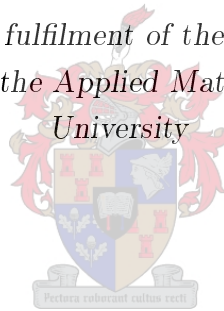


Shark Identification Using the Notches in the Dorsal Fin

by

Tessa Marais

*Thesis presented in partial fulfilment of the requirements for the degree
of Master of Science in the Applied Mathematics at Stellenbosch
University*



Department of Mathematical Sciences
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Supervisors:

Dr. S.J. Van der Walt Prof. B. Herbst

December 2016

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date:

Copyright © 2016 Stellenbosch University
All rights reserved.

Abstract

Shark Identification Using the Notches in the Dorsal Fin

T. Marais

Department of Mathematical Sciences

University of Stellenbosch

Private Bag X1, 7602 Matieland, South Africa

Thesis: MSc (Applied Mathematics)

December 2016

In order to protect endangered species, such as the Great White shark, a reliable estimate of the population is needed. In counting animals, it is necessary to be able to distinguish between individuals. The current procedure is manual photo identification, which is a time consuming process. By automating this process, when a shark is spotted, it can easily be matched to an existing shark in the database or added to the database as a new shark.

Photo identification software is already available for animal species such as penguins, elephants and dolphins. DARWIN, used for identification of dolphins was tested on sharks, but found to be unsuccessful in matching an unknown shark fin to the correct shark in the database. DARWIN also requires extensive user input, which is what we are trying to eliminate or greatly reduce in the identification of sharks. In this thesis, Hidden Markov models (HMM) is used to develop software to identify individual sharks. The results of the HMM was then compared to software using Dynamic Time Warping (DTW) to do the matching. The DTW program was able to correctly match 80% of the images within a rank of twenty and 62% with a rank of two or less. Using HMM, 84% of the photographs were correctly matched with a rank of twenty or less, but only 56% with a rank of two or less and 64% with a rank of five or less.

Although the HMM does not perform as well as the DTW, much better performance is expected from the HMM software by building up a quality database through the visual inspection and inclusion of photographs that will lead to more consistent models.

Uittreksel

Shark Identification Using the Notches in the Dorsal Fin

T. Marais

Department of Mathematical Sciences

University of Stellenbosch

Private Bag X1, 7602 Matieland, South Africa

Tesis: MSc (Applied Mathematics)

Desember 2016

In die bewaring van bedreigde spesies, is dit belangrik om 'n betroubare skatting van die populasie te hê. Wanneer die diere getel word, is dit nodig om hulle uit mekaar te kan ken. Huidiglik word foto identifikasie met die hand gedoen, wat 'n baie langdurige proses is. As hierdie proses programmaties gedoen kan word, sou dit moontlik wees om onmiddelik 'n haai uit te ken vanaf 'n foto.

Foto identifikasie sagteware is reeds beskikbaar vir pikkewyne, olifante en dolfyne. DARWIN, wat gebruik word vir die identifikasie van dolfyne, was getoets op haaie, maar was onsuksesvol om 'n onbekende haai te identifiseer deur dit te pas met die regte haai in die databasis. DARWIN vereis ook ekstensiewe invoer van die gebruiker en dit is juis wat ons probeer uitskakel of verminder in die identifikasie van haaie. In hierdie tesis, word die verskuilde Markov-modelle (HMM) gebruik om sagteware te ontwikkel wat 'n individuele haai kan identifiseer. Die resultate van hierdie sagteware word dan vergelyk met die resultate wat verkry is van die sagteware wat dinamiese tydsverbuiging (DTW) gebruik in die identifisering. DTW kon 80% van die haaivinne korrek identifiseer met 'n rang van 20 of minder en 62% met 'n rang van twee of minder. Hierteenoor kon HMM 84% korrek identifiseer met 'n rang van 20 of minder, maar slegs 56% met 'n rang van twee of een en 64% met 'n rang van vyf of minder.

Alhoewel HMM nie so goed soos DTW nie, sal die HMM sagteware heelwat beter vaar wanneer 'n goeie databasis opgebou word deur slegs die mees onlangse, hoë-kwaliteit foto's in die opleiding in te sluit.

Acknowledgements

First and foremost I offer my sincerest gratitude to my supervisor, Dr Stefan van der Walt, for his knowledge and patience during the first part of this project. His encouragement and support were invaluable. I would also like to thank my second supervisor, Professor Ben Herbst, for his extensive knowledge and the opportunity to learn from him. For his ever willingness to offer help and words of encouragement, I am grateful for Dr. Pieter Holtzhauzen. Besides programming knowledge, he also showed me the best mountain bike routes in Jonkershoek. These are all experts in their fields and provided much needed input in my steep learning curve.

I would especially like to thank Sara Andreotti, Marine Biologist (PhD) at the Department of Botany and Zoology (Evolutionary Genomics Group), Stellenbosch University, without whom this project would not have been possible. She supplied me with an abundance of sharkfin photographs, taken over several years and she also set this whole project in motion. Kerry Moss, MSc student at the time, also at the Department of Botany and Zoology, performed the laborious task of testing DARWIN. Thank you, Kerry.

Lize Cillie, doing her Honours project in Applied Mathematics on the segmentation of the sharkfin, I would like to thank for the collaboration and friendship in the first year of this project.

Last, but not least, I would like to extend my deepest gratitude to my parents for their continuous encouragement and support, and my husband for his patience, for lightening the load of the everyday mundane tasks so that I could focus on this project and his unwavering love and support.

Dedications

To my parents, my lifelong fans.

Contents

Declaration	i
Abstract	ii
Uittreksel	iii
Acknowledgements	iv
Dedications	v
Contents	vi
List of Figures	ix
List of Tables	xiv
List of Abbreviations	xv
List of Symbols	xvi
1 Introduction	1
2 Identification Methods	3
2.1 Photo identification	3
2.1.1 Advantages	3
2.2 Manual identification	3
2.2.1 The Rutzen method	3
2.2.2 Disadvantages	4
2.3 Semi-automatic identification software	4
2.3.1 Identification software for different animal species	4
2.3.2 The dorsal fin as distinguishing feature	6
2.4 The DARWIN software	6
2.4.1 Tracing the outline of the dorsal fin	8

<i>CONTENTS</i>	vii
2.4.2 Feature extraction	9
2.4.3 Matching	9
3 Preparation of the data	11
3.1 Preparing the database	11
3.1.1 Expected difficulties	11
3.1.2 Manual pre-processing	14
3.2 Edge extraction	14
3.2.1 Tracing the outline	14
3.2.2 Extracting the edge	15
3.2.3 Orientation	18
3.3 Normalization	18
3.3.1 Mean normalization	19
3.3.2 Singular Value Decomposition (SVD)	20
4 Dynamic programming	21
4.1 Dynamic Time Warping (DTW)	21
4.1.1 Matching unaligned series	21
4.1.2 Constraints	22
4.1.3 The cost matrix	23
4.2 The software	24
4.2.1 Summary	25
5 Hidden Markov Model	27
5.1 Gaussian Mixture Models	27
5.2 Building the HMM software	31
5.2.1 Sequential data	31
5.2.2 Markov models	32
5.2.3 Hidden Markov models (HMM)	33
5.2.4 The parameters	34
5.2.5 The Viterbi algorithm	35
5.2.6 Training	36
5.2.7 The expectation maximization (EM) algorithm	37
6 Training and Testing	38
6.1 Database	38
6.1.1 Gathering of the data	38
6.1.2 Sorting the data	41
6.1.3 Extraction of fin edges and normalization	42
6.2 Training	45
6.2.1 GMM	50

<i>CONTENTS</i>	viii
6.2.2 Training using the HMM	50
6.3 Testing	51
6.3.1 Control	51
6.3.2 Random photographs	51
6.3.3 Evaluating measures	51
6.3.4 Optimizing the algorithm	52
6.3.5 Summary	52
6.4 Results	52
6.4.1 HMM	53
6.4.2 Testing of DARWIN	53
6.4.3 Results of testing DTW	59
6.5 Summary	60
7 Conclusions and future work	61
7.1 Recommendations	61
7.2 Future work	62
Bibliography	64

List of Figures

2.1	The Rutzen method involves dividing the image of the shark fin into three sections and counting the notches in the serrated edge of the fin for all three sections.	4
2.2	The morphology of a shark's fin is shown in the sketch at the top, showing the free rear tip which might lead to distortion on a photograph of the lower part of the fin due to bending. The rectangle in the photograph at the bottom left shows the free rear tip clearly.	5
2.3	Photo identification. The green line in the image at the top traces the edge of the elephant's ear, which is used as distinguishing feature. The red circles on the photograph of the penguin indicate the black markings that are unique to every individual penguin.	7
2.4	The difference in anatomy of a shark's fin (on the left) to a dolphin's fin (on the right).	8
2.5	The extraction of the outline of the dorsal fin of a dolphin. The original photograph of the fin (top left), thresholding (top right), morphological processing to remove unrelated parts (bottom left), and finally the extraction of the fin outline (bottom right).	9
2.6	Mapping the outline of the unknown fin to a fin in the database is shown here.	10
3.1	Examples of photographs to exclude are shown here. At the top left is a blurred photograph and the top right shows a photograph of a bent fin. The bottom left photograph is of very bad quality, however the extraction function as developed later in this thesis still manages to extract the correct and entire fin edge. Finally, in the photograph at the bottom right, the reflection on the water distorts the clarity of the fin edge. In the case of the blurred photograph (top left), the function <code>route_through_array</code> is unable to define the notches in their entirety.	12

LIST OF FIGURES

x

3.2	Examples of bad quality images. In the photograph at the top, the silhouette of the fin is reflected on the water. The photograph at the bottom left shows a shark fin partially submerged. At the bottom right a photograph shows two unwanted qualities: an out of focus fin and sunlight reflecting off the trailing edge of the fin. These type of photographs make extraction by the software difficult.	13
3.3	The two Sobel kernels, for the x direction (on the left) and the y direction (on the right).	15
3.4	The steps involved in edge extraction includes applying a Sobel operator to get an estimate of the gradient, representing the edges in both the x and y directions (left), and inverting the image to present the edges by the lowest values (right).	16
3.5	The problem encountered with extraction using <code>route_through_array</code> is shown in the image to the left. The <code>route_through_array</code> function fails to trace the entire depth of the deeper notches. The image in the middle shows the extraction of the fin when thresholding (Otsu's method) is applied before the <code>route_through_array</code> function extracts the fin. The image to the right is the binary, convoluted and inverted image that is used to extract the edge. The circle in the image shows where part of the edge of the fin is lost and traced on the background.	16
3.6	Some more examples where Otsu's method failed to successfully define the edge between the foreground and the background is shown at the top. In these images, it is almost impossible to discern the shape of the fin by eye, because too much of the background information is included and some of the edge detail is lost. The images at the bottom are the binary, convoluted and inverted images using adaptive thresholding. Adaptive thresholding results in very noisy images, as can be seen here.	17
3.7	Both edges are traced in order to determine the orientation of the shark in the image.	19
4.1	The difference between matching using the Euclidean distance, or direct point-for-point matching, and Dynamic time warping (DTW). The figure on the left shows the mismatch when using Euclidean distance and the figure on the right shows correct matching using DTW.	22

LIST OF FIGURES

xi

4.2	The grid matching two time series. The two series to match are shown on the horizontal and vertical axes and the path in the grid shows the matching points. This is the minimum cost path for the matching.	23
5.1	The density approximation of a two component Gaussian mixture model. The green and red curve shows the Gaussians for the two components and the dashed curve is the combination of the two into a Gaussian mixture model. The histogram is also shown as the bars.	30
5.2	A graph without links, representing sequential data where the observations are treated as independent and identically distributed.	32
5.3	A first order Markov chain of observations.	32
5.4	A second order Markov chain.	32
5.5	The representation of sequential data where latent variables form the Markov chain. Each observation is determined by the state of the corresponding latent variable.	33
6.1	Examples of how a shark's fin bend while swimming, leading to different extracted edge arrays as for straight fins.	39
6.2	Plotting the different extracted fin paths for two different sharks. In the top image, the paths extracted from the four different photographs of a shark, all follow the same pattern. The bottom image shows the paths extracted from three different photographs of the same fin. Here, however, the arrays does not fit on top of each other.	40
6.3	The photographs above show how a shark's fin can change over the years. The photograph on the left was taken in June 2009 and the photograph on the right was taken in October 2010. It can clearly be seen in the photograph on the right that a big notch appeared where previously there wasn't one. In this case the software might still be able to match the photograph to the right model, because the rest of the fin is still closely related.	42

6.4	The sharkfins in the photographs above almost seem to belong to two different sharks. The photograph on the left was taken in May 2009 and the photograph on the right was taken in October 2010. In the photograph on the right it can clearly be seen that the notches are more pronounced. The orientation of the sharkfin in the photographs are also very different leading to very different plots of the path of the trailing edge of the dorsal fin. Training on both of these fins, will lead to an inconsistent model and trying to correctly match one of these to such a model will be very difficult to nearly impossible.	43
6.5	The type of features that are very distinguishable (at the bottom) will result in a better match than less pronounced features (at the top).	44
6.6	How the fin extraction fail on blurred images.	44
6.7	Examples of the most common failures or problems encountered with the extraction of the edge. The user will mark the top of the fin as the start point and even though the lower part of these fins are underwater or occluded by splashing water, the user will mark where the end of the trailing edge is or is supposed to be, so as to define the entire trailing edge, albeit the lower part will not be accurately defined. The software for matching can then be written so as to only include the top two thirds of the fin for matching. This way, we are certain that the part of the fin that we use for matching is accurately defined in its entirety.	46
6.8	Examples of the fin extraction function defining the fin edge on the water is shown here. It is important to note that up until the lower part of the fin where the water occludes the fin, the fin extraction function is very successful in defining the edge accurately.	47
6.9	Here examples are shown of successful extracted edges. Most of these photographs are ideal in the sense that there is a stark contrast between the shark fin and the water in the background. All of these fins, however, have deep notches and the fin extraction function still manages to define these successfully.	48
6.10	More examples of successful extractions are shown here. In most of these photographs, there are considerable detail in the background water, such as reflections or splashing water. The fin extraction function defines the fin edges successfully even in these cases.	49
6.11	The confusion matrix for testing done on a small dataset consisting of 119 trained fin edges and trying to match 20 “unknown” photographs.	50

*LIST OF FIGURES***xiii**

- 6.12 The green dots in the photograph at the top is difficult to nearly impossible to see, leading to human error in the tracing of the outline of the fin, in cases where the user is required to adjust the outline of the fin. The circle in the image at the bottom shows where the user failed to adjust the outline of the fin because the dots on the photograph were too unclear. 55
- 6.13 Photographs of bad quality that DARWIN was unable to trace. These photographs were successfully traced by `route_through_array` used in the HMM software. 56
- 6.14 This image shows part of a fin that was traced manually, but the software returns points along the edge that does not include the notches in the fin edge, as can be seen in the part encircled on the image above. 57
- 6.15 The matching of a newly introduced photograph of an “unknown” shark is shown here. 58

List of Tables

6.1	Results from five different tests using Hidden Markov models. . . .	53
6.2	Summary of results comparing DARWIN, DTW and HMM. . . .	60

List of Abbreviations

DARWIN	digital analysis and recognition of whale image on a network
DP	dynamic programming
DTW	dynamic time warping
EM	expectation maximization
GMM	gaussian mixture models
GUI	graphical user interface
HMM	hidden Markov model
PCA	principal component analysis
SVD	singular value decomposition

List of Symbols

Constants:

$\pi =$ 3.1415926535897932384626433832795

$e =$ 2.7182818284590452353602874713526

Preprocessing:

G_x sobel x kernel

G_y sobel y kernel

GM gradient magnitude

μ mean

m number of values

s scaling factor

σ standard deviation

A matrix

U, V orthogonal matrices

Σ diagonal matrix

R rotation matrix

Gaussian Mixture models:

K number of clusters

z discrete variable

σ^2 variance

Σ covariance

P probability

γ mixture component probability

N number of data values

π mixing coefficients

Hidden Markov models:

N	number of observations
S	number of states
$a_{i,j}$	transition probability from state i to state j
A	transition matrix
t, T	time
π	initial state distribution
ϕ	emission probabilities
$b_j(x_t)$	emission probability for state j and observation x at time t
θ	set of parameters describing the HMM
X	data set
K	initial number of portions
S^*	state sequence
γ	latent variable marginals
δ	maximum probability
ψ	array to keep track of path

Chapter 1

Introduction

Many shark species are endangered as a result of the international shark trade as well as a low reproductive rate [1]. In order to protect the sharks, more information is needed regarding the sharks and their migratory routes. Other relevant information include the areas where sharks prefer to mate and nurse their young, and at what age they mature. Key to the survival of the species is accurate population estimates [27].

Photo identification is preferable to the method of attaching an electronic tag device to the shark fin, as it is less invasive and more economical than tagging the species. The method of photo identification is used on animal species where certain features can be distinguished. The serrated pattern along the trailing edge on the dorsal fin of a shark is unique to every individual shark, therefore serving as a “fingerprint” [32]. Manual photo identification is very time consuming as it involves manual inspection of thousands of photographs of shark fins, and human error is prevalent.

Recent advances in computer vision allow the development of automated identification systems. It has the added advantage that a database can be maintained that is accessible from anywhere in the world and it is possible to make an immediate identification of an animal with minimal user input [13]. Researchers can access and contribute to this database, making it possible to study world wide migration routes and track individual animals across the world.

