longer, vertical half axis and underestimates the distance in direction of smaller, horizontal half axis. (b) The same is true for d_β. (c) In contrast, d_γ does not over- or underestimate the distance in direction of the half axes. This causes the hourglass shaped isolines. The distance is slightly underestimated between those directions, however.

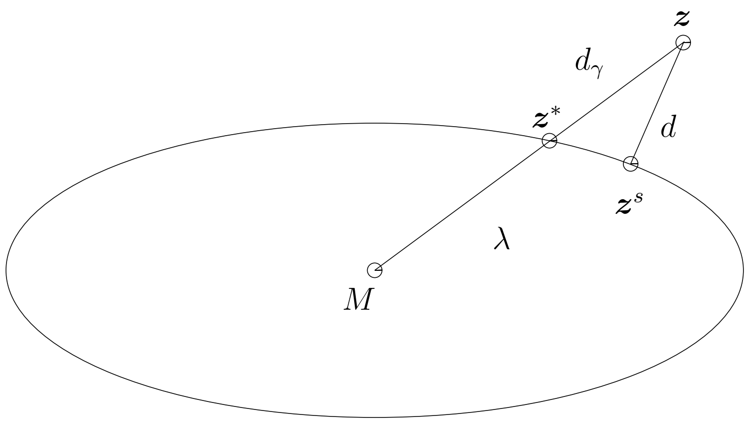


Figure 4.3 – Approximation of the real distance between a point and a superellipse via the ray-to-centre approach. The ideal case would be finding the nearest point $\boldsymbol{z}^s$ on the superellipse to calculate the true distance d of $\boldsymbol{z}$ from the given superellipse. Since this is not feasible, d is approximated by d_γ, the distance between the intersection of the superellipse and a line connecting $\boldsymbol{z}$ and the origin of the superellipse M.

for the scaling factor w leads to the definition of the second distance function as

$$d_\beta(\boldsymbol{z}, C^s) = |1 - w| = \left|1 - \left(\left(\frac{x}{a}\right)^{\frac{2}{\varepsilon}} + \left(\frac{y}{b}\right)^{\frac{2}{\varepsilon}}\right)^{\frac{\varepsilon}{2}}\right|. \tag{4.10}$$

Since at the flat side of a superellipse a higher scaling factor is needed to reach a certain distance than on the pointed side, the isolines do not have a constant distance to the zero level similar to the first approach. However, the distance between consecutive isolines remains constant in all normal directions of the superellipse as can be seen in Figure 4.2(b).

The last approach is dubbed ray-to-centre approach: Given an arbitrary point $\boldsymbol{z}$, instead of finding the nearest point on the superellipse $\boldsymbol{z}^s \in C^s$, it is approximated by the intersection point $\boldsymbol{z}^* = (x^*, y^*)^T \in C^s$ of the superellipse with a line that is drawn from the point $\boldsymbol{z}$ to the centre of the superellipse. Figure 4.3 illustrates this for an example. Then the distance is approximated by

$$d_\gamma(\boldsymbol{z}, C^s) = \|\boldsymbol{z}^* - \boldsymbol{z}\|. \tag{4.11}$$

With the ray-to-centre approach the approximation $\boldsymbol{z}^*$ can be calculated by using the parametrised superellipse equations (4.2) which lead to

$$\frac{y}{x} = \frac{y^*}{x^*} = \frac{b \sin(\vartheta^*)^\varepsilon}{a \cos(\vartheta^*)^\varepsilon} \Rightarrow \vartheta^* = \tan^{-1}\left(\left(\frac{ay}{bx}\right)^{\frac{1}{\varepsilon}}\right) \Rightarrow \begin{aligned} x^* &= a\cos(\vartheta^*)^\varepsilon \\ y^* &= b\sin(\vartheta^*)^\varepsilon, \end{aligned} \tag{4.12}$$

and, therefore, $\boldsymbol{z}^* = (x^*, y^*)$ which can be used in (4.11). The discontinuity for $x = 0$ and $y \neq 0$ is handled by setting $\boldsymbol{z}^* = (0, b)$ and, therefore, $d_\gamma(\boldsymbol{z}, C^s) = b$. If $\boldsymbol{z} = (0, 0)$ the distance is the minimum of a and b.

An alternative approach is using that there is a $\gamma \geq 0$ for which

$$\boldsymbol{z} = \gamma \frac{\boldsymbol{z}^*}{\|\boldsymbol{z}^*\|}. \tag{4.13}$$

With this entering $\boldsymbol{z}$ in the implicit superellipse formula (4.1) gives

$$F(\boldsymbol{z}) = F\left(\gamma \frac{\boldsymbol{z}^*}{\|\boldsymbol{z}^*\|}\right) = \left(\frac{\gamma}{\|\boldsymbol{z}^*\|}\right)^{\frac{2}{\varepsilon}} \overbrace{F(\boldsymbol{z}^*)}^{=1} = \left(\frac{\gamma}{\|\boldsymbol{z}^*\|}\right)^{\frac{2}{\varepsilon}} \tag{4.14}$$

$$\Rightarrow F(\boldsymbol{z})^{\frac{\varepsilon}{2}} = \frac{\gamma}{\|\boldsymbol{z}^*\|}. \tag{4.15}$$

Eventually, this leads to

$$d_\gamma(\boldsymbol{z}, C^s) = \|\boldsymbol{z}^* - \boldsymbol{z}\| = \left\| \boldsymbol{z} \frac{\|\boldsymbol{z}^*\|}{\gamma} - \boldsymbol{z} \right\| = \|\boldsymbol{z}\| \left| F(\boldsymbol{z})^{-\frac{\varepsilon}{2}} - 1 \right|, \tag{4.16}$$

thus, directly calculating the approximated distance.

In Figure 4.2(c) the resulting isolines to the given red superellipse are shown. At a first glance, the figure might suggest that the ray-to-centre method is not a good choice, since the isolines are not superellipses anymore as one naively suspect. However, the important part of an isoline is that they are also isolines in Euclidean distance, i.e., the Euclidean distance is constant for all points on the isoline. This is at least approximately true in the ray-to-centre case near the half axis points. Due to this, the isolines are no longer superellipses. This approach is also better with equidistant consecutive isolines, albeit not perfect.

All three approximations showed some problems with approximating the real distance to the zero level. The ray-to-centre method, however, is at least correct in the direction of both half axes. The experiments in [36] showed that in case of SAS images it is indeed the superior choice. Therefore, the ray-to-centre method will be used in this chapter to approximate the distance to a superellipse.

4.3 Computationally efficient superellipse fitting

As previously stated, in the first approach an already extracted contour $C = \left\{\boldsymbol{z}_i = (x_i, y_i)^T \text{ for } i = 1, \dots, N\right\}$ is used in combination with the superellipse assumption. This is done by fitting a superellipse to C. As usual, the fitting will be done by minimizing a fitting error q between the contour and a given superellipse. All six parameters in $\boldsymbol{\nu}$ (see (4.5)) could be used as minimising variables. But since the complexity of a minimisation problem grows non-linear with the number of variables, it was decided to split the problem in two parts. Even though the exact shape of the extracted contour is distorted by noise, with enough contour points and a Principal Component Analysis (PCA) the position and rotation of the shadow or highlight can be inferred reliably. Thus, the translation T and rotation θ can be calculated before the minimisation. The fitting problem simplifies to

$$\min_{a,b,\varepsilon,\tau} q(C, a, b, \varepsilon, \tau). \tag{4.17}$$

As stated above, the translation and rotation was already calculated via PCA and compensated, hence the x_i and y_i have zero mean and the maximum variance of the $\boldsymbol{z}_i$ is found in the direction of the y-axis.

Intuitively, the fitting error should be defined as the sum of squared distances for each point to the superellipse in Euclidean distance. As already discussed in Section 4.2.1 this is computationally expensive, especially due to the iterative nature of fitting. Thus, the ray-to-centre distance approximation d_γ is used instead. Hence, the fitting error is defined as

$$q(C, a, b, \varepsilon, \tau) = \sum_{\boldsymbol{z}_i \in C} d_\gamma(\boldsymbol{z}_i, C^s)^2, \tag{4.18}$$

with $d_\gamma(\boldsymbol{z}_i, C^s)$ defined as in (4.16).

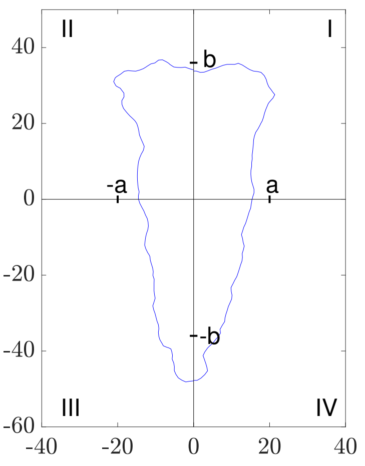
The straight forward version after having defined the error q is just fitting one complete superellipse to the whole contour. Hereby, the four parameters must be found that minimize the fitting error in (4.18). Any iterative solver can be chosen to solve this optimisation problem. The authors of [18] used this complete superellipse for the whole contour approach.

Looking at man-made objects on images, though, it is noticeable that the contours are not always perfectly symmetrical. Therefore, it might be an advantage to loosen the superellipse shape constraint a little by allowing piecewise superellipse contours. This means, instead of four parameters for all data points, every quadrant gets its own four parameters [44]. To ensure a continuous graph of the resulting approximation, the semi axes are linked so that

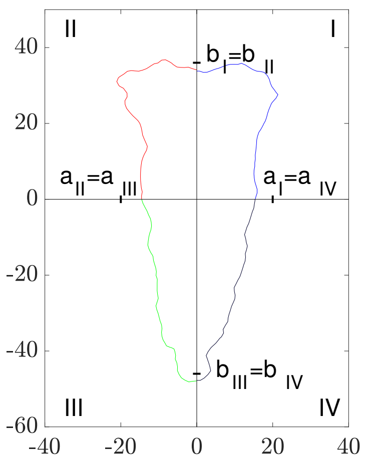
$$a_I = a_{IV}, \qquad a_{II} = a_{III}, \qquad b_I = b_{II}, \qquad b_{III} = b_{IV}, \tag{4.19}$$

with the indices indicating the quadrant. With these constraints, the total number of parameters of the minimization problem is 12. Thus, albeit the adaptability of the fitting contour increased, the drawback is the high amount of adjustable parameters. Therefore, a third possible approach is applying the superellipse fitting for each quadrant independently, without any coupling between the semi axes. And afterwards, the appropriate semi axes are averaged to obtain the final 12 parameters [44]. Figure 4.4 illustrates the three different views on the contour fitting problem.

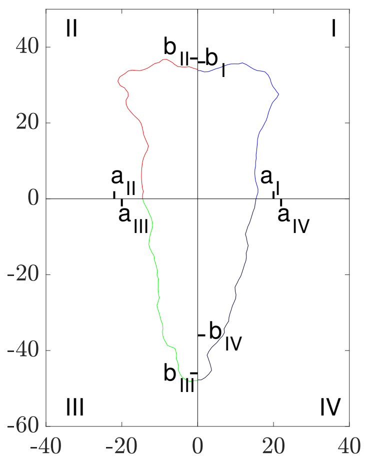
As previously stated, the complexity of a minimisation problem grows non-linearly with the number of variables. Therefore, it can be beneficial to think about how to transform the minimisation of (4.18) from a four variable minimisation to an iterative minimisation with less variables. This can be done by linearising (4.18) with respect to a and b. Strictly speaking, the dependence of ϑ on a and b makes the formulation non-linear, as well as the fact that the tapering deformation is dependent



(a) a, b, ε, τ



(b) $a_I = a_{IV}$, $a_{II} = a_{III}$, $b_I = b_{II}$, $b_{III} = b_{IV}$, ε_I, ε_{II}, ε_{III}, ε_{IV}, τ_I, τ_{II}, τ_{III}, τ_{IV}.



(c) $\{a_I, b_I, \varepsilon_I, \tau_I\}$, $\{a_{II}, b_{II}, \varepsilon_{II}, \tau_{II}\}$, $\{a_{III}, b_{III}, \varepsilon_{III}, \tau_{III}\}$, $\{a_{IV}, b_{IV}, \varepsilon_{IV}, \tau_{IV}\}$

Figure 4.4 – The three approaches to superellipse fitting. (a) The whole contour is approximated with one complete superellipse. Optimization over four variables. (b) The contour points are sorted by quadrant. Each quadrant is simultaneously fitted with a superellipse while the half axis parameters are coupled. Optimization over twelve variables. (c) The contour points are again sorted by quadrant. Each quadrant is fitted with a superellipse in sequence. Only after the fitting for all quadrants is finished the appropriate half axes are averaged to get a continuous contour. Four optimisation over four parameters.

on b. Consequently, two assumptions are made. Firstly, there is no tapering, i.e., $\tau = 0$. Secondly, the ratio between a and b in (4.12) is fixed within one iteration. Thus, ϑ is only dependent on ε. The first assumption means a trade-off between accuracy and speed. The experiments will show how big the impact of neglecting the additional shape parameter is. The impact of the second assumption, however, is small if the initial guess is sufficiently good. Only the ratio between a and b is needed in (4.12) which changes slowly with respect to small changes of $a \gg 1$ and $b \gg 1$.

Recalling the parametrised definition of a superellipse (4.2), and thanks to the assumptions just made - and only with them - the fitting parameters can be found via the overdetermined equation system

$$\boldsymbol{A}(\varepsilon)\boldsymbol{\rho} = \boldsymbol{x} \tag{4.20}$$

with

$$\boldsymbol{\rho} = \begin{pmatrix} a \\ b \end{pmatrix}, \qquad \boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \\ y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \qquad \boldsymbol{A}(\varepsilon) = \begin{pmatrix} \cos(\vartheta_1)^\varepsilon & 0 \\ \cos(\vartheta_2)^\varepsilon & 0 \\ \vdots & \vdots \\ \cos(\vartheta_N)^\varepsilon & 0 \\ 0 & \sin(\vartheta_1)^\varepsilon \\ 0 & \sin(\vartheta_2)^\varepsilon \\ \vdots & \vdots \\ 0 & \sin(\vartheta_N)^\varepsilon \end{pmatrix}, \tag{4.21}$$

which is linear in $\boldsymbol{\rho}$. The standard method for solving this kind of problem is the least squares method

$$q(\boldsymbol{\rho}, \varepsilon) = (\boldsymbol{x} - \boldsymbol{A}(\varepsilon)\boldsymbol{\rho})^T((\boldsymbol{x} - \boldsymbol{A}(\varepsilon)\boldsymbol{\rho})), \tag{4.22}$$

which is equivalent to the fitting error (4.18) with the assumptions described before. Minimizing (4.22) with respect to $\boldsymbol{\rho}$ leads to

$$\nabla_{\boldsymbol{\rho}} q = 0 \qquad \Rightarrow \qquad \boldsymbol{A}(\varepsilon)^T\boldsymbol{A}(\varepsilon)\boldsymbol{\rho} = \boldsymbol{A}(\varepsilon)^T\boldsymbol{x} \tag{4.23}$$

Formally the solution is

$$\boldsymbol{\rho} = \left(\boldsymbol{A}(\varepsilon)^T\boldsymbol{A}(\varepsilon)\right)^{-1}\boldsymbol{A}(\varepsilon)^T\boldsymbol{x}. \tag{4.24}$$

If (4.24) is inserted in (4.22), the fitting error becomes

$$q_{\boldsymbol{\rho}}(\varepsilon) = q(\boldsymbol{\rho}, \varepsilon) = \boldsymbol{x}^T(Id - \boldsymbol{P}(\varepsilon))\boldsymbol{x} \tag{4.25}$$

$$\boldsymbol{P}(\varepsilon) = \boldsymbol{A}(\varepsilon)\left(\boldsymbol{A}(\varepsilon)^T\boldsymbol{A}(\varepsilon)\right)^{-1}\boldsymbol{A}(\varepsilon)^T. \tag{4.26}$$

Algorithm 3 shows how to calculate the fitting parameters with this linearisation. Depending on the initial guess for a/b, the algorithm needs only very few iteration steps. For the following experiments a and b were chosen by fitting an ellipse to the given contour points, which is a by-product of the PCA. Alternatively, the linearisation could be used with ε^* considered constant equal 1. With this second initialising scheme via ellipse fitting just two iteration steps are sufficient.

The presented linearisation approach so far covered the case where one complete superellipse is used to approximate the whole contour. But two other versions were also introduced, namely dividing the contour in for separate quadrants with either coupled or independent half axis parameters a and b. It is easy to see how this approach can be transferred to the case where each quadrant of the contour is treated independently. For the version with the coupled half axis parameters, however, $\boldsymbol{A}$, $\boldsymbol{\rho}$ and $\boldsymbol{x}$ have to be redefined. For the first quadrant the x- and y-coordinates are

$$\boldsymbol{x}_I = (\boldsymbol{x}_{I1}, \boldsymbol{x}_{I2}, \dots, \boldsymbol{x}_{IN_I})^T \; \boldsymbol{y}_I = (\boldsymbol{y}_{I1}, \boldsymbol{y}_{I2}, \dots, \boldsymbol{y}_{IN_I})^T. \tag{4.27}$$

Furthermore, let $\boldsymbol{c}_I$ and $\boldsymbol{s}_I$ be defined as

$$\boldsymbol{c}_I(\varepsilon) = \begin{pmatrix} \cos(\vartheta_{I1})^\varepsilon \\ \cos(\vartheta_{I2})^\varepsilon \\ \vdots \\ \cos(\vartheta_{IN_I})^\varepsilon \end{pmatrix}, \qquad \boldsymbol{s}_I(\varepsilon) = \begin{pmatrix} \sin(\vartheta_{I1})^\varepsilon \\ \sin(\vartheta_{I2})^\varepsilon \\ \vdots \\ \sin(\vartheta_{IN_I})^\varepsilon \end{pmatrix}. \tag{4.28}$$

After repeating these definitions for the other quadrants accordingly, $\boldsymbol{A}$, $\boldsymbol{\rho}$ and $\boldsymbol{x}$ can be redefined as

$$\boldsymbol{\rho} = \begin{pmatrix} a_1 \\ b_1 \\ a_2 \\ b_2 \end{pmatrix}, \quad \boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_I \\ \boldsymbol{x}_{II} \\ \boldsymbol{x}_{III} \\ \boldsymbol{x}_{IV} \\ \boldsymbol{y}_I \\ \boldsymbol{y}_{II} \\ \boldsymbol{y}_{III} \\ \boldsymbol{y}_{IV} \end{pmatrix} \quad \boldsymbol{A}(\varepsilon) = \begin{pmatrix} \boldsymbol{c}_I(\varepsilon_I) & 0 & 0 & 0 \\ 0 & \boldsymbol{s}_I(\varepsilon_I) & 0 & 0 \\ 0 & 0 & \boldsymbol{c}_{II}(\varepsilon_{II}) & 0 \\ 0 & \boldsymbol{s}_{II}(\varepsilon_{II}) & 0 & 0 \\ 0 & 0 & \boldsymbol{c}_{III}(\varepsilon_{III}) & 0 \\ 0 & 0 & 0 & \boldsymbol{s}_I(\varepsilon_I) \\ \boldsymbol{c}_{IV}(\varepsilon_{IV}) & 0 & 0 & 0 \\ 0 & 0 & 0 & \boldsymbol{s}_{IV}(\varepsilon_{IV}) \end{pmatrix}. \tag{4.29}$$

After the redefinition, the computation of $q_{\boldsymbol{\rho}}$ is equivalent to the steps (4.22)-(4.26).

4.3.1 Experimental results

In a first experiment to test the supperellipse fitting presented above, each fitting approach is tested on 86 contours of object shadows on SAS images. The main idea behind object classification on sonar images is that the three-dimensional geometrical shape of an object defines the shape of the highlight and shadow it produces on a sonar image. Figure 2.4 in the introduction showed already examples of real SAS images. Based on the shape of highlight and shadow, a trained human operator can determine that the real object that generated the particular highlight and shadow

Algorithm 3: Linearisation of the minimization of the fitting error q.

```
Data: Extracted contour points $\boldsymbol{x}$
Result: Superellipse fitting parameters $a, b, \varepsilon$
Initial guess for ratio $^a/_b$;
do
    Solve $\varepsilon^* = \arg\min_\varepsilon q_p(\varepsilon)$;
    Insert $\varepsilon^*$ in (4.24) to receive $\boldsymbol{\rho}^*$;
    $(a^*, b^*) = \boldsymbol{\rho}^*$;
    Update ratio $^a/_b$;
while Changes in $q(\boldsymbol{\rho}^*, \varepsilon^*)$ are big.;
```

shapes on, e.g., Figure 2.4(a) has to be cylindric. Consequently, the automatic distinction between different objects is usually based on features that are derived from the shapes of the objects' shadows and highlights in the sonar images [49]. This makes a good shape extraction via segmentation crucial.

In the literature the shadow is considered to be more discriminative. The shape of the detected object, and therefore its object class, is inferred from the shape of the shadow [7], [64]. The contour of the shadow is extracted via segmentation. Afterwards, contour based features are computed for a classification. Obviously, the quality of the contour features depend heavily on the quality of the extracted contour. Unfortunately, even with state-of-the-art sonar systems, the resulting images suffer from a lot of distortions. These distortions can be caused either internally by errors in the signal processing or externally by the environment. Natural structures like a rocky seabed can veil the distinctive shapes of the targets. A strong seabed reverberation with respect to the object or so called speckle noise can further degenerate the image quality.

Independent of the source of the noise the question is how the original, noise-free shapes of highlight and shadow can be restored. On this note, on an ideal sonar image the shape of the shadow of a target object should be ellipsoid, rectangular or in-between those two. This fits perfectly into our approach of limiting the extracted contours to superellipses. The assumption is that all extracted shadow contours of man-made objects would be superellipses without the presence of noise.

Before applying the superellipse fitting, a PCA was used to put the origin of the coordinate system for each contour in its centre, as well as to align the y-axis with the main orientation of the contour. The Nelder-Mead algorithm [58] was used as iterative solver whenever required. The starting values are calculated by ellipse fitting.

Figure 4.5 shows the fitting results made by the different approaches for one example. As expected, the three parameter approach provides the poorest approximation. Tapering, however, helps to better adapt to the given contour. Nevertheless, dividing the extracted contour in the four quadrants enables the fitting contour to properly represent the bullet like shape and the average error per point drops.

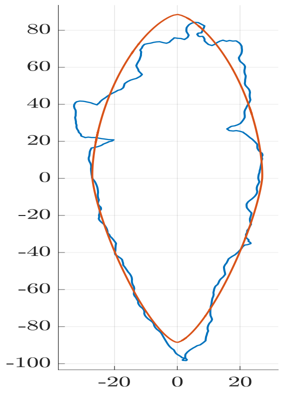
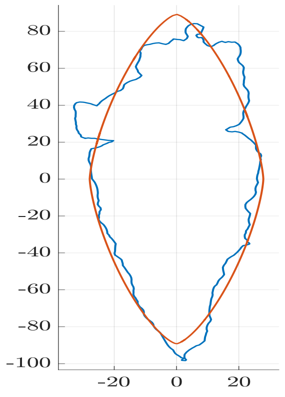
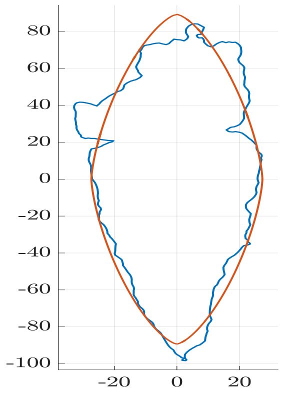
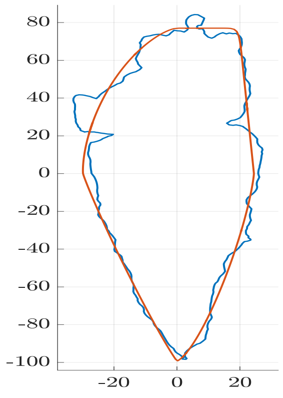
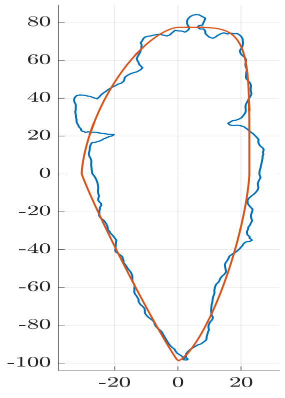
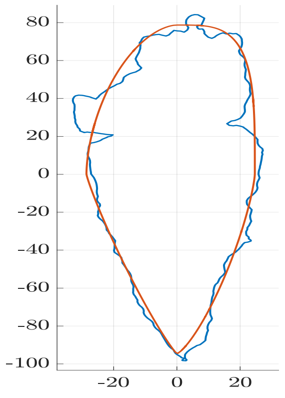
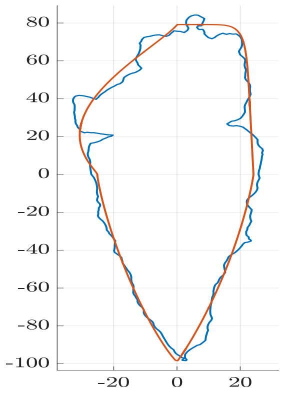
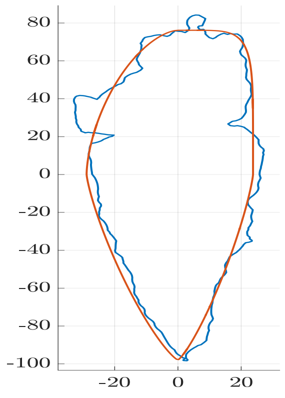
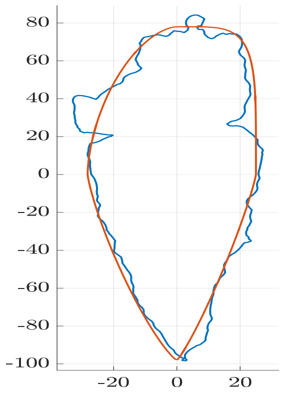
	Tapering	Standard	Linearised
One superellipse	 (a) error: 4.06	 (b) error: 4.42	 (c) error: 4.43
Four coupled superellipses	 (d) error: 2.75	 (e) error: 2.9	 (f) error: 3.07
Four independent superellipses	 (g) error: 2.84	 (h) error: 3.4	 (i) error: 3.49

Figure 4.5 – All nine superellipse fitting results for one example. Row-wise it is the one complete superellipse for the whole contour approach, the for each quadrant one superellipse but coupled over the half axis approach, and finally the one superellipse for each quadrant independently fitted with averaging the half axis parameters afterwards approach. Column-wise it ordered decreasing by computational effort. First the full supperellipse fitting with tapering, then without tapering and finally linearised without tapering.

