Efficient statistical analysis of video and image data

Ludwig Bothmann

München 2016

Efficient statistical analysis of video and image data

Ludwig Bothmann

Dissertation an der
Fakultät für Mathematik, Informatik und Statistik
der Ludwig-Maximilians-Universität München

vorgelegt von
Ludwig Maximilian Bothmann
am 12.07.2016
in München

Erstgutachter:	Prof. Dr. Göran Kauermann
Zweitgutachter:	Prof. Dr. Roland Langrock
Tag der Disputation:	27.09.2016

Für Charlotte und Benjamin

Danksagung

Bei Göran Kauermann möchte ich mich dafür bedanken, dass ich während meiner Doktorandenzeit an zwei überaus spannenden Forschungsthemen arbeiten durfte. Er gab mir den Freiraum, eigenständig zu arbeiten und meinen Weg zu finden, und stand mir gleichzeitig bei jeder Frage sehr schnell zur Seite. Seine moderne Herangehensweise an die Statistik, bei der die ursprünglich motivierende Anwendung der zu entwickelnden Methoden immer im Auge behalten wird, wird auch weiterhin ein Vorbild für mich sein.

Mein Dank gilt außerdem Roland Langrock, der sich freundlicherweise bereit erklärte, das Zweitgutachten für diese Arbeit zu übernehmen.

Bei Manuel Langkau bedanke ich mich herzlich für die Bereitstellung der Sonarfilme für das Fischprojekt und für das Beantworten zahlreicher praktischer Fragen. Ihm und Marc Schmidt danke ich darüber hinaus für die Einladungen nach Münster zu sehr angenehmen und produktiven Arbeitstreffen, an denen auch Andreas Hoffmann großen Anteil hatte.

Ebenso gilt mein Dank Joachim Tischbierek, der das User-Interface der EtWas-Software implementierte, dabei mein R-Programm auf Herz und Nieren testete und nicht locker ließ, bis nicht auch die letzte Fehlermeldung verschwunden war.

Bei Carolin Maier bedanke ich mich für ihr intensives Testen meines R-Codes zum Fischprojekt und ihr großes Engagement bei der Bearbeitung ihrer Masterarbeit.

Michael Bothmann danke ich für die Implementierung des Floodfill-Algorithmus in C++. Die daraus resultierende beträchtliche Beschleunigung der Analysen hatte einen großen Anteil daran, eine Echtzeitanalyse der Sonarfilme zu ermöglichen.

Annette Menzel, Nicole Estrella, Michael Matiu und Christian Schunk bin ich sehr dankbar für die produktive und sympathische Zusammenarbeit am Phänologieprojekt. Bjoern Menze danke ich für wertvolle Anregungen beim Verfassen des resultierenden Papers. Florian Radlherr gilt mein Dank für die freundliche Genehmigung zur Nutzung der Bilddaten von foto-webcam.eu.

Michael Windmann danke ich für die angenehme Zeit in unserem gemeinsamen Büro und die gute Zusammenarbeit am Fischprojekt. Ebenso möchte ich allen anderen Mitarbeitern des Instituts für Statistik danken, die zu einer sehr angenehmen Arbeitsatmosphäre beitrugen. Dieser Dank gilt insbesondere den Mitgliedern des Institutschors und ihrer Leiterin, Christa Jürgensonn. Darüber hinaus möchte ich mich bei Brigitte Maxa, Elke Höfner und Iris Burger für ihre vielfältige Unterstützung bedanken.

Den vielen Studierenden, die während meiner Doktorandenzeit Lehrveranstaltungen von mir besuchten, danke ich für ihr großes Interesse an der Statistik und ihr unermüdliches Nachfragen. Durch die notwendigen Erklärungen einführender und fortgeschrittener statistischer Themen habe ich selbst immer wieder viel dazu gelernt und neu verknüpft.

Bei Ludwig Fahrmeir möchte ich mich herzlich für seine Beratung und Unterstützung in entscheidenden Momenten meines Studiums bedanken, ohne die es wahrscheinlich nicht zu dieser Arbeit gekommen wäre.

Bei Nora Fenske bedanke ich mich von Herzen für ihren intensiven Beistand während der gesamten Doktorandenzeit, zuletzt sogar unter stark erschwerten Bedingungen.