The DARWIN package is a digital identification software program that identifies dolphins based on the shape of their fins, by tracing the fin and extracting features that make it possible to distinguish between individual dolphins. The identification of sharks using DARWIN was tested in at least two independent studies by different researchers and found to be unreliable as it produces matches with unacceptably high levels of error [5]. The testing of DARWIN was repeated and found to be very time consuming as extensive user input is needed in order

to extract the fin edges. We will look at the results of these tests in more detail in Chapter 6.

It is therefore the goal of this thesis to develop a software program for the automatic and accurate identification of individual sharks using Hidden Markov models as classification technique. This software program will then be tested on a prepared database of sharkfin photographs. Software using Dynamic Time Warping has been developed by Dr Holthauzen, specifically for the identification of the great white shark [3] and tested as part of one the independent studies mentioned in the previous paragraph. These results will be used, together with the results from the DARWIN tests, to compare to the results achieved in this thesis using Hidden Markov Models.

We will start in Chapter 2 with a discussion of manual photo identification and then look at current software available for the identification of different animal species. This will include a discussion of the DARWIN package as photo identification software. Chapter 3 looks at what needs to be done before the actual identification can take place; this includes preparation of the database, the extraction of the fin edges and normalization. Chapters 4 and 5 deal with the algorithms used, Dynamic Time Warping in the software developed by Dr Holtzhausen and Hidden Markov models, as used in the development of the software in this thesis, respectively. In Chapter 6, the testing of the different software is described and the results are discussed. The conclusions and recommendations are summarized in Chapter 7.

Chapter 2

Identification Methods

2.1 Photo identification

Photo identification is the technique of taking photographs of distinctive features of an animal and comparing it to photographs in an existing database in order to identify individual animals. Patterns or scars on the skin or fur of the animal provide the distinctive characteristics that make it possible to distinguish between individual animals. When identifying cetaceans, the dorsal fin or tail fluke is used, because the trailing edge of the dorsal fin has nicks and notches that are unique to every individual animal, much as the tail fluke of a whale has dark markings that are different for every individual animal. Photo identification has been done manually in the past, but with technological advances, semi-automatic photo identification is now possible.

2.1.1 Advantages

Photo identification is advantageous, because it is cheap, non-invasive and it can be done by non-experts. It is also accessible world-wide and allow studies of global migratory routes [35]. Photo identification allows scientists to identify and track individual animals as well as providing a more accurate estimation of the population.

2.2 Manual identification

2.2.1 The Rutzen method

The manual method used until recently for the identification of sharks is known as the “notches code”, referred to as the “Rutzen method (RM)” by O’Connell



Figure 2.1: The Rutzen method involves dividing the image of the shark fin into three sections and counting the notches in the serrated edge of the fin for all three sections.

[23], shown in Figure 2.1. This method identifies and uses the most prominent marks in the edge of the dorsal fin. Each new photograph has to be inspected and analysed and is then classified into a group according to these marks.

A standardised three-section grid is used to divide a high-resolution photograph of the dorsal fin into three parts. Notches in the trailing edge of the fin are counted and given a score, for each grid section [5]. The top two thirds are most reliable for making an accurate identification. This is mostly because the lower part of the fin is usually underwater and not visible on the photographs. Also, the lower part of the trailing edge of the fin consists of a free rear tip which is detached from the body of the shark, as shown in Figure 2.2, and may be subject to distortion.

2.2.2 Disadvantages

The Rutzen method is done manually and as there are thousands of photographs to compare against, this is very time consuming and includes an element of human error.

2.3 Semi-automatic identification software

2.3.1 Identification software for different animal species

There are existing software for digitally identifying animals such as whales, dolphins and elephants, and even penguins. The need for this type of software

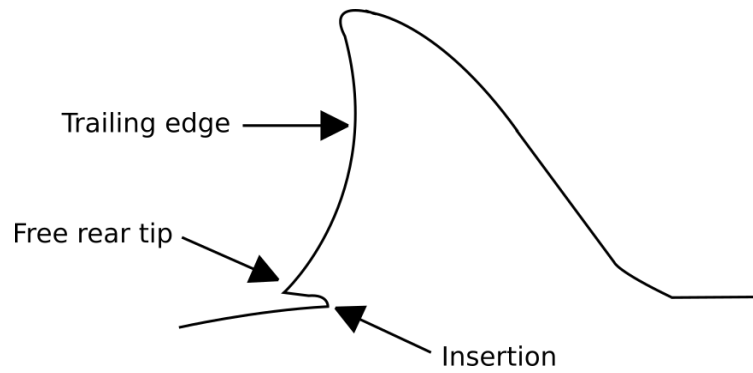


Figure 2.2: The morphology of a shark's fin is shown in the sketch at the top, showing the free rear tip which might lead to distortion on a photograph of the lower part of the fin due to bending. The rectangle in the photograph at the bottom left shows the free rear tip clearly.

arises in order to track animals by non-invasive means.

In the case of elephants, the distinguishing feature is provided by the shape of the nicks in their ears. Researchers in Italy published a paper in 2007 describing their software in the photo identification of wild elephants. In their program, the user is required to input several reference points [6].

Whales can be identified using the black markings on their tailfins. “Fluke Matcher” is a program released in 2010 that greatly increases efficient identification of humpback whales. User input is needed to identify the five major control points and the program then lists all the images in the database from most likely match to least likely match [22].

African penguins are distinguished by the pattern of black spots on their torsos. In 2004, students from the University of Bristol published a paper on the automated visual recognition of individual African penguins [11]. Researchers at the University of Bristol are using the software to track and study African penguins at Robben Island in South Africa [28].

Figure 2.3 shows two of these animals, an elephant and a penguin, with the distinctive features highlighted in yellow and red, respectively.

2.3.2 The dorsal fin as distinguishing feature

For both dolphins and sharks, the distinguishing feature is the dorsal fin, mostly because of the notches in the trailing edge of the fin. In the case of dolphins, the shape of the dorsal fin also varies among individuals, but it is often curved back. Thus the entire fin is important and used in identification software for dolphins.

However, for sharks, the trailing edge of the dorsal fin contains the most unique information, and is extracted for use in identification software. Here we find an important difference in the feature that is used for identification of these two different cetacean species. Figure 2.4 demonstrates the difference in the anatomy of a shark’s fin from a dolphin’s fin. We will now discuss software that was developed for the identification of bottlenose dolphins, the DARWIN package.

2.4 The DARWIN software

DARWIN is an acronym for “Digital Analysis and Recognition of Whale Images on a Network”. This software was developed in 1993 by John Stewman at Eckerd College in St. Petersburg, Florida [30]. DARWIN is open-source and faculty and undergraduate students from Eckerd College have improved on the

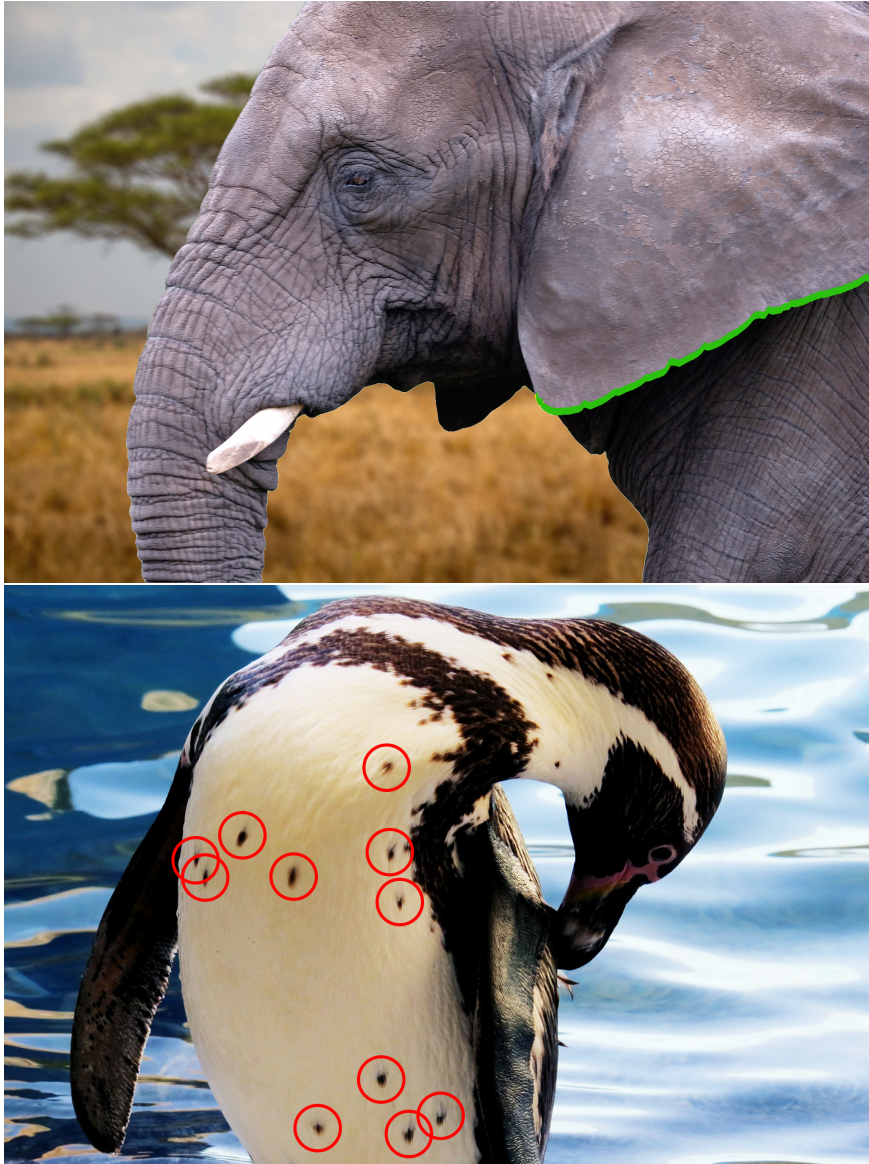


Figure 2.3: Photo identification. The green line in the image at the top traces the edge of the elephant's ear, which is used as distinguishing feature. The red circles on the photograph of the penguin indicate the black markings that are unique to every individual penguin.



Figure 2.4: The difference in anatomy of a shark's fin (on the left) to a dolphin's fin (on the right).

software since its debut. In 2000, Zach Roberts added chain codes and affine transformations and in 2008, Scott Hale introduced auto-tracing of the fin outline. The 2008 paper is the most recent publication listed on the DARWIN website [17]. DARWIN was developed for the use of researchers to identify individual bottlenose dolphins. The fin outline is approximated semi-automatically. DARWIN not only contains a collection of dorsal fin images, but also provides information on the individual dolphins and their sighting data. The dolphin images can also be sorted according to sighting location, sighting date or the extent of damage to the dorsal fin.

We have established that the distinguishing feature used in the identification of dolphins is the dorsal fin; the entire shape of the fin as well as the trailing edge containing the notches. First, the fin has to be isolated from the background in order to find its detailed shape. It is thus necessary to define the boundary between the fin and the water. By extracting this boundary, or edge, the matching between different fins can be done, in order to find the fin that is the best match. We will now look at how DARWIN handles both these issues.

2.4.1 Tracing the outline of the dorsal fin

The DARWIN software has a function called “Fin Trace” that automatically draws the outline of the edge of the dorsal fin. DARWIN software uses thresholding to segment the dorsal fin from the background in the image. The intensity threshold is selected using an unsupervised minimum error. Morphological processing then removes unrelated parts and the outline of the fin can be extracted. The points of this traced outline are moved to the actual positions that describe the edge of the fin, using active contours [21]. The user has the option of manually tracing the outline of the fin, before this repositioning of the outline [30]. Further user input may be required to move individual points in the case where

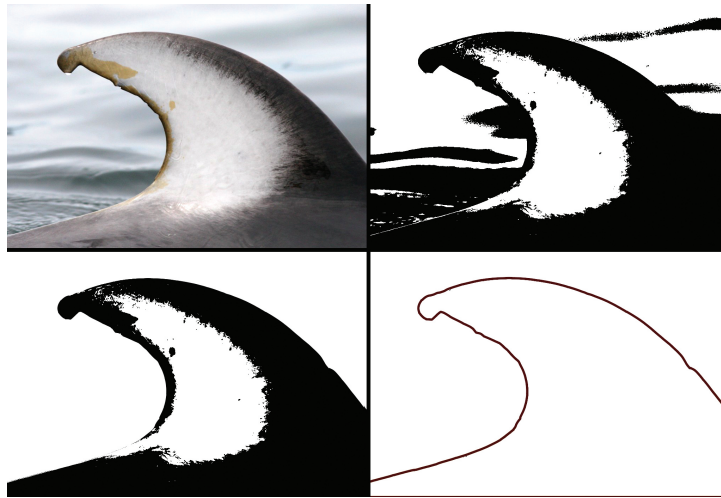


Figure 2.5: The extraction of the outline of the dorsal fin of a dolphin. The original photograph of the fin (top left), thresholding (top right), morphological processing to remove unrelated parts (bottom left), and finally the extraction of the fin outline (bottom right).

reflection occurs on the edge of the fin. The use of active contours may also cause the software to miss nicks and notches in the edge of the fin [5]. More input from the user is then required to move certain points in the outline of the edge to better define the nicks and notches. A representation of the automatic part of the process of extracting the outline of the dorsal fin is shown in Figure 2.5.

2.4.2 Feature extraction

The chain code representation is used to encode the outline of the dolphin fin. The chain code algorithm starts at the first point of the leading edge and computes the angular direction to every successive point along the edge. These points are taken at regular length intervals [2]. The software computes the directions between equally spaced points on the fin edge. Features of interest on the fin, such as the fin tip and certain notches would show up as a change in these directions and is used to solve for a transformation matrix in order to map one fin to another. This is a linear transformation such that the original measurements of ratios of the distances between points are maintained [2].

2.4.3 Matching

DARWIN compares the trace of the edge of the dorsal fin of a new photograph to the traces of edges existing in the database to make a match. This comparison

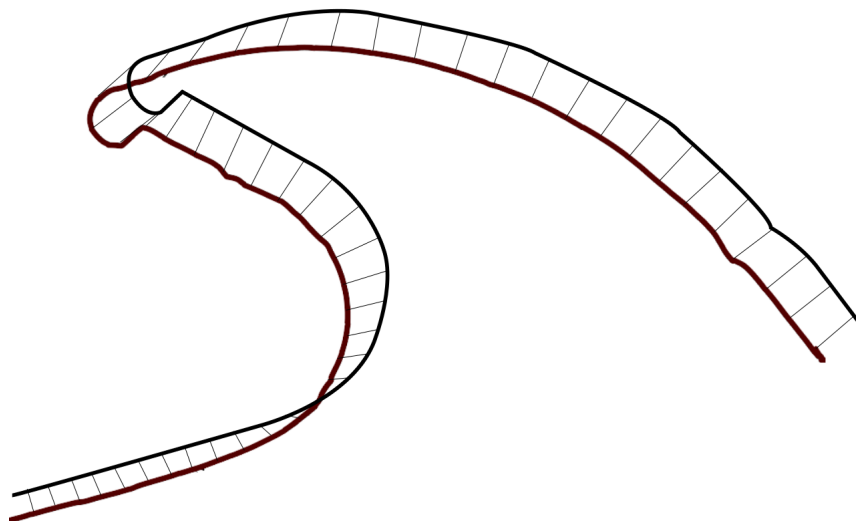


Figure 2.6: Mapping the outline of the unknown fin to a fin in the database is shown here.

is done by the function “Fin Matching” in the DARWIN software. The extracted fin is stored as a chain code type representation, highlighting important features in the fin outline. Using wavelet decomposition, the most prominent notch is located on the extracted edge. The points from one edge is mapped to other fin edges in the database by means of a transformation matrix. The most probable match is the fin edge in the database that produces the smallest mean squared error when mapped to the unknown fin edge [5]. The photographs in the database are ranked according to probability of match. A match of a new trace to a trace in the database of the exact same shark, has a rank of one [4].

Chapter 3

Preparation of the data

For both the Hidden Markov model software, as developed in this thesis, as well as the Dynamic Time Warping developed by Dr Holthauzen, we need to extract the outline. The pre-processing in this chapter is mostly specific for our data, but also includes standard pre-processing such as normalization, and will be applied to both software techniques, unless otherwise specified.

3.1 Preparing the database

3.1.1 Expected difficulties

The main difficulty faced by any automated recognition system is the quality of the photographs. The photographs are taken in imperfect conditions. Since it is difficult to capture the animals while they are swimming, the images may be blurred or the fin may be bent. In addition there are several other factors that cannot readily be controlled that severely impact on the accuracy of the recognition. These include unpredictable lighting and different photographic angles. When the sharks are spotted from far away and it is necessary to zoom to take their photograph, significant quality loss may occur [2]. In Figure 3.1, bad quality and blurred photographs are shown, as well as a photograph of a bent fin and the reflection of the sunlight on the water in another photograph. Figure 3.2 shows more such difficulties, such as the reflection of the fin on the water, the fin being partially occluded by water and reflected light on the trailing edge of the fin.

It is therefore necessary to manually inspect each and every photograph and identify such photographs as shown in Figure 3.2, where not enough fin is visible for identification, the lighting is not sufficient or blurring will make edge extraction difficult, and exclude these from the database. In this way, a



Figure 3.1: Examples of photographs to exclude are shown here. At the top left is a blurred photograph and the top right shows a photograph of a bent fin. The bottom left photograph is of very bad quality, however the extraction function as developed later in this thesis still manages to extract the correct and entire fin edge. Finally, in the photograph at the bottom right, the reflection on the water distorts the clarity of the fin edge. In the case of the blurred photograph (top left), the function `route_through_array` is unable to define the notches in their entirety.