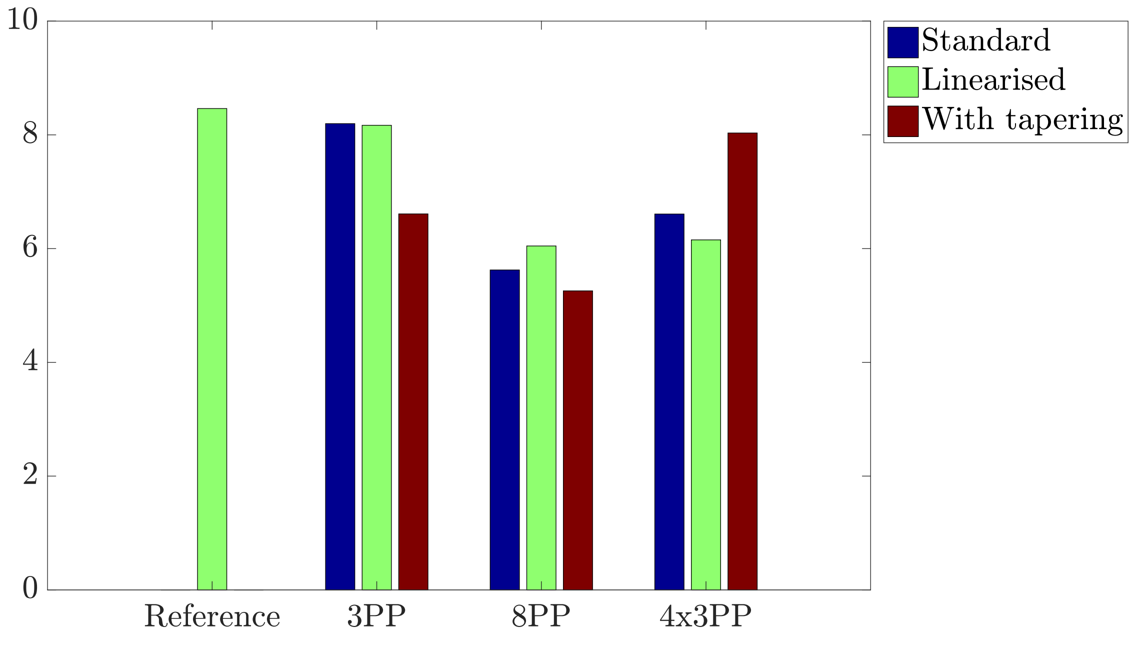
Interestingly, the difference between the Standard, i.e., non-linear fitting without tapering, and the linearised version is each time relatively small and visually barely visible. This suggests that the error introduced by the linearisation is small. To investigate this further, the mean point-wise error was averaged over all results for the 86 contours. Figure 4.6(a) shows the result.

The ellipse starting contour is named reference to show the improvement of introducing superellipses. Another reference point is the three parameter problem (3PP) with Standard since it is the approach used by [18]. As expected, the error drops from the one complete superellipse for the whole contour to the other two approaches. But also just adding tapering improved the fitting accuracy by a good margin. Within the category of four coupled superellipses (8PP), the relative values are as expected. Linearising the approach slightly decreased the accuracy while adding tapering improved the result. For the approach of fitting the four superellipses independently (4x3PP), however, the result is not as expected. Even though the accuracy is always lower than their coupled analogue, it is interesting to see that in the 4x3PP-case linearisation actually increased the accuracy compared to Standard. At the same time adding tapering increased the error significantly. The reason for this is probably the fact that the half axis parameter b is also used in the tapering deformation (cf. (4.3)). So by averaging the half axis parameters not only the geometrical extend is changed, but also the tapering deformation.

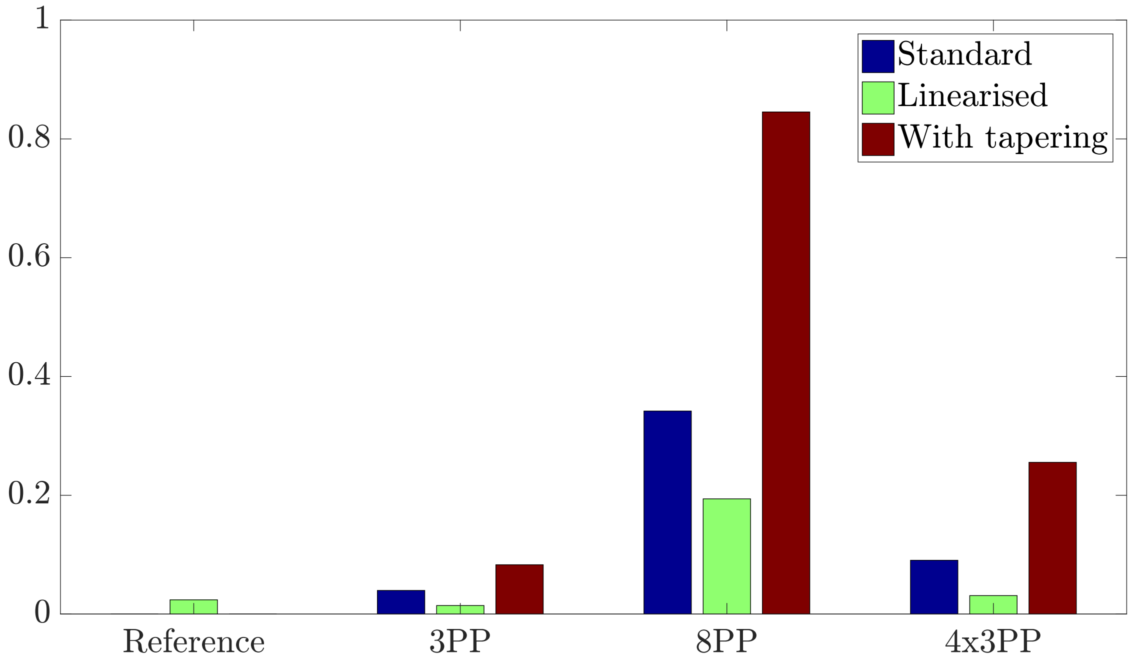
An important factor, besides the accuracy, is the computational effort a fitting algorithm yields. Figure 4.6(b) shows the average computation time for each approach. As desired, the linearisation decreased the computation time for each approach individually. Adding tapering on the other hand increased the computation time, especially for the quadrant-wise approaches. Dividing the contour in the four quadrants and coupling the half axis roughly doubled the computation time. The independent quadrant-wise fitting increased the computation time with respect to the one complete superellipse for the whole contour approach only slightly.

When looking on both the computation time and the fitting accuracy at the same time, it is notable that the quadrant-wise independent fitting has nearly the same low error rate as the coupled quadrant-wise fitting, while at the same time having fast computation of the one complete superellipse for the whole contour approach. Moreover, independent of the used approach the linearisation decreased the computation time significantly, while no noticeable trade-off was made regarding the accuracy. Considering tappering, the choice is not that apparent. In fact, the coupled fitting with tapering achieved the lowest average error. However, the accuracy gain to the second best version without tappering is minimal, while at the same time the computation time roughly doubled. Therefore, if time is a constraint, linearised, quadrant-wise independent fitting without tapering seems to be the best choice, otherwise coupled quadrant-wise fitting with tapering gives a slightly higher accuracy.

Since the data set of real examples is limited, a model contour with added noise is used in a second experiment. The two model contours and for each one a corrupted example can be seen in Figure 4.7. The noise is generated by a random walk of a



(a) Mean distance per point



(b) Mean calculation time in seconds

Figure 4.6 – Result for superellipse fitting on 86 extracted shadow contours. In order to make the individual results comparable the the minimization criteria was divided by the number of contour points.

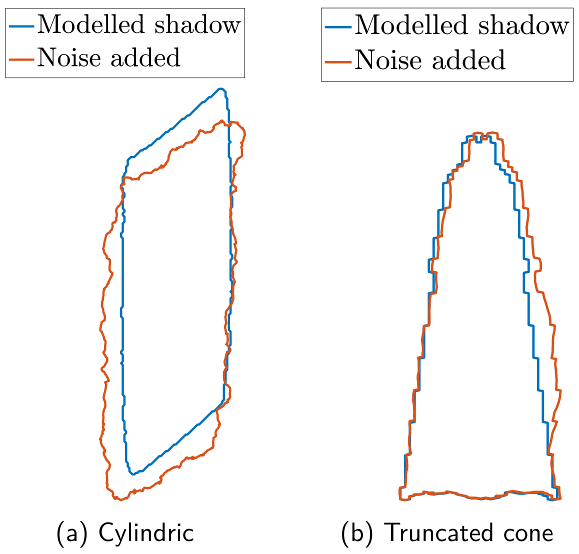


Figure 4.7 – A synthetic shadow is used to generate several instances by adding random noise.

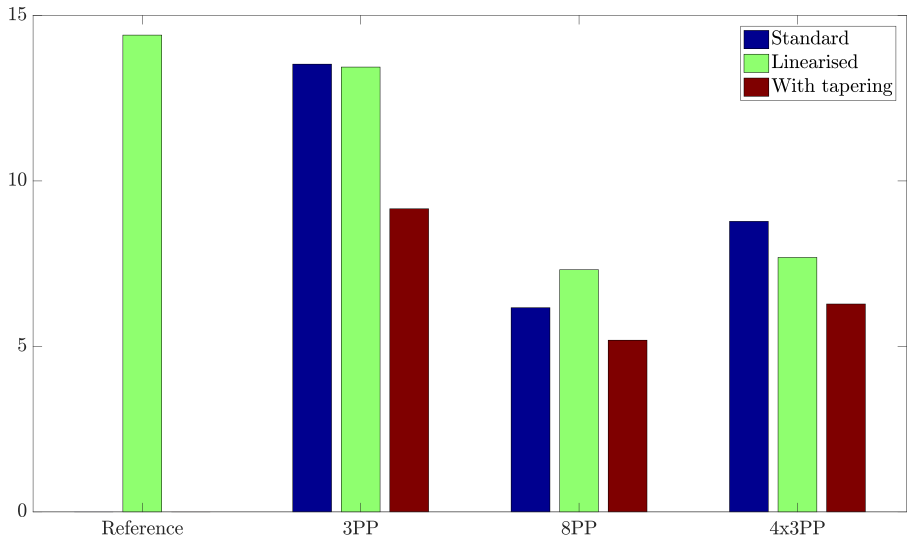
scaled Gaussian pseudorandom variable with mean 0 and standard deviation 1. To smooth the noise, a digital filter is applied. As previously, a PCA is used on every synthetic contour to estimate the translation and rotation.

This time, the real contour is known, thus, after the different fitting algorithms are employed, the real difference between the model contour and the approximation can be calculated. The average results for 1000 runs, 500 for the cylindric and 500 for the truncated cone, can be found in Figure 4.8. There was no qualitative difference between the two types of contours, i.e., the cylindric and truncated cone. So the individual results were averaged.

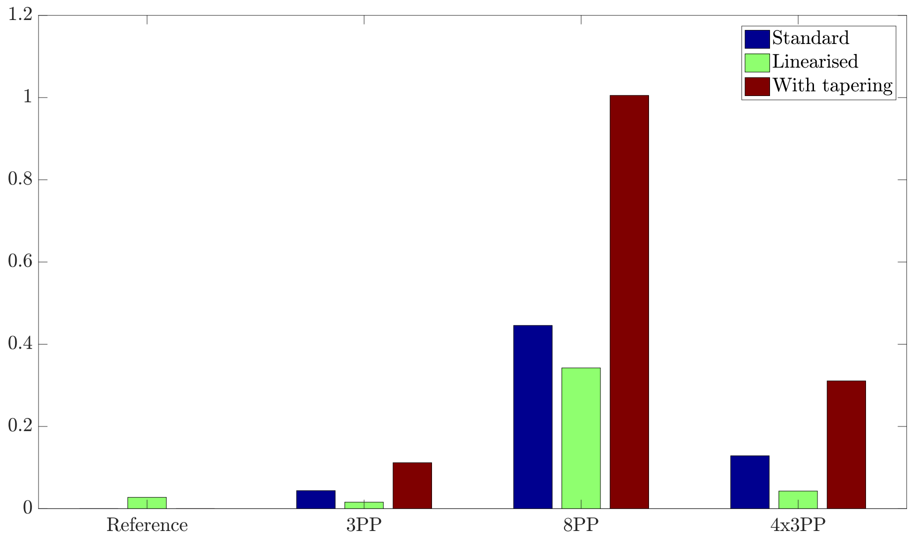
Compared to the results on real contours, the results on the synthetic shadows are for the most part qualitatively the same. Different is the four quadrant independent fitting with tapering which has in contrast to the experiment on real data a better accuracy than the standard or linearised versions. The reason for this lies probably in the symmetry of the synthetic shadows. As stated before, the averaging of the half-axis parameter b has an effect on the shape of the contour if tapering is used, since tapering has b as an input parameter. But since especially for the truncated cone the length of the shadow is approximately equal for the left and right side, averaging b does not change the value that much. Hence, no instability for the tapering is introduced.

4.3.2 Conclusion on superellipse fitting by post-processing

In this section, new methods of fitting a superellipse onto a given contour were presented. Improvements were made with respect to accuracy compared to the standard version which is one complete superellipse fitted to the whole contour.



(a) Mean distance per point



(b) Mean calculation time in seconds

Figure 4.8 – Average result for superellipse fitting on 500 simulated cylindric and 500 simulated truncated cone shadow contours combined. In order to make the individual results comparable the minimization criteria was divided by the number of contour points.

Experiments suggest that with the help of the introduced linearisation, a significant speed up in the computation time can be achieved, while the resulting approximation of the contour is equally accurate. Adding tapering improved the approximation, albeit only slightly. Unfortunately, at the same time the computational effort increased. With this in mind, there are two options. If there is no time constraint and the approximation of the contour has to be as good as possible, one should choose the 8 parameter approach with tapering. However, if a time constraint exists, the linearised independent approach that nearly achieves the same fitting quality for only a fraction of computational effort should be favoured.

Overall, the results of the true contour reconstruction via superellipse fitting as post-processing step are promising. Yet, one problem occurred repeatedly; sometimes the contour extraction failed due to extensive noise. In this cases, recovering the true contour with a superellipse fitting was futile. Consequently, in the next section a way of incorporating the superellipse assumption already in the contour extraction step is discussed.

4.4 Active contours segmentation with superellipse shape constraints

Active contours segmentation is a standard image processing technique based on the assumption that there exists a positive functional that takes the image and a segmentation of the image as input and gives a measure of *goodness* of the segmentation as output. To be more precise, the function shall be zero for the optimal segmentation and greater than zero otherwise with a roughly ordinal scale. When one out of two possible segmentations is significantly worse, the output of the functional corresponding to the better segmentation should be smaller to indicate it is nearer to the perfect segmentation. The segmentation of an image is then a minimisation problem for which the wide body of existing general optimisation techniques can be used.

The core challenge for the implementation of an active contour method is the construction of the functional. There are basically two different approaches to this problem. A good segmentation is often characterised by a contour right on the edge that marks the border between an object and the background. This leads to edge-based active contour models where the functional measures the edge strength on the border between the different segments, e.g., snake models [30]. Unfortunately, there are many imaging techniques where edges can be rather weak. Examples are diagnostic sonography and sonar. On sonar images, especially the shadows often rather fade out than having clear crisp edges. A second alternative aspect of a good segmentation is that the different areas generated by the segmentation should be quite homogeneous. This leads to region-based active contours approaches [9] where the functional measures the homogeneity within the different areas according to some statistics. This family of approaches is more suited for images with poor quality

[68]. Therefore, a region-based active contours model will be presented, since images made underwater usually fall under the poor image quality category independent of the imaging system.

For region-based models, the functional for two regions can be written as

$$f(u,l) = \int_\Omega g_1(u(\boldsymbol{z}))\Theta(l(\boldsymbol{z}))\mathrm{d}\boldsymbol{z} + \int_\Omega g_0(u(\boldsymbol{z}))\Theta(-l(\boldsymbol{z}))\mathrm{d}\boldsymbol{z}. \tag{4.30}$$

As previously stated, u is the image and $\boldsymbol{z} \in \Omega$ is a pixel position in the image domain Ω. Heaviside functions are denoted as Θ. The essential parts of (4.30) are the levelset function l and g_1 and g_0. The level set function implicitly defines the segmenting contour by being the signed distance to the segmenting contour. It is important to note that l is positive within the object region marked by 1 and negative in the background region 0. The functions of the pixel intensities g_1 and g_0, on the other hand, measure the dissimilarity between the pixel intensity at a certain point $\boldsymbol{z}$ and what is expected in the regions 1 or 0.

Figure 4.9 shows an illustration of (4.30) in an ideal case where the functions g_1 and g_0 are 0 when a pixel belongs to the respective area and greater zero otherwise. For simplicity it is set to 1 in this example. For this perfect scenario, the functional f gives the number of pixel on the wrong side of the segmenting contour. Obviously, f then becomes 0 if and only if the contour encloses the object exactly.

The challenge now is to find a suitable g. Actually, if g were given, as it is the case in the simplistic example above, the segmentation task would be trivial. But before going into details how to properly derive suitable estimates of the g, we will firstly adjust (4.30) on this more general level to be more suitable for a wider range of segmentation cases.

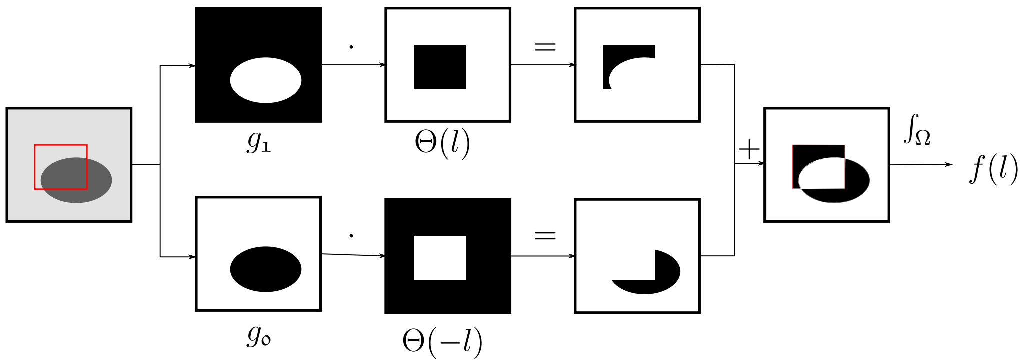


Figure 4.9 – Illustration of (4.30) with ideal g_0 and g_1. On the left we see an example image with a dark grey object, light grey background, and a suggested segmenting contour in red. If g_0 is 0 (white) for background pixel and 1 (black) otherwise and g_1 is 0 (white) for object pixel and 1 (black) otherwise, the functional (4.30) will give the number of wrongly segmented pixels.

Often (4.30) is accompanied by a penalty term that encourages particular properties of the segmenting contour, e.g., one that is smooth. At the same time, there are no constraints on the level set function besides that it is a signed distance function of the segmenting contour. Thus, the contour can have an arbitrary shape albeit with properties that the penalty term targets at. Alternatively, shape constraints can be directly incorporated in the level set function so that a penalty term is not needed. Since we are interested in geometric shapes, the level set function can be a signed distance function to a given superellipse. No penalty term is needed.

Furthermore, the two-region-segmentation given by (4.30) can be modified to be able to extract more than one object simultaneously [78]. The disadvantage of having only one object contour when there are actually two different object is as follows. Assuming that two distinct objects can be found in an image and one is considered as area 1, with (4.30) the function g_0 now has to model background and the second object at the same time. With simple models this will probably fail. But if instead, there is a g_{00} for the background, a g_{01} for the first object and a g_{10} for the second object, each g could most likely easier represent its own area. Generally speaking, in order to divide the image in 2^n regions, n level set functions need to be used. To demonstrate this, two level set functions are used from here. This modifies (4.30) to

$$\begin{aligned} f(u, l_0, l_1) = \int_\Omega & g_{11}(u(\boldsymbol{z}))\Theta(l_0(\boldsymbol{z}))\Theta(l_1(\boldsymbol{z}))\mathrm{d}z \\ & + \int_\Omega g_{10}(u(\boldsymbol{z}))\Theta(l_0(\boldsymbol{z}))\Theta(-l_1(\boldsymbol{z}))\mathrm{d}z \\ & + \int_\Omega g_{01}(u(\boldsymbol{z}))\Theta(-l_0(\boldsymbol{z}))\Theta(l_1(\boldsymbol{z}))\mathrm{d}z \\ & + \int_\Omega g_{00}(u(\boldsymbol{z}))\Theta(-l_0(\boldsymbol{z}))\Theta(-l_1(\boldsymbol{z}))\mathrm{d}z. \end{aligned} \tag{4.31}$$

With this setting, the image is divided into four regions 00 = background, 01 = object 1, and 10 = object 2. Additionally, since the positive areas of the level set functions can intersect, there is 11 = overlap. Those areas correspond to the signs of $l_0(\boldsymbol{z}) = l(\boldsymbol{\nu}_0, \boldsymbol{z})$ and $l_1(\boldsymbol{z}) = l_1(\boldsymbol{\nu}_1, \boldsymbol{z})$ as shown in Figure 4.10. The parameter vectors $\boldsymbol{\nu}_0$ and $\boldsymbol{\nu}_1$ are the parameters that define the parametrised level set function. Here those are the parameters of the superellipses leading to the respective zero level set (see (4.5)). But of course, also any other parametrised level set function could be used if it better suits the application. Please note that the area 11 may be empty if the superellipses for object 1 and object 2 do not intersect.