München, Juli 2016 Ludwig Bothmann

Zusammenfassung

In Folge des schnellen technologischen Fortschritts steht die Statistik heute vor zahlreichen methodischen Herausforderungen. In vielen Bereichen werden Video- und Bilddaten erhoben, die wegen ihrer großen Menge nicht mehr per Hand analysiert werden können. Die vorliegende Arbeit beschäftigt sich mit zwei komplexen Fragestellungen aus dem Bereich der angewandten Statistik, die in Zusammenhang mit dem Klimawandel motiviert sind. In beiden interdisziplinären Projekten werden statistische Methoden entwickelt, um aus Video- bzw. Bilddaten effizient und mit wenig manuellem Aufwand Informationen zu gewinnen.

Das erste Projekt stammt aus dem Bereich der Fischökologie: Mit dem Ausbau erneuerbarer Energien werden immer mehr Wasserkraftwerke gebaut, die Fischen bei der Wanderung große Probleme bereiten. Vor diesem Hintergrund wird in der vorliegenden Arbeit ein System entwickelt, mit dem automatisch und in Echtzeit Fische gezählt und klassifiziert werden können, die von einer Unterwasser-Sonarkamera vor einem Wasserkraftwerk gefilmt wurden. Mit der Information über Anzahl und Art der Fische können Maßnahmen durchgeführt werden, die Fischen bei der Durchwanderung von Flüssen helfen. Das entwickelte Verfahren der Fischerkennung beginnt mit einer umfassenden Vorverarbeitung der Bilder, bei der Fische auf den Videos erkannt und verfolgt werden. Anschließend werden die Fische vermessen und Merkmale berechnet, mit denen sie im letzten Schritt mit statistischen Klassifikationsverfahren den Klassen *Aal*, *anderer Fisch* und *Treibgut* zugeordnet werden. Anhand von Beispielvideos wird die hohe Klassifikationsgüte des Verfahrens demonstriert. Im Rahmen des Projekts wurde außerdem eine Software implementiert, mit welcher das entwickelte System im laufenden Betrieb eines Wasserkraftwerks angewandt werden kann.

Das zweite Projekt ist im Bereich der Phänologie angesiedelt: Eine Hypothese von Klimaforschern besagt, dass sich die Zeitpunkte der Jahreszeitenwechsel aufgrund der Erderwärmung verschieben. Um diese Zeitpunkte für eine große Anzahl verschiedener Orte herauszufinden, können Bilder von Webcams genutzt werden. In diesem Zusammenhang wird in der vorliegenden Arbeit eine Methode entwickelt, mit der die Zeitpunkte der Jahreszeitenwechsel für gegebene Bilder einer Webcam automatisch bestimmt werden können. Dazu werden zunächst datengesteuert Bereiche der Bilder identifiziert, die viel Information über saisonale Variation haben. Im zweiten Schritt werden basierend auf der Zeitreihe der relativen Grünintensität in diesen Bereichen die Zeitpunkte der Jahreszeitenwechsel bestimmt. Hierfür werden Strukturbruchmethoden für Zeitreihen aus der Literatur benutzt. Zusätzlich wird ein Ansatz überwachter Klassifikation vorgestellt, der auf einer Varianzzerlegung der Bilder in Eigenbilder basiert und ebenfalls die Zeitpunkte der Jahreszeitenwechsel bestimmt. Das Funktionieren der entwickelten Methoden wird an Bildern zweier wissenschaftlicher Webcams demonstriert sowie an Bildern dreier Webcams, deren Daten öffentlich im Internet verfügbar sind. Außerdem wird anhand einer frei zugänglichen Webcam-Datenbank von über 13000 Webcams gezeigt, dass sich die entwickelten Methoden auch für vollautomatische Analysen großer Datenmengen eignen.

Alle entwickelten Methoden sind im statistischen Programmpaket R implementiert und in den R-Paketen `sonar` und `phenofun` frei verfügbar.

Summary

Due to the fast technological progress of our days, statistics is faced with various methodological challenges. In many different areas, video and image data are collected and cannot be analyzed manually due to the large data volume. The present thesis deals with two complex problems of applied statistics which are motivated related to the global climate change. In both interdisciplinary projects, statistical methods are developed to extract relevant information from video and image data efficiently and with low manual effort.