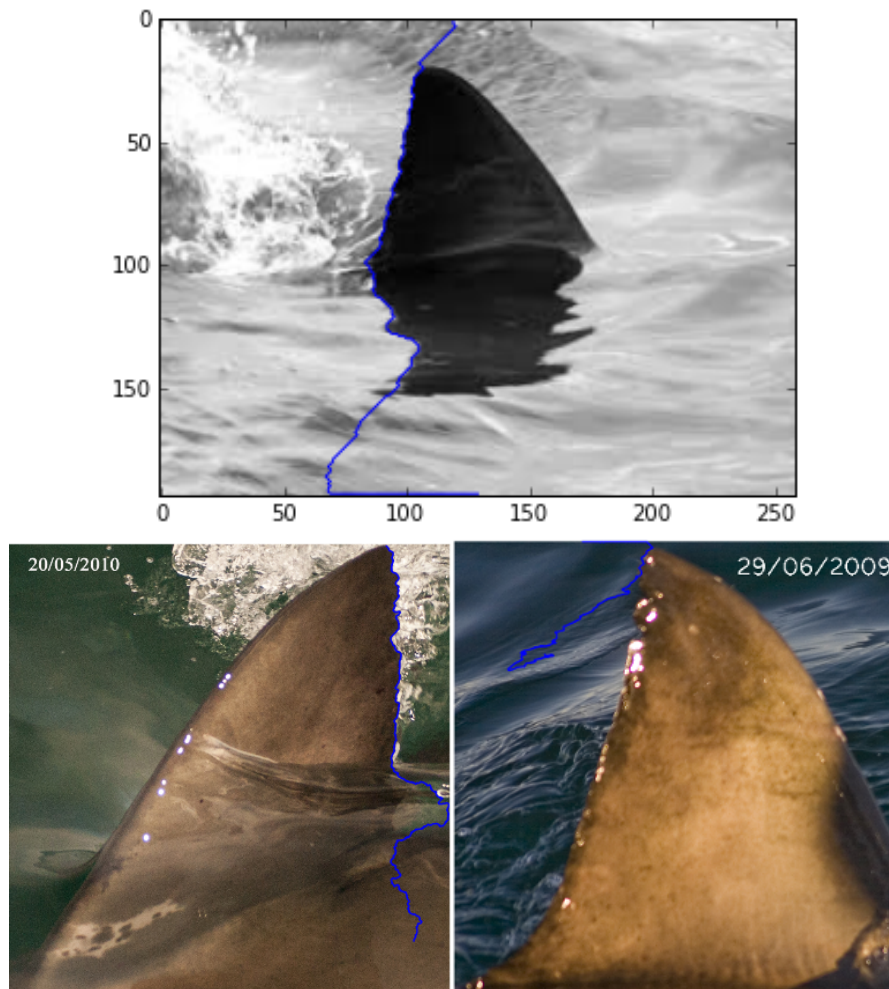


Figure 3.2: Examples of bad quality images. In the photograph at the top, the silhouette of the fin is reflected on the water. The photograph at the bottom left shows a shark fin partially submerged. At the bottom right a photograph shows two unwanted qualities: an out of focus fin and sunlight reflecting off the trailing edge of the fin. These type of photographs make extraction by the software difficult.

carefully curated database is set up that will increase our chances of accurate matching.

3.1.2 Manual pre-processing

The photographs are edited manually by cropping the shark fin so that it fills the whole image and as much of the unnecessary background is removed. This will be standard procedure in most recognition software, as it makes the extraction of the important features easier.

Some sharks are photographed from the left and some from the right. This poses a problem for the matching part in the software, because the mirror image of a photograph will look like an entirely different photograph to the matching software. The orientation of the sharkfin can be automated by tracing both edges and returning the edge with the most variance, thereby knowing if the photograph should be flipped or not. This solution of orientating the fin is discussed in more detail in Section 3.2.3 and the disadvantages of doing the orientation programmatically is listed. However, since manual intervention is such a crucial part of the pre-processing, orientating the fin will also form part of this pre-processing so that all sharks are orientated in the same default direction. Manual intervention as described here is feasible, because it will reduce computing time and prevent possible errors. The cropping and flipping of images are basic editing options in any photo editing software.

In the case of the DTW software, the photographs are not flipped. Instead, both the flipped and the unflipped versions are matched against the photographs in the database.

3.2 Edge extraction

The edge extraction is the first step in the program and is very crucial to the success of the algorithm. We need to extract the correct edge successfully in order to do the comparisons. Since identification relies on the notch pattern in the trailing edge of the fin, identification will be impossible if the edge is not accurately extracted. We will now look at how the extraction of the fin edges are done for use in both the HMM and DTW software.

3.2.1 Tracing the outline

First of all, the fin needs to be isolated from the rest of the image. We perform edge detection using a Sobel filter. The Sobel filter uses two 3×3 kernels, shown in Figure 3.3, one for the x -direction and one for the y -direction, to convolve with the image in order to approximate the derivatives in both directions. These

-1	0	+1	+1	+2	+1
-2	0	+2	0	0	0
-1	0	+1	-1	-2	-1
Gx			Gy		

Figure 3.3: The two Sobel kernels, for the x direction (on the left) and the y direction (on the right).

approximations of the derivatives are combined in the following manner, to get the gradient magnitude (GM), that describes the edges in both directions in the image [34]

$$GM = \sqrt{G_x^2 + G_y^2} \quad (3.1)$$

In Figure 3.4, the image on the left, shows the result of the convolution, highlighting the edges in the image. The edge most clearly visible is the edge separating the foreground from the background and is the edge that we are interested in.

3.2.2 Extracting the edge

In order to extract this edge from the image, we use an algorithm in Scikit Image, called `route_through_array`. The function takes an array as input as well as a start and end point, calculates the path through the array with the minimum cost, then returns a list of n-d index tuples that describes the path. The total cost of this minimum cost path is returned as well [24]. An optional parameter “`fully_connected`” allows one to set the algorithm to permit diagonal moves or only axial moves. A simple sum of the values along the path is used to calculate the costs. We invert the image so that the edge is presented by black (which has the lowest value) and the rest of the image is white, having the highest value. A weight is also applied, by taking the square of the edge strength, in order to further enhance the contrast between the foreground and background. The image to the right in Figure 3.4 shows the convoluted image with inverted edges. Taking the cube of the edge strength instead of the square, led to better results. This weight is called the edge affinity and describes how likely the path is to stick to an edge. It must be greater than 1, a typical value is around 3, and is also the default value.

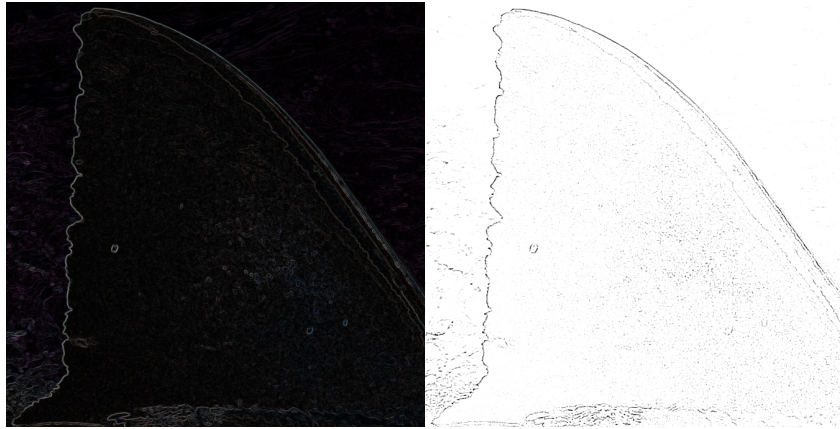


Figure 3.4: The steps involved in edge extraction includes applying a Sobel operator to get an estimate of the gradient, representing the edges in both the x and y directions (left), and inverting the image to present the edges by the lowest values (right).

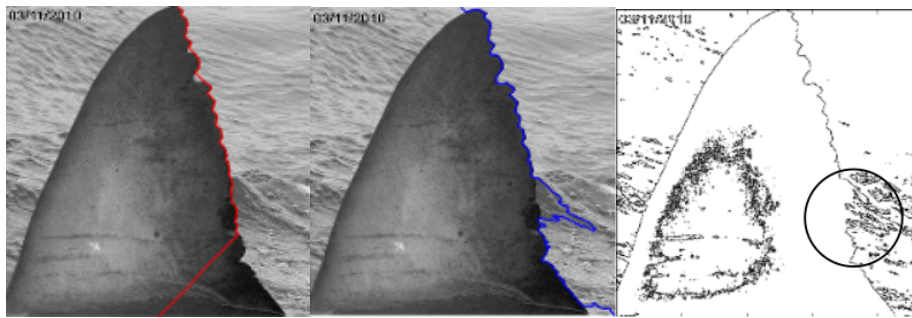


Figure 3.5: The problem encountered with extraction using `route_through_array` is shown in the image to the left. The `route_through_array` function fails to trace the entire depth of the deeper notches. The image in the middle shows the extraction of the fin when thresholding (Otsu's method) is applied before the `route_through_array` function extracts the fin. The image to the right is the binary, convoluted and inverted image that is used to extract the edge. The circle in the image shows where part of the edge of the fin is lost and traced on the background.

Initial problems with tracing using `route_through_array`

Using the function `route_through_array` to extract the edge, with the edge affinity as 2, we notice some shortcoming in the function for our purpose. The function seems to underdefine some of the deeper notches in the edge of the fin, ie. it doesn't trace the entire depth of some of the notches, as shown in Figure 3.5. We want to extract the edge of the fin as accurately as possible, in order to make the best matching.

In an attempt to solve for this shortcoming in the `route_through_array` function, we apply thresholding before the Sobel operator is used. Thresholding

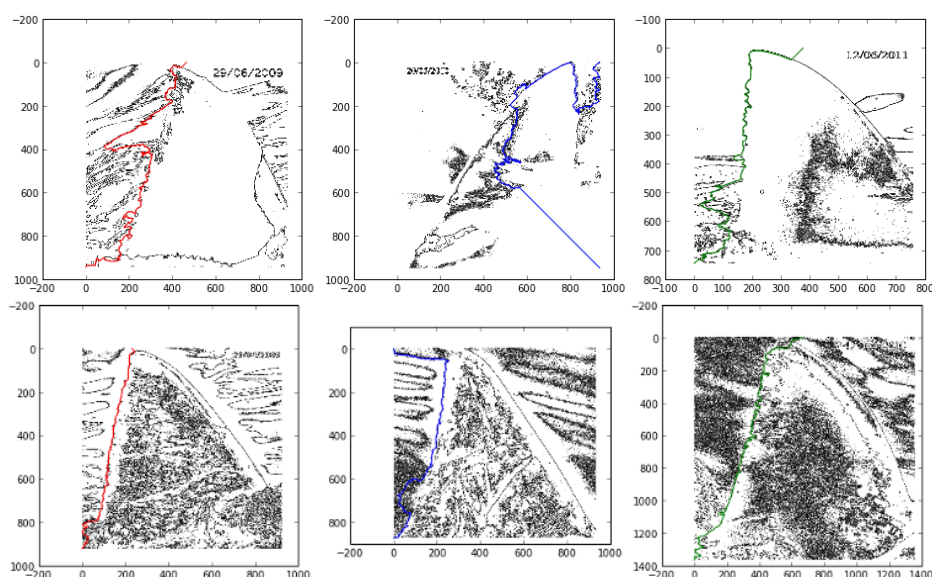


Figure 3.6: Some more examples where Otsu's method failed to successfully define the edge between the foreground and the background is shown at the top. In these images, it is almost impossible to discern the shape of the fin by eye, because too much of the background information is included and some of the edge detail is lost. The images at the bottom are the binary, convoluted and inverted images using adaptive thresholding. Adaptive thresholding results in very noisy images, as can be seen here.

is one of the simplest methods of image segmentation. A binary image is created by changing the intensity of a pixel to black if it is below the chosen threshold or white if above the threshold. For completely automated thresholding, the threshold needs to be selected by the computer.

The two thresholding algorithms available in the Scikit Image library is `threshold_adaptive` and `threshold_otsu`. Otsu's method is a clustering-based thresholding method that calculates and then applies the optimal threshold. Using Otsu's method, the function `route_through_array` is able to define in entirety the deeper notches. We notice however, that by using Otsu's method, the edge is not as clearly defined everywhere along the edge as was the case without any thresholding. Some of the detail of the edge is lost when thresholding is applied.

Testing Otsu's method on 50 images, we find that complete extraction of the fin edge failed for almost half of the images. Here we abandoned any further attempt to make use of Otsu's method for thresholding. In Figure 3.6, at the top, more examples of how Otsu's method fails to successfully define the edge between foreground and background are shown.

Adaptive thresholding is a technique that changes the threshold dynamically over the image to adapt to the changing lighting conditions in the image. Fig-

ure 3.6, at the bottom, shows the binary, convoluted and inverted images where adaptive thresholding was applied. It is clear that adaptive thresholding results in very noisy images and as can be seen in the middle of the bottom images, the function `route_through_array` finds part of the path on the water, because the boundary between fin and water is not clear at all. Attempts to use thresholding is completely abandoned. It becomes clear that the function `route_through_array` is sufficient in its extraction of the edge of the fin without thresholding. Even though the function does not define the deep notches entirely, convolution with the Sobel operator and inversion of the image (without thresholding), clearly defines the boundary between fin and background, and `route_through_array` is able to extract a more precise edge than with thresholding, although not entirely accurate. This will most probably not be a problem, because the same fin in a different photograph will be extracted in the same way, because of the good repeatability of the `route_through_array` function. Setting the edge affinity to 3, more accurately defined the fin edges.

3.2.3 Orientation

As stated in Section 3.1.2, the orientation is only fixed for the HMM software so that all the sharks in the photographs swim in the same direction. Detecting the orientation of the trailing edge can be automated in a straightforward manner. To do this, we call the `route_through_array` function twice to trace both edges of the sharkfin. In order to ensure that both edges are captured, we supply the end of the path to be in the lower left bottom of the image in one call to the function and the end to be in the lower right bottom of the image in the other. In both instances we supply the start point as the middle of the top of the image. Then we find the path with the most variance, as this will be the serrated, trailing edge of the fin and we can flip the image accordingly so that the shark is swimming in the chosen default direction. In Figure 3.7, we can see the two paths that are extracted by the Scikit Image function, `route_through_array`.

The trailing edge on some fins, however, are very smooth and in these cases the algorithm will have difficulty in determining which edge is the trailing edge and will therefore not be very accurate. We therefore conclude to include the orientation as part of the manual pre-processing step.

3.3 Normalization

Normalization is a common requirement for machine learning algorithms. The goal is to have standard normally distributed data and to scale features in a way so that they have comparable ranges of values. Usually this is done so that the

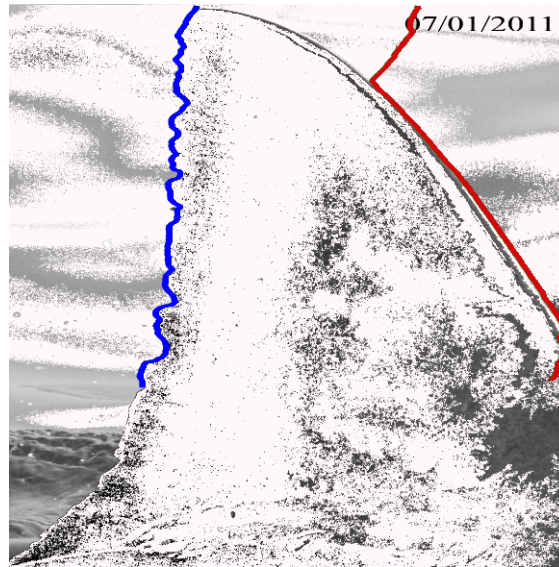


Figure 3.7: Both edges are traced in order to determine the orientation of the shark in the image.

data have Gaussian zero mean and unit variance, thereby normalizing position and scale.

The extracted fin edges come from different sized photographs, in which the fins are oriented at different angles and appear at different positions on the photographs. It is thus necessary to normalize with respect to size, position and in-plane rotation. The extracted edge is an ordered (x, y) array, which is the coordinates giving the position. We need to normalize with respect to position in both the x and y direction, using mean normalization. Mean normalization is used to normalize with respect to position for both the HMM and the DTW software. Normalizing the in-plane rotation and scale in the case of DTW, is done by applying Principal Component Analysis and then normalizing according to the image size and fin maximum. Singular value decomposition is used to normalize in the case of HMM for both the in-plane rotation and the scale.

3.3.1 Mean normalization

Mean normalization is when the data is adjusted by subtracting the mean of the data, in order to normalize position. We normalize in both the x and y direction. The calculation of the mean is simply the sum of all the values divided by the number of values in the feature vector,

$$\mu_x = \frac{1}{m} \sum_{i=1}^m x_i \quad (3.2)$$

where m is the number of values in the vector. The mean vector is then given by

$$\mu = \{\mu_x, \mu_y\} \quad (3.3)$$

Every value in the vector is then replaced with $x_i - \mu$.

3.3.2 Singular Value Decomposition (SVD)

The next step is to normalize the scale and the in-plane rotation, in order to be able to compare the data in a sensible way. Using SVD, we can easily achieve this. Given the extracted trailing edge of the fin, as a two-dimensional matrix, M , the SVD is given by

$$M = U \Sigma V^T \quad (3.4)$$

where U and V are orthogonal matrices and Σ is a diagonal matrix that contains the singular values of M , sorted in descending order of magnitude. The SVD then gives a decomposition of the matrix, M , into three simple transformations: the initial rotation, V^T , the scaling of the coordinate axes, Σ , and the final rotation, U [36]. The transformed vector is given by

$$\text{path}_{normalized} = \frac{\sqrt{n} \text{diag}(\Sigma) \bullet V}{\Sigma_{0,0}} \quad (3.5)$$

where n is the length of the feature vector and $\text{diag}(\Sigma)$ is the singular values of M . Combining SVD with mean normalization, the extracted edge is normalized with respect to position, scale and in-plane rotation.