As already mentioned from a more abstract viewpoint, we deal with the minimisation of a functional with respect to the functions l_0 and l_1 or, since parametrised level set functions are used, with respect to the parameters $\boldsymbol{\nu}_0$ and $\boldsymbol{\nu}_1$. The gradient descent equation to find the minimising $\boldsymbol{\nu}_0$ and $\boldsymbol{\nu}_1$ is given by

$$\boldsymbol{\nu}_i^{n+1} = \boldsymbol{\nu}_i^n - \lambda \nabla_{\boldsymbol{\nu}_i} f \text{ for } i = 0, 1 \tag{4.32}$$

where λ is the step size of the iteration. Strictly speaking, λ is a scalar and equal for both i as in the definition. But due to the different nature of the variables it

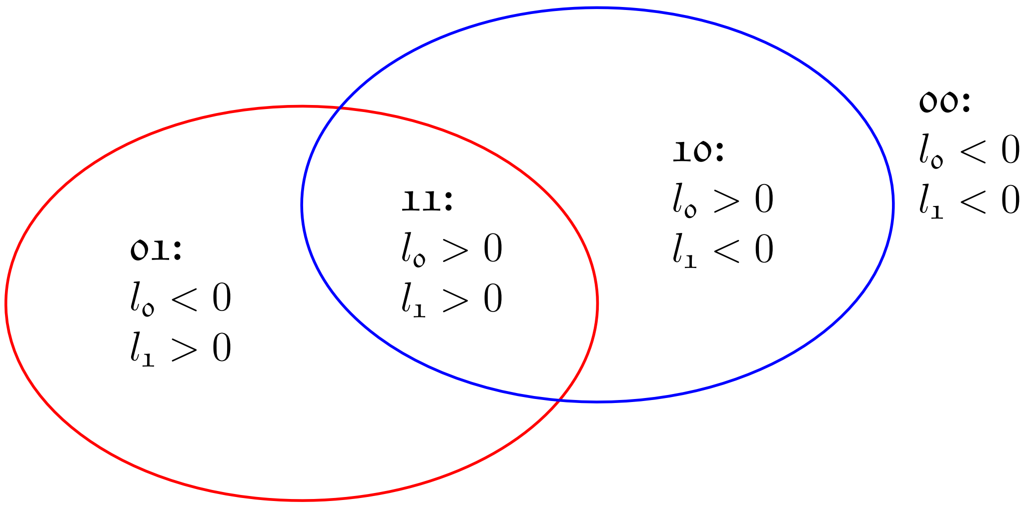


Figure 4.10 – The image gets segmented into four areas according to the signs of the level set functions. The red line indicates $l_1 = 0$ and the blue line is $l_0 = 0$.

was favourable to apply individual step sizes for the different deformations. The derivative of f is

$$\nabla_{\nu_0} f = \int_\Omega \left[(g_{11} - g_{01})\,\Theta(l_1) + (g_{10} - g_{00})\,\Theta(-l_1)\right]\delta(l_0)\nabla_{\nu_0} l_0 \mathrm{d}\boldsymbol{z} \tag{4.33}$$

and

$$\nabla_{\nu_1} f = \int_\Omega \left[(g_{11} - g_{10})\,\Theta(l_0) + (g_{01} - g_{00})\,\Theta(-l_0)\right]\delta(l_1)\nabla_{\nu_1} l_1 \mathrm{d}\boldsymbol{z} \tag{4.34}$$

with δ is the Dirac function. Coming from the abstract mathematical view back to the segmentation level, this means that iteratively the contour is pushed and transformed towards a better segmentation with respect to f. This connection between segmentation and the gradient descend equation will become clearer once (4.33) and (4.34) are slightly modified at end of Section 4.4.2.

For now, (4.33) and (4.34) are still not usable since neither the weight functions g nor the level set functions l are defined. We will start with the level set function.

4.4.1 Superellipse level set function

As stated previously, the goal is to extract superellipses. Consequently, the level set function used is not defined over an arbitrary contour but by

$$l(\boldsymbol{z}, \boldsymbol{\nu}) = \tilde{d}(a, b, \varepsilon) \circ q_{tap}^{-1}(\tau) \circ q_{trans}^{-1}(T) \circ q_{rot}^{-1}(\theta)(\boldsymbol{z}) \tag{4.35}$$

with $\boldsymbol{\nu} = \{a, b, \varepsilon, \tau, T, \theta\}$ as before. The q's denote the transformations tapering, translation and rotation. Please note that the transformations are inverted. The signed distance function $\tilde{d}$ measures the signed distance between a given superellipse and the *un*-transformed coordinates $\tilde{\boldsymbol{z}}$ of a point $\boldsymbol{z}$. For the rest of this section it is assumed that the inverted transformations have already been applied and just use $\boldsymbol{z}$ to keep the notation less cumbersome. The point to contour distance is typically defined as the signed distance between $\boldsymbol{z}$ and the nearest point on the contour

$$\tilde{d}(\boldsymbol{z}) = \begin{cases} d(\boldsymbol{z}, C^s) & \text{if } \boldsymbol{z} \text{ is inside of } C^S \\ -d(\boldsymbol{z}, C^s) & \text{else,} \end{cases} \tag{4.36}$$

where d is as before the Euclidean distance. As previously stated in Section 4.3, there is unfortunately no closed-form solution for the distance to a superellipse and an iterative approach is too expensive in terms of computation. Preliminary test again showed that the ray-to-centre approach is the superior choice to approximate the distance between $\boldsymbol{z}$ and C^s by the distance between $\boldsymbol{z}$ and $\boldsymbol{z}^*$ (see Figure 4.2). The reason for this is due to the fact that the Dirac function in (4.48) and (4.49) needs to be regularized to get meaningful results. Otherwise, nearly no discrete pixel would be evaluated since virtually none is exactly on the zero level set. For the regularization we follow the example in [9] and replace Θ by

$$\Theta_\xi(l) = \left(\frac{1}{2}\left(1 + \frac{2}{\pi}\arctan\left(\frac{l}{\xi}\right)\right)\right) \tag{4.37}$$

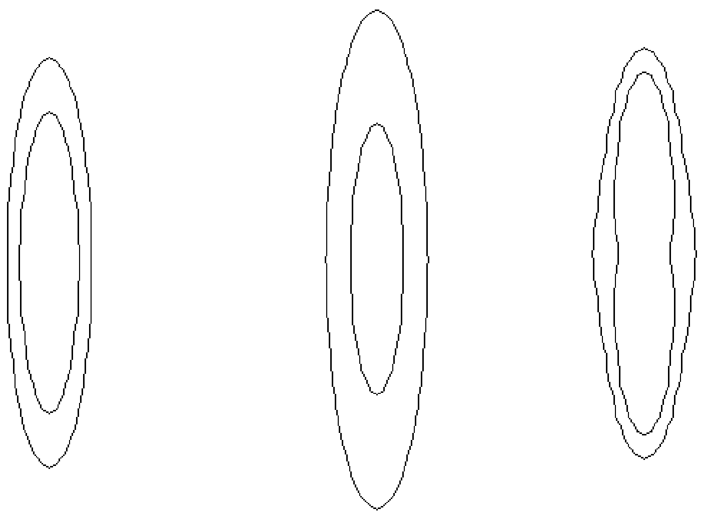
and accordingly δ by

$$\delta_\xi(l) = \frac{\mathrm{d}\Theta_\xi(l)}{\mathrm{d}l} = \frac{1}{\pi}\frac{\xi}{\xi^2 + l^2}. \tag{4.38}$$

Those regularizations now enable the algorithm to also consider pixels close to the zero level. Their weight or impact on (4.48) and (4.49), however, is governed by δ_ξ. In an ideal case, all points with the same distance to the zero level set would get the same value after applying the regularized Dirac function. However, since approximations for the distance function are used, this is not true anymore. Figure 4.11 illustrates the effect of the different distance approximations. In the first and second case the up down direction is significantly broader then the left right direction. For the segmentation this means that the up down direction has a broader area that impacts resulting values in (4.33) and (4.34). Here the third approach which is the ray-to-centre distance approximation - albeit not perfect - has not a so big directional bias. This is why $\tilde{d}$ is defined via d_γ

$$\tilde{d}(\boldsymbol{z}) = \begin{cases} d_\gamma(\boldsymbol{z}, C^s) & \text{if } \boldsymbol{z} \text{ is inside of } C^S, \\ -d_\gamma(\boldsymbol{z}, C^s) & \text{else.} \end{cases} \tag{4.39}$$

As discussed in Section 4.2.1 there are two ways to calculate d_γ, namely via (4.12) or (4.13). However, the computation of the derivatives of the subsequent d_γ for the evolution equations (4.48) and (4.49) tend to be unstable if the approach of (4.12) that leads to an $\tan^{-1}$ is followed. So, even though both lines of action lead in theory to exactly the same result, the second one corresponding to (4.13) is used since the derivative is numerically more stable. An additional bonus is that it is computationally faster.



(a) $\delta_\xi(d_\alpha) = 0.1$ (b) $\delta_\xi(d_\beta) = 0.1$ (c) $\delta_\xi(d_\gamma) = 0.1$

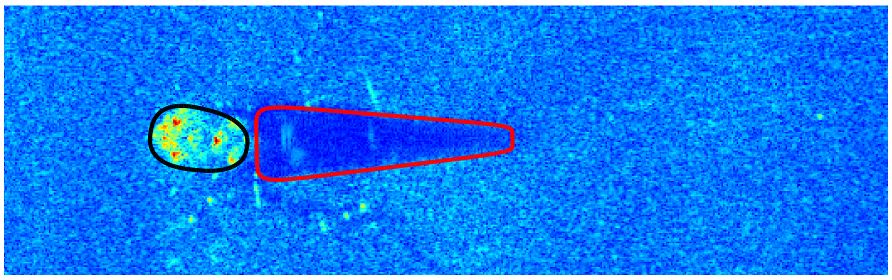
Figure 4.11 – The lines indicate where the regularized Dirac function equals 0.1 for the three different level set functions for the same superellipse.

4.4.2 Weight functions

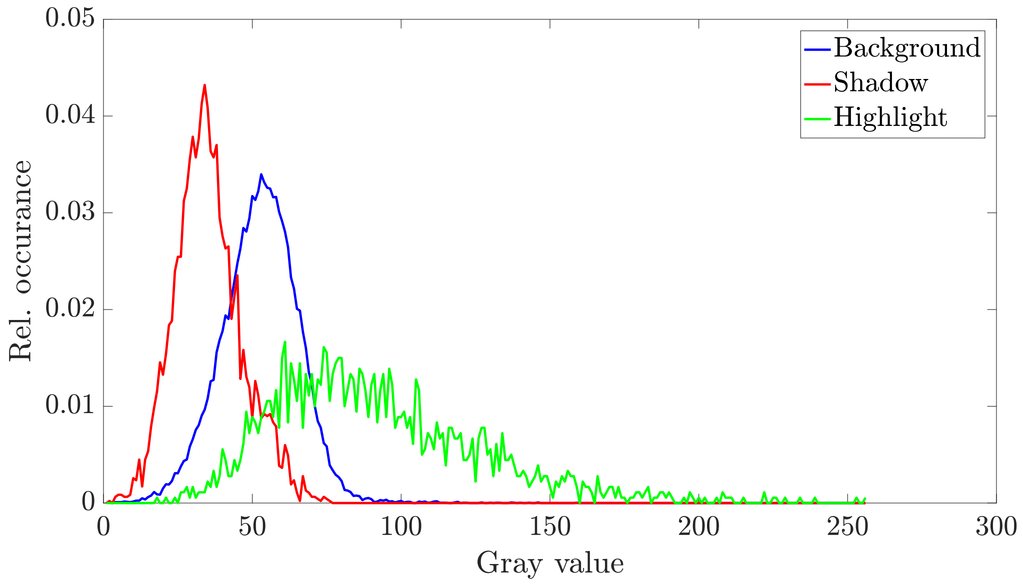
The standard version of the weight function or dissimilarity function for area based active contours algorithms is given by

$$g_{\mathfrak{a}}(u) = \|u - \hat{\mu}_{u,\mathfrak{a}}\|^2 \tag{4.40}$$

where again $u = u(\boldsymbol{z})$ is the pixel intensity at position $\boldsymbol{z}$ and $\hat{\mu}_{u,\mathfrak{a}}$ is the estimated average pixel intensity for each respective area $\mathfrak{a} = \{00, 01, 10, 11\}$. Usually, the real average pixel intensity $\mu_{u,\mathfrak{a}}$ is not given. Thus, an estimation based on the pixel intensities that are *currently* in the respective areas is used. This formulation is especially suited for applications where each object and the background have a distinct average pixel intensity with little variance. Wrongly segmented pixels will have a big squared distance to the mean, while correct segmented pixels are close to the mean. Unfortunately, this is not always the case. Looking at the example of a SAS image, if this applies then the histograms for the areas highlight, shadow, and background would need to have three narrow spikes that do not overlap. Especially the highlight area in sonar images usually features a broad range of pixel intensities. Background and shadow pixel intensities have a smaller variance, but tend to overlap. Figure 4.12 shows the typical normalized histograms of the different image regions. To investigate possible substitutions to the standard weight function (4.40), the



(a) Segmentation



(b) Normalized Histograms

Figure 4.12 – An example segmentation of an object on a SAS image with a black boundary around the highlight and a red boundary along the shadow region is shown on top. A normalised histogram of each individual area can be seen below.

equation can be reformulated as

$$g_{\mathfrak{a}}(u) = \|u - \hat{\mu}_{u,\mathfrak{a}}\|^2 = -\ln\left(\exp\left(-(u-\hat{\mu}_{u,\mathfrak{a}})^2\right)\right) = -\ln\left(\exp\left(-\frac{(u-\hat{\mu}_{u,\mathfrak{a}})^2}{2\cdot\frac{1}{2}}\right)\right). \tag{4.41}$$

Looking at the argument of the logarithm, it can be seen that it is a Gaussian density with variance $\sigma^2_{u,\mathfrak{a}} = 1/2$ and without the normalization factor, i.e., the kernel of a Gaussian density. With this insight and the illustration of the normalized histograms in Figure 4.12, it becomes obvious that at least in theory the standard weight function (4.40) is not suitable. Even a three standard deviation neighbourhood would only encompass five pixel intensities since $\sigma_{u,\mathfrak{a}} \approx 0.7$. Thus, in order to model the potentially broader range of pixel intensities in the different sonar image areas, $\sigma^2_{u,\mathfrak{a}}$ would need to be bigger and ideally not equal for each $\mathfrak{a}$. Instead of choosing fixed values that would fit in a specific case but could change for the next image, the variance can also be inferred from the data. To do so, a Gaussian distribution is assumed

$$p(u|\hat{\mu}_{u,\mathfrak{a}}, \hat{\sigma}^2_{u,\mathfrak{a}}) = \frac{1}{\sqrt{2\pi\hat{\sigma}^2_{u,\mathfrak{a}}}} \exp\left\{-\frac{(u-\hat{\mu}_{u,\mathfrak{a}})^2}{2\hat{\sigma}^2_{u,\mathfrak{a}}}\right\}, \tag{4.42}$$

where the parameter $\hat{\sigma}^2_{u,\mathfrak{a}}$ and $\hat{\mu}_{u,\mathfrak{a}}$ can be calculated with maximum log-likelihood estimation

$$\hat{\mu}_{u,\mathfrak{a}} = \frac{1}{N_{\mathfrak{a}}} \sum_{z\in\omega_{\mathfrak{a}}} u, \tag{4.43}$$

$$\hat{\sigma}^2_{u,\mathfrak{a}} = \frac{1}{N_{\mathfrak{a}}} \sum_{z\in\omega_{\mathfrak{a}}} (u-\hat{\mu}_{u,\mathfrak{a}})^2. \tag{4.44}$$

Thus, a definition like

$$g_{\mathfrak{a}}(u) = -\ln\left(p(u|\hat{\mu}_{u,\mathfrak{a}}, \hat{\sigma}^2_{u,\mathfrak{a}})\right) = -\ln\left(p(u|\mathfrak{a})\right) \tag{4.45}$$

could be made, i.e., $g_{\mathfrak{a}}$ gives the negative logarithm of the probability that a pixel intensity u occurs in area $\mathfrak{a}$. Compared to the standard definition (4.40) it is now possible to take into account the variance of the pixel intensities considered part of a certain region. Still, there are disadvantages. Figure 4.13(a) shows example approximations of the pixel intensity distributions via Gaussian probability density functions. Recalling that $g_{\mathfrak{a}}$ should be 0 for a pixel intensity that belongs in the respective region and greater then 0 otherwise, seems to be fulfilled at first glance. But if for example u is a high pixel intensity like 200, it can safely be assumed that it represents a highlight pixel, even though the concrete value 200 might appear rarely in the highlight. Hence, $g_{\mathfrak{a}}$ should return a small value or in best case zero. But $p(200|\mathfrak{10})$ itself is nearly zero. Consequently, the negative logarithm of $p(200|\mathfrak{10})$ is far bigger than zero. The same happens at the lower end of the pixel intensities for the shadow, albeit not that extreme. Moreover, even at the mode of the distribution the value is fairly low as can be seen in Figure 4.12(b). So even though the change from a fixed variance in the standard version to a data inferred one made it possible to encompass a broader range of values, boundary cases and the mode are not properly handled. With $p(u|\mathfrak{a})$ one piece of information is neglected; there is no *unknown* area or with other words

$$p(\mathtt{00}|u) + p(\mathtt{10}|u) + p(\mathtt{01}|u) + p(\mathtt{11}|u) = 1 \quad \forall u. \tag{4.46}$$

This train of thought leads to a modification of (4.45) to

$$g_{\mathfrak{a}}(u) = -\ln\left(p(\mathfrak{a}|u)\right). \tag{4.47}$$

Figure 4.13 illustrates the implications of this difference. Continuing the earlier example, $p(\mathtt{10}|200)$ is nearly 1. Hence, $g_{\mathfrak{a}}$ will be nearly 0 as expected. Moreover, the heights of the modes are also in a much more reasonable range.

Substituting the general g with (4.47) in (4.33) and (4.34) and using the Bayes Theorem gives

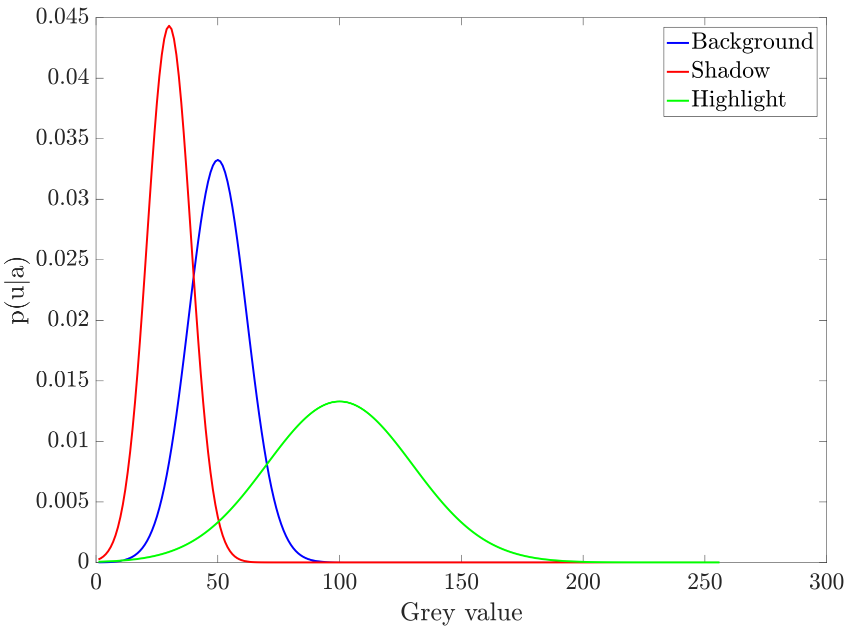
$$\nabla_{\nu_0} f = \int_\Omega \left[\ln\left(\frac{p(u|\mathtt{01})p(\mathtt{01})}{p(u|\mathtt{11})p(\mathtt{11})}\right)\Theta(l_1) + \ln\left(\frac{p(u|\mathtt{00})p(\mathtt{00})}{p(u|\mathtt{10})p(\mathtt{10})}\right)\Theta(-l_1)\right]\delta(l_0)\nabla_{\nu_0} l_0 \mathrm{d}\boldsymbol{z} \tag{4.48}$$

and

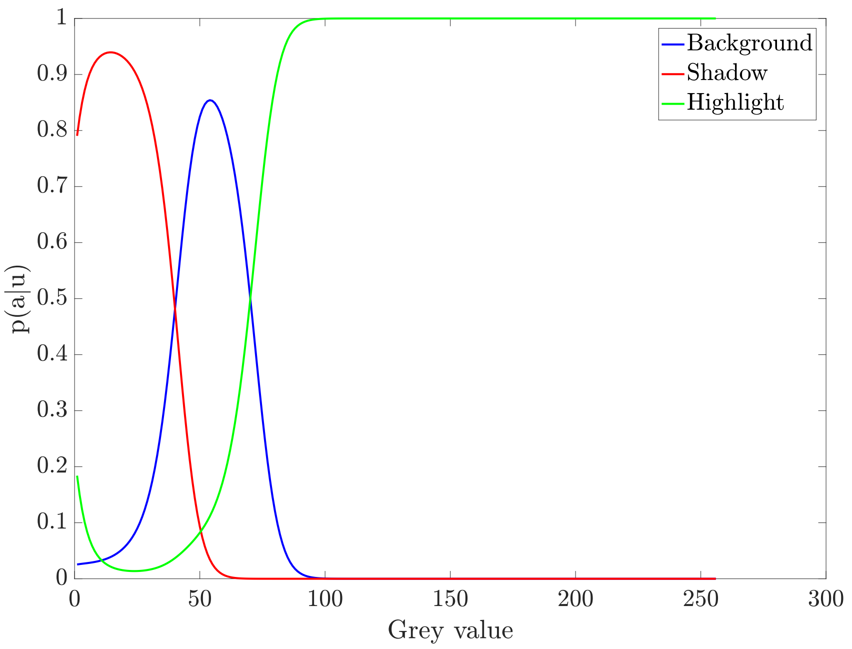
$$\nabla_{\nu_0} f = \int_\Omega \left[\ln\left(\frac{p(u|\mathtt{10})p(\mathtt{10})}{p(u|\mathtt{11})p(\mathtt{11})}\right)\Theta(l_0) + \ln\left(\frac{p(u|\mathtt{00})p(\mathtt{00})}{p(u|\mathtt{01})p(\mathtt{01})}\right)\Theta(-l_0)\right]\delta(l_1)\nabla_{\nu_0} l_1 \mathrm{d}\boldsymbol{z}. \tag{4.49}$$

Applying the logarithmic laws changed the subtraction to a division and $p(u)$ vanished since it is in both numerator and denominator. The probability of being in a certain area $p(\mathfrak{a})$ can either be assumed to be the same for each region, which has the effect that these probabilities vanish from the equation, or it can be used to favour a certain area over another, e.g., by setting $p(\mathtt{11})$ to (nearly) 0 to avoid overlapping contours.

The equations (4.48) and (4.49) were derived from a mathematical view point of a minimisation problem and might seem disconnected from the purpose of segmentation. To motivate that the derivations are also reasonable in this context, one can take a closer look at (4.48). In the minimum, i.e., when the best segmentation with respect to the chosen cost is found, the derivative (4.48) should be 0. The integral kernel is 0 for most points in Ω since the Dirac functions limits the potential non-zero entries to the zero-level of l_0. Hence, only the points on the current contour represented by l_0 are relevant. The value for each of those points depends on whether the two superellipses intersect $\Theta(l_1)\delta(l_0)$ (see blue line in Figure 4.14) or not $\Theta(-l_1)\delta(l_0)$ (see



(a) $p(u|\mathfrak{a})$



(b) $p(\mathfrak{a}|u)$

Figure 4.13 – Comparison between using $p(u|\mathfrak{a})$ and $p(\mathfrak{a}|u)$. High pixel intensities belong most likely to the highlight. The $p(u|\mathfrak{a})$ approach does not reflect this which in turn would result in a high g_{10} value, even though it should be near zero. In this example it is assumed that there is no overlap area 11.

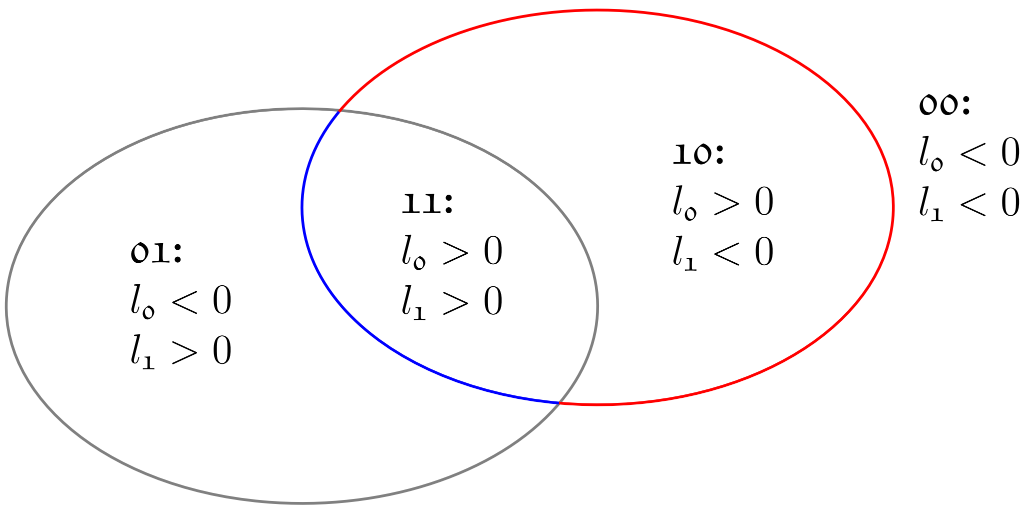
red line in Figure 4.14). Since the natural logarithm is only 0 for input 1, it can be seen that in order to have a zero derivative of f,

$$p(u|\mathfrak{01})p(\mathfrak{01}) \overset{!}{=} p(u|\mathfrak{11})p(\mathfrak{11}) \Leftrightarrow p(\mathfrak{01}|u) \overset{!}{=} p(\mathfrak{11}|u) \quad \text{for } l_{\mathfrak{0}}(\boldsymbol{z}) = 0,\ l_{\mathfrak{1}}(\boldsymbol{z}) > 0 \tag{4.50}$$

and

$$p(u|\mathfrak{00})p(\mathfrak{00}) \overset{!}{=} p(u|\mathfrak{10})p(\mathfrak{10}) \Leftrightarrow p(\mathfrak{00}|u) \overset{!}{=} p(\mathfrak{10}|u) \quad \text{for } l_{\mathfrak{0}}(\boldsymbol{z}) = 0,\ l_{\mathfrak{1}}(\boldsymbol{z}) < 0. \tag{4.51}$$

In other words, this is the case when the contour is on the line where the probability for belonging to one of the two adjacent areas is equal. The analogue is true for (4.49). This clearly fits with the idea that the segmenting contour should be right at the border between the different areas.