The first project originates from the area of fisheries ecology: With the expansion of renewable energies, more and more water power plants are constructed and make fish migration difficult. Motivated by this fact, in this thesis a system is developed which allows to count and classify fish seen on underwater sonar videos in front of water power plants automatically and in realtime. With the information about number and species of fish, protection measures can be taken to help the fish to migrate through rivers. The developed method of fish detection starts with a thorough preprocessing of the images which detects and tracks fish on the videos. Then, features are computed for each fish, and finally, the fish are assigned to the classes *eel*, *other fish* and *debris* using the computed features and standard statistical classification methods. The high classification accuracy of the method is shown for example videos. Within the scope of this project, a software was implemented which allows to apply the developed system at a water power plant during operation.

The second project originates from the area of phenology: The climate change research community is interested in the question if season onset dates change due to the global warming. To observe season onset dates for a large amount of different locations, webcam images can be used. In this thesis, a method is developed which allows to automatically extract season onset dates from webcam images. Therefore, regions of interest on the images are defined in a data-driven way, i.e., areas containing pixels with high information about seasonal variation. Subsequently, season onset dates are derived from the time series of percentage greenness in these regions using structural change point methods for time series from the literature. Additionally, a supervised classification approach based on a variance decomposition of the images in eigenimages is presented which also determines season onset dates. The usefulness of the developed methods is demonstrated with data from two scientific webcams and three webcams with data publicly available from the internet. Moreover, by analyzing images from a publicly available webcam database of over 13000 webcams it is shown that the developed methods can be applied completely automatically to large data volumes as well.

All developed methods are implemented in the statistical software package R and publicly available in the R packages `sonar` and `phenofun`.

Contents

Chapter 0: Outline

This thesis results from two interdisciplinary research projects involving the *Institut für Statistik* at the *Ludwig-Maximilians-Universität München*. The first project originates from the field of fisheries ecology and connects fisheries biologists, computer scientists and statisticians. It is motivated by the question how fish can pass a water power plant when migrating downstream in a river. With this purpose, an approach shall be developed which allows to count and classify fish based on underwater sonar videos in realtime. Additionally, a user-friendly and fast software shall be provided based on this approach. The second project originates from the field of phenology and connects phenologists, computer scientists and statisticians. It is motivated by the question how season onset dates change with the warming of the global climate. In this project, a method shall be developed which allows to automatically identify season onset dates based on webcam images. The underlying idea is to analyze the images of a large number of webcams located all over the world simultaneously and in short time.

Both projects share the methodological property that a large amount of image data shall be analyzed automatically and that manual analyses would be possible but unfeasible due to the large data volume. The term *efficient* in the title of this thesis refers to the requirement that all developed methods have to deliver good results at low computational complexity. This means that minimal improvements of classification accuracy can be sacrificed when in return a high gain in computing time or computing complexity is obtained.

The two main parts of this thesis can be read independently. Detailed motivations and outlines of the different projects are given in the respective introductory Chapters 1 and 7. In this chapter, we shortly sketch the frames and goals of the two projects:

Realtime classification of fish in underwater sonar videos

On behalf of the *Bezirksregierung Düsseldorf, Obere Fischereibehörde, Germany,* and *RWE Innogy, Hydro Power & New Technologies, Essen, Germany*, an interdisciplinary working group cooperated in this research project. Apart from statisticians from the *Institut für Statistik* at the *Ludwig-Maximilians-*

Universität München, biologists from the *Büro für Umweltplanung, Gewässermanagement und Fischerei, Bielefeld, Germany* and from the *LFV Hydroakustik GmbH, Münster, Germany* were involved in this project, as well as a computer scientist from *jTi-Soft, Gütersloh, Germany*.

The overall goal of this project was the development of a realtime warning system (*EtWas – Echtzeit-Warnsystem*) for the arrival of fish at a water power plant. The project started at the beginning of 2012 and was successfully completed in April 2014. As main output, a software was implemented for the usage at a water power plant. This software announces the arrival of fish in front of the power plant and warns the operator to take measures for the protection of the fish.

One sub-project was the automatic detection, counting and classification of fish based on videos produced by an underwater sonar camera. The first part of this thesis treats this sub-project and is mainly based on Bothmann *et al.* (2016b). However, this thesis extends the paper in several ways: Each individual step of the analysis is explained and illustrated thoroughly, this was not completely possible in