With the trailing edges normalized, the next step is to develop an algorithm for comparing them, but before we discuss Hidden Markov models, we will first look at Dynamic Time Warping.

Chapter 4

Dynamic programming

When doing the matching of the edges, we need to take their nonlinear deformation into account. To do this, we will use dynamic time warping.

Dynamic time warping (DTW) is used when comparing two time series with each other. It has been used for speech recognition to be able to match words that are expressed at different speeds. Speaker recognition and signature recognition are other applications of the DTW algorithm. It works iteratively to align the two sequences by warping the time axis to find the best match [16].

4.1 Dynamic Time Warping (DTW)

4.1.1 Matching unaligned series

In the case of a time series, the two given sequences are warped non-linearly in the time dimension to measure their similarity. When dynamic sequences or coordinates as in our case are considered, the non-linear warping is usually done on the axis that measures how far along the point is on the sequence, usually the x -axis. This non-linear warping on the x -axis is shown in Figure 4.1. A direct comparison is not possible since the two sequences in general experience nonlinear deformations with respect to one another. In order to compare the two sequences, this nonlinear deformation has to be determined. One of the earliest algorithms for doing this, is the so-called Dynamic Time Warping algorithm, essentially Dijkstra's algorithm [19].

The algorithm essentially finds the smallest warping path, W , relating the two sequences, that minimizes the total warping cost,

$$\text{dist}(W) = \sum_{n=1}^N \text{dist}(w_{ni}, w_{nj}) \quad (4.1)$$

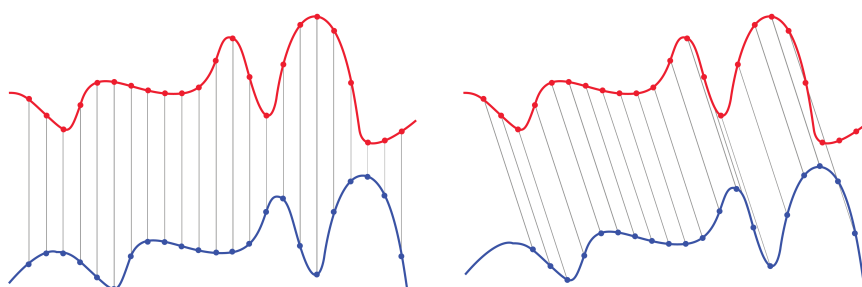


Figure 4.1: The difference between matching using the Euclidean distance, or direct point-for-point matching, and Dynamic time warping (DTW). The figure on the left shows the mismatch when using Euclidean distance and the figure on the right shows correct matching using DTW.

As we are interested in comparing each value of one sequence to each value of the other sequence, we map one sequence onto the other in a grid. Assuming that the endpoints match, the lowest cost path runs from the lower left to upper right hand corner as shown in Figure 4.2, with the diagonal indicating only a linear deformation, $M \neq N$.

4.1.2 Constraints

The following constraints are assumed when building the model using dynamic time warping. They refer to the path that is tracked on the grid matching the two time series and limit the number of paths that need to be considered:

- The path is not allowed to go backwards, only forwards.
- Continuity is important, the path is not allowed to skip an index.
- The start should be at the bottom left and the end at the top right. This ensures that the entire sequences are considered.
- As we know that similar signals will result on a path close to the diagonal, we can add a warping window to exclude paths that stray too far from the diagonal, in order to reduce the computational cost.
- The gradient of the path on the grid is also constrained to keep from aligning paths that differ too much in length. This prevents the matching of a very short sequence to a very long one, making the assumption that the two sequences that we want to match won't differ in length too much.

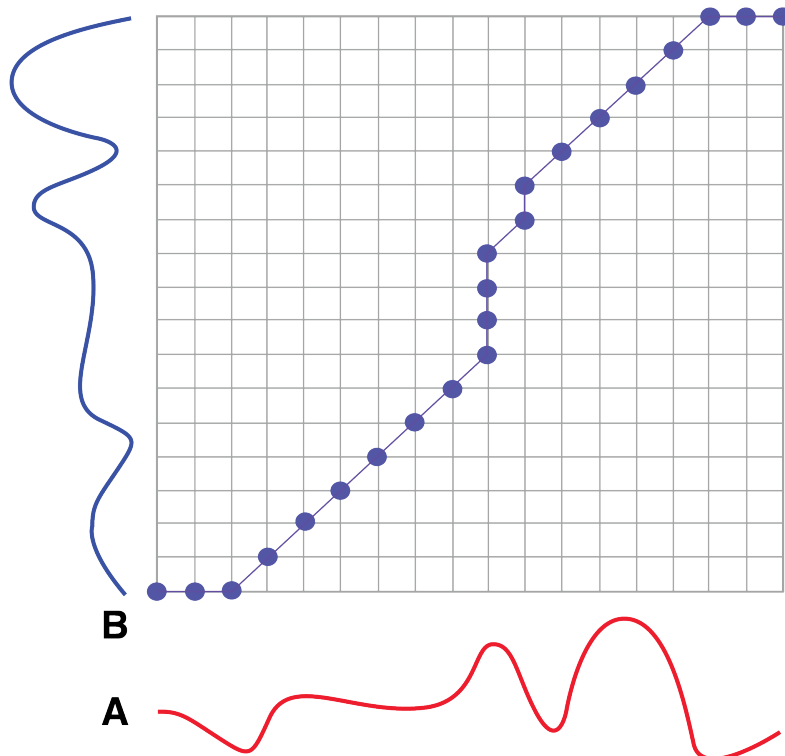


Figure 4.2: The grid matching two time series. The two series to match are shown on the horizontal and vertical axes and the path in the grid shows the matching points. This is the minimum cost path for the matching.

4.1.3 The cost matrix

Suppose two sequences are to be matched. One sequence has length N and the other has length M . We construct a matrix of size $N \times M$, where each cell represents the distance between the i -th element of the first sequence and the j -th element of the second sequence. The function used to calculate this distance metric depends on our application of the dynamic time warping. The Euclidean distance is commonly used.

On the grid, the best alignment between the two sequences is a path from the bottom left to the top right of the matrix. We need to calculate this path.

The cost of the path is calculated by simply summing the local cost in each cell up to that point. We want to find the path that minimizes the overall cost between the two sequences. There are numerous possible paths to follow from the bottom left to the top right of the matrix. To find the lowest cost path, we could use brute force and calculate the cost of each possible path. This would be computationally very expensive. We therefore make use of dynamic

programming and added constraints as explained in Section 4.1.2.

Dynamic Programming

The essence of the algorithm is to calculate the lowest cost path to every grid point, saving the total cost of the path up to that point, as well as the previous grid point on the path. The total cost of the optimal path to grid point (i, j) is given by

$$d = \|x_i - y_i\| + \min \begin{cases} d(i, j - 1) \\ d(i - 1, j) \\ d(i - 1, j - 1) \end{cases} \quad (4.2)$$

where d is the distance between the two points. Note the important assumption that (i, j) can only be reached from one of three previous connected points. This is the first two constraints listed in Section 4.1.2, stating that the path is only allowed to go forwards and that the path is not allowed to skip an index. This prevents extreme warping when matching the two edges. To find the best alignment, we backtrack from the top right cell. The top right cell itself contains the cost of the path. Thus, a lower cost equates to a better match or alignment between two sequences.

4.2 The software

Dr Holtzhausen developed an algorithm using Dynamic Time Warping to match an unknown photograph to shark fin photographs of identified sharks in a database. He also developed a full management system in order to use the software [3].

The advantage of this package is that only minimal user input is needed and the fact that the user input assists the algorithm in finding the correct edge of the fin. The top five matches returned, however, may include multiple photographs of the same individual shark. What we would actually like to see are five different sharks in the top five possible matches. In this way, if the software was unable to match the shark correctly with a rank of one, we might be able to choose the correct match in the remaining four possible matches.

The user is required to click on the start and end point of the trailing edge of the fin. Defining the start and end point of the trailing edge of the fin in this way, ensures that the path finding algorithm follows the edge on the correct side of the fin and also keeps the algorithm from running past the end point and including possible reflections of the fin on the water. The Sobel operator is

applied to the image and the `route_through_array` function is used to find the trailing edge of the fin.

As the top of the fin is the easiest feature to distinguish and mark, the top of the fin is used as the starting point in the DTW software. The top two thirds from the top of the edge of the fin are also the most reliable part in matching as discussed in Section 2.2.1. The input from the user ensures that the starting point is correctly defined as the top of the fin. Any fins that are too deeply submerged under the water would have been excluded in the pre-processing step, so we can assume that only photographs of complete fin edges will be included in the extraction and matching part of the software. As we will see later, some of the fins in the photographs are somewhat submerged or occluded by water splashes. In these cases, the user should still mark the end of the fin where it would have been, in order for the extracted edges to be of the correct length and the edges to match up. The assumption is therefore made that the endpoint defined by the user will in most cases be the correct end of the trailing edge of the fin.

The basic Dynamic Time Warping algorithm is used in the DTW software, with normalization applied before matching different variations of the fin. The fin edge is normalized with respect to in-plane rotation by using PCA. Principal Component Analysis (PCA) is calculated in order to rotate the fin edge to align to the horizontal axis.

As stated previously, the sharks swim in both directions in the photographs, both left and right. In the case of the DTW software, the photographs in the database are not flipped so that the sharks swim in only one default direction. To account for this, the fin edge to be matched is flipped and both the original (unflipped) and flipped versions of the fin are compared to the fin edges in the database. Another variation is to take only the first two thirds of the fin edge to be matched and match this against the first two thirds of every fin edge in the database. In this variation, both the unflipped and flipped versions are used for matching.

All these matching variations are scored and a combination of the scores is used to assign ranks to the fin edges in the database from most probable match (rank 1) to least probable match. The fin in the database to provide the lowest cost, is the most probable match. Only the five best ranked matches are returned in order.

4.2.1 Summary

In conclusion, the following is to be noted with DTW:

- The assumption is made that the start and end point of the two fin edges

that are to be compared, match up. The start point is straightforward, as this is the top of the fin and in all cases, this will be clear on the photograph. The end point is not so clear, as in some photographs, the bottom of the fin is occluded by water. In these cases, the user is required to mark the real end point as far as the user is able to guess where the end point should be. Because matching is also done by using only the top two thirds of the fin edge, the water occluded part of the fin will be excluded and won't have an effect on the matching.

- In order to account for differences in image size and fin orientation in the photograph, the edge array is normalized with respect to position, scale and in-plane rotation using mean normalization and PCA.
- The advantage of DTW is that two fin edge arrays can be compared directly, ie. there is no need to do training.

Chapter 5

Hidden Markov Model

When using the hidden Markov models, we need a chosen number of states. We do however not know where one state ends and where the next begins. In order to find the separations of the states, we will use Gaussian mixture models. The Gaussian mixture models are described in short first before discussing the Hidden Markov models, which is the main part of the matching algorithm.

5.1 Gaussian Mixture Models

A Gaussian or normal distribution has a bell-shaped curve. A single normal distribution, uni- or multi-variate, is very limited in that it can only describe uni-modal data. A more general model is the Gaussian Mixture models, shown in Figure 5.1, that can potentially model any data distribution and is also useful for clustering. Multivariate Gaussian distributions are used for more than one variable. Gaussian mixture models are used for classification.

Gaussian mixture models can be expressed using discrete latent variables and is trained using maximum likelihood estimation. Latent variables allow us to represent complicated distributions with simple components [10]. Our data does not come with class labels, but we need to divide the data into a number of smaller datasets in a meaningful way. We divide the data into k clusters, where k is known. In order to do this, we use a special version of the EM algorithm.

It is useful to introduce the discrete variable, z , that has a 1-of- K representation. A given element $z_k = 1$ indicates component k , while all other elements are zero. The constraints $z_k \in \{0, 1\}$ and $\sum_k z_k = 1$ are satisfied. The vector z can thus be in any one of K states, depending on the element that is nonzero.

Partially observed data

Generally, classification assumes that the training data is fully observed, ie. each observation is provided with a class label. In our case, the data is partially observed, ie. the data does not come with class labels, therefore the labels need to be inferred from the observed data. The labels in this case are represented by discrete latent variables.

Because we use the left-to-right model, we can simply divide the edge array into equal-sized segments, using the number of states to determine the number of segments to divide the edge array into.

Mixture Models

The most commonly used distribution for continuous variables is the Gaussian distribution. The two parameters that describe the Gaussian are the mean μ and the variance σ^2 . For a vector that is D -dimensional, the mean will also be a D -dimensional vector and the covariance will be a $D \times D$ matrix, Σ . The covariance must be symmetric and positive-definite. This Gaussian is called a multivariate Gaussian distribution [10].

Univariate Gaussian

The probability density function of the normal distribution is given by the equation

$$f(x|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (5.1)$$

A simplified notation describing the Gaussian distribution is $N(\mu, \sigma^2)$ and the elements of the sequence vector is given by x .

Multivariate Gaussian

For the multivariate Gaussian, the probability density function is of the form

$$f_x(x_1, \dots, x_k) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right) \quad (5.2)$$

Gaussian Mixture Models

The Gaussian mixture distribution is given by

$$p(x) = \sum_{k=1}^K \pi_k N(x|\mu_k, \Sigma_k) \quad (5.3)$$

We need to estimate the various parameters, which is the mixture component probabilities, the mean vector and the covariance matrix. We start by calculating the probability that an observation x belongs to component k , ie. we need to calculate $\gamma(z_k) = p(z_k = 1|x)$. Bayes' theorem immediately gives

$$\gamma(z_k) = \frac{p(z_k = 1)p(x|z_k = 1)}{\sum_{j=1}^K p(z_j = 1)p(x|z_j = 1)} \quad (5.4)$$

If this is written in terms of Gaussians, we find that

$$\gamma(z_k) = \frac{\pi_k N(x|\mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j N(x|\mu_j, \Sigma_j)} \quad (5.5)$$

The prior probability when $z_k = 1$, is π_k , and the corresponding posterior probability after observing x , is $\gamma(z_k)$. The posterior probability is also called the responsibility that component k contributes to the observation x [10].

Maximum likelihood

We want to fit a GMM to N given data values, $X = \{x_1, \dots, x_N\}$. The log likelihood is given by

$$\ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k N(x_n|\mu_k, \Sigma_k) \right\} \quad (5.6)$$

The partial derivative of the log-likelihood 5.6 is set to zero with respect to μ_k , giving the mean vector

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) x_n \quad (5.7)$$

We also get the covariance matrix as

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (x_n - \mu_k)(x_n - \mu_k)^T \quad (5.8)$$

and the mixing coefficients

$$\pi_k = \frac{\sum_{n=1}^N \gamma(z_{nk})}{N} \quad (5.9)$$

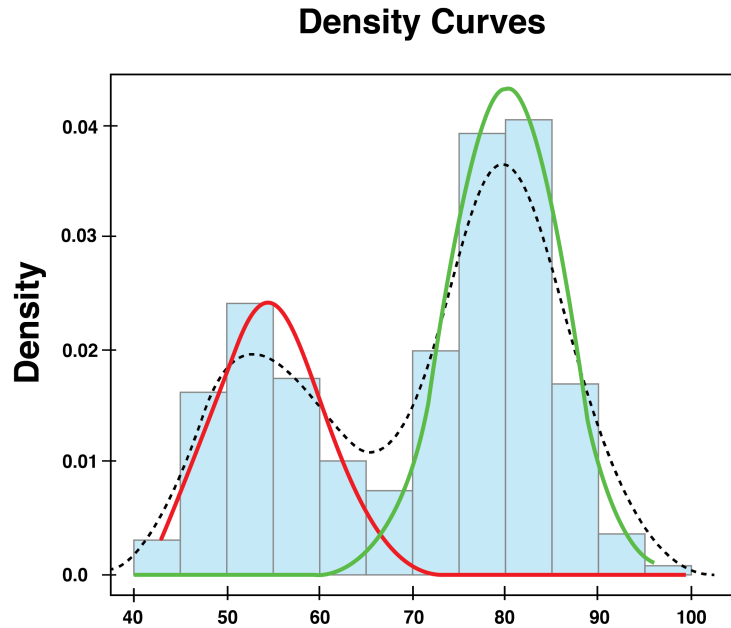


Figure 5.1: The density approximation of a two component Gaussian mixture model. The green and red curve shows the Gaussians for the two components and the dashed curve is the combination of the two into a Gaussian mixture model. The histogram is also shown as the bars.

Expectation Maximization

The expectation maximization (EM) for Gaussian mixtures is described here. We know that the EM algorithm comprises two steps, evaluating some measure with respect to the parameters and then the re-estimation of the parameters using the measure calculated in the first step. The goal is to maximize a likelihood function. In the case of the Gaussian Mixtures, the parameters that will be re-estimated are the means, covariances and the mixing coefficients for the components.

The steps involved in the EM algorithm for Gaussian mixtures

1. Find the initial values for the means μ_k , covariances Σ_k and mixing coefficients π_k .
2. Calculate the initial value of the log likelihood as defined in 5.6.
3. Using the parameter values calculated in Step 1, and the equation for the responsibilities, 5.4, evaluate the contribution of each data point to the components. This is the E-step.

4. The responsibilities calculated in the previous step, is now used to re-estimate the parameters.
5. Again, we calculate the log likelihood from 5.6, using the new parameters.
6. The last step is to test for convergence. We can do this by either testing for no change in the parameters, or the value of the log likelihood. If convergence has not been reached, we repeat from Step 3.