$\mathfrak{00}$: $l_{\mathfrak{0}} < 0$ $l_{\mathfrak{1}} < 0$
$\mathfrak{10}$: $l_{\mathfrak{0}} > 0$ $l_{\mathfrak{1}} < 0$
$\mathfrak{11}$: $l_{\mathfrak{0}} > 0$ $l_{\mathfrak{1}} > 0$
$\mathfrak{01}$: $l_{\mathfrak{0}} < 0$ $l_{\mathfrak{1}} > 0$

Figure 4.14 – Only the points on the blue line, i.e., where $\Theta(l_{\mathfrak{1}})\delta(l_{\mathfrak{0}}) = 1$, and the red line, i.e., where $\Theta(-l_{\mathfrak{1}})\delta(l_{\mathfrak{0}})$, are evaluated.

4.4.3 Probability density function estimation

In the previous section, the standard distance to mean approach for the $g_{\mathfrak{a}}$ (4.40) was analysed and then substituted with a more suitable probabilistic approach that used Gaussian distributions to approximate the pixel intensity distributions. Eventually, this led to the equations (4.48) and (4.49). Re-examining the steps taken, it becomes obvious that the only reason the essential part of $g_{\mathfrak{a}}$ is defined with a Gaussian distribution, is because it is close to the standard version. To compute the evolution steps in (4.48) and (4.49) $p(u|\mathfrak{a})$ is needed. The only requirement is that $p(u|\mathfrak{a})$ models the pixel intensity distributions of region $\mathfrak{a}$. But this does not necessarily imply a Gaussian distribution. Actually, it does not even has to be the same distribution to model all four areas. Hence, any reasonable distribution that could after sampling result in graphs as can be seen in Figure 4.12 can be used to model the pixel intensities. As before, the real distribution is not given. So it or its parameters have to be estimated based on the segmentation of the current iteration.

With respect to what distribution model to apply, the first idea can be to use the normalized histogram directly. Especially for the background of an image, there should be enough good samples to properly represent the true distribution. Still, for the most likely much smaller object regions, this approach might not be the best choice. The smaller sample size combined with the probably non-optimal starting position could skew the approximation. In early iterations the contour of one object could overlap with the area of another object which leads to a fragmented $p(\mathfrak{a}|u)$. Instead of having three separate intervals of representing object 1, object 2 and background as expected, they would be intermixed.

An alternative to the normalised histogram is the use of parameter-based models $p(\cdot|\hat{\eta}_\mathfrak{a})$ where the sampled pixel intensities are used to derive the parameters $\hat{\eta}_\mathfrak{a}$ via maximum log-likelihood estimation. To keep the symbols less cumbersome $\eta = \hat{\eta}_\mathfrak{a}$ or equivalent will be used from here on for the parameters. But please keep in mind that all the derived parameters are just for one region $\mathfrak{a}$ at a time and just estimates based on the samples from the current segmentation.

The use of a Gaussian distribution was already discussed as it is an intuitive reformulation and extension of the standard weight function. Yet, since the segmentation method is intended for a wide variety of image types, alternative probability density functions (pdfs) related to the respective imaging system can be considered besides the Gaussian distribution. Since the main evaluation data set consists of SAS images, the literature was searched for suitable pdfs for the SAS context. The first candidate is the Gamma distribution as suggested as the underlying pixel intensity distribution in [6] which is given by

$$p(u|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp\{-\beta u\} \tag{4.52}$$

where Γ is the gamma function. Solving the maximum log-likelihood equation for the maximising α and β yields

$$\frac{\alpha}{\beta} = \frac{1}{N_\mathfrak{a}} \left(\sum_{z\in\omega_\mathfrak{a}} u \right) \tag{4.53}$$

and

$$\Psi(\alpha) - \ln(\beta) = \frac{1}{N_\mathfrak{a}} \left(\sum_{z\in\omega_\mathfrak{a}} \ln(u) \right), \tag{4.54}$$

where Ψ is the logarithmic derivative of the gamma function, also called Digamma function. Solving (4.53) for β and inserting into (4.54) leads to

$$\Psi(\alpha) - \ln(\alpha) = \frac{1}{N_\mathfrak{a}} \sum_{z\in\omega_\mathfrak{a}} \ln(u) - \ln\left(\frac{1}{N_\mathfrak{a}} \sum_{z\in\omega_\mathfrak{a}} u \right). \tag{4.55}$$

There is no closed form solution to (4.55) but the Newton-Raphson method can be used to get the iteration scheme

$$\alpha_t = \alpha_{t-1} - \frac{\Psi(\alpha_{t-1}) - \ln(\alpha_{t-1}) - \frac{1}{N_\mathfrak{a}} \sum_{z \in \omega_\mathfrak{a}} \ln(u) + \ln\left(\frac{1}{N_\mathfrak{a}} \sum_{z \in \omega_\mathfrak{a}} u\right)}{\Psi(\alpha'_{t-1}) - \frac{1}{\alpha_{t-1}}}. \tag{4.56}$$

For a starting value, the left hand side of (4.55) can be substituted for the approximation

$$\Psi(\alpha) - \ln(\alpha) \approx -\frac{1}{2\alpha}\left(1 + \frac{1}{6\alpha + 1}\right) \tag{4.57}$$

so that the resulting quadratic equation can be solved for α.

Alternatively, the Rayleigh distribution is also sometimes used to describe the pixel intensity distribution in sonar images [4]. It is given by

$$p(u|\lambda) = \frac{u}{\gamma^2} \exp\left\{\frac{-u^2}{2\gamma^2}\right\}. \tag{4.58}$$

The maximum log-likelihood estimation of γ^2 is

$$\gamma^2 = \frac{2}{N}\left(\sum_{z \in \omega_\mathfrak{a}} u^2\right). \tag{4.59}$$

As third and last option, the Weibull distribution can be considered, which is a close relative of the Rayleigh distribution. It is given by

$$p(u|\upsilon, \varpi) = \frac{\varpi}{\upsilon}\left(\frac{u}{\upsilon}\right)^{\varpi - 1} \exp\left\{-\left(\frac{u}{\upsilon}\right)^{\varpi}\right\}. \tag{4.60}$$

The maximum likelihood estimation yields

$$\upsilon^\varpi = \frac{1}{N_\mathfrak{a}} \sum_{z \in \omega_\mathfrak{a}} u^\varpi \tag{4.61}$$

and

$$\frac{1}{\varpi} = \frac{\sum_{z \in \omega_\mathfrak{a}} u^\varpi \ln u}{\sum_{z \in \omega_\mathfrak{a}} u^\varpi} - \frac{1}{N_\mathfrak{a}} \sum_{z \in \omega_\mathfrak{a}} \ln u, \tag{4.62}$$

which again needs to be solved iteratively. In most cases even a simple trial and error approach with linear interpolation is enough [11].

4.4.4 Probability density function selection

For the theoretical analysis, it was advantageous to look at the algorithm from a more abstract point of view. But to actually use it for segmentation it has to be assessed which pdfs are the most suitable for each area respectively. In Section 4.4.3 five options to model $p(u|\mathfrak{a})$ were introduced: normalized histogram, Gaussian distribution, Gamma distribution, Rayleigh distribution, and Weibull distribution. For the proposed segmentation approach, it has to be decided which pdf is best suited to model the respective areas of the image.

Since real data is used for these experiments where no information about the real pixel intensity distributions is available, an alternative has to be used. As preparation for determining the most suitable combination all combinations of pdfs for the different areas in the image are generated. Then the proposed algorithm is applied with each combination of pdfs on every image. That means there are *number-of-pdf-combinations* segmentations per image. To evaluate now which pdf combination is the most suitable, two methods come to mind.

The first idea is that the cost function (4.31) can be used to evaluate a pdf combination at the end of the segmentation. If a specific model of the pixel intensity distribution is a good fit to the actual pixel intensities in a region, the respective $g_{\mathfrak{a}}(u)$ should be small for all pixels in this region. If the assumed distribution is unable to properly represent the real distribution, the $g_{\mathfrak{a}}(u)$ should return high values even in the final iteration. A comparison of the mean final cost values over all segmentations, i.e., fixed pdf combination averaged over all samples, can hence be used to rank the different pdf combinations.

As a second option, the resulting superellipse parameters η_0 and η_1 for a specific pdf combination can be taken as input parameters to train a naive Bayes classification. Given η_0 and η_1 it predicts which object is depicted on an image, i.e., a cylindrical object, a truncated cone, a wedge shaped object, or a rock. The estimated prediction error of the classifier is used as measure to judge the specific $p(u|C_a)$ combination. The reasoning behind this is that a good segmentation is an important prerequisite for a successful classification. Hence, if the classification is successful, this prerequisite should be fulfilled and it implicitly suggests that the choice for the pdfs was good.

Our data set contains $N = 213$ SAS images each showing an object of one of the $Y = 4$ target classes. For every image the ground truth $y_i \in \{$cylindrical object, truncated cone, wedge shaped object, rock$\}$, $i = 1, \dots, N$ is known. There is one label for the whole image. Since some target classes have only a few observations, a bootstrap approach is used [27]. From the data set consisting of N example images, M samples are uniformly drawn with replacement, with $M = N$. On this bootstrap sample set a classifier is trained. This is repeated B times. This means there are B training sets B_b with $b = 1, \dots, B$. Applying a naive Bayes classifier training on each training set yields B classifiers and, consequently, B class labels $\widehat{y}_i^b$ with $b = 1, \dots, B$ for the i-th image. A new sample can now be classified by majority vote.

For the prediction error estimation the ".632+" estimator $\widehat{Err}^{(.632+)}$ is used [27].

For this purpose, the leave-one-out bootstrap error $\widehat{Err}^{(1)}$ is calculated. The leave-one-out bootstrap error is computed by first calculating the average loss L_i for the i-th example image by classifying it with the bootstrap classifiers that were *not* trained with example image i. Assuming that $B^{-i} \subset \{1, \ldots, B\}$ contains the indices of those classifiers that were not trained with the i-th example, that $|B^{-i}|$ is the cardinality of B^{-i}, and that the indicator function $\mathbf{1}$ is used as loss function.

$$\mathbf{1}(u, v) = \begin{cases} 0 & u = f \\ 1 & u \neq v \end{cases} \tag{4.63}$$

Then the average loss L_i for the i-th example is given by

$$L_i = \frac{1}{|B^{-i}|} \sum_{b \in B^{-i}} \mathbf{1}(y_i, \widehat{y}_i^b). \tag{4.64}$$

After having the average loss for the i-th example image, the leave-one-out bootstrap error $\widehat{Err}^{(1)}$ is given by averaging over all N example images.

$$\widehat{Err}^{(1)} = \frac{1}{N} \sum_{i=1}^{N} L_i = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|B^{-i}|} \sum_{b \in B^{-i}} \mathbf{1}(y_i, \widehat{y}_i^b) \tag{4.65}$$

It is ensured that the estimate does not suffer from overestimating the classifiers performance by evaluating the classifiers only with samples that were not used for training and by not using the whole data set for training. Due to the random sampling with replacement a certain image i has the probability of

$$p\{\text{sample } i \in B_b\} = 1 - \left(1 - \frac{1}{N}\right)^M \approx 0.632 \tag{4.66}$$

to be in a certain training set B_b. Consequently, $|B^{-i}| \approx 0.368B$.

Actually, the risk is that $\widehat{Err}^{(1)}$ is too high. Each classifier is trained on $\approx 0.632M$ unique samples. The performance difference between a classifier trained with the full training data set consisting of M unique examples compared to one trained with only $\approx 0.632M$ unique samples could be significant. Assuming $\widehat{y}_i$ is the class label from a classifier trained on the full data set. Then the training error $\overline{err}$ is given by

$$\overline{err} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}(y_i, \widehat{y}_i). \tag{4.67}$$

The authors of [19] discussed that the true error is somewhere between $\widehat{Err}^{(1)}$ and the training error $\overline{err}$ of the full data set. The point $\widehat{w}$ between $\widehat{Err}^{(1)}$ and $\overline{err}$ can be estimated via the *no-information error rate* γ

$$\widehat{\gamma} = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbf{1}(y_i, \widehat{y}_i), \qquad \widehat{R} = \frac{\widehat{Err}^{(1)} - \overline{err}}{\widehat{\gamma} - \overline{err}}, \qquad \widehat{w} = \frac{0.632}{1 - 0.368\widehat{R}}. \tag{4.68}$$

This leads to

$$\widehat{Err}^{(.632+)} = (1 - \widehat{w})\overline{err} + \widehat{w}\widehat{Err}^{(1)}. \tag{4.69}$$

The combinations of pdfs for highlight, shadow, overlap, and background can now be ranked with $\widehat{Err}^{(.632+)}$. Iteratively, a certain combination of pdfs is used to segment all $N = 213$ SAS images. The resulting superellipse parameters $\eta_{0,i}$ and $\eta_{1,i}$ with $i = 1, \dots, N$ for this specific pdf combination are used to train a classifier via bootstrap as described above. The ".632+" estimator $\widehat{Err}^{(.632+)}$ is then used to rank all pdf combinations. The lowest error indicates the best pdfs for highlight, shadow, overlap, and background in the context of using the superellipse parameters for classification.

4.4.5 Experiments

The data used for the first experiments was collected during the Colossus II sea trial in 2008 conducted by the Centre for Maritime Research and Experimentation (CMRE). The sonar data was generated by the MUSCLE AUV equipped with a 300 kHz sonar. It consists of 212 SAS images each with a size of 526 pixels by 160 pixels and a resolution of 18 mm by 25 mm. The images show several instances of cylindrical objects (CY)(67), truncated cones (TR)(70), and wedge shaped objects (WE)(47) which pose as targets. Moreover, there are images of non-MMOs like mine-sized rocks (RO)(28). Since an initial guess for the active contours algorithm is needed, the fast template matching of Chapter 3 is used; the position and geometrical extent of the best fitting template defines the starting superellipses for highlight and shadow ($\varepsilon = 1$).

Segmentation

For the two approaches outlined in Section 4.4.4, all 212 examples were segmented with each of the $5^4 = 625$ pdf combinations. Figure 4.15 illustrates the significant segmentation improvement that can be made by choosing the right combination of superellipse distance approximation and pdf. In the end, the lowest cost value on average was achieved by the combination found in the second column of Table 4.1. The best combination with respect to the classification success approach can be found in the third column of Table 4.1.

Comparing the segmentation results of both optimum combinations visually, one can see that – generally speaking – the combination derived from the cost function produces better segmentations. Figure 4.16(a)-(d) were produced with this approach. The best .632+ segmentation produced in some cases wrong segmentations of cylindric objects, albeit the error was somewhat consistent for this kind of object class. Figure 4.17 shows two examples of this behaviour. Apparently, the for this case consistent wrong segmentation made it easier for the classification algorithm to distinguish between the examples.

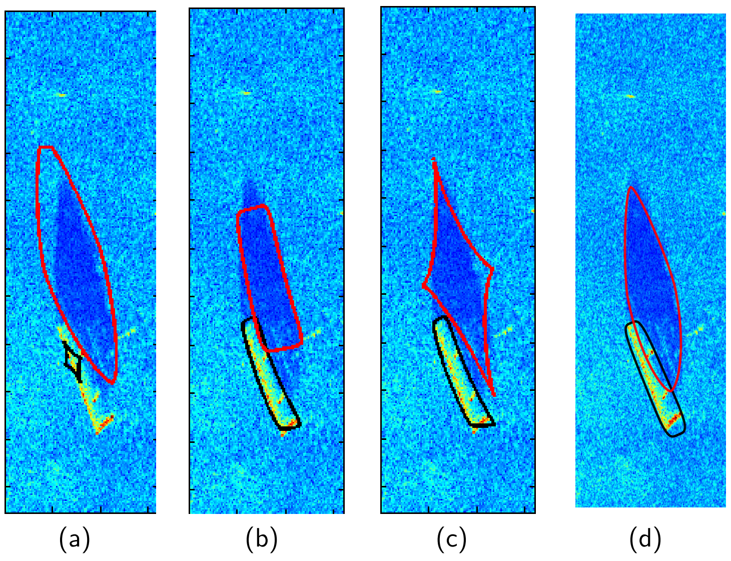


Figure 4.15 – For the first image the level set function based on the standard distance measure for superellipses (4.8) and the standard active contour weight functions (4.40) are used for all areas. For the second and third image the weight functions given by second column of Table 4.1 but the inferior level set functions (4.8) and (4.10) were used. The forth image shows the best case, i.e., ray-to-centre level set function (4.16) and the weight functions given by second column of Table 4.1.

Table 4.1 – The best choices for the pixel intensity estimations regarding cost function minimisation and classification result. A bootstrapped naive Bayes is used to in both cases to calculate the number of true positives.

	$f(u, l_o, l_1)$	.632+
background	Weibull	Gaussian
shadow	norm. hist.	norm. hist.
highlight	Rayleigh	Rayleigh
overlap	Weibull	Rayleigh
tp classification	172	181

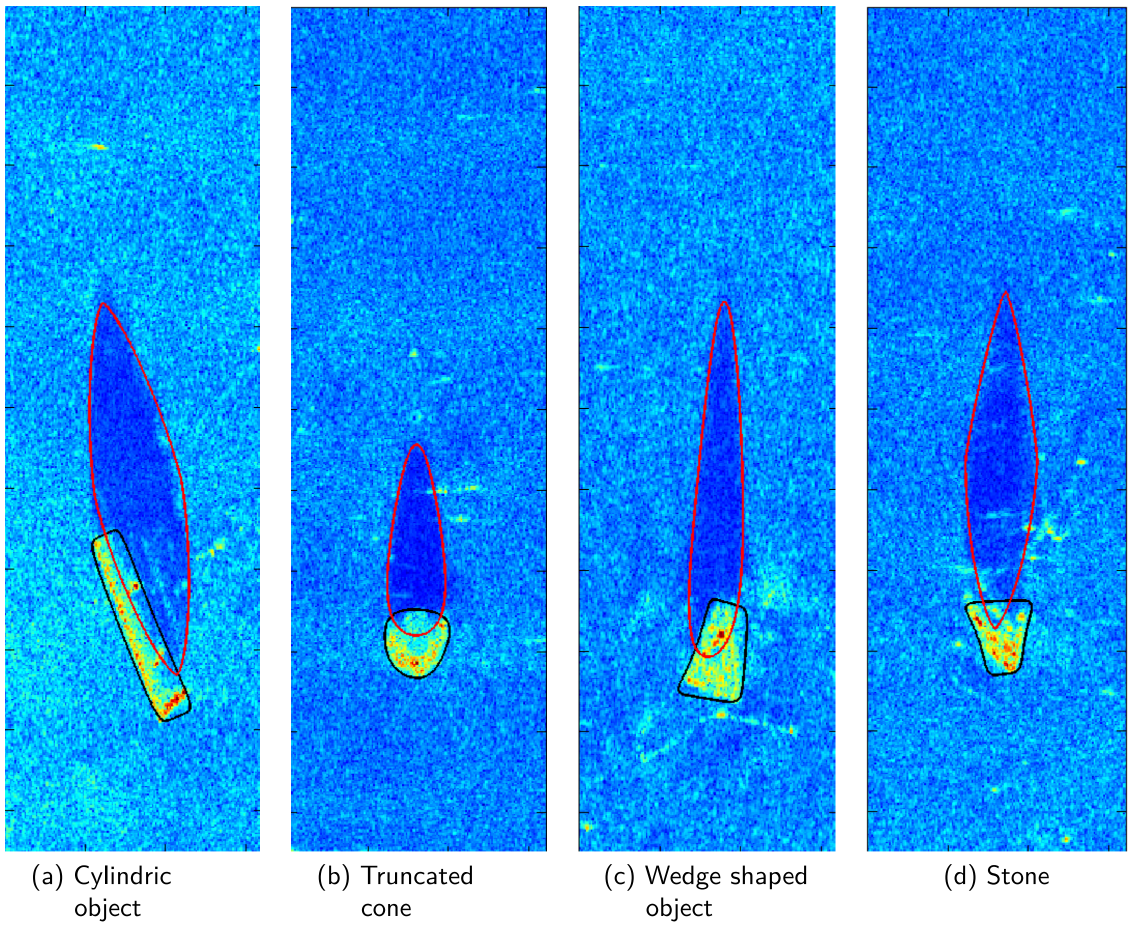


Figure 4.16 – Examples of superellipse guided segmentation.

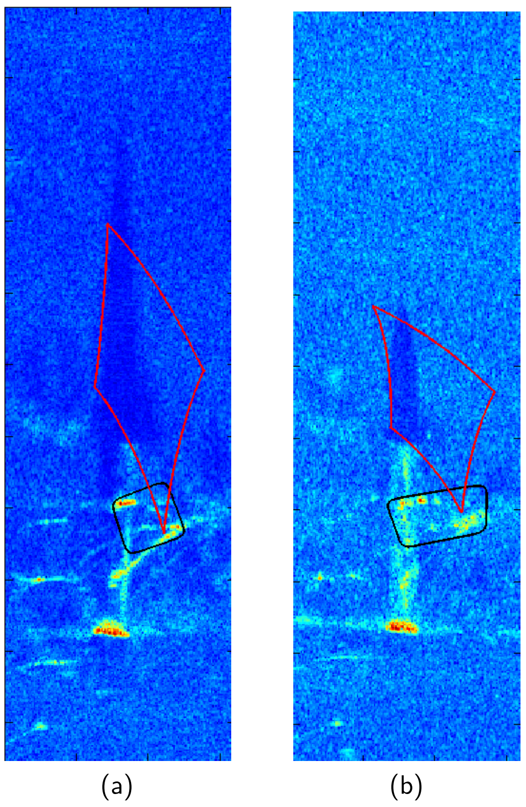


Figure 4.17 – In both cases the segmentation is not successful. Nevertheless, the extracted parameters are still of value for the classification since the error is consistent over different examples of the same type in the same position.

But there are two problems with this trade-off between segmentation accuracy for a higher detection rate. On the one hand is the training size too limited to make reliable predictions if this consistent wrong segmentation would persist on a bigger dataset. On the other hand will this be a drawback for the next step discussed in the following chapter. There the accurate generation of a simulated image of the object based on the segmentation is the core part. This is not possible with a segmentation like one can see in Figure 4.17. This is why the third column of Table 4.1 is discarded and only the results gained from the cost function approach that let to the second column of Table 4.1 are used for comparison and further processing in Chapter 5.

In a second experiment, the robustness of the segmentation towards additive noise was tested. This was done by first simulating a SAS image without noise and then applying noise with varying strength. How the simulation is done will be explained in detail in Section 5.1.1 in the next chapter. The noise is applied by using the MATLAB function *awgn* that adds white Gaussian noise to a signal. The function uses the image and a Signal-to-Noise Ratio (SNR) as input. The SNR is applied to the whole image.

One has to keep in mind that the segmentation algorithm takes the pixel intensity distribution into account. Even though the goal is to approximate a real SAS image with the simulation, the pixel intensity distribution will be different for this experiment compared to the real SAS images. Hence, once again one has to figure out the best combination of pixel intensity distribution. But instead of repeating the analysis over the best cost function value or the best classification result, the fact that a simulation is used allows a much more direct segmentation quality measure.

The ground truth segmentation from the simulation is easily obtainable via threshold since there is no noise. Hence, it is possible to directly quantify the quality of the segmentation. Figure 4.18 shows an example of a simulated cylindric object with the threshold based ground truth. Furthermore, it shows the segmentation results after different levels of noise are added. Even though especially the shadow becomes barely recognizable in the last images, the segmentation only fails with an SNR of $-10\,\mathrm{dB}$.