The maximum likelihood for the Gaussian mixture model can result in singularities, ie. the likelihood can become infinite. In terms of the mixture model, a singularity means that one of the mixture components' mean is equal to a data point in the mixture component, $\mu_j = x_n$, and the variance is zero. The contribution of this data point to the log likelihood function goes to infinity. This results in the likelihood being infinity, clearly incompatible with maximizing the likelihood. This problem does not occur in the case where a Bayesian approach was used [10]. Alternatively, to avoid singularities, it is necessary to change the mean to a random value and the covariance to a large value in the case where the mean is equal to a data point in the mixture component. Optimization can then be run to completion.

5.2 Building the HMM software

5.2.1 Sequential data

If one is dealing with independent and identically distributed (i.i.d.) data, the likelihood function can be expressed as the product over all the data points of the probability distribution. Figure 5.2 shows a graph without links, representing the case of modelling a sequence as independent observations [10]. If however, the data is not i.i.d., the joint probability density function no longer factorizes over the different observations. Other than time series, the sequence of characters in a sentence are also sequential data.

In sequential data, such as that describing the shark fin edge, the present observation depends on past observations and is not i.i.d. These dependencies need to be built into the probabilistic model. For our problem, a left-to-right, first-order Hidden Markov model is appropriate.

Recognizing the correlation between consecutive observations, allows us to create a probabilistic model, described by the joint distribution for a sequence of N observations as

$$p(x_1, \dots, x_N) = \prod_{n=1}^N p(x_n | x_1, \dots, x_{n-1}) \quad (5.10)$$

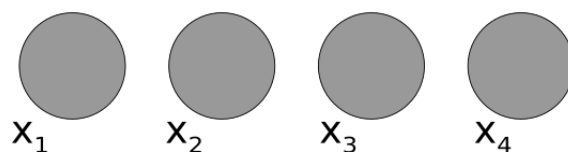


Figure 5.2: A graph without links, representing sequential data where the observations are treated as independent and identically distributed.

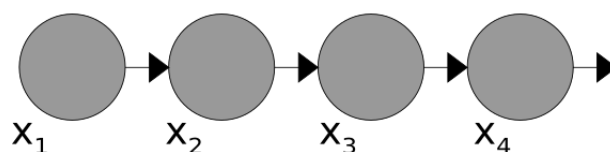


Figure 5.3: A first order Markov chain of observations.

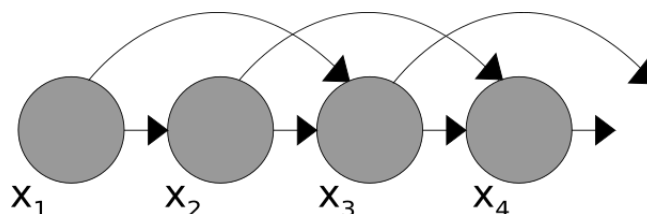


Figure 5.4: A second order Markov chain.

5.2.2 Markov models

It is expected that the most recent observations will give more relevant information about possible future values and that more historical values will be less relevant. However, if future values were to depend on all previous observations, the model's complexity would grow as more observations are made. Thus, we make the assumption that future values only depend on most recent observations. This assumption leads us to Markov models, and in Figure 5.3 the first order Markov chain is shown, where the distribution of an observation is dependent only on the previous observation [10].

The joint distribution for a sequence of observations is expressed using the product rule. For a first order Markov chain, the joint probability distribution factorizes as,

$$p(x_1, \dots, x_N) = p(x_1) \prod_{n=2}^N p(x_n | x_{n-1}) \quad (5.11)$$

where x_n is the current observation and N is the number of observations. This first order Markov model is restrictive and we expect that additional important information may be contained in earlier observations. If earlier observations

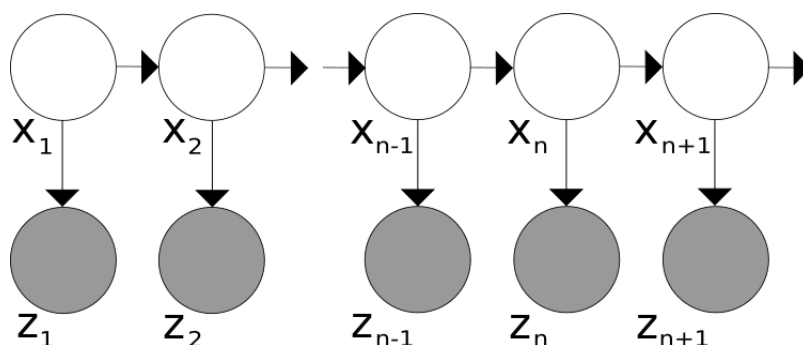


Figure 5.5: The representation of sequential data where latent variables form the Markov chain. Each observation is determined by the state of the corresponding latent variable.

directly influence the present observation, then higher order Markov models may be appropriate. The second order Markov chain is shown in Figure 5.4 and the joint distribution is given by

$$p(x_1, \dots, x_N) = p(x_1) p(x_2|x_1) \prod_{n=3}^N p(x_n|x_{n-1}, x_{n-2}) \quad (5.12)$$

Increasing the order of the Markov model, will result in a larger number of parameters in the model, thus the complexity has now also increased. Making the order of the Markov model too high, will result in an impractical model [10].

5.2.3 Hidden Markov models (HMM)

In many applications, the state described by a random variable is not directly observable, ie it is a latent variable. It is however indirectly observed. If there is a dependency between the latent variable and the observed variable, one can infer information about the latent variable from the observation. In HMMs, the latent variables are discrete. Latent variables are simple components used to construct more complex models in which the latent variables form the Markov chain. For hidden Markov models, the latent variables are discrete and the model is described by directed graphs. The observed variables can be either discrete or continuous [26].

Each observation x_n has a discrete latent variable z_n associated with it. The latent variables form the Markov chain. The probability distribution of z_n is dependent on the state of the previous variable z_{n-1} according to $p(z_n|z_{n-1})$. The joint distribution factorizes as

$$p(x_1, \dots, x_N, z_1, \dots, z_N) = p(z_1) \left[\prod_{n=2}^N p(z_n | z_{n-1}) \right] \prod_{n=1}^N p(x_n | z_n) \quad (5.13)$$

Thus, for a HMM both the transition probabilities, $p(z_n | z_{n-1})$, as well as the observation probabilities, $p(x_n | z_n)$, has to be known or learned from the data.

5.2.4 The parameters

States

The hidden Markov model is in 1 of S number of states at a given time, ie. z can take on 1 of S discrete values, however, we do not know in which state. This is why this type of Markov models are called hidden. Each individual state can be modelled as a naive Bayes or i.i.d. model, with a probability density function to describe its observations. The HMM can thus be seen as a series of naive Bayes models with probabilistic transitions between them [19]. The transition probabilities are given by $a_{i,j} = P(s_{t+1} = j | s_t = i)$, where s_t is used to denote a state at time t , and it describes the probability of transitioning to state j , given that the current state is i . The conditions $0 \leq A_{i,j} \leq 1$ and $\sum_j A_{i,j} = 1$ hold. Looking at the transition matrix, the rows represent the current state and the columns the state at the next time step, ie. the destination state. The transition matrix is stochastic, meaning the rows sum to one. We want to impose a left-to-right model, and this is done by setting the elements, $a_{i,j}$, of the matrix to zero if $s < j$ [29].

Non emitting states

We also define two additional states without any density functions, an initial state and a terminating state. These states are called non emitting states. The initial state, or s_0 has no links entering it, ie. no transition takes place to this state, and the final state, s_{N+1} , has no links leaving it. We always start in state s_0 and we always terminate in state s_{N+1} . We need to know, or learn from the data, the transition probabilities from state s_0 to states (s_1, \dots, s_N) and the transition probabilities from states (s_1, \dots, s_N) to state s_{N+1} .

Emission probabilities

The conditional distributions of the observed variables are defined by $p(x_n | z_n, \phi)$, where ϕ is a set of parameters that describe the distribution, also known as emission probabilities. For the case where the observations in x are continuous

variables, the set of parameters can be given by mixtures of Gaussians with the conditional distribution in the form

$$p(x|z) = \prod_{k=1}^K N(x|\mu_k, \Sigma_k)^{z_k} \quad (5.14)$$

If the elements of x are discrete, the set of parameters describing the distribution, is given by conditional probability tables [10].

5.2.5 The Viterbi algorithm

We want to find the most probable sequence of hidden states for an observation sequence. We could try to consider all the possible paths and choose the one with the highest probability, but this would be computationally expensive, as the number of possible paths to consider grows exponentially with the length of the observation sequence. The Viterbi algorithm searches more efficiently, so that the computation cost only grows linearly with sequence length. The Viterbi algorithm is based on dynamic programming methods.

Each path's probability is calculated by summing up the products of the transition and emission probabilities continuously. At any given time step t , there are a number of paths leading up to the current node, but we only need to keep track of the path with the highest probability. When the final time step T , is reached, we can backtrack from the current state to find the complete path with the highest probability [10].

It is important to note that this most probable sequence of hidden states are not the same as the set of states that are individually the most probable. We can find the latter by using the forward-backward algorithm to find the latent variable marginals $\gamma(z_t)$ and maximizing them individually [29].

Calculation

First, we need to define the best score along a single path, which is the highest probability up to time t ,

$$\delta_t(i) = \max_{s_1, s_2, \dots, s_{t-1}} P(s_1, s_2, \dots, s_t = i, x_1, x_2, \dots, x_t | \theta) \quad (5.15)$$

Induction gives us

$$\delta_{t+1}(j) = (\max_i \delta_t(i) a_{ij}) \cdot b_j(x_{t+1}) \quad (5.16)$$

It is for the maximization of 5.16, that we need to keep track of an argument. We create a new array, $\psi_t(j)$ to store this argument. The Viterbi

algorithm has four steps, initialization, recursion, then termination and finally the backtracking step in order to find the optimal state sequence. For $1 \leq i \leq S$, the initialization step involves

$$\delta_1(i) = \pi_i b_i(x_1) \quad (5.17)$$

where $b_i(x_1)$ is the observation probability at state S_i for the observation element at time t . Initialize the new array, $\psi_1(i) = 0$. In the recursion step, we find the maximum value as follows

$$\delta_t(j) = \max_{1 \leq i \leq N} (\delta_{t-1}(i) a_{ij}) b_j(x_t) \quad (5.18)$$

where t starts at time step 2. The same applies to

$$\psi_t(j) = \operatorname{argmax}_{1 \leq i \leq N} (\delta_{t-1}(i) a_{ij}) \quad (5.19)$$

Then we reach the termination step at time T , the likelihood is given by

$$P^* = \max_{1 \leq i \leq N} (\delta_T(i)) \quad (5.20)$$

and the final state is found as the last index

$$s_T^* = \operatorname{argmax}_{1 \leq i \leq N} (\delta_T(i)) \quad (5.21)$$

After the termination step, we can find the most probable state sequence (path) by backtracking, using the array we created to keep track of the path

$$s_t^* = \psi_{t+1}(s_{t+1}^*) \quad (5.22)$$

where $t = T - 1, T - 2, \dots, 1$ [26].

5.2.6 Training

In order to have a fully defined HMM, we first need to learn the transition probabilities and the observation probability density functions.

Given an observed data set $X = \{x_1, \dots, x_N\}$, and the set of parameters, $\theta = \{\pi, A, \phi\}$ governing the model, the optimal sequence of hidden states can be estimated and the likelihood of this sequence to fit the model can be calculated. In order to estimate the set of parameters, given that we have the observed data set X , we use the expectation maximization algorithm to maximize the likelihood function. As this estimate only converges to a locally optimal set of parameters, it is not necessarily the best possible solution [26].

5.2.7 The expectation maximization (EM) algorithm

If the state sequences were known, it would have been possible to calculate the model parameters, and if the model parameters were known, we could have calculated the optimal state sequence. Both of these are unknown and that is why we resort to the EM algorithm.

Initialization

In the first, or initialization step, we assign a state sequence to the observation sequence. Clustering provides good initial labels for a fully connected model. If the left-to-right model is used, the observation, or training sequence can be divided into K equal portions. From here we simply count the occurrences of state transitions in the following manner

$$a_{i,j} = \frac{\text{expected number of transitions from state } S_i \text{ to state } S_j}{\text{expected number of transitions from state } S_i} \quad (5.23)$$

The probability density function for each state is calculated by grouping all the observation elements associated with each state i together and calculating the mean vector and variance matrix of a Gaussian mixture model. The initial state probabilities are estimated as described in Section 5.2.4.

Re-estimation

The re-estimation part consists of two steps, the expectation step and the maximization step. For the set of parameters estimated in the initialization step, we apply the Viterbi algorithm on the training sequence to calculate the log likelihood, $p(X, S^*)$ where S^* is the expected state sequence. We keep track of this likelihood in order to check for convergence later. The state sequence is now used to re-estimate the set of parameters in the same way as described in the initialization step above.

Termination

The re-estimation step is repeated until the score obtained from the expectation (E) step is within the specified tolerance. Here the process is terminated [19].

Chapter 6

Training and Testing

In this chapter we will look at the different steps followed in the experimentation of this project. First is the database phase: gathering of the data, sorting the data (exclusion of certain photographs), extraction of the edges and normalization and finally storing all the edges belonging to the same shark in one array. Next is the training phase: dividing the array into segments of equal size according to the number of states, Gaussian mixture models for advanced clustering of the different states in order to find the emission probabilities, and using the Hidden markov model to train a model for each shark. The testing phase includes using the Viterbi algorithm on the new “unknown” fin edge to find the best sequence of states as well as the likelihood of this fin edge of belonging to each trained model in the database. Finally in the evaluation phase we use the confusion matrix to visually assess our results.

6.1 Database

6.1.1 Gathering of the data

Dr. Andreotti, postdoctoral researcher at the Department of Botany and Zoology at the University of Stellenbosch, has gathered photographs of shark fins over a few hundred days, in the area of Dyer Island Nature Reserve, at Klein Bay (Gansbay), South Africa. The sharks were attracted to the boat and then lured to the surface in order to take high-definition photographs of good quality [5]. Dr Andreotti set up the database for manual photo identification, containing more than four thousand photographs. Certain criteria were adhered to in the decision of which photographs to include in the reference database. One criterion is that the complete fin should be exposed on the photograph. Also, as the dorsal fin of the shark is very flexible, only photographs of fins in which

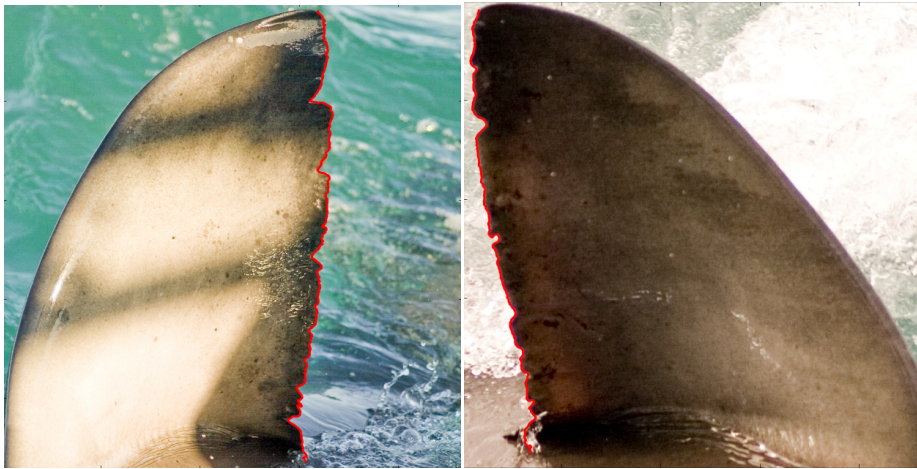


Figure 6.1: Examples of how a shark's fin bend while swimming, leading to different extracted edge arrays as for straight fins.

the fins are not bent and the fins are perpendicular to the surface of the sea as well as the camera lens were included [5]. All of these criteria will be adhered to in the setting up of our database for training and testing. The categorization of the sharks included information about resighting of the sharks on different days.

Database of photographs

The database used for the testing of the Hidden Markov models consists of a total of 509 photographs. The number of different sharks and therefore the number of models trained, are 117. For most of the sharks, 48 of them, there are only two different photographs. There are 34 sharks with three photographs in the database, and progressively less sharks with each increasing count of number of photographs. Only one shark in our database has eight photographs. Including more photographs does not necessarily lead to a better model, but including better quality and carefully curated photographs will lead to the best model possible.

Figure 6.2 shows the plots of the extracted edges, after normalization, for two different sharks. In the top image, the fin paths of four different photographs of the same shark are closely related and fit almost exactly on top of each other. The image in the bottom however, shows how much one photograph can differ from the other two, resulting in a very different plot. This is usually the case where a photograph of a bent fin, as shown in Figure 6.1 was included in the database, together with the photographs of straight fins of the same shark.

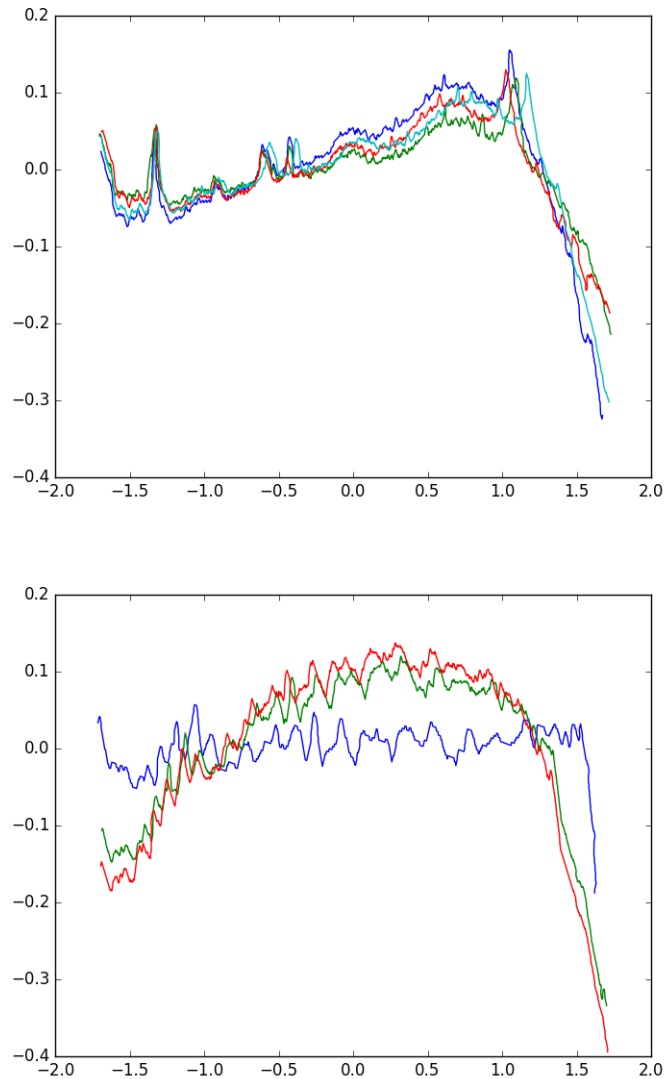


Figure 6.2: Plotting the different extracted fin paths for two different sharks. In the top image, the paths extracted from the four different photographs of a shark, all follow the same pattern. The bottom image shows the paths extracted from three different photographs of the same fin. Here, however, the arrays does not fit on top of each other.

6.1.2 Sorting the data

In Section 3.1.1, we discussed the type of photographs that will be excluded from our database because of the quality of the photographs or the photographic angle, as well as where occlusion of the sharkfin or reflection on the sharkfin occurs. In this section of the chapter, we will look at difficulties encountered pertaining to the sharkfins itself. This includes how a sharkfin may change with time, the bend or orientation of the sharkfin in the photograph, as well as the type of features in the trailing edge of the fin.

Changes over years

A problem in the training data is the fact that a shark's dorsal fin from two years ago won't necessarily look the same as the same shark's dorsal fin in the present, especially if the shark is still young. In a shark's lifetime, he will be involved in fights with other sharks and get injured, this might cause injuries to the shark's dorsal fin, changing the nicks and notches in the fin, as shown in Figure 6.3. In the case where a shark's fin has changed over the years, the software might not be able to make a correct match, because it will still be trained on the anatomy of the earlier shark fin. In the case of Figure 6.3, the trained model using both these photographs might still be sufficient, because the fin up to the new notch looks very similar in both photographs. But it might be a good idea to only include the most recent photographs of a certain shark, or to visually confirm that there wasn't any major changes in the shark's fin, before including all the available photographs. However, as can clearly be seen in the photographs at the top in Figure 6.4, the fins almost look completely different and all the notches are more pronounced in the second photograph. In this case, our software might not be able to make a successful match.

Orientation

Another problem that is encountered with the data, is the orientation or bend of the shark fin in the photograph. The two photographs at the top of Figure 6.4 show the same shark. The images at the bottom of the figure show the plots of the normalized extracted edges of both of these shark fins. Looking at these two plots, they do not look at all related. Normalizing the extracted edges, further distorts the paths that are plotted. The bend of the sharkfin in the second photograph results in a convex plot, whereas the fin in the first photograph has a concave plot. The first photograph shows the normal orientation or bend of the sharkfin and is preferred to the second photograph. We can see, however, that there are two photographs that lead to this graph as seen in the second photograph. This means that we can easily train both these types and have

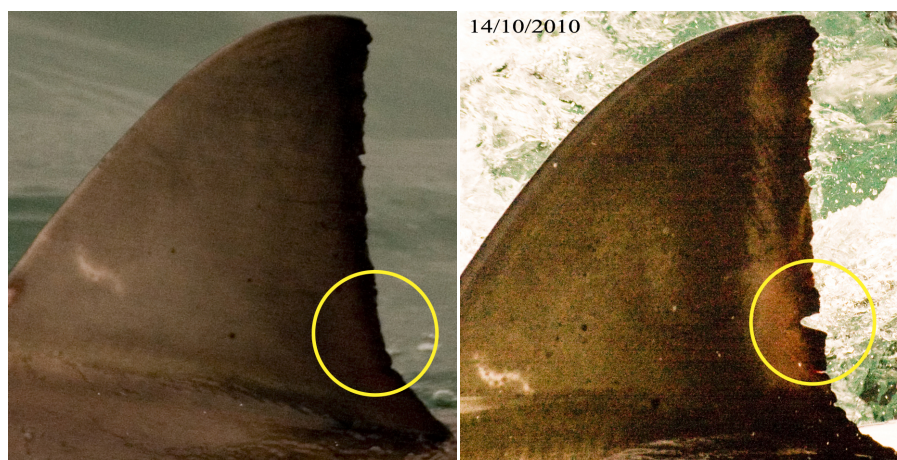


Figure 6.3: The photographs above show how a shark's fin can change over the years. The photograph on the left was taken in June 2009 and the photograph on the right was taken in October 2010. It can clearly be seen in the photograph on the right that a big notch appeared where previously there wasn't one. In this case the software might still be able to match the photograph to the right model, because the rest of the fin is still closely related.

two models for a shark, to increase our chances of matching this shark. It is preferable to the method of including all the photographs, even when they differ, or simply excluding those that differ. It was however stated at the beginning of this chapter that photographs in which the sharkfin is bent or is not 90 degrees to the water and the camera, will be excluded from the database.

Type of features

The more pronounced the notches in the trailing edge of the fin, the better the chances of a good match. Looking at the four photographs in Figure 6.5, it is immediately clear that the photographs of the two fins at the bottom are not from the same shark. It is not so clear at first glance and without further inspection that the two photographs at the top does not belong to the same shark. The software will also find it easier to distinguish or match a shark fin in the case where it has more pronounced features, such as deeper or more noticeable and unique notches. The trailing edges of the two fins in the photographs at the top of Figure 6.5 is almost completely smooth and will be difficult to match correctly with a small rank.

6.1.3 Extraction of fin edges and normalization

The method for extraction was described in 3.2.2. The same approach as used in the DTW algorithm program is used here, where the user selects the start and

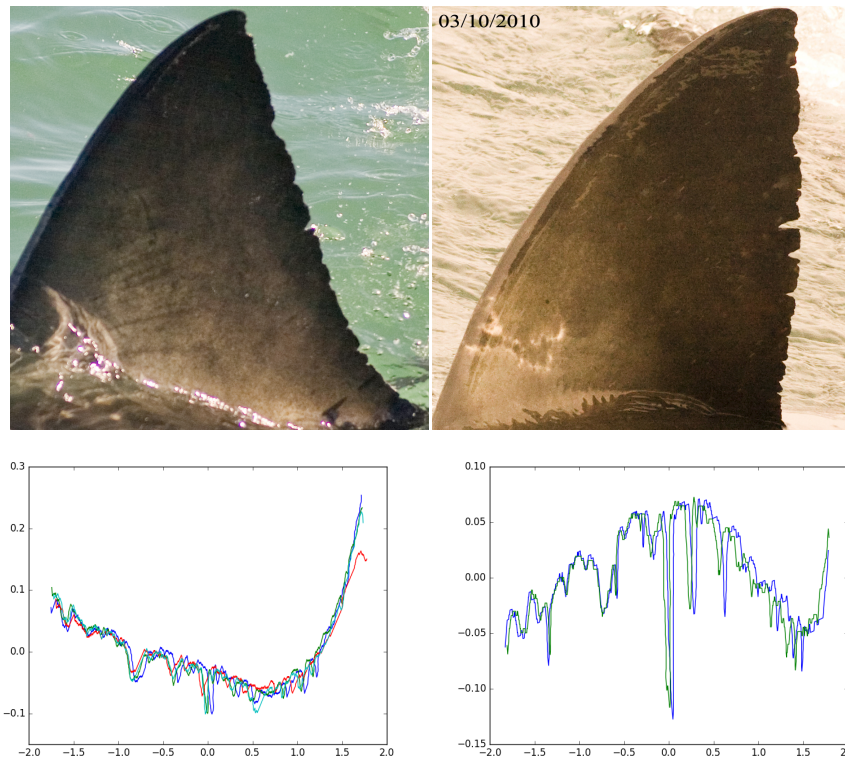


Figure 6.4: The sharkfins in the photographs above almost seem to belong to two different sharks. The photograph on the left was taken in May 2009 and the photograph on the right was taken in October 2010. In the photograph on the right it can clearly be seen that the notches are more pronounced. The orientation of the sharkfin in the photographs are also very different leading to very different plots of the path of the trailing edge of the dorsal fin. Training on both of these fins, will lead to an inconsistent model and trying to correctly match one of these to such a model will be very difficult to nearly impossible.

end point of the fin. The scikit algorithm, `route_through_array`, then finds the path from the start to the end point describing the trailing edge of the shark's dorsal fin.

Figure 6.6 shows how the function `route_through_array` fails on blurred images. The blurring causes the fin to fade into the background and the edge between fin and background is not clear enough for the edge extraction. It was previously stated that blurred photographs should not be included in the database and here we find the proof to sustain that statement.

The photographs of fins in Figure 6.7 and Figure 6.8 show the most common extraction mistakes made by the function `route_through_array`. In all of these cases the bottom part of the fin is covered by water or water splashes across the bottom part of the fin. The edge is not entirely visible in these areas and



Figure 6.5: The type of features that are very distinguishable (at the bottom) will result in a better match than less pronounced features (at the top).



Figure 6.6: How the fin extraction fail on blurred images.

the extraction function defines either a reflection of the fin on the water, or the water droplets as part of the fin. One option is to mark the end of the fin just above the water line or water droplets, but this will have the result of fins of different lengths and is not desirable for good matching. It is better to have the same length of fins extracted and rather only use up until a certain point of these extracted edges, like only the top two thirds. It is therefore advisable to mark the end of the fin on the photograph where it is supposed to be, even though it is not visible. Thereby, if only a part of the fin is used for matching, it will be the same amount of fin that is matched. It is clear from Figure 6.7 and Figure 6.8, that at least the top two thirds of the fins are accurately defined by the extraction software.

In conclusion, the function `route_through_array` is able to successfully extract the fin edges in most cases, even with poorer quality photographs or photographs with considerable detail in the background. The extraction function does however fail when applied to a blurred image in the sense that it does not clearly define the notches in its entirety. These are the photographs that will not be included in the database. To account for the cases where part of the fin is occluded by water, we can use only part of the fin, most likely the first two thirds of the fin, to do the training and the matching.

Each individual extracted edge is then normalized as discussed in Section 3.3, with respect to position, scale and in-plane rotation. Mean normalization is first applied by subtracting the mean from the edge array, thereby normalizing with respect to position. Using Singular Value Decomposition, three simple transformations is applied: the initial rotation, the scaling of the coordinate axes, and the final rotation. The edge array is now also normalized with respect to scale and in-plane rotation.

After normalization, all the edges extracted from photographs that are included in the database for an individual shark, are added to the same array. This is because we train one model per shark, not per photograph. The data is now ready to start training.

6.2 Training

In training, we will learn a model for every individual shark by using all the photographs available for that shark, but excluding the 50 photographs that will be used for testing. Learning a model means that we learn the transition probabilities and the observation probability density functions, as explained in Chapter 5. The model always starts in the initial state and ends in the terminating state. We need to determine with which probabilities the model will move between the states in between the initial and terminating states.



Figure 6.7: Examples of the most common failures or problems encountered with the extraction of the edge. The user will mark the top of the fin as the start point and even though the lower part of these fins are underwater or occluded by splashing water, the user will mark where the end of the trailing edge is or is supposed to be, so as to define the entire trailing edge, albeit the lower part will not be accurately defined. The software for matching can then be written so as to only include the top two thirds of the fin for matching. This way, we are certain that the part of the fin that we use for matching is accurately defined in its entirety.

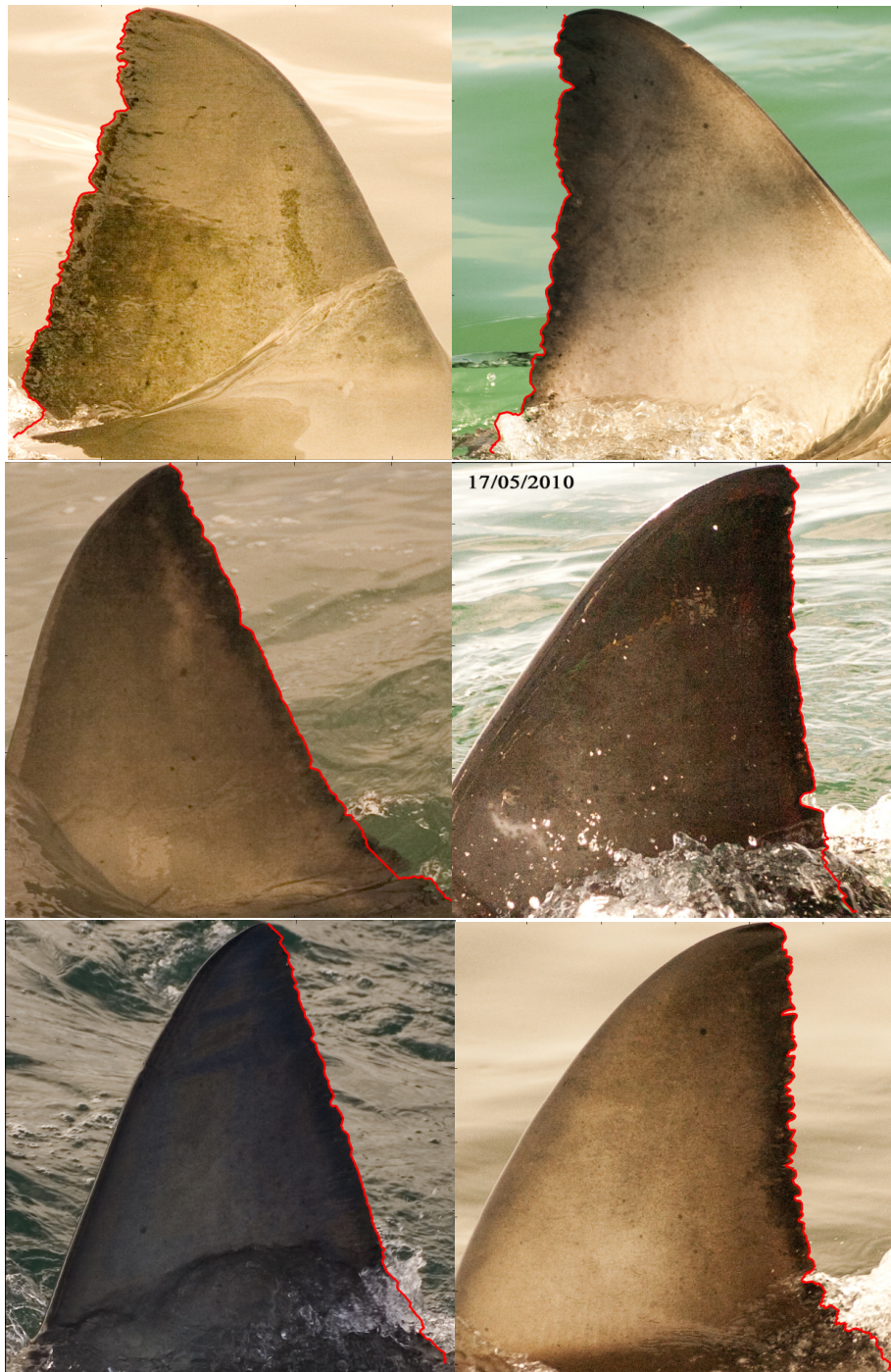


Figure 6.8: Examples of the fin extraction function defining the fin edge on the water is shown here. It is important to note that up until the lower part of the fin where the water occludes the fin, the fin extraction function is very successful in defining the edge accurately.



Figure 6.9: Here examples are shown of successful extracted edges. Most of these photographs are ideal in the sense that there is a stark contrast between the shark fin and the water in the background. All of these fins, however, have deep notches and the fin extraction function still manages to define these successfully.