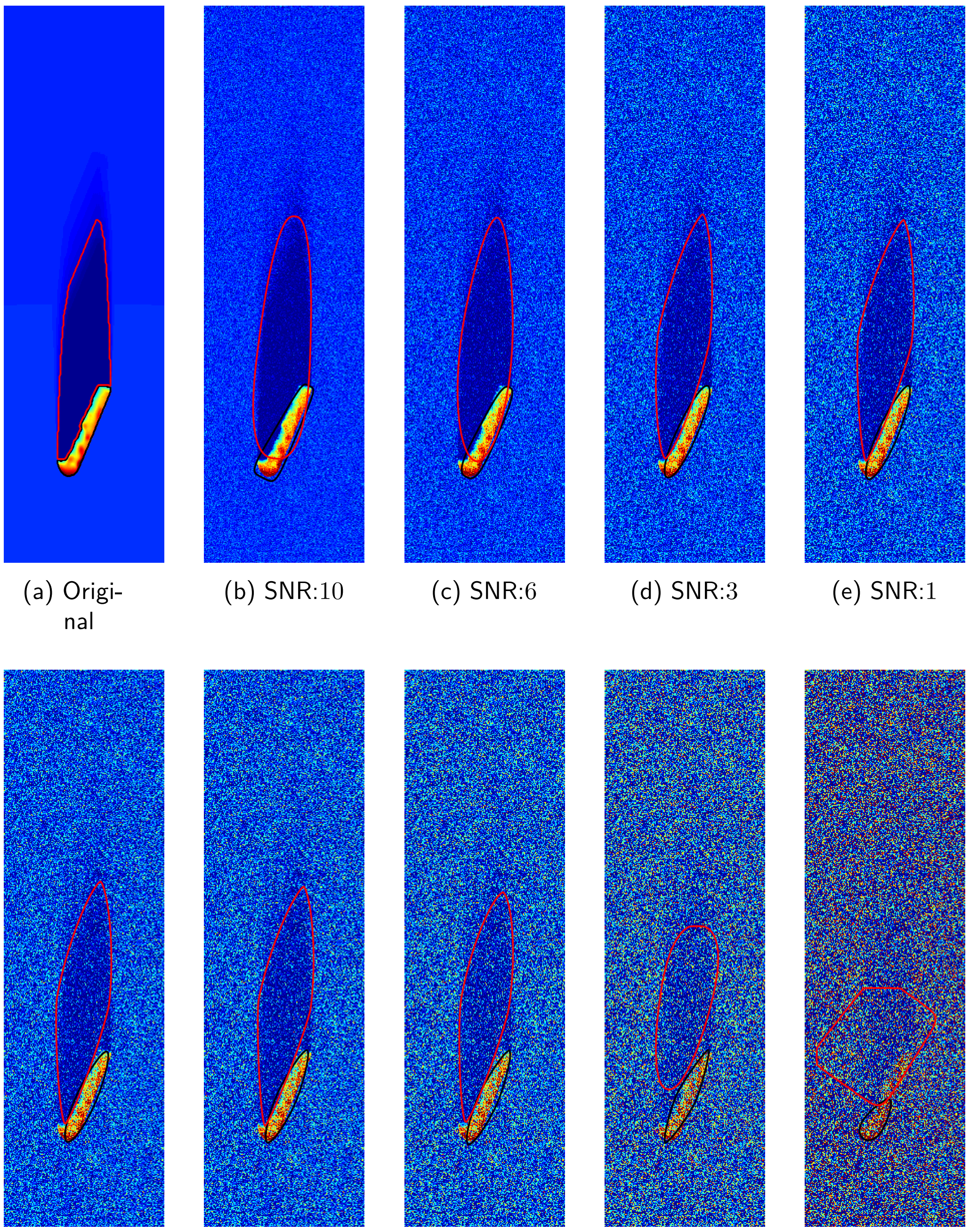
After having a closer look at the segmentation results one can notice that it is not uncommon to have an overlap between shadow and highlight. Especially, when the shadow area is concave due to the highlight. In some extreme cases the shadow segmentation overlapped the complete highlight area, so that in principle no highlight area was segmented out, only overlap. Figure 4.18(b) and Figure 4.18(c) show this for example, albeit less extreme.

The superellipse parameter based classification presented next will not account for those cases and should not be affected anyway. But to make the segmentation visually more intuitive or if other following processing steps depend on a clear distinction between the areas, several approaches can be used to resolve this ambiguity. The question here is to which area the overlap area can be counted. A simple straight forward approach is first calculating the mean value of highlight, shadow, and background, and then assign the overlap area based on similarity between mean values. A more sophisticated approach can be made by using the probability density distribution that were computed during the segmentation. With those the log-likelihood can be calculated that the pixels of the overlap belong to the three other areas. In praxis both approaches worked reasonably fine.

Superellipse parameter for classification

Revisiting the flowchart in Figure 2.5 shows that the processing step after the detection is actually called *Feature Extraction*. Usually, the segmentation is only the first step of this. But in fact, with the superellipse the feature extraction is already a by-product of the segmentation. The half axis parameters give the length and width, the squareness parameter describes the shape. Moreover, the rotation and the tapering parameters can be used as features. As previously mentioned, [18] dealt with superellipse fitting of MMOs in SAS images; they also used the features as input for a classification later on. Unlike the approach proposed in this section, the superellipse assumption was not integrated in the segmentation.

Table 4.2 shows the comparison between the classification results using the



(a) Original (b) SNR:10 (c) SNR:6 (d) SNR:3 (e) SNR:1

(f) SNR:0 (g) SNR:-1 (h) SNR:-3 (i) SNR:-6 (j) SNR:-10

Figure 4.18 – Simulated cylindric object with threshold based ground truth simulation. The results of the segmentation after different levels of noise are added. The starting contours and all parameters are in all cases equal.

Table 4.2 – Comparison with older approaches. In each case a bootstrapped naive Bayes classifier is used. The classifier is only trained on the respective extracted superellipse parameters.

Approach	Overall	Cylindric	Trunc. cone	Wegde	non-MMO
Ground truth	212	67	70	47	28
Dura etal [18]	136	48	59	24	5
4x3PP standard	143	42	58	38	5
from [36]	170	62	65	35	8
Classification	172	65	65	35	7

superellipse parameters as input for a bootstrapped Naive Bayes classifier. The row after the ground truth shows the performance of a classifier trained on the superellipse parameters; the parameters were gained by fitting to an already extracted contour as was done in [18]. This approach was modified in the beginning of the chapter to allow an individual fitting to the contour for each quadrant. The best performing version with respect to classification was when the half axis were not coupled. The classification result is shown in the third row. An improvement is already visible.

In contrast to these two post-processing methods, in the second half of this chapter an alternative superellipse based approach was introduced. This time, the superellipse fitting was directly introduced in the segmentation algorithm. An earlier version of the presented algorithm proposed in [36] can be found in the third row. Eventually, the result of the further improved version from this chapter is presented in the last row. The main difference between the last two is the use of $p(\mathfrak{a}|u)$ instead of $p(u|\mathfrak{a})$ in the weight functions as explained in Section 4.4.2. Even though the quadrant-wise formulation of the problem improved the result for the post-processing approach, this clearly shows that integrating the superellipse constraint in the segmentation algorithm makes the feature extraction more robust.

Overall, Table 4.2 indicates that the presented approach leads to some progress in the classification. But it is also apparent that wedge shaped objects and rocks are still challenging targets. As the confusion matrix in Table 4.3 shows, wedge shaped objects and rocks are most often mislabeled as truncated cones. The main difference between the wedge shaped objects and truncated cones on SAS images are the roundish shape of the highlight for truncated cones versus the sharper triangular shape of the highlight of the wedges (c.f. Figure 2.4(b) versus Figure 2.4(c)). Sometimes those edges are not that prominent. This explains why wedges are mislabeled as truncated cones and not vice versa.

The problem for rocks is the big variety in shapes. Therefore, learning a general *rock shape* is not trivial. Especially considering the limited size of the data set.

Table 4.3 – Confusion Matrix of the classification phase. Classification rate 81.1%

		Predicted class				
		CY	TR	WE	RO	Recall
Class	CY	65	0	1	1	0.970
	TR	0	65	3	2	0.929
	WE	2	8	35	2	0.745
	RO	8	10	3	7	0.25
	Precision	0.867	0.783	0.833	0.583	

4.4.6 Superellipse guided active contours segmentation on underwater visual data

The superellipse guided active contours segmentation is not limited to the use case of segmenting the highlight and shadow of objects on sonar images. In Chapter 2 an alternative application for object recognition in the context of deep sea ROV operations was introduced. As previously stated, an important element within the overall scenario is the recognition of the panel and the state of its handles.

Figure 4.19 shows an panel with different handles in different states, i.e., rotations. To properly represent the panel in the simulation these states have to be inferred from the perceptional data. In this case this means the monocular video stream of a stereo camera. It is assumed that the position of the panel within each frame of the video stream is approximately known. This can be realised for example with artificial markers on the panel which can be seen in Figure 4.19. Since the positions and types of handles on the panel are known, their position within each image frame can be computed based on the pose of the panel in the image frame.

Given this information the image given by the camera can be divided into several ROIs where each shows an individual handle. Consecutively, in each region the superellipse guided active contours algorithm can be applied to segment it into background and the round-shaped handle. Now the region of interest is reduced to only the part of the image showing the handle with the lever. Here again the superellipse guided active contours algorithm can be applied to segment out the rectangular lever.

To test the proposed approach a new method of evaluation was used. The whole setup of the DexROV project is modelled in a simulation as depicted in Figure 4.20. With this setting it is possible to have either the simulated sensor data or the recorded real sensor data as input for the various processing algorithms. This can be done for each sensor individually as well as parts of the recognition pipeline.

The idea behind this approach is that with real data it is oftentimes difficult to tell whether an algorithm itself is flawed or if the input data is not good enough due to noise or faulty preprocessing. If one is able to evaluate each building block of the perception pipeline with various levels of *optimal* input data then its possible to pinpoint the strengths and weaknesses more thoroughly [24]. Moreover, the ground truth of simulated data is usually easily available.

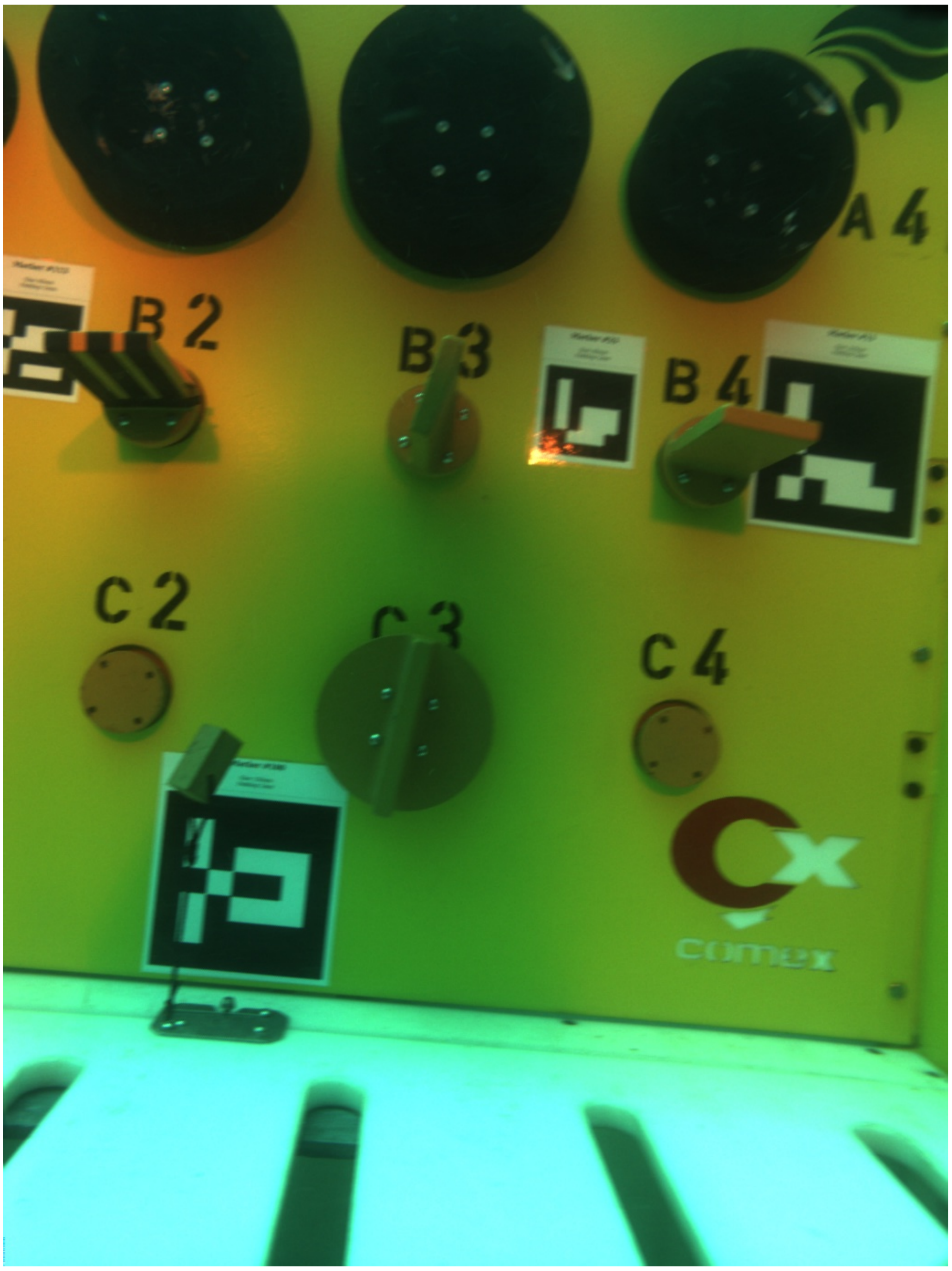


Figure 4.19 – Mock-up panel with handles that an ROV needs to operate, i.e., turn.

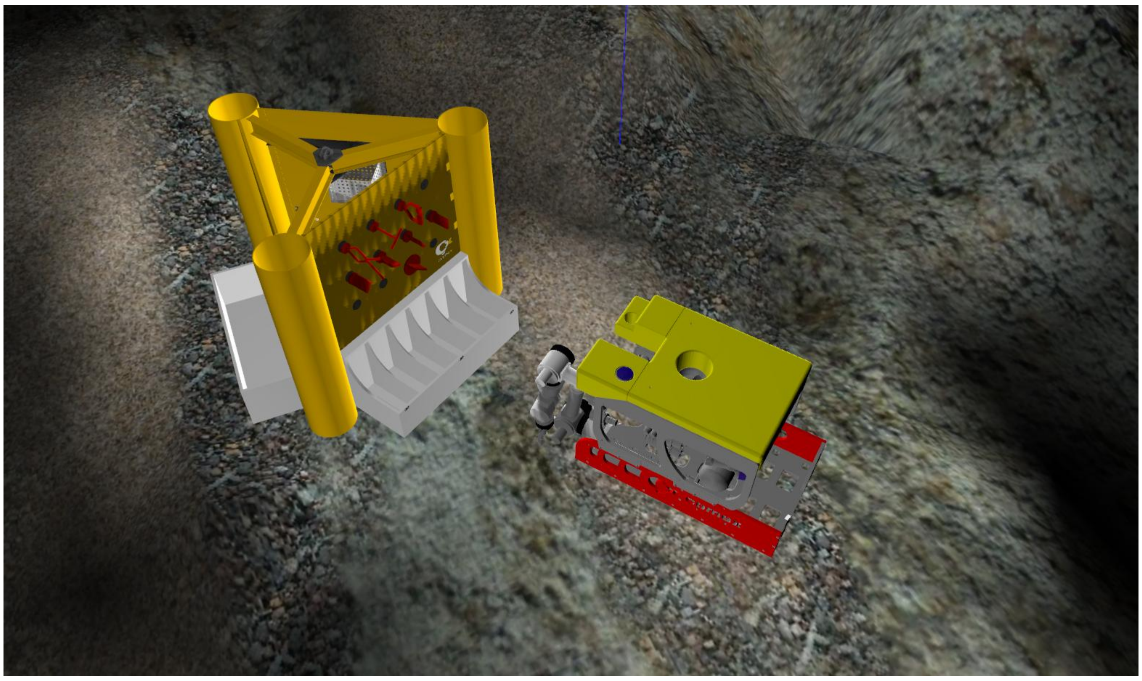
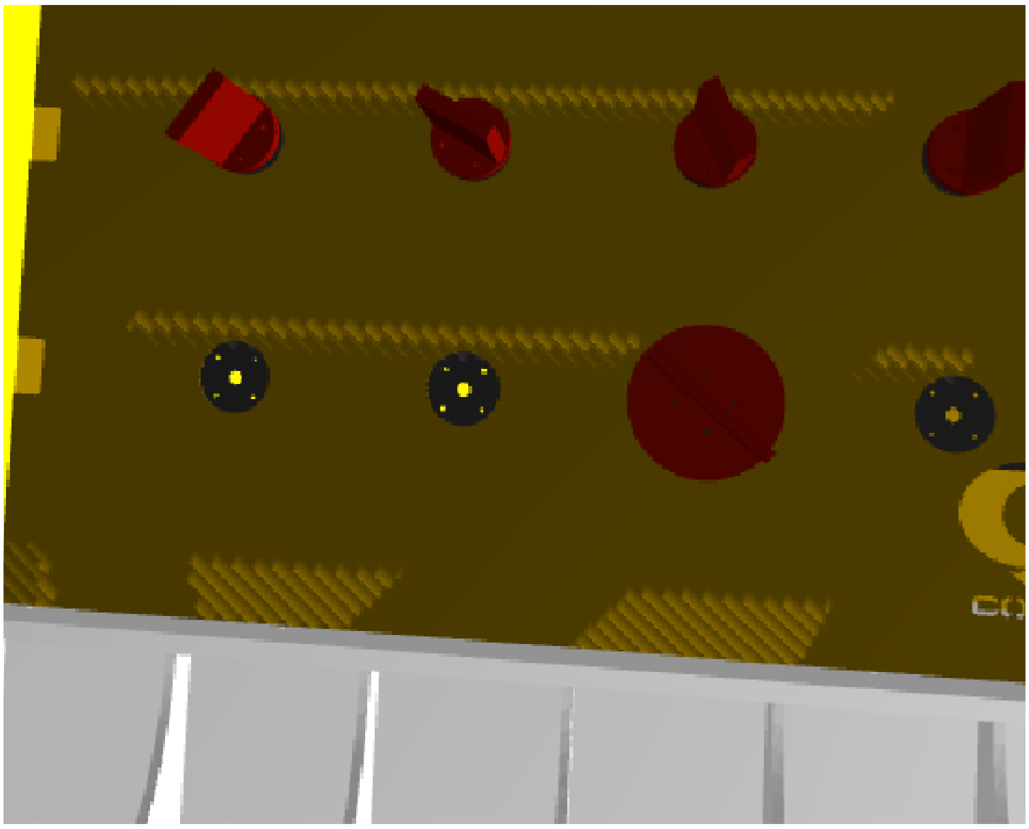


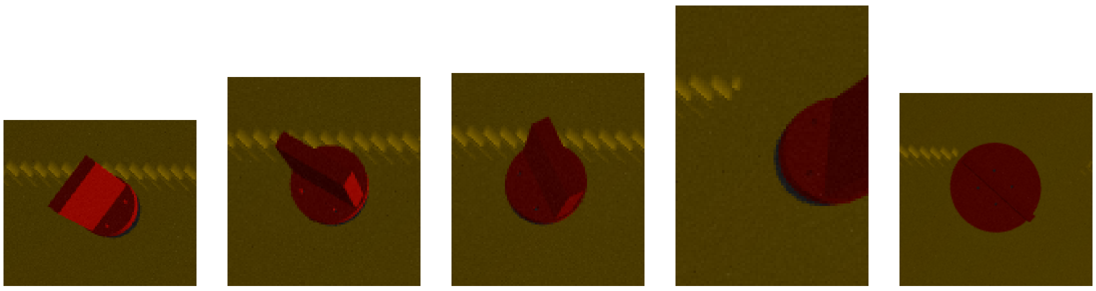
Figure 4.20 – Simulated panel and ROV in the physics simulator Gazebo.

To follow this line of thought the starting configuration uses a simulated optimal panel detection with simulated camera input. This means the input of the superellipse guided active contours segmentation looks like Figure 4.21(a). The image shown is in colour for easier interpretation. The actual input image is converted to greyscale via weighted addition of each channel. From this input image the ROIs are generated containing the individual handles. Since the simulated panel detection is optimal, the location of the handles is exactly known and, hence, the ROIs will be optimal. Figure 4.21(b) shows the ROIs. Figure 4.21(c) then shows the pixels that the segmentation classified as object. Here a Gaussian distribution was assumed for the object and background pixel intensity distributions. The assumption here is that object and background have each a distinct grey value and the camera adds some Gaussian noise to that.

Apparently, the algorithm was able to segment background and handle reasonably well. Unfortunately, employing the segmentation algorithm again to segment out the rectangular levers did not succeed. The region based approach fails here since most parts of handle and the lever on it have the same greyscale value divided by a darker shadow part or a lighter reflecting part. Therefore, the second step was replaced with a Hough transformation for lines.

Usually, the Hough transformation is computationally expensive. But it is feasible in this real time or near real time application because the number of points is small enough after the first segmentation with only handle pixels remaining. The resulting lever state estimations can be seen in Figure 4.22. The estimated angles are off by usually only 5-10 degrees compared to the ground truth values.


(a) Input image


(b) Regions of interest

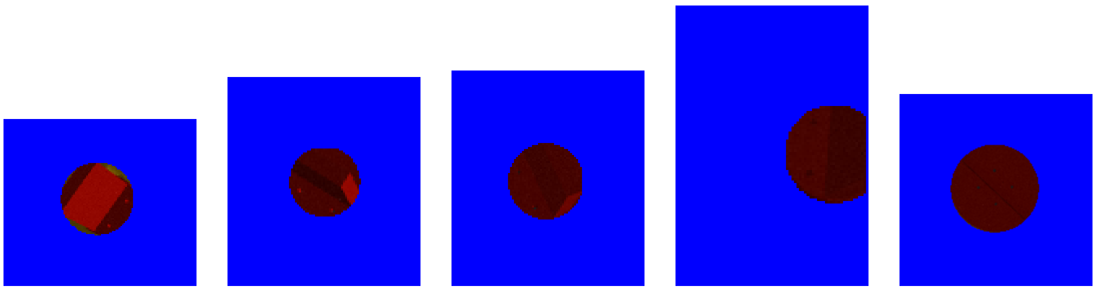

(c) Segmented out handles

Figure 4.21 – Valve pose estimation on a simulated panel.

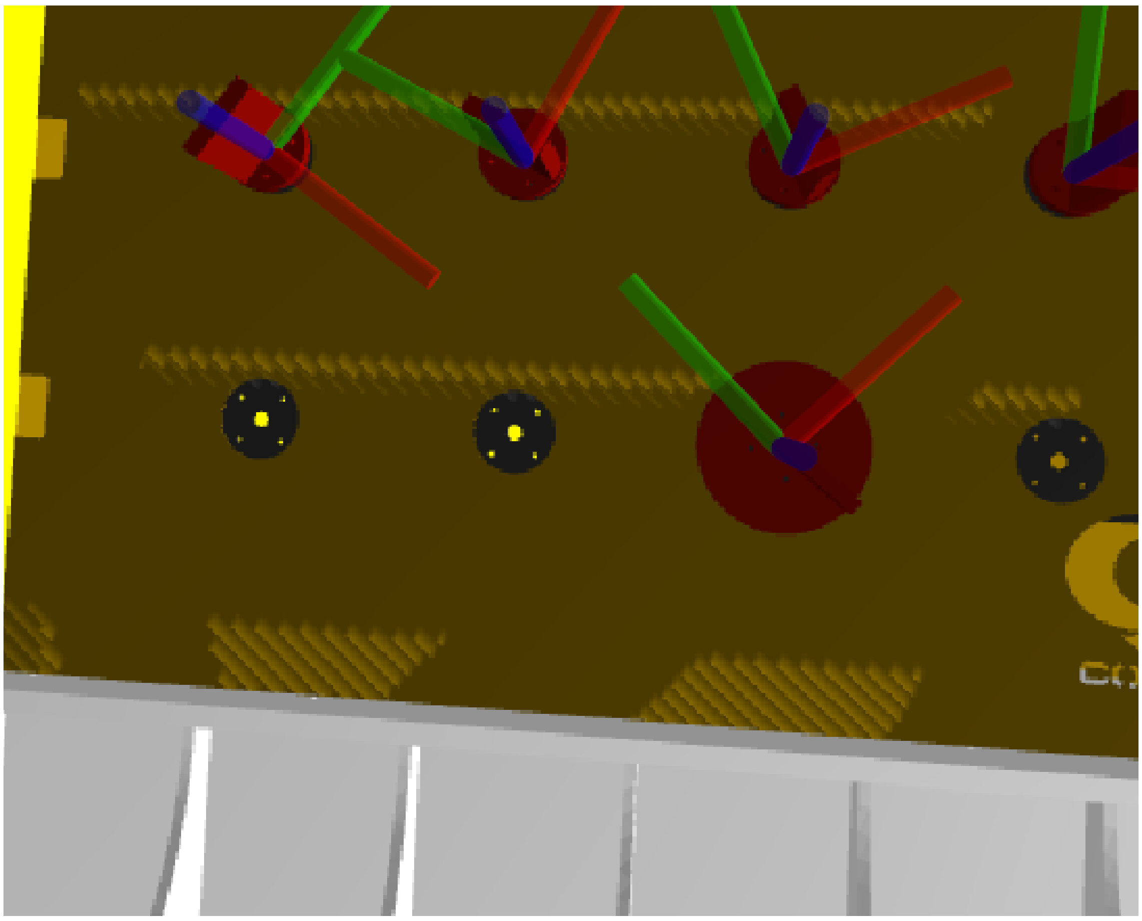
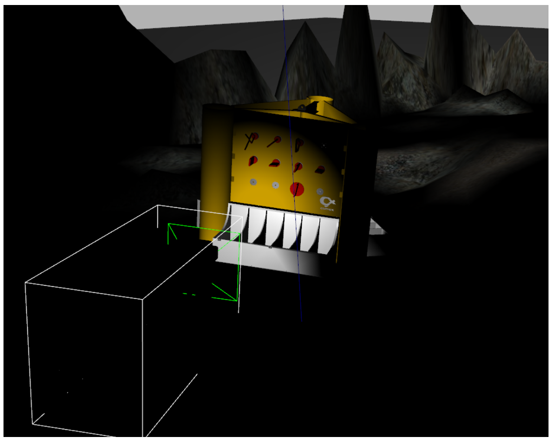


Figure 4.22 – Final result of the handle pose estimation on a simulated panel. The red axis represents the x-axis, green the y-axis, and blue the z-axis. Zero degree rotation is a handle in vertical position, i.e., the green axis points up.


(a) Simulated environment

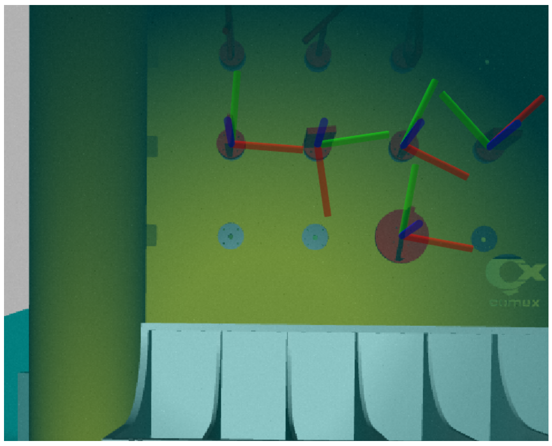

(b) Simulated input with detected handle states overlayed

Figure 4.23 – In the second simulated scenario the only light source is connected to the ROV. Figure (a) shows the environment with a white bounding box around the un-illuminated ROV for visualization. Figure (b) shows the resulting input image for the handle state detection already overlayed with the result.

The simulation used so far employs ambient light that illuminates the whole scene. As a next step the simulation is changed so that the only light source is coming from a simulated light attached to the ROV. The opening angle of the light cone is chosen so that not the whole panel can be illuminated evenly. In addition to that a range depended haze was applied to the visual data. This should increase the difficulty of recognizing the handle states on the simulated visual data as well as make it more realistic. In the real scenario, especially in the deep sea, one also has to cope with the uneven illumination provided by a ROV's light.

Figure 4.23 shows a view in the simulator and the resulting image that will be used as input for the handle state detection. On the depicted input image Figure 4.23(b) one can see that, on the one hand, the handles near the centre of the illuminated area have a better visible lever position due to the higher contrast shadow. On the other hand, handles to the side of the illuminated area are reduced to dark blobs. Nevertheless, the algorithm is still able to recognize the handle state most of the time in this more challenging scenario.

Since the preliminary tests of the algorithm on simulated data indicate that the proposed algorithm is capable of detecting the handle state in the DexROV scenario, the next challenge is using real data. This is done by replacing the video stream from the simulated camera with the playback of data gathered with a real camera during trials at sea. Thus, the input image looks like 4.21(a). Again the first steps are the recognition of the panel pose via artificial markers, which in turn makes it possible to generate ROIs for the handle positions. Due to localization noise on the data the ROIs can not be expected to be as exactly focused on the handles as before.

Figure 4.24(b) shows the ROIs the superellipse guided active contours segmentation uses as input. As one can see, the objects are not exactly in the middle of the ROI any more. This means the starting contours for the the superellipse guided active contours segmentation which assume an object in the middle of the input image are not as optimal as before. Figure 4.24(c) then shows the handles segmented out. Apparently, the algorithm has difficulties with assigning the printed handle numbers to the background. Nevertheless, the handles are still in the segmented area, albeit not as exactly enclosed as before in the simulation. In case of the *C3* handle, the label is hardly visible and the algorithm has no problems segmenting the handle perfectly.

Finally, Figure 4.25 shows the resulting pose estimation superimposed over the real image. The slight offset in the position is due to inaccuracies in the camera model and the marker detection. It works as intended for the handles *B1, B3* and *C3*. Even though the handles *B1* and *B3* were not perfectly segmented out, the most dominant line in the segmented region was still given by the lever.

In case of *B2* it is obvious why the Hough transformation could not succeed, even though the handle itself is segmented out relatively good. The markings on the lever give the dominant lines and do not correspond to the lever angle. For the handle *B4* the segmentation algorithm had problems with removing the artificial markers from the ROI as can be seen in Figure 4.21(c). Consequently, the straight vertical lines of the markers were found by the Hough transformation.

For a more thorough test all three setups were tested for various ROV positions. For this the recordings from a DexROV sea trial were used. Whenever the panel was in frontal view and the artificial markers were properly recognized, the simulated ROV was placed in the same position in front of the simulated panel, i.e., the simulation re-enacts the same scene as was present at the time of the recording. This means a real stereo image *and* a simulated stereo image of the same situation are available. The proposed algorithm expects a monocular image as input. Thus, two images from different perspectives are available for every position. Both are used to estimate the lever angles for the visible handles.

The panel itself was detected 303 times, i.e., 606 simulated and real images are available, respectively. However, not all images contain all handles at the same time. On average each handle was in view in about 380 images. Table 4.4 shows the resulting average detected angle compared with the ground truth. Unfortunately, the exact ground truth values for the handle rotation seen in the real data were not available for the experiments. Therefore, the ground truth values were manually determined from the available visual data. Afterwards the simulated handles were set to those approximate ground truth values. For the experiments this means that the noted ground truth values are exact for the two simulated scenarios and approximated for the real image.

The results over multiple views are in line with the observations made on the example images Figure 4.22, Figure 4.23b, and Figure 4.25. The average detected angles are very close to the ground truth values for the simulated scenario with ambient light. The slight errors can be explained by the not perfectly frontal view.

Table 4.4 – Mean recognized orientation of the different handles for different experiment setups. Zero rotation means the handle lever is vertical.

Handle	Views	Ground truth	Simulation with ambient light	Simulation with sim. ROV light	Real image data
C3	366	-10.0	-10.4	-9.6	-9.9
B1	460	-5.0	-1.1	-3.6	-6.2
B2	419	-85.0	-82.3	-81.7	-60.5
B3	361	-20.0	-27.3	-23.6	-27.5
B4	312	85.0	100.7	57.7	119.2

In the second simulated scenario with the simulated ROV lights, the rotations of the handles *C3*, *B1*, *B2*, and *B3* were again detected reasonable fine with only small deviations from ground truth. But the state of handle *B4* is not detected very well any more. As in the example image in Figure 4.23(b) the handle was not properly illuminated a number of times. In those cases there were no shadow lines visible to deduce the rotation.

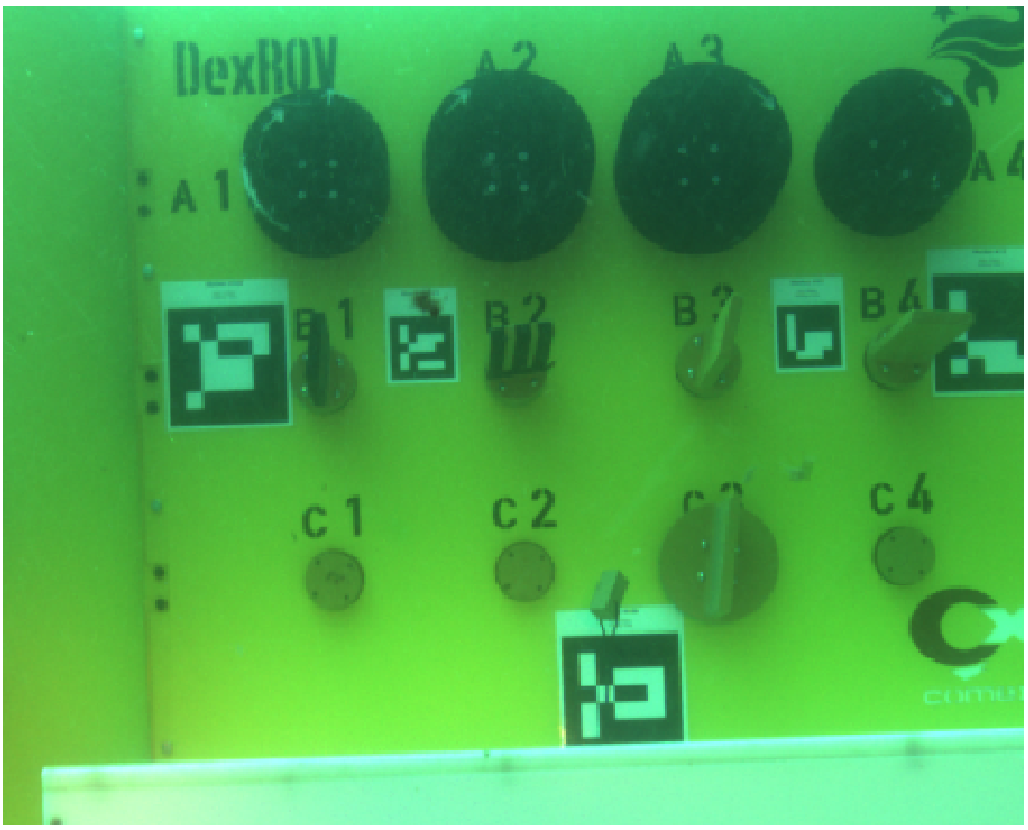
Finally, the experiment was done on the real images. Here the angle of *C3* was again correctly detected. The lever angle estimation performed also well for *B1*. Actually, it was better than in all other cases. The black marking on the lever was apparently an advantage that made it easier to detect the orientation. For *B2* the problem with the vertical stripes persisted throughout the experiment. On *B3* one can see that compared to *B1* the missing black marking made it more difficult for the algorithm to detect the correct angle. Eventually, for the last handle *B4* the superellipse guided active contours algorithm usually failed to extract the handle due to the artificial markers.

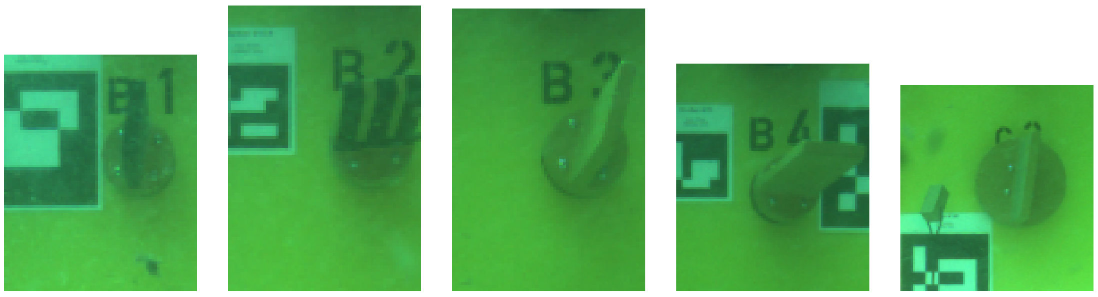
Overall, the experiments on the DexROV visual data showed that the superellipse guided active contours algorithm is not limited to its originally intended sonar application but can also be applied to visual data where man-made objects of known shape need to be extracted. It also showed that a proper preparation of the objects, e.g., a black marked lever, can support the algorithm.

4.5 Conclusion

The topic of this chapter was the shape-based segmentation of images in order to extract meaningful features for the classification. This was motivated by the observation that if there would be no noise, the shapes of man-made objects on an image can often be described by a limited set of possible shapes called superellipses. This insight sparked the idea to exploit this prior knowledge for better segmentation results. Two distinct approaches followed.

The first implemented the idea by using superellipse fitting on an already extracted contour. Several different settings for the post-processing approach were presented


(a) Input image


(b) Regions of interest

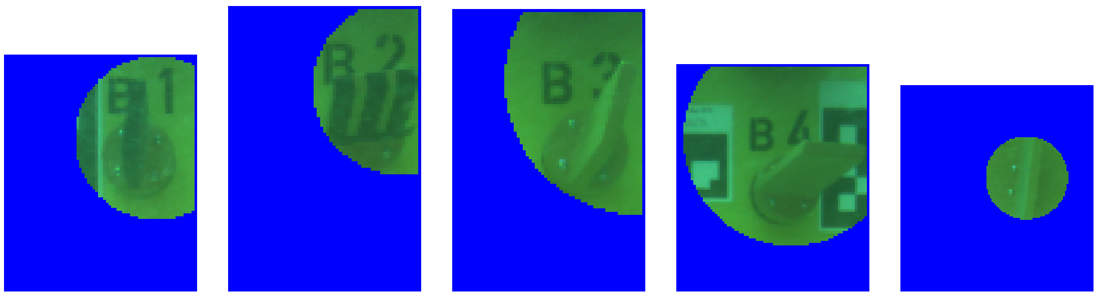

(c) Segmented out handles

Figure 4.24 – Valve pose estimation on a real panel

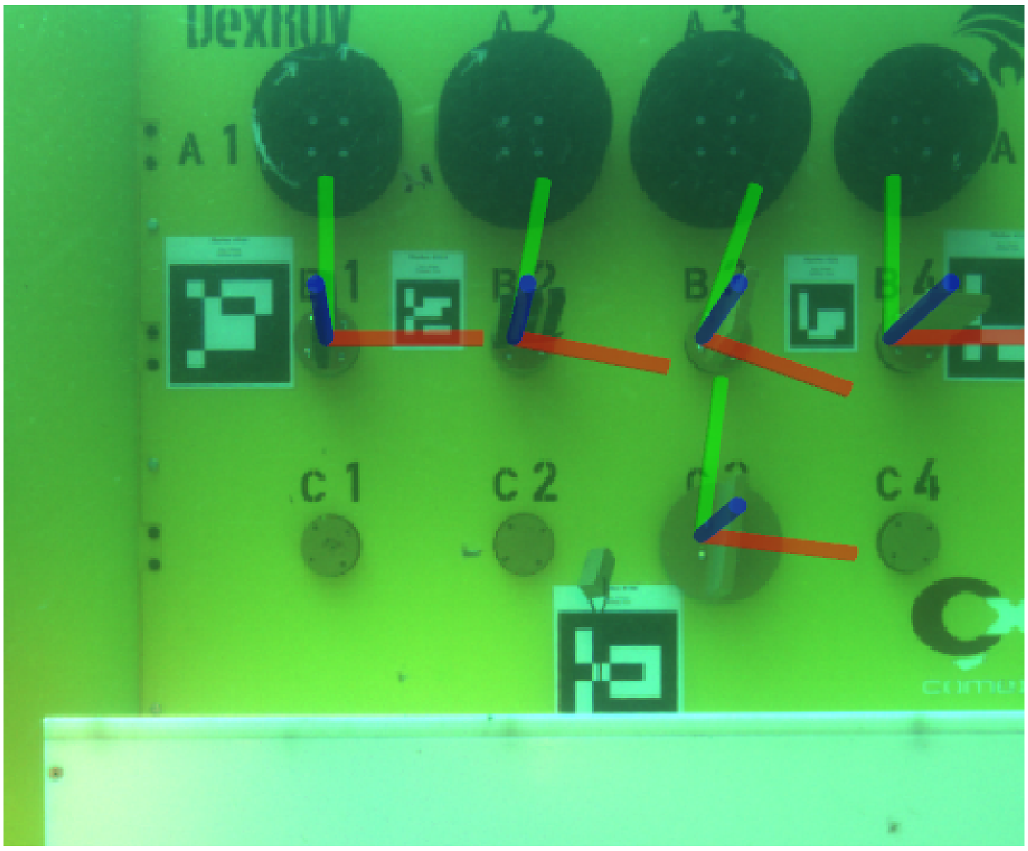


Figure 4.25 – Final result of the handle pose estimation on a real panel. Good results for the handles *B1*, *B3*, and *C3*. The algorithm failed for *B2* and *B4*.

and a novel linearisation of the fitting equation was introduced. To evaluate this approach, experiments on object shadow contours extracted from real SAS images were done. The shown modifications to the reference of one superellipse for the whole contour not only were able to much better approximate the contour, but also improved the processing time significantly.

Nevertheless, there were still problems. Due to the dependence on an independent segmentation algorithm, the superellipse fitting algorithm was not able to recover the true shape if the input was too distorted. Therefore, it was decided to investigate a means to integrate the superellipse assumption directly into the segmentation algorithm. Accordingly, a novel superellipse-driven active contours method was presented.

It was shown how to derive this method from standard active contours theory, i.e., how to embed the superellipse fitting into the segmentation. Moreover, the reformulation allowed for several additional enhancements. A probabilistic approach towards the pixel intensities allows to go beyond just Gaussian pixel distributions.

The original algorithm tailored to photography implicitly assumed a Gaussian pixel intensity distribution with a fixed variance within the different image areas. With a novel reformulation of the equations that govern the segmentation it was possible to change this underlying distribution assumption into better suitable distributions, with parameters that can be calculated from the image itself. The results of experiments on SAS images suggest that, instead of having the same intensity distribution model for the whole image, each region should be treated differently. For the presented SAS data, the intensity distribution of the highlight region is best modelled by Rayleigh distribution, where for the shadow region it is the normalised histogram and for background pixels the Weibull distribution.

To further support the choice of a superellipse constraint, additional experiments showed that the superellipse feature of the contours of highlight and shadow in a SAS image can be used immediately for classification. Having the superellipse constraint integrated directly in the segmentation algorithm improved the feature extraction compared to post-processing approaches. Choosing the empirically determined intensity distribution models for the segmentation, the classification based on the superellipse segmentation was able to correctly classify 172 objects out of 212.

The presented general framework of superellipse based segmentation with active contours is not only applicable to SAS images but can also be adapted to other application areas, e.g., medical application where ellipse [75] and even superellipse [70] shape representations can be applied. The main adaption to be made is to adjust the point-density functions to match the respective imaging system. To support this claim, the algorithm was also applied to the task of segmenting objects on underwater camera images. To be exact the task was recognizing the orientation of a lever on top of a handle. The experiments showed that the proposed algorithm is suitable for segmenting out the round handles which allows in most cases the lever orientation recognition.

Despite the advancements made in this chapter regarding classification, there can still be no guarantee that the recognized class is correct 100% of the time. At the same time it is not uncommon that it is obvious to a human when a classification is wrong. A human might make the simple loop from the result back to the beginning, noting that the result does not match the input image. The next chapter of this thesis is dedicated to this top-down classification verification.

CHAPTER 5

Classification Verification

5.1 Introduction

Despite having developed a seemingly complete workflow from receiving the SAS image over the fast detection of objects of interest towards classification via superellipse guided segmentation and feature extraction, an important addition can still be made. A crucial feature of a classification algorithm is a measure of how much the result can be trusted. Especially for the task of safety or security related tasks, it is essential that an object classified as stone is actually a stone. A *maybe stone* that triggers a second closer inspection is a lot better than a *stone* classification that causes the miss of an object of interest.

Some classification algorithms like the previously used Naive Bayes are able to output not only classifications but also scores for the different classes. Yet, this does not change the underlying problem of only depending on the result of the classifier. Moreover, even if the classification works perfectly, the whole workflow is a concatenation of complex algorithms. An error in for example the segmentation can lead to an erroneous feature extraction, which even a perfectly working classifier sometimes can not compensate. Therefore, an independent consistency check of the results is needed.

A top-down approach that uses the classification result at the end of the process and compares it to the original input image can give this independent judgement, whether the classification result is reasonable or not. The idea is that, besides the class, also position and orientation information were extracted with the superellipse guided active contours without edges approach. Combining these makes it possible to simulate how the image should look like. The expectation is that if no error occurred during the processing, the original SAS input image and the simulation should look alike. This concept is illustrated in Figure 5.1.

The general idea of using a top-down analysis as basis for classification is of course not completely new. From a broad perspective, any template matching approach to some extend simulates different configurations of highlight and shadow areas and matches them with the original image. In contrast to the proposed approach though, it is used as a direct means of classification.

An example in the context of naval mine classification is [67] where all possible shadows for each object type in every reasonable pose are calculated and a similarity value of the most similar simulation for each class is used as classification input. The obvious main drawback of this work is the need for an exhaustive search over all poses and classes. Even if there is only a small amount of object types and the orientation, i.e., the aspect angle of the object is discretised very roughly, the shadow length is dependent on the distance to the sonar and the height over ground. This increases the number of needed simulations significantly and makes the computation very time consuming. This is not the case for our approach: the bottom-up processes, i.e., the segmentation and classification, provide a type and a pose hypotheses and only a single so-to-say template match has to be computed.

The remainder of this chapter will describe how the simulation is implemented and subsequently, how the simulated image and the input image are compared.

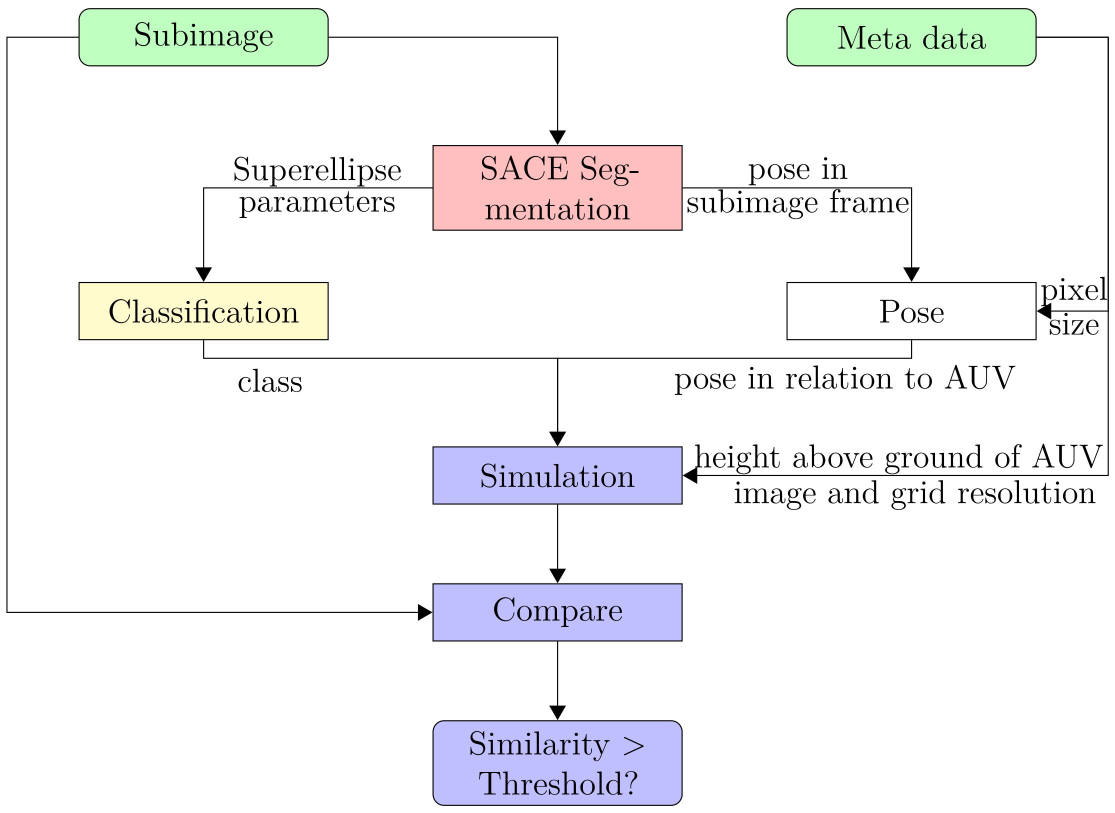


Figure 5.1 – Flowchart of the proposed approach. The image showing the potential object and the SAS image meta data are the input. Applying the Superellipse guided Acitive Contours without Edges (SACE) algorithm yields the highlight and shadow shape describing superellipse parameters. These parameters are used to classify the object and to estimate its pose. This class and pose information, in conjunction with the SAS image meta-data, is used to generate a simulated version of the image, which is compared to the original image.