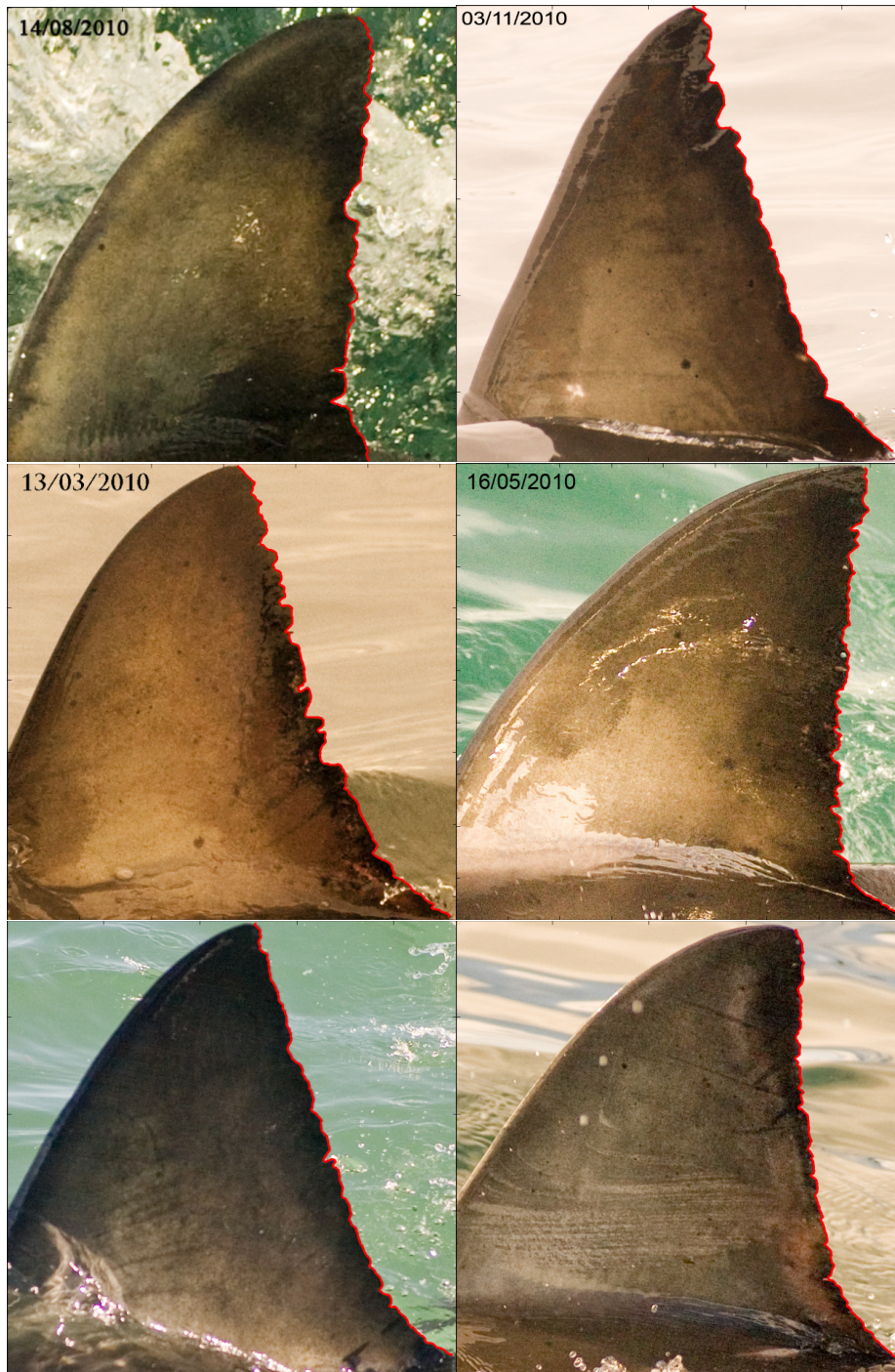


Figure 6.10: More examples of successful extractions are shown here. In most of these photographs, there are considerable detail in the background water, such as reflections or splashing water. The fin extraction function defines the fin edges successfully even in these cases.

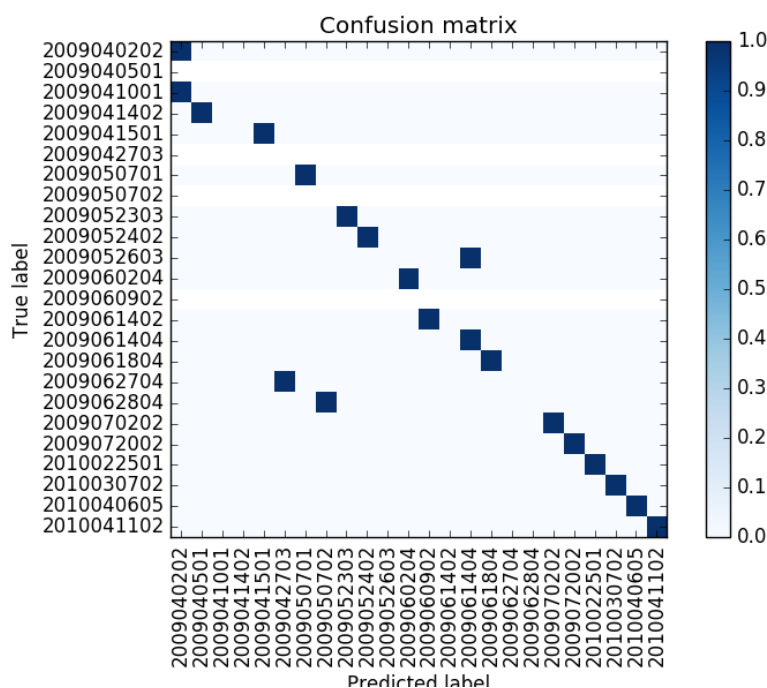


Figure 6.11: The confusion matrix for testing done on a small dataset consisting of 119 trained fin edges and trying to match 20 “unknown” photographs.

6.2.1 GMM

First we need to define the different states for every individual model and decide on the number of states that will be used. Optimization of the algorithm in terms of the number of states will be done later by trial and error. We divide the edge array into the chosen number of states simply by dividing the edge array into equal parts. Next we use the Gaussian mixture models to optimally do a soft assignment of each element of the edge array to a state.

6.2.2 Training using the HMM

These soft assignments of Gaussian mixtures are our emission probabilities. To find the transition probabilities, we simply count the occurrences of state transitions, as given in 5.23 in Section 5.2.7. Section 5.2.7 also describes the EM-algorithm, which is used to find the above mentioned parameters at convergence. Diagonal covariances are used.

6.3 Testing

When a new photograph of an unknown shark is captured and we need to identify the individual shark, we compare it against the photographs in our database. In terms of our program, we will compare the extracted and normalized edge of the fin in the new photograph to the trained models in our database and use the Viterbi algorithm to find the most probable state sequence as well as the likelihood that this is the correct sequence of states.

6.3.1 Control

We perform a control test in order to ensure that the algorithm is doing what we expect it to.

We do training by including all the photographs and training a model for each individual shark instead of for every photograph. Here we will also include the 50 random photographs that we want to match. This control test will tell us how good the data is for the testing. Again we hope for a 100% match. This is however not the case. Only 45 of the 50 photographs are matched to the correct model, resulting in a 90% match. Looking at the photographs that is not matched to the correct model, we find that the problem shown in Figure 6.4 and described in Section 6.1.2 is the reason for the mismatches. This confirms our hypothesis that including photographs of bent fins in a model where photographs of straight fins are also included, will confuse the software.

6.3.2 Random photographs

When we do the matching of a photograph in practise, the new photograph would not have been seen by the database. Thus we need to remove the photographs that we want to match from the database before we do the training. Fifty photographs are randomly chosen and removed from the database and then the remaining photographs are used to build a model for each shark.

6.3.3 Evaluating measures

To estimate the quality of the predictions of the model, quantifying measures are needed. Scikit Learn [24] provides a metrics module with the functions that we will use. Since a model is trained for each class, ignoring the properties of other classes, we are using a generative approach [19]. We will use the confusion matrix as a way to evaluate the models.

In a confusion matrix, the rows represent the predicted classes and the columns represent the true or actual classes. The confusion matrix shows where the classification system mislabeled or misclassified classes. The diagonal of the

table is the correct labeling, outside of the diagonal lies the errors [33]. The diagonal indicates the correct predictions. The entries in the confusion matrix are converted to probabilities.

6.3.4 Optimizing the algorithm

We find the optimal number of states by trial and error, we test repeatedly on fewer photographs and try to get the matching as close to 100% as possible. Starting with 58 photographs of 19 sharks, using 13 states and matching 10 random photographs, a 90% match is made with a rank of one. Next we increase the number of sharks to 38 by adding more photographs, up to a total of 119 photographs. We progressively increase the number of states until the matching is at its best and a further increase in the number of states leads to a decrease in the number of fins successfully matched with a rank of one. The optimal number of states is found to be 12 and matching 20 fins to the database, results in fourteen out of the 20 randomly chosen photographs to be matched with a rank of one. The confusion matrix for this test is shown in Figure 6.11. Nineteen out of the 20 photographs are correctly matched with a rank of 20 or less. These tests were done on a smaller database in order to be able to repeat these tests a few times for every different value chosen for the number of states. The tests were then repeated once for every different value using the complete database to confirm the results achieved in the smaller database tests.

6.3.5 Summary

In summary, the following points are repeated:

- The HMM can potentially improve on the results seen with DTW because the training of the fin edges includes all variations of the extracted edge.
- The disadvantage is that more training data is required, which might not always be available, as with our case.
- The case of using only one photo was also investigated and the results compared with DTW's results.

6.4 Results

In our classification, false positives are more acceptable than false negatives. We would rather include samples that were assigned the wrong label, than not including the right sample. Because the output would contain more than just the most likely match, we return the top five most probable matches and it is

Rank	1	2 or less	5 or less	20 or less	more than 20
1	0.42	0.56	0.64	0.84	0.16
2	0.44	0.54	0.60	0.88	0.12
3	0.34	0.42	0.62	0.80	0.20
4	0.42	0.58	0.64	0.76	0.24
5	0.42	0.54	0.58	0.84	0.16
Average	0.41	0.53	0.62	0.82	0.18

Table 6.1: Results from five different tests using Hidden Markov models.

possible to choose the true match manually. This ensures that the correct match was made, without having to inspect all the photographs in the database.

6.4.1 HMM

Perfect match

The software was tested by choosing 50 random photographs and comparing these to the trained models. The total number of photographs used was 509. A total of 56% of the random photographs were correctly matched with a rank of two or less. Including a rank of up to five, 64% of the random photographs were correctly matched. The HMM could match 84% of the photographs with a rank of twenty or less. Thus only 16% of the photographs were matched with a rank of more than twenty.

Five different tests were done, each time 50 different photographs were randomly chosen to exclude from the database and use to test the software. The results from these five tests are shown in Table 6.1.

6.4.2 Testing of DARWIN

A study was done in 2006-2008 in order to estimate the white shark population at sites off Central California [12]. Manual photo identification was done on 321 photographs with sufficient quality and 130 unique individuals were identified. The fin photographs were correctly matched with 98% success using manual photo identification. Their attempts to use DARWIN was unsuccessful, as it resulted in unacceptable levels of error (T. K. Chapple 2009, unpublished data) [12].

In 2014, DARWIN was again tested to validate its accuracy [5]. To do the testing, a database of 426 high quality white shark photographs of already identified sharks was used to identify 53 unique sharks from 122 additional photographs. In the cases where “Fin Trace” failed to automatically trace an accurate edge of the dorsal fin, the edges were manually adjusted in order to

improve the matching done by the software. Manual intervention in the tracing of the outline of the dorsal fin was needed for each photograph [4].

The above mentioned studies both failed to use DARWIN for successful matching and extensive user input was required for the extraction of the fin edges. Therefore, instead of repeating this experiment, the results from the second study will be presented here.

A third attempt was made to test DARWIN for use in semi-automatic identification of sharks. First to be noted is that the latest update to the DARWIN code was in 2012 and this was only for a minor change. The latest publication listed on the website for DARWIN is of an article from 2008. It is not clear if the code is actively maintained.

DARWIN's function "Fin Trace" should be able to automatically trace the outline of the fin. If it is unable, it will ask the user to manually trace the outline of the fin. The outline is then repositioned to the actual positions that describe the edge of the fin. This means that it is not necessary for the user to trace the fin perfectly, which is quite difficult using the cursor. The user has the option of adjusting certain points after this repositioning. The user also has the option of adding points in order to better describe the nicks and notches present in the fin. These points shown as dots on the image are very small and not very clear against the colours in the photograph. The colour can be changed under Settings, but not the size of the dots. The small size of the dots makes it quite difficult to see where the outline of the fin is defined and this greatly increases the possibility for human error, as shown in Figure 6.12. In the photograph at the top of Figure 6.12, the green dots inside the circle are very unclear and were missed by the user when the outline was adjusted. This resulted in the traced outline shown at the bottom of Figure 6.12, which is clearly wrong. In this case, the user should go back and correctly retrace the outline. Therefore, it is advisable that the user check the accurateness of the outline after every tracing and do further adjustments if necessary, which increases the user input in the extraction of the fin edge.

The photographs that DARWIN could not automatically trace, were photographs of bad quality, ie. the sharkfin in the photograph was blurred or the resolution was low. These fin edges were successfully traced by the `route_through_array` function used in the development of the HMM software in this thesis. The size of the dots also makes it problematic to adjust the outline of the fin edge, because it is difficult to click on such a small dot.

So now the user traced the outline of the fin manually, and the software adjusted the traced outline to the actual points describing the edge of the outline. The software now gives this outline in terms of points along the fin edge. These points describing the outline are quite a few pixels apart from one another and

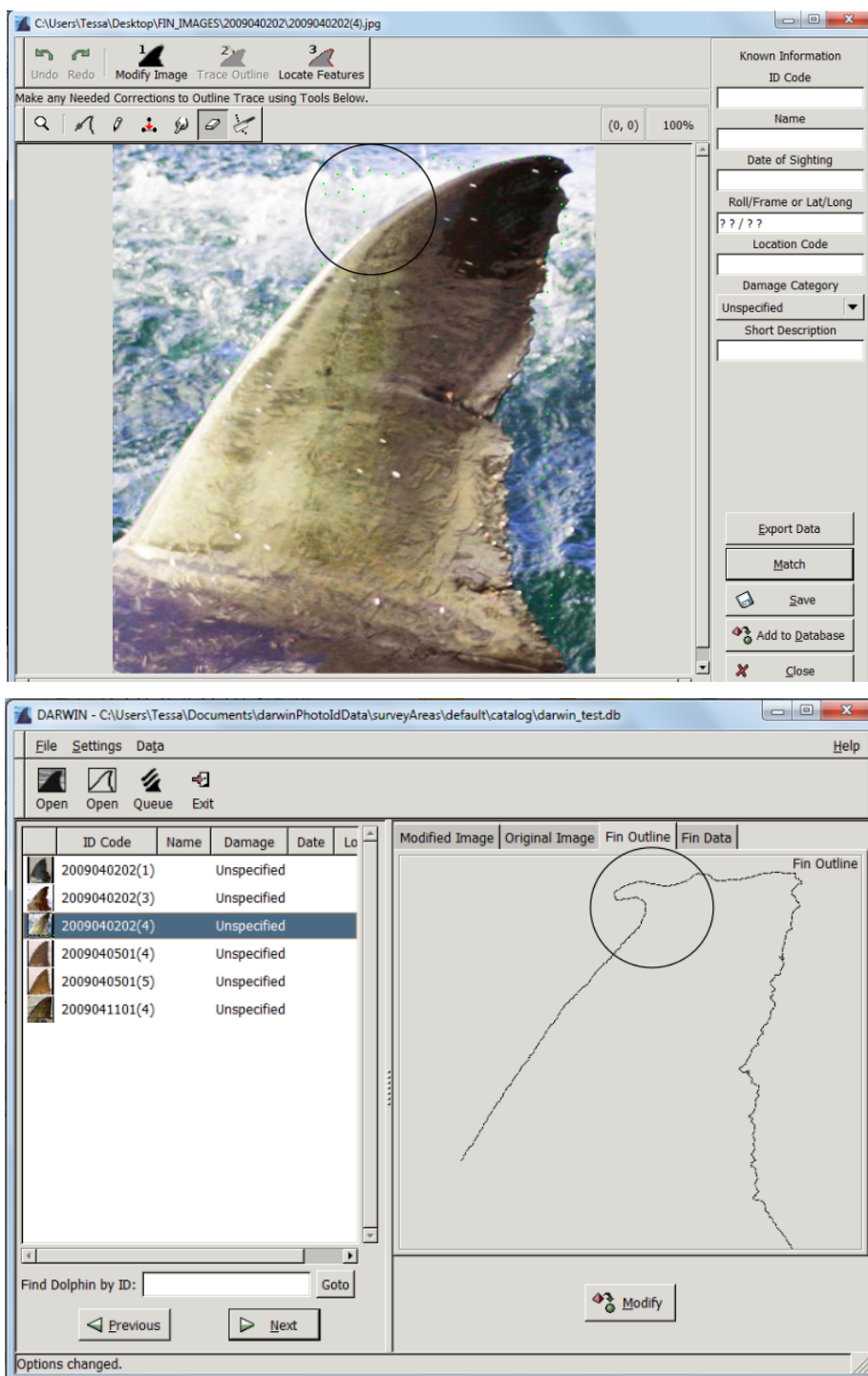


Figure 6.12: The green dots in the photograph at the top is difficult to nearly impossible to see, leading to human error in the tracing of the outline of the fin, in cases where the user is required to adjust the outline of the fin. The circle in the image at the bottom shows where the user failed to adjust the outline of the fin because the dots on the photograph were too unclear.



Figure 6.13: Photographs of bad quality that DARWIN was unable to trace. These photographs were successfully traced by `route_through_array` used in the HMM software.

even though the fin outline was traced including the notches in the fin, these points does not seem to define the notches in the fin. It is therefore also necessary for the user to add more points in this case in order to successfully define the important features in the outline of the fin.

The tracing as described above: manually tracing the outline of the fin, adjusting the traced fin outline and adding points where necessary along the outline, took about five minutes. The objective of this thesis is to make photo identification of sharks as automatic as possible. The amount of user input needed in the extraction of a fin edge using the DARWIN software, as described above, is very extensive and the DARWIN program is therefore not very promising as automatic photo identification software. The extraction of fin edges using DARWIN is a laborious, strenuous task and human error is prevalent. Using `route_through_array` for the extraction of the entire fin in all the photographs, would make it possible to test the matching part of DARWIN against the HMM software.

If finally the fin edges in the photographs are correctly traced, excluding those kept aside for testing, a new photograph of an “unknown” shark to be identified is introduced. This fin edge should now also be traced as was done for the setting up of the database as described above. In the DARWIN program, the “Match” function is now selected. The software then fits this new outline on each traced outline in the database and determines and returns the best match for this new photograph. It ranks all the outlines in the database according to rank from best match to the “unknown” fin.