5.1.1 Simulation of SAS images for top-down hypotheses verification

The simulation of sonar images in general is a non-trivial topic. Simulating the physical process can be done [4], [12] but usually comes at a great computational cost. Therefore, to keep the computation time in a reasonable time frame, in this chapter the simulation of SAS images based on object and pose hypotheses is implemented in a rather simple way with the ray tracing program POV-Ray [61].

Even though POV-Ray simulates light- rather than sound-based image generation, simplified simulations are often considered to be a good enough approximations of a sonar image of a real target [15], [67]. As mentioned, the decisive advantage is the much lower simulation time compared to actual SAS image simulations.

Figure 5.2(a) shows the general setting. The height of the illumination source h_l and the distance to the object d_o can be derived by the height of the AUV over the seafloor and the field of view of the sonar, i.e., the height h_c of the simulated camera is chosen such that in combination with the viewing angle the resulting simulated image has the same size and resolution as the original SAS image. Figure 5.2(b)-(d) show the models of the objects from a side-view. Figure 5.2(e) shows an example SAS image. The classifier labels this example as a cylindrical object based on the superellipse parameters. In addition, they provide estimates for the localization, i.e., its position and its rotation. Based on this information, the simulated image shown in Figure 5.2(f) is generated.

5.1.2 Similarity measures between the original and the simulated image

Eventually, the question arises of how to decide whether the simulated image and the original input image show the same object and the hypothesis of the classifier is therefore accepted, or if both images are too different and the decision of the classifier therefore has to be rejected.

For this purpose, several strategies have been investigated. For each strategy, the core of the comparison itself is a normalized 2D-cross-correlation. However, the strategies differ in the preprocessing of the original image and the simulated image. Please note that the following options are not applied in sequence as the labels might suggest but one has to decide which method to use.

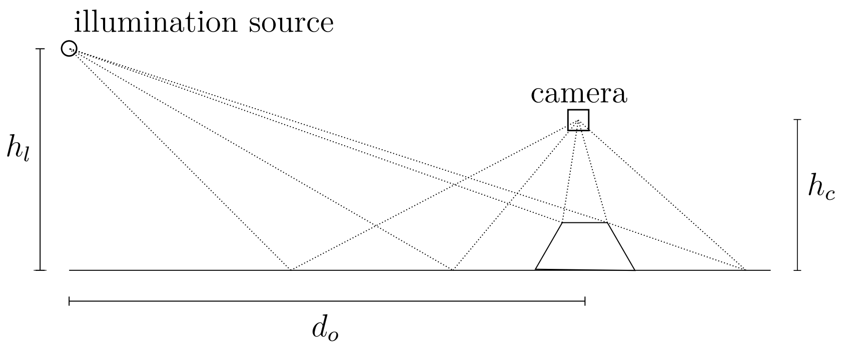
A Both the original image and the simulated target are filtered with a Gaussian kernel.

B One output of the SACE segmentation are the pdfs of the highlight, shadow and background areas. Moreover, the simulated target can be easily segmented via thresholding. This can be used to replace the pixels in the highlight, shadow and background area in the simulated image with pixels sampled from the respective pdfs of the real image. Afterwards, both the simulated image with the according pixel intensity distribution and the original image are filtered with a Gaussian kernel.

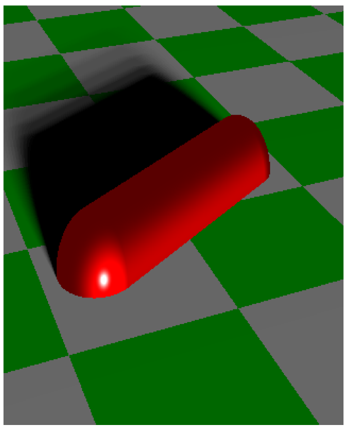
C The SACE segmentation of the original image is compared to the segmented, i.e., thresholded simulated image.

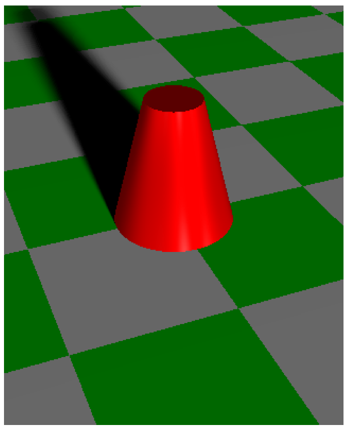
D The simulated image is processed as in B and afterwards segmented via SACE. The result is compared to the SACE segmentation of the original image.

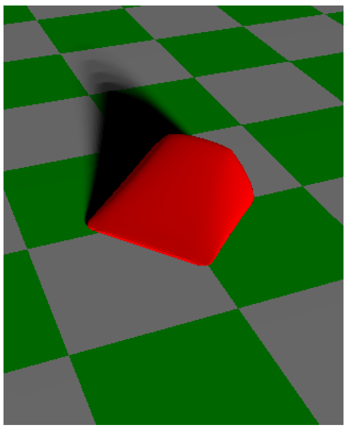
E The simulated image is segmented via a threshold. For the original image $p(\mathfrak{a}|u)$ is used to classify each pixel according to which region has the highest probability, i.e., $\hat{\mathfrak{a}} = \arg\max_{\mathfrak{a}}(p(\mathfrak{a}|u))$. A median filter is applied afterwards to reduce noise.

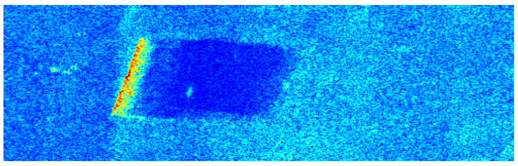


(a) Setting


(b) Cyl. model


(c) Trunc. cone model


(d) Wedge model


(e) Original Image

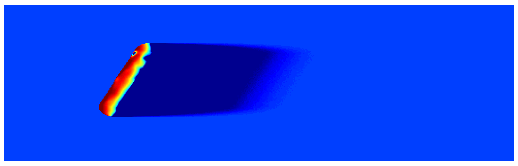

(f) Respective simulated image based on class and pose information gained by segmentation and classification

Figure 5.2 – Setting of the simulation of a SAS image via POV-Ray.

Figure 5.3 shows the results of the different preprocessing strategies. The normalized 2D-cross-correlation is then calculated with the preprocessed real image (Figure 5.3, left) as target and the preprocessed simulation result (Figure 5.3, right) as template.

Instead of using the whole simulated image which has the same dimensions as the original SAS image, only the content of a bounding box around the simulated object is used as template. This lessens the effect of clutter near the target on the correlation and it has the additional advantage of decreasing the processing time. The bounding box is illustrated by a red dashed line in Figure 5.3. It is calculated by finding the first and last non-background pixel in x- and y-direction and by applying a small margin.

For the last three cases of preprocessing (C, D, E), it is also possible to measure the distance between the segmenting contours with the modified Hausdorff distance [17] in addition to cross-correlation; these cases are denoted with CH, DH and EH.

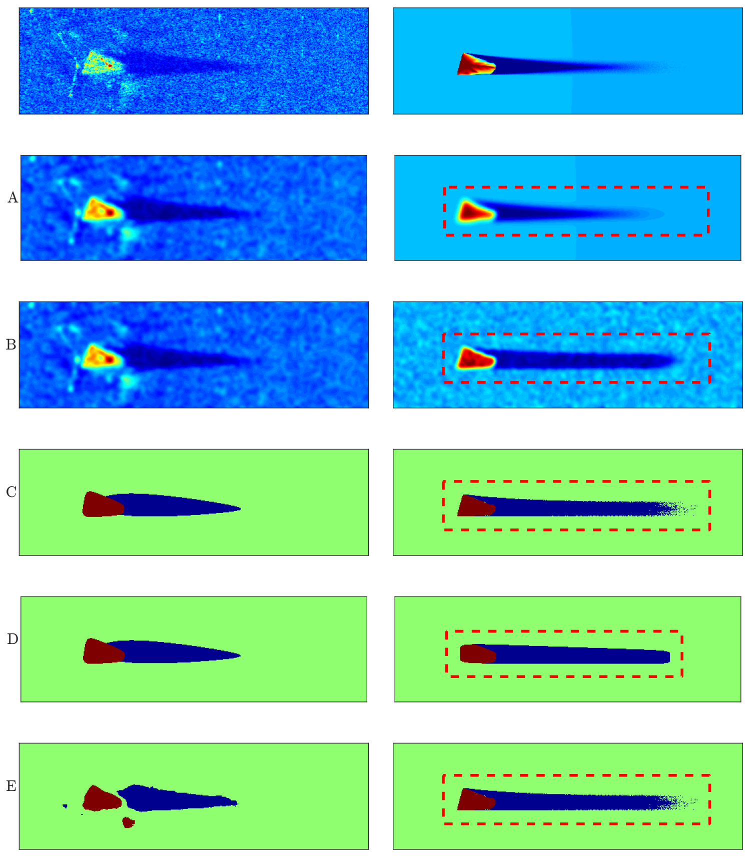


Figure 5.3 – First column: original SAS image (top) and results of the preprocessing methods A to E. Second column: simulated target (top) and results of the preprocessing methods A to E including the template bounding box.

5.2 Experiments

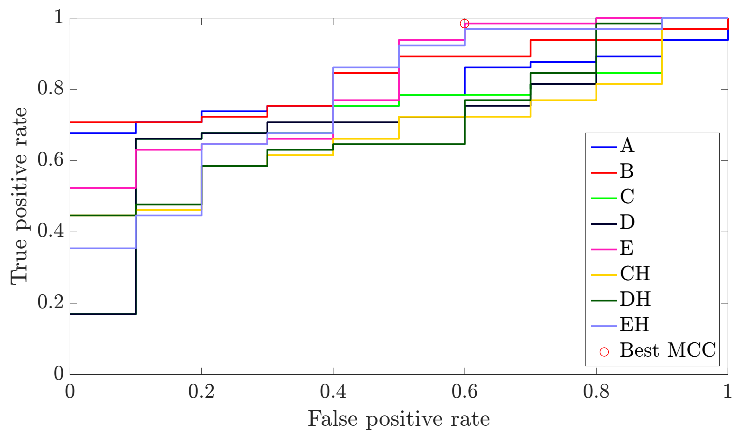
As previously, the data used for the experiments was collected during the Colossus II sea trial in 2008 conducted by the CMRE. The sonar data was generated by the MUSCLE AUV equipped with a 300 kHz sonar. It consists of 212 SAS images each with a size of 526 $\times$ 160 pixels and a resolution of 18 mm $\times$ 25 mm. The images show several instances of CYs(67), TRs(70), and WEs(47) which pose as targets. Moreover, there are images of non-MMOs like ROs(28). Since an initial guess for the active contours algorithm is needed, the fast template matching of Chapter 3 is used; the position and geometrical extent of the best fitting template defines the starting superellipses for highlight and shadow ($\varepsilon = 1$).

5.2.1 Comparison of proposed similarity measures

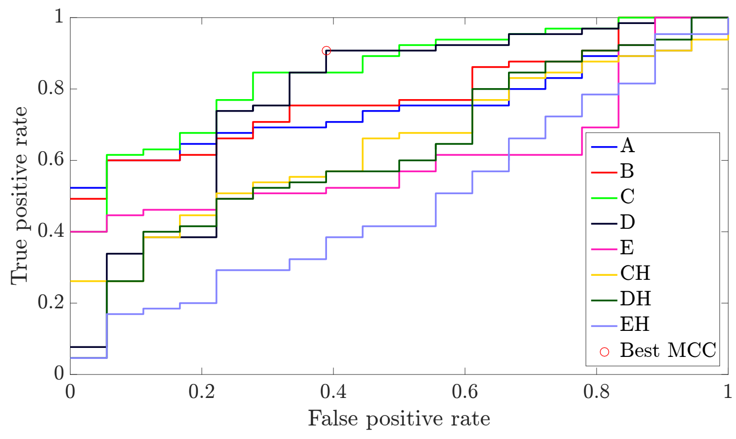
In Section 5.1.2 different methods of comparing the original image and simulated image, i.e., calculating their similarity are introduced. But since computation time is always a concern only the most suitable should be computed. Moreover, a threshold for when two images can be considered similar is also still needed for the different similarity measures. Therefore, all measures are calculated and for each measure a Receiver Operator Characteristic (ROC) is generated by setting the threshold to different values.

The data-set is first segmented by the SACE algorithm to derive the shape-describing superellipse parameters of the shadow and the highlight areas. Using the results of the last chapter, an appropriate decision is made on which of the four pdfs for the different image areas are chosen (see second column of Table 4.1). On the resulting superellipse parameters, a bootstrapped naive Bayes classifier is generated over B subsets consisting out of M samples drawn uniformly and with replacement. The general procedure to generate the ROC is:

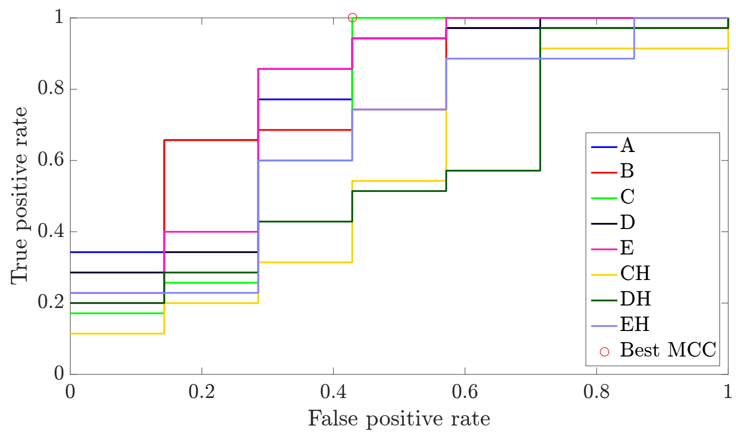
0. Set the models for the pdfs approximation of the different image areas (see second column of Table 4.1).
1. For all images in the training data set.
   (a) Run the SACE algorithm on the image with the shape defining parameters ν_1 and ν_2 for highlight and shadow as output.
   (b) Classify the example with the bootstrapped naive Bayes classifiers by majority vote (where only classifiers that were not trained on the particular training example have a vote).
   (c) With the class and pose information, simulate a sonar image.
   (d) Compute the similarity between the original and the simulated image according for all of the methods discussed in Section 5.1.2.
2. Generate ROC curves for each of the similarity measures. Differentiate by class.



(a) Cylindric: Highest AUC = 0.84 with B, highest Matthews Correlation Coefficient (MCC) = 0.52 with E.



(b) Trunc. cone: Highest AUC = 0.85 with C, highest MCC = 0.53 with D.



(c) Wedge: Highest AUC = 0.80 with A, highest MCC = 0.73 with C.

Figure 5.4 – ROC curve and Area Under the ROC Curve (AUC) for the three object types.

The analysis of which similarity measure to choose revealed that it depends on the simulated object type. Figure 5.4 shows the ROC curves for each object type and for each version of preparation discussed in Section 5.1.2. The optimal threshold was chosen according to the MCC [52] as suggested in [62]. Alternatively, one could argue that a false classification should never be accepted and hence, the optimal threshold should produce a false positive rate of 0. This, however, would in general also lead to a low true positive rate.

Another alternative are commonly used measures like Precision, Recall and derived thereof the F-factor. The drawback of said measures is that the performance prediction is influenced by the prevalence of positive examples in the test data and the bias of the classification model towards positive examples [62]. Since the test data for the verification phase is very unbalanced, assuming the classification worked reasonable well, the actual performance of a certain threshold could be overestimated.

For the cylindrical objects, the best performance according to the MCC is achieved by method E with around 98% true positive rate and 60% false positive rate in the verification. For the truncated cones method, D gave the best results (true positive rate/false positive rate = 91%/39%). For the wedge shaped objects, the highest score was achieved by method C, which compared the three-colour-image of the original image gained by SACE segmentation with the three-colour-image of the simulation; here a 100% true positive rate can be achieved with 43% false positive rate. Overall, one can note that the comparison based on three-colour-images, i.e., shape based, yields better results than the methods A and B.

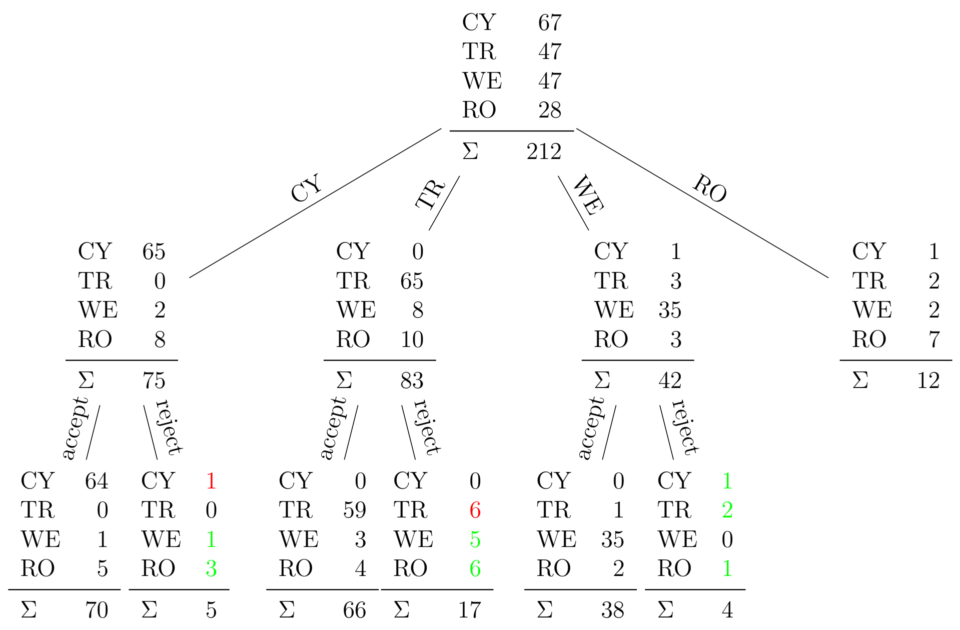
5.2.2 Summary of the algorithm

Combining the results of the previous and this chapter, Algorithm 4 arises. Figure 5.5 illustrates the result of using the whole algorithm described in Algorithm 4 with the pdfs combination chosen in the previous chapter on the training data set. On the left part of the tree, one can see that 75 objects are classified as cylindrical, albeit only 65 are actually cylindrical and 10 are misclassified. Of the correctly classified objects, 64 have a high enough similarity with the simulated cylindrical object to accept the hypothesis of correct classification. Four of the falsely classified examples and one actually correctly identified object are rejected. As discussed in the previous chapter, considering just the classification, 172 of the 212 training samples were classified correctly.

Unfortunately, but not surprisingly, the verification step decreased the true positive rate so that out of the 165 correctly classified MMOs, i.e., cylindrical, truncated cones and wedges, only 158 were accepted. At the same time, however, the number of incorrectly positive classified objects went from 35 to 16. One can see the effect of the validation step by comparing precision and recall first after the classification and then after validation. Table 5.1 shows the confusion matrix for classification only; Table 5.2 shows the confusion matrix after the validation. The recall value drops for cylindrical objects and truncated cones, but at the same time the precision improves significantly for all three MMOs types. The values for RO

Algorithm 4: SAS image classification and verification

```
Data: SAS image
Result: Class label with verification
Set the models for the pdfs approximation of the different image areas (c.f.
second column of Table 4.1);
Run the SACE algorithm on the image;
→ the shape defining parameters $\nu_1$ and $\nu_2$ for highlight and shadow;
Classify the example with the bootstrapped naive Bayes classifiers by
majority vote;
→ Class label;
With the class and pose information (via $\nu_1$ and $\nu_2$), simulate a sonar image;
→ Simulated image;
Compute the similarity between the original and the simulated image
according to the results of Section 5.2.1;
→ class label accept or reject;
```



CY 67 TR 47 WE 47 RO 28 Σ 212
CY
TR
WE
RO
CY 65 TR 0 WE 2 RO 8 Σ 75
CY 0 TR 65 WE 8 RO 10 Σ 83
CY 1 TR 3 WE 35 RO 3 Σ 42
CY 1 TR 2 WE 2 RO 7 Σ 12
accept
reject
CY 64 TR 0 WE 1 RO 5 Σ 70
CY 1 TR 0 WE 1 RO 3 Σ 5
CY 0 TR 59 WE 3 RO 4 Σ 66
CY 0 TR 6 WE 5 RO 6 Σ 17
CY 0 TR 1 WE 35 RO 2 Σ 38
CY 1 TR 2 WE 0 RO 1 Σ 4

Figure 5.5 – This tree shows the performance of the classification and verification step. For the verification the best similarity measure and threshold according to the ROC curves in Figure 5.4 was used.

Table 5.1 – Confusion Matrix of the classification phase. True positive rate 81.1%, false positive rate 18.9%.

		Predicted class				
		CY	TR	WE	RO	Recall
Class	CY	65	0	1	1	0.970
	TR	0	65	3	2	0.929
	WE	2	8	35	2	0.745
	RO	8	10	3	7	0.25
	Precision	0.867	0.783	0.833	0.583	

Table 5.2 – Confusion Matrix after the validation phase. Rejected examples are categorized as unknown(UK). True positive rate 77.8%, false positive rate 9.9%.

		Predicted class					
		CY	TR	WE	RO	UK	Recall
Class	CY	64	0	0	1	2	0.955
	TR	0	59	1	2	8	0.843
	WE	1	3	35	2	6	0.745
	RO	5	4	2	7	10	0.25
	Precision	0.914	0.894	0.921	0.583	-	

stay the same since they are not simulated.

A higher precision value means that a higher confidence can be put into the classification label. At the same time, assuming the worst case that a correctly classified object gets labelled as unknown(UK), it just means that an ambiguous object needs to be re-evaluated. Ultimately, this also leads to a better estimation of the confidence in each classification.

Table 5.3 shows again the comparison to older approaches. This time it includes the verification step.

5.3 Summary of the results on the top-down classification verification

The final step of a three-step approach for shape-based object recognition in sonar images was presented, which was applied to the challenging task of object classification on SAS images. After the superellipse driven segmentation and classification described in the previous chapter, the classification results are verified in a top-down process by means of simulation. For this, the class and pose information of the analysed object gained by previous steps is used to generate a simulated SAS image. Comparing this to the original SAS image yields a similarity value which is used to verify the class prediction of the classifier.

Table 5.3 – Classification and verification performance comparison with older approaches.

Approach	Overall	Cylindric	Trunc. cone	Wegde	Rocks
Ground truth	212	67	70	47	28
Dura etal [18]	136	48	59	24	5
4x3PP standard	143	42	58	38	5
from [36]	170	62	65	35	8
Classification	172	65	65	35	7
Verification	158/165	64	59	35	-

In the experiment section it was analysed which similarity measure achieves the best performance with respect to comparing the original and the simulated SAS image. It turned out that in general cross-correlating both images outperformed contour comparison via Hausdorff distance. The best pre-processing before the cross-correlation depended on the object type. Applied to the training data, 158 out of the 165 correctly classified objects were verified while 19 of the incorrectly classified got rejected.

CHAPTER 6

Summary and Conclusion of the Thesis

In this thesis, the object detection and classification of underwater objects was investigated. Usually, the shape of the objects of interest are known beforehand. The central idea of this thesis is that exploiting this a priori knowledge can lead to a more robust and successful detection and classification. The thesis begins with the introductory Chapter 1 where the content of this thesis with the advancements made and related work is described.

Chapter 2 introduced the data used throughout the experiments sections in this thesis. Besides the data itself also the general setting is introduced in which the data is gathered and why in these settings detection and classification of underwater objects is needed.

With Chapter 3 the introduction of new algorithms starts. The focus was on the detection of Man-Made Objects (MMOs) on Synthetic Aperture Sonar (SAS) images. A typical Autonomous Underwater Vehicle (AUV) has a coverage rate of $\geq 600\,\mathrm{m}^2\,\mathrm{s}^{-1}$. Since the goal is real time object recognition, i.e. during the mission, the whole processing chain from image generation to the final classification has to be efficient enough to process the incoming SAS data before new data arrives.

In Chapter 3 an efficient template matching based screening method was presented to rapidly reduce the Region Of Interest (ROI) on the SAS image. For this a method was adapted that originally was intended for face detection. This face detection method features a special image representation called integral images and a cascaded classification method called AdaBoost. The integral image allows for efficient template matching with constant computation time, while the cascaded AdaBoost algorithm does automated feature selection and classifier training.

Modified templates and template matching features based on statistical tests were introduced to make it more suitable for the task of object detection on SAS data. The experiments made on real data suggest that the new introductions are a valuable addition.

In Chapter 4, the central idea was to use the a priori knowledge that noise free object contours of the MMOs of interest can be described by superellipses. This led to two distinct approaches. In the first half of Chapter 4, the aim was to reconstruct an already extracted but noisy shadow contour via superellipse fitting. In the experiment section, it was shown that doing the fitting quadrant-wise lowered the fitting error significantly even when the fitting was done independently.