In Figure 6.15 at the top, the photograph of the fin to be matched is shown to

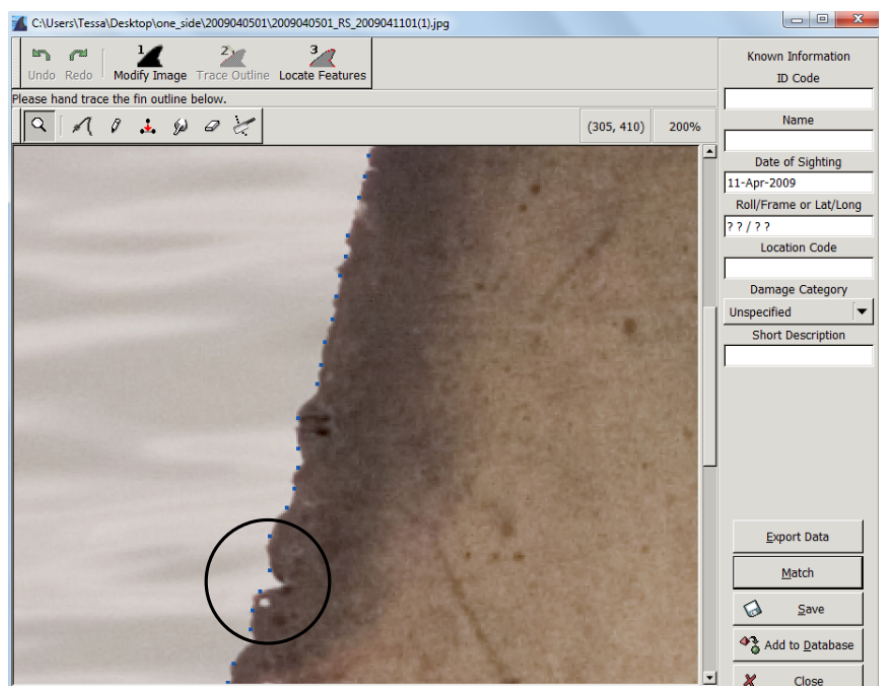


Figure 6.14: This image shows part of a fin that was traced manually, but the software returns points along the edge that does not include the notches in the fin edge, as can be seen in the part encircled on the image above.

the left and the photograph that is the best match is shown to the right. In the bottom left corner, the fin outlines in the database is listed according to rank. In the bottom right corner, the outline of the fin from the new photograph is fitted on top of the outline of the best match photograph from the database. The image at the bottom of Figure 6.15 is an expansion of this and shows the outlines of the two fins as well as the chain code representation showing the important features in both fins and how well they align. The matching done by DARWIN seems to work quite well, however, it should be pointed out that the outline produced by this specific fin is very unique and the notches is very pronounced and DARWIN might not work as well for other fins. Even if DARWIN is able to successfully match a sufficient amount of unknown fins, and we can say with confidence that DARWIN's matching capabilities are acceptable, the fact remains that extensive user input is necessary for the tracing of the fin outlines and therefore DARWIN fails miserably in the task of automating the identification of unknown sharks.

Fin trace

DARWIN allows for manual adjustment of the trace outline if the software couldn't accurately trace the outline to include the very important notches in

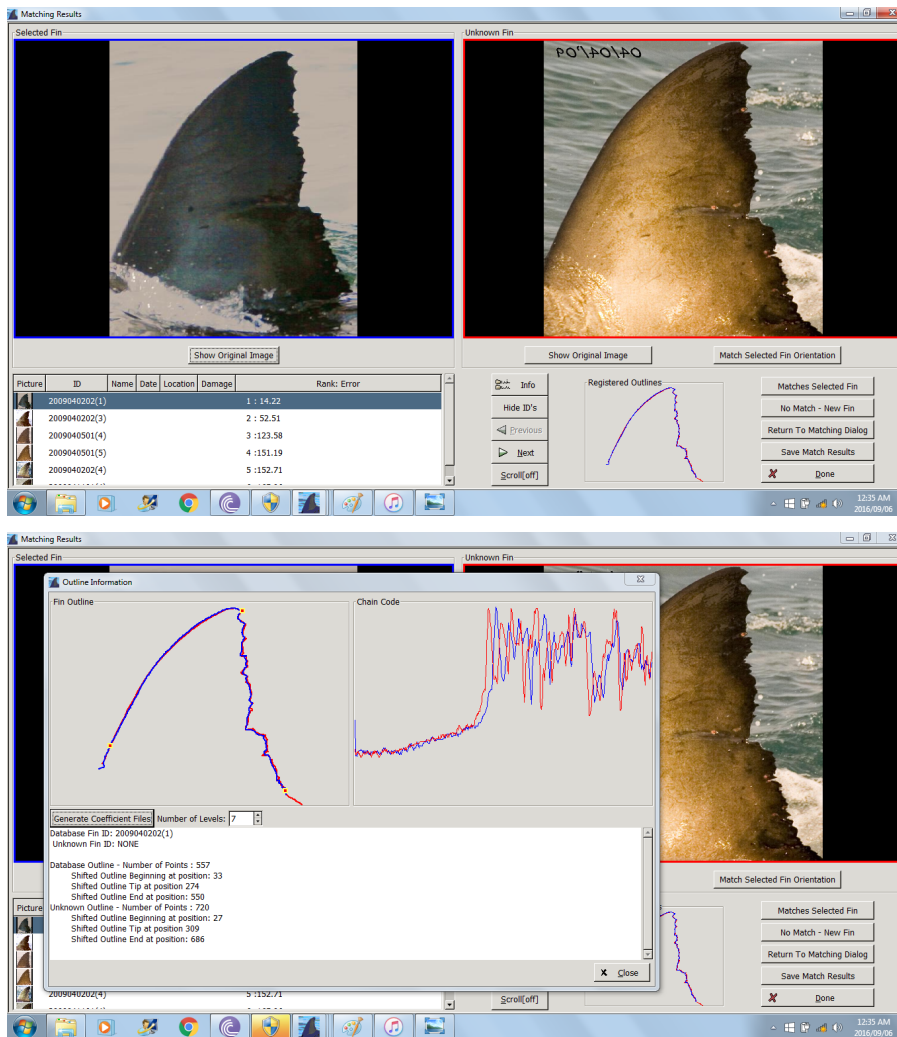


Figure 6.15: The matching of a newly introduced photograph of an “unknown” shark is shown here.

the trailing, serrated edge. A total of 426 high quality photographs were tested. Manual adjustment of the trace outline was needed in all 426 cases to accurately and optimally reflect the dorsal fin.

Control

A control set of 50 photographs was used to test if DARWIN would be able to match the exact same photograph. These 50 photographs are duplicates of photos contained in the database in DARWIN. We expect these photographs to be matched with rank 1 to their duplicates in 100% of the cases. This is not the case, however, only 86% of the control photographs were correctly matched with rank 1.

Perfect match

Fifty photographs were then randomly selected from the entire collection of photographs and the matching tested. DARWIN was only able to match 10 photographs (20%) with a rank of 1. If we accepted matches with a ranking of more than 1, up to a rank of 20, another 20% of the photographs could be matched. Thus, 60% of the photographs had a rank of more than 20.

Conclusion

The conclusion reached by Dr Andreotti, based on a poor paired t-test result, was that DARWIN is unable to successfully match the dorsal fin of a great white shark to the correct shark in the database. This is the same conclusion as reached by Chapple in 2009.

6.4.3 Results of testing DTW

Perfect match

The software was tested by randomly selecting fifty photographs to match against the 744 images in the database. The algorithm could correctly match 80% of the images within a rank of twenty. Of the total, 62% of the images were correctly matched with a rank of two or less. Only seven images couldn't be matched [3].

Conclusion

The average difference between the first rank and the match made by the software, is 66.34 (with a 95% confidence) [4].

	DARWIN	DTW	HMM
Control matched	86%	-	90%
Rank of 1	20%	-	42%
Rank of 2 or less	-	62%	56%
Rank of 20 or less	40%	80%	84%

Table 6.2: Summary of results comparing DARWIN, DTW and HMM.

For shark identification using DTW, the paired t-test result confirmed the hypothesis that software using dynamic time warping is able to successfully find the match of an unknown shark fin photograph with a rank of 20 or less.

Conclusion for testing using HMM

The HMM performs better than DTW when we include results up to a rank of twenty. Matching with a rank of one or two is less than, but still comparable with the results for DTW. It is expected that the HMM software will have improved performance in the case where the database of photographs was carefully constructed in order to only include photographs that will lead to consistently trained models.

6.5 Summary

The results for the three different methods discussed in this thesis are summarized in Table 6.2. The control test for the HMM is less than 100%, which is undesirable. A test result of 100% for the control will invoke more confidence in the database of trained models. Even though the HMM does not perform as well as DTW within a rank of two, it does show better results when matches with up to a rank of twenty are included.

Chapter 7

Conclusions and future work

7.1 Recommendations

In the testing of the HMM software, some very important conclusions were drawn about the photographs in the database. A few recommendations can be made so as to improve the matching ability of the software developed in this thesis.

Taking of photographs

If at all possible with data collection, when taking photographs of shark fins, it should be ensured that the shark fin is viewed at a ninety degrees angle and the fin is as straight as possible. It might also be a good idea to take multiple photographs of the same shark at one sighting. This might be difficult, because the shark is constantly moving, but using the burst mode on the camera might be helpful in this case.

Including photographs in training

When choosing which photographs to include in the training when creating the database of models, different photographs of the same shark should be inspected to ensure that they are comparable. An easy way to do this, would be to extract the trailing edges of the fins from the different photographs, applying normalization and plotting the results on the same plot. The edges following the same trend on the plot should be included and the rest excluded from training. Before this is done, manual inspection of the photographs would also be helpful in removing older photographs in the case where the trailing edge of the fin has changed significantly over time. Only in the case where the dorsal fin of the

shark hasn't changed much over the course of the years in which it was tracked, should all the photographs be included.

This process might be time consuming, but it would only have to be done once in the creation of a quality database, and it might only be necessary to update one fin at a later time, in the case where the trailing edge of the fin has changed.

Results returned

Returning the results of matching a shark fin, it is advisable to return the top five matches. The photograph of the unknown shark can then be visually matched to one of the five photographs with the top five ranks. This increases the chances of finding the correct match even if the software wasn't able to match the photograph to the correct model with a rank of one.

7.2 Future work

The Hidden Markov model software developed as part of this thesis is found to be successful in matching an unknown photograph to the correct model in the database of trained models. It is believed that the HMM itself is sufficient to do successful matching, but that the database of photographs is lacking in quality and consistency. Steps should be taken in order to create a database of trained models that are more suited to assist the model in finding a correct match with more accuracy. This would be the most efficient way to improve the performance of the software.

Firstly, all the available photographs should be collected. The fin edges should then be extracted and normalized. All the extracted and normalized edges belonging to an individual shark should be plotted on the same graph. Visually inspecting this graph should be able to indicate whether all or only some of the photographs for that shark should be included. In the case where the plot of an edge deviates from or differs too much from the majority of edges, it should be excluded from training.

Also, if by inspection of the photographs it becomes clear that the trailing edge of the shark fin has significantly changed over the course of it being tracked and photographed, only the most recent and relevant photographs should be included. To increase the chances of making a positive identification of an unknown shark fin, it is recommended to take multiple photographs at a sighting in order to ensure that a good quality photograph is available to do the matching with. The database of trained models should also be updated if it should be discovered that the trailing edge of a certain shark has changed since the

previous sighting.

Using this newly created database, a model for each shark can be trained and the HMM software tested in order to compare it with the results as found in this thesis. Only then can the HMM software be revised in order to optimally improve the software itself.

It is recommended to use the function `route_through_array` for the extraction of all the fins and then use these extracted edges to test DARWIN's matching algorithm part and see how it weighs up to the HMM and DTW algorithms.

In the case of HMM, it might be worthwhile to include another variation of the matching in the software. In this variation, a test can be done using only the top two thirds of the fin and combine this result with the score from matching with the entire fin.

Bibliography

- [1] Endangered species - sharks. Website: <http://earth.com/endangered-species-sharks>.
- [2] Computing dolphin fin photo ids. *American Scientist*, Volume 100, November-December 2012.
- [3] S Andreotti, P. Holtzhausen, M Rutzen, M Meyer, S. Van der Walt, B. Herbst, and CA Matthee. Semi-automated Software for Dorsal Fin Photographic Identification of Marine Species: Application to White Sharks.
- [4] S. Andreotti, M. Rutzen, S. Van der Walt, S. Von der Heyden, R. Henriques, M. Meyer, H. Oosthuizen, and CA. Matthee. An Integrated Mark-Recapture and Genetic Approach to Estimate the Population Size of White Shark in South Africa. *Marine Ecology Progress Series 552*, pages 241–253, 2016.
- [5] S Andreotti, M Rutzen, PL Wesche, CP O’Connell, M Meyer, H Oosthuizen, and CA Matthee. A novel categorisation system to organise a large photo identification database for white sharks *Carcharodon carcharias*. *African Journal of Marine Science*, 36(1):56–67.
- [6] A. Ardovini, L. Cinque, F. Della Rocca, and E. Sangineto. A Semi-automatic Approach to Photo Identification of Wild Elephants. *Lecture Notes in Computer Science*, pages 226–232, June 2007.
- [7] D. Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.
- [8] Yoshua Bengio, Renato De Mori, Giovanni Flammia, and Ralf Kompe. Neural Network - Gaussian Mixture Hybrid for Speech Recognition or Density Estimation. *Advances in Neural Information Processing Systems 4 (NIPS’91)*, pages 175–182, 1992.

- [9] Jeff A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. *International Computer Science Institute*, 4:126, 1998.
- [10] C Bishop and N Nasrabadi. *Pattern Recognition and Machine Learning*, volume 16. 2007.
- [11] T. Burghardt, B. Thomas, P.J. Barham, and J. Calic. Automated Visual Recognition of Individual African Penguins. *Fifth International Penguin Conference*, September 2004.
- [12] TK. Chapple, SJ. Jorgensen, SD. Anderson, PE. Kanive, AP. Klimley, LW. Botsford, and BA. Block. A First Estimate of White Shark, *Carcharodon Carcharias*, abundance off Central California. *Biology Letters* 7, pages 581–583, 2011.
- [13] Enrice Gennari W. Hermann Oosthuizen Deon Kotze Mike Meyer David W. Sims Catherine S. Jones Leslie Robert Noble Chrysoula Gubili, Ryan Johnson. Concordance of genetic and fin photo identification in the great white shark, *carcharodon carcharias*, off mossel bay, south africa. *Springer-Verlag*, June 2009.
- [14] K R Debure and a S Russell. Feature extraction for content-based image retrieval in DARWIN (digital analysis and recognition of whale images on a network). *Proceedings of First ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 470 –, 2001.
- [15] Bengt Fornberg and Ben Herbst. Modeling in applied mathematics. *University of Stellenbosch , Department of Applied Mathematics*, 2000.
- [16] GenTXWarper. Dtw algorithm, 2015. [Online; accessed 02-December-2015].
- [17] DARWIN Research Group. Darwin, 2016. [Online; accessed 20-August-2016].
- [18] Scott a. Hale. Unsupervised threshold for automatic extraction of dolphin dorsal fin outlines from digital photographs in DARWIN (Digital Analysis and Recognition of Whale Images on a Network). *eprint arXiv:1202.4107*, pages 3–7, 2012.
- [19] B. Herbst, J. Du Preez, and S. Kroon. Machine Learning. pages 92–128.
- [20] William Y Huang and Richard P Lippmann. HMM Speech Recognition with Neural Net Discrimination. *Advances in Neural Information Processing Systems* 2, pages 194–202, 1990.

- [21] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contours models. *International Journal of Computer Vision*, pages 321–331, 1988.
- [22] E. Kniest, D. Burns, and P. Harrison. Fluke Matcher: a computer-aided matching system for humpback whale (*Megaptera novaeangliae*) flukes. *Marine Mammal Science*, 26(3):744–756, 2010.
- [23] CP. O’Connell, S. Andreotti, M. Rutzen, M. Meyer, and P. He. *Ocean and Coastal Management*, August.
- [24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [25] Michael Picheny, Bhuvana Ramabhadran, and Stanley F. Chen. Lecture 3 Gaussian Mixture Models and Introduction to HMM’s. (September), 2012.
- [26] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.
- [27] Michael Scholl. Michael scholl – finprinting, 2015. [Online; accessed 6-December-2015].
- [28] R.B. Sherley, T. Burghardt, P.J. Barham, N. Campbell, and I.C. Cuthill. Spotting the difference: towards fully-automated population monitoring of African penguins *Spheniscus demersus*. *Endangered Species Research*, 11:101–111, March 2010.
- [29] Mark Stamp. A revealing introduction to hidden Markov models. *Department of Computer Science San Jose State*, pages 1–20, 2004.
- [30] John Stewman, Kelly Debure, Scott Hale, and Adam Russell. Iterative 3-D pose correction and content-based image retrieval for dorsal fin recognition. *Iciar 2006, Lncs 4141*, pages 648–660, 2006.
- [31] Hao Tang, Mark Hasegawa-Johnson, and Thomas S. Huang. Toward robust learning of the Gaussian mixture state emission densities for hidden Markov models. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 5242–5245, 2010.
- [32] Michael C. Scholl Thomas P. Peschak. *South Africa’s Great White Shark*. Number p. 37-38. Struik, South Africa, 2006.

- [33] Wikipedia. Confusion matrix — wikipedia, the free encyclopedia, 2015. [Online; accessed 10-November-2015].
- [34] Wikipedia. Sobel operator — wikipedia, the free encyclopedia, 2015. [Online; accessed 8-December-2015].
- [35] Wikipedia. Wildlife photo-identification — wikipedia, the free encyclopedia, 2015. [Online; accessed 6-December-2015].
- [36] Wikipedia. Singular value decomposition — wikipedia, the free encyclopedia, 2016. [Online; accessed 14-August-2016].