Moreover, a novel linearisation approach of the fitting error minimisation was introduced that sped up the computation time needed for the fitting. In fact, the computation time of independently fitting one superellipse per quadrant was in the linearised formulation even lower than the previous standard of fitting one superellipse to the complete contour. At the same time, it was significantly better able to represent the contour.

In the second half of Chapter 4, the superellipse a priori knowledge idea was taken one step further. Instead of relying on a possibly erroneous extracted contour that needed to be corrected, the contour extraction itself should be limited to only superellipses. The active contours without edges framework was modified to implement this idea. With the modification it is guaranteed that the segmenting contour is a superellipse.

Moreover, the core equation of the active contours without edges algorithm originally developed for photographies was reformulated and elevated to a more abstract level. This in turn made it possible to properly adjust the equations to account for the pixel intensity distributions of a given imaging system.

In the experiments section the proposed Superellipse guided Acitive Contours without Edges (SACE) algorithm was first evaluated on a set of real SAS images. It was shown that the extracted superellipse parameters can be used to classify the objects. The performance of the superellipse fitting on an already extracted contour from the first half of Chapter 4, the SACE algorithm that integrates the superellipse fitting directly in the segmentation, and a method from the literature that also separates segmentation and fitting, were evaluated. The performance measure was based on the ability to classify a test data set based on the extracted superellipse features. The used classifier was a bootstrapped naive Bayes classifier.

The SACE approach outperformed the best fitting approach from the first part of the chapter as well as the fitting approach from the literature. Even though the main use case in this thesis were SAS images, this enhanced algorithm is thanks to the abstract view point easily transferable to other imaging systems. This is why the algorithm could also be successfully applied to segment objects on underwater camera images. Possible other applications could be medical ultrasound where image quality is similarly poor and ellipses and superellipse are already used as shape priors.

Eventually, in Chapter 5 the central question was how to verify the result of the classification. The potentially available confidence value of the classifier is not enough to judge the credibility of the result, due to the concatenation of complex algorithms where the classification is only the last step in a likely long chain. Slight errors in the beginning of the recognition process may accumulate and ultimately could lead to wrong classifications even though the classifier has high confidence in the classification.

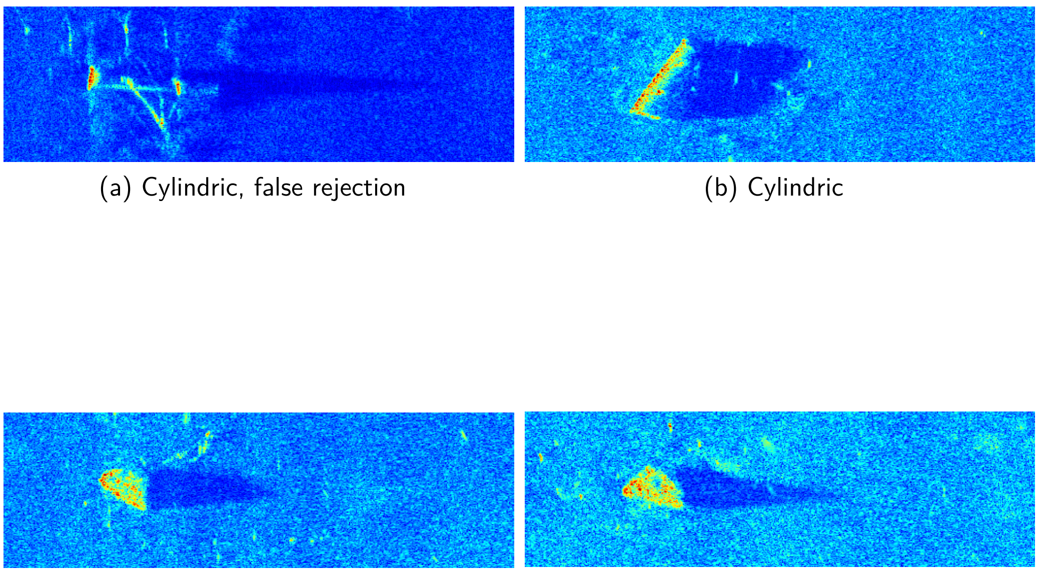
Therefore, a novel verification approach for object classification on SAS images was proposed that uses the class label from the classification and the pose estimation based on the extracted superellipse parameters to predict via simulation how the original input SAS image should have looked like. After the simulated image generation, a similarity measure between the simulated SAS image and the original SAS image is then used to determine whether to accept or reject the class label proposed by the classifier. A tool for approximating sound based imaging via a ray tracer for light based images was introduced to implement this idea.

The advantage of using a light based approach is that the process of simulating one image with the proposed tool is much faster than if the actual physical process of the sonar signal processing would be simulated. The approach of simulating the SAS image based on the precessing results and then comparing the simulated SAS image with the original SAS image was tested on a real SAS data set that was previously classified by the SACE approach described in Chapter 4. Out of the 172 correctly classified objects, 165 were verified while 19 of the incorrectly classified got rejected.Thus, the false positive rate was reduced by more than 50%.

Future work Even though the classification of the MMO worked reasonably well with the proposed approach, the results are still far from the desired 99+% accuracy. Especially object-sized stones on SAS images are a big challenge for every automatic system due to their arbitrary shape. Moreover, analysing examples where the proposed approach in the thesis fails reveals the limits of only SAS image based classification. Figure 6.1 shows (a) one example of a cylindrical object that was incorrectly rejected next to (b) an actual cylindrical object for comparison. Moreover, (c) a rock that was incorrectly verified as a wedge shaped object next to (d) an actual wedge shaped object.

The first two images highlight again the aspect-angle-dependence of an object on an SAS image. In this case, the cylindrical object lies perpendicular to the travel direction of the AUV. The only reason why this cylindrical object is rather easy to spot for a human operator are the attached cables. Unfortunately, the cables are not a reliable feature for identifying real objects, due to the fact that the cables are only attached to recover the test objects. However, if the same object is scanned from another angle it may look like the example in (b) and the ambiguity vanishes. This again emphasises the value of the pose estimation. With pose estimation, for example based on the superellipse parameters extracted with the SACE algorithm, an optimal angle for the next scan can be calculated to increase the probability of a successful classification. An interesting topic in this direction would be how to not just process all the different images independently but instead fuse the information gathered from multiple SAS images.

The second pair of images shows the limits of an SAS image based approach that works on individual images. It is doubtful that even a better angle would make the rock look less wedge-shaped. Therefore, future work has to focus on incorporating complementary information besides the SAS image. A starting point could be an additional processing of the SAS data via interferometry to get the object height [5]. Eventually, 3D data could be used to acquire the relevant missing information to robustly determine the object class. Interestingly, the same principle idea of Chapter 4 would still be applicable; the 3D data is inherently noisy and fitting a 3D superellipse, i.e. superquadric, could be of great benefit to suppress this noise [77].



(a) Cylindric, false rejection

(b) Cylindric

(c) Rock, false aceptance as wedge

(d) Wedge shaped

Figure 6.1 – Examples of incorrectly rejected/accepted objects and typical representatives for comparison.

Bibliography

[1] M. Atherton, *Echoes and Images: The Encyclopedia of Side-scan and Scanning Sonar Operations.* OysterInk Publications, 2011.

[2] S. Baglio, G. Muscato, and N. Savalli, “Tactile measuring systems for the recognition of unknown surfaces”, *Instrumentation and Measurement, IEEE Transactions on*, vol. 51, no. 3, pp. 522–531, 2002.

[3] J. Behn and D. Kraus, “Kernel based target detection for mine hunting in sonar images”, in *Proc. of the 4th International Conference on Underwater Acoustic Measurements: Technologies & Results (UAM 2011)*, Kos, Greece, 2011.

[4] J. Bell and L. Linnett, “Simulation and analysis of synthetic sidescan sonar images”, *IEE Proceedings - Radar, Sonar and Navigation*, vol. 144, no. 4, p. 219, 1997.

[5] P. Berkel and S. Leier, “A comparison of different techniques for object height determination – SAS interferometry versus shadow length”, in *UDT 2017*, 2017.

[6] P. Blondel, *The Handbook of Sidescan Sonar.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, p. 344.

[7] D. Boulinguez and A. Quinquis, “Classification of underwater objects using fourier descriptors”, in *Image Processing And Its Applications, 1999. Seventh International Conference on (Conf. Publ. No. 465)*, vol. 1, Jul. 1999, 240–244 vol.1.

[8] H. Büning, “Robust and adaptive tests for the two-sample location problem”, *Operations-Research-Spektrum*, vol. 16, no. 1, pp. 33–39, 1994.

[9] T. Chan and L. Vese, “Active contours without edges”, *Image Processing, IEEE Transactions on*, vol. 10, no. 2, pp. 266–277, Feb. 2001.

[10] A. G. Chavez, J. Fontes, P. Afonso, M. Pfingsthorn, and A. Birk, “Automated species counting using a hierarchical classification approach with haar cascades and multi-descriptor random forests”, in *IEEE Oceans*, 2016.

[11] A. C. Cohen, “Maximum likelihood estimation in the weibull distribution based on complete and on censored samples”, *Technometrics*, vol. 7, no. 4, pp. 579–588, 1965.

[12] E. Coiras and J. Groen, “Simulation and 3d reconstruction of side-looking sonar images, advances in sonar technology”, in, S. R. Silva, Ed. InTech, 2009, ch. 1, p. 232.

[13] J. Coronado-Vergara, G. Avina-Cervantes, M. Devy, and C. Parra, “Towards landmine detection using artificial vision”, in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, 2005, pp. 659–664.

[14] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database”, in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255.

[15] K. Denos, M. Ravaut, A. Fagette, and L. Hock-Siong, “Deep learning applied to underwater mine warfare”, in *Oceans ’17 MTS/IEEE, Aberdeen*, 2017.

[16] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, “Decaf: A deep convolutional activation feature for generic visual recognition”, *CoRR*, vol. abs/1310.1531, 2013.

[17] M.-P. Dubuisson and A. Jain, “A modified hausdorff distance for object matching”, *Proceedings of 12th International Conference on Pattern Recognition*, vol. 1, no. 1, pp. 566–568, 1994.

[18] E. Dura, J. Bell, and D. Lane, “Superellipse fitting for the recovery and classification of mine-like shapes in sidescan sonar images”, *IEEE Journal of Oceanic Engineering*, vol. 33, no. 4, pp. 434–444, Oct. 2008.

[19] B. Efron and R. Tibshirani, “Improvements on cross-validation: The .632+ bootstrap method”, *Journal of the American Statistical Association*, vol. 92, no. 438, pp. 548–560, 1997.

[20] A. Elbergui, I. Quidu, B. Zerr, and B. Solaiman, “Model based classification of mine-like objects in sidescan sonar using the highlight information”, in *ECUA 2012*, Edinburgh, United Kingdom, Jul. 2012.

[21] G. L. FORESTI and S. GENTILI, “A vision based system for object detection in underwater images”, *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 14, no. 02, pp. 167–188, 2000.

[22] Y. Freund and R. E. Schapire, “Experiments with a new boosting algorithm”, in *Proceedings of the Thirteenth International Conference in Machine Learning*, San Francisco, 1996, pp. 148–156.

[23] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: A statistical view of boosting”, *Annals of Statistics*, vol. 28, p. 2000, 1998.

[24] T. Fromm, C. A. Mueller, M. Pfingsthorn, A. Birk, and P. di Lillo, “Efficient Continuous System Integration and Validation for Deep-Sea Robotics Applications”, in *OCEANS 2017 - Aberdeen*, 2017.

[25] J. Gancet, P. Weiss, G. Antonelli, M. Pfingsthorn, S. Calinon, A. Turetta, C. Walen, D. Urbina, S. Govindaraj, P. Letier, X. Martinez, J. Salini, B. Chemisky, G. Indiveri, G. Casalino, P. di Lillo, E. Simetti, D. de Palma, A. Birk, T. Fromm, C. Mueller, A. Tanwani, I. Havoutis, A. Caffaz, and L. Guilpain, “Dexterous Undersea Interventions with Far Distance Onshore Supervision: the DexROV Project”, in *IFAC Conference on Control Applications in Marine Systems*, 2016.

[26] M. Geilhufe, W. A. Connors, S. A. V. Synnes, Ø. Midtgaard, T. O. Sæbø, R. E. Hansen, and S. Dugelay, “Assessment of mine hunting performance evaluation parameters across multiple side-looking sonar systems and frequencies”, in *OCEANS 2015 - MTS/IEEE Washington*, Oct. 2015, pp. 1–9.

[27] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2nd ed., ser. Springer Series in Statistics. New York, NY: Springer New York, 2009, p. 764.

[28] S. Hirose, S. Yokota, A. Torii, M. Ogata, S. Suganuma, K. Takita, and K. Kato, “Quadruped walking robot centered demining system - development of titan-ix and its operation”, in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 1284–1290.

[29] A. M. Kaneko, M. Marino, and E. F. Fukushima, “Humanitarian demining robot gryphon: New vision techniques and optimization methods”, in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 228–233.

[30] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models”, *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[31] K. Kato and S. Hirose, “Proposition and basic experiments of shape feedback master-slave arm-on the application for the demining robots”, in *Robotics and Automation, 2000. Proceedings. ICRA ’00. IEEE International Conference on*, vol. 3, 2000, 2334–2339 vol.3.

[32] K. Kato and S. Hirose, “Proposition of the humanitarian demining system by the quadruped walking robot-adaptability for various tasks using the foot-end-effecter changing mechanism”, in *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 1, 2000, 769–774 vol.1.

[33] D. Köhntopp, B. Lehmann, and D. Kraus, “Computational efficient object detection exploiting advanced templates”, in *Proceedings of the 1st international conference and exhibition on Underwater Acoustics (UA2013)*, 2013.

[34] D. Köhntopp, B. Lehmann, and D. Kraus, “Efficient superellipse fitting based contour extraction for mine-like shape recognition”, in *Proceedings of the 2nd international conference and exhibition on Underwater Acoustics (UA2014)*, 2014.

[35] D. Köhntopp, B. Lehmann, D. Kraus, and A. Birk, “Classification and localization of naval mines with superellipse active contours (accepted)”, *IEEE Journal of Oceanic Engineering*, 2018.

[36] D. Köhntopp, B. Lehmann, D. Kraus, and A. Birk, “Segmentation and classification using active contours based superellipse fitting on side scan sonar images for marine demining”, in *International Conference on Robotics and Automation (ICRA)*, IEEE Press, 2015.

[37] D. Köhntopp, B. Lehmann, D. Kraus, and A. Birk, "Autonomous mine recognition using AUV and ATR", in *UDT 2017*, 2017.

[38] D. Köhntopp, B. Lehmann, D. Kraus, and A. Birk, "Seafloor classification for mine counter measure operations on synthetic aperture sonar images", in *Oceans '17 MTS/IEEE*, Aberdeen, 2017.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks", in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105.

[40] S. Larionova, L. Marques, and A. T. de Almeida, "Toward practical implementation of sensor fusion for a demining robot", in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, 2004, 3039–3044 vol.3.

[41] S. Larionova, L. Marques, and A. T. de Almeida, "Features selection for sensor fusion in a demining robot", in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 3175–3180.

[42] S. Larionova, L. Marques, and A. T. de Almeida, "Detection of natural landmarks for mapping by a demining robot", in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2006, pp. 4959–4964.

[43] D. Lee, G. Kim, D. Kim, H. Myung, and H.-T. Choi, "Vision-based object detection and tracking for autonomous navigation of underwater robots", *Ocean Engineering*, vol. 48, no. Supplement C, pp. 59–68, 2012.

[44] B. Lehmann, "Beiträge zur automatisierten objekterkennung und insbesondere zur bild-segmentierung in hochaufgelösten sonarbildern", PhD thesis, Universität Wuppertal, 2013.

[45] B. Lehmann, D. Kraus, and A. Kummert, "Coupled curve evolution equations for ternary images in sidescan-sonar images guided by lame curves for object recognition", in *19th IEEE International Conference on Image Processing*, Sep. 2012, pp. 2553–2556.

[46] B. Lehmann, K. Siantidis, I. Aleksi, and D. Kraus, "Efficient pre-segmentation algorithm for sidescan-sonar images", in *2011 7th International Symposium on Image and Signal Processing and Analysis (ISPA)*, Sep. 2011, pp. 471–475.

[47] S. Leier, R. Fandos, and A. M. Zoubir, "Motion error influence on segmentation and classification performance in sas-based automatic mine countermeasures", *IEEE Journal of Oceanic Engineering*, vol. 40, no. 1, pp. 57–70, Jan. 2015.

[48] R. Lienhart, A. Kuranov, and V. Pisarevsky, “Empirical analysis of detection cascades of boosted classifiers for rapid object detection”, in *Pattern Recognition: 25th DAGM Symposium, Magdeburg, Germany, September 10-12, 2003. Proceedings*, B. Michaelis and G. Krell, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 297–304.

[49] O. Lopera and Y. Dupont, “Automated target recognition with sas: Shadow and highlight-based classification”, in *2012 Oceans*, Oct. 2012, pp. 1–5.

[50] T. Luczynski and A. Birk, “Underwater image haze removal with an underwater-ready dark channel prior”, in *IEEE Oceans*, IEEE press, 2017.

[51] T. Luczynski, M. Pfingsthorn, and A. Birk, “The pinax-model for accurate and efficient refraction correction of underwater cameras in flat-pane housings”, *Ocean Engineering, Vol. 133, pp. 9-22, March 2017*, vol. 133, pp. 9–22, 2017.

[52] B. W. Matthews, “Comparison of the predicted and observed secondary structure of t4 phage lysozyme”, *BBA - Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.

[53] M. Mignotte, C. Collet, P. Perez, and P. Bouthemy, “Three-class markovian segmentation of high-resolution sonar images”, *Computer Vision and Image Understanding*, vol. 76, no. 3, pp. 191–204, 1999.

[54] Y. Mori and H. Tokuni, “Excavation depth and crush process for an excavation-type demining robot”, in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, 2005, pp. 505–510.

[55] C. A. Mueller, T. Fromm, A. G. Chavez, D. Köhntopp, and A. Birk, “Robust Continuous System Integration for Critical Deep-Sea Robot Operations Using Knowledge-Enabled Simulation in the Loop (submitted)”, in *International Conference on Intelligent Robots and Systems (IROS)*, IEEE Press, 2018.

[56] C. A. Mueller, T. Fromm, H. Buelow, A. Birk, M. Garsch, and N. Gebbeken, “Robotic bridge inspection within strategic flood evacuation planning”, in *IEEE Oceans*, IEEE press, 2017.

[57] M. Nakjem, R. Fardal, and B. Haugsvær, “Future norwegian unmanned maritime mine countermeasures”, in *UDT 2017*, 2017.

[58] J. A. Nelder and R. Mead, “A simplex method for function minimization”, *The Computer Journal*, vol. 7, no. 4, p. 308, 1965.

[59] F. J. Pereda, H. G. de Marina, J. F. Jimenez, and J. M. Giron-Sierra, “A development project of autonomous marine surface vehicles for sea demining”, in *Control Automation Robotics and Vision (ICARCV), 2010 11th International Conference on*, 2010, pp. 573–578.

[60] F. J. Pereda, H. G. de Marina, J. F. Jimenez, and J. M. Giron-Sierra, “Sea demining with autonomous marine surface vehicles”, in *Safety Security and Rescue Robotics (SSRR), 2010 IEEE International Workshop on*, 2010, pp. 1–6.

[61] Persistence of Vision Pty. Ltd. (2004), *Persistence of Vision Raytracer (Version 3.6).*

[62] D. Powers, “Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation”, *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[63] H. Proenca and S. Filipe, “Combining rectangular and triangular image regions to perform real-time face detection”, in *2008 9th International Conference on Signal Processing*, Oct. 2008, pp. 903–908.

[64] I. Quidu, J. Malkasse, G. Burel, and P. Vilbe, “Mine classification using a hybrid set of descriptors”, in *OCEANS 2000 MTS/IEEE Conference and Exhibition*, vol. 1, 2000, 291–297 vol.1.

[65] Y. Rachlin, J. M. Dolan, and P. Khosla, “Efficient mapping through exploitation of spatial dependencies”, in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, 2005, pp. 3117–3122.

[66] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “CNN features off-the-shelf: An astounding baseline for recognition”, *CoRR*, vol. abs/1403.6382, 2014.

[67] S. Reed, Y. Petillot, and J. Bell, “Automated approach to classification of mine-like objects in sidescan sonar using highlight and shadow information”, *Radar, Sonar and Navigation, IEE Proceedings -*, vol. 151, no. 1, pp. 48–56, Feb. 2004.

[68] S. Reed, Y. Petilot, and J. Bell, “A model based approach to mine detection and classification in sidescan sonar”, in *OCEANS 2003. Proceedings*, vol. 3, Sep. 2003, 1402–1407 Vol.3.

[69] P. Rosin, “Fitting superellipses”, *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 7, pp. 726–732, Jul. 2000.

[70] L. Saroul, O. Bernard, D. Vray, and D. Friboulet, “Prostate segmentation in echographic images: A variational approach using deformable super-ellipse and rayleigh distribution”, in *Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium on*, May 2008, pp. 129–132.

[71] J. Sawas, Y. Petillot, and Y. Pailhas, “Cascade of boosted classifiers for rapid detection of underwater objects”, in *Proceedings of the European Conference on Underwater Acoustics*, 2010.

[72] J. D. A. Silva, “Proposition and development of a robot manipulator for humanitarian demining”, in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 1209–1214.

[73] C. Spampinato, D. Giordano, R. Di Salvo, Y.-H. J. Chen-Burger, R. B. Fisher, and G. Nadarajan, “Automatic fish classification for underwater species behavior understanding”, in *Proceedings of the First ACM International Workshop on Analysis and Retrieval of Tracked Events and Motion in Imagery Streams*, ser. ARTEMIS ’10, Firenze, Italy: ACM, 2010, pp. 45–50.

[74] D. Sudac, S. Majetic, K. Nad, J. Obhodas, and V. Valkovic, “Improved system for inspecting minefields and residual explosives”, *Nuclear Science, IEEE Transactions on*, vol. 61, no. 4, pp. 2195–2203, 2014.

[75] M. Taron, N. Paragios, and M.-P. Jolly, “Border detection on short axis echocardiographic views using a region based ellipse-driven framework”, English, in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2004*, ser. Lecture Notes in Computer Science, C. Barillot, D. Haynor, and P. Hellier, Eds., vol. 3216, Springer Berlin Heidelberg, 2004, pp. 443–450.

[76] H. G. Urban, *Handbuch der Wasserschalltechnik*. Bremen: STN Atlas Elektronik, 2000.

[77] N. Vaskevicius and A. Birk, “Revisiting superquadric fitting: A numerically stable formulation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, pp. 1–1, 2017.

[78] L. Vese and T. Chan, “A multiphase level set framework for image segmentation using the mumford and shah model”, English, *International Journal of Computer Vision*, vol. 50, no. 3, pp. 271–293, 2002.

[79] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features”, in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2001, pp. 511–518.

[80] D. P. Williams and A. J. Hunter, “Multi-look processing of high-resolution sas data for improved target detection performance”, in *Image Processing (ICIP), 2015 IEEE International Conference on*, Sep. 2015, pp. 153–157.

[81] D. P. Williams and A. J. Hunter, “On the relationship between sas image resolution and target-detection performance”, in *OCEANS 2015 - Genova*, May 2015, pp. 1–5.

[82] D. P. Williams, “Bayesian data fusion of multiview synthetic aperture sonar imagery for seabed classification”, *IEEE Transactions on Image Processing*, vol. 18, no. 6, pp. 1239–1254, Jun. 2009.

[83] D. Williams and J. Groen, “Multi-view target classification in synthetic aperture sonar imagery”, in *Proceedings of the 3rd International Conference and Exhibition on Underwater Acoustic Measurements: Technologies and Results*, Nafplion, Greece, 2009, pp. 21–26.

[84] D. Williams, “Fast target detection in synthetic aperture sonar imagery: A new algorithm and large-scale performance analysis”, *IEEE Journal of Oceanic Engineering*, vol. 40, no. 1, pp. 71–92, Jan. 2015.

[85] D. Williams and J. Groen, “A fast physics-based, environmentally adaptive underwater object detection algorithm”, in *OCEANS, 2011 IEEE - Spain*, Jun. 2011, pp. 1–7.

[86] S. C. Wong and B. A. MacDonald, “A topological coverage algorithm for mobile robots”, in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 2, 2003, 1685–1690 vol.2.

[87] V. Yordanova and H. Griffiths, “Rendezvous point technique for multivehicle mine countermeasure operations in communication-constrained environments”, *Marine Technology Society Journal*, vol. 50, no. 2, pp. 5–16, Mar. 2016.

[88] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?”, *CoRR*, vol. abs/1411.1792, 2014.

[89] Y. Zhiyang, “Finding efficient robot path for the complete coverage of a known space”, in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, 2006, pp. 3369–3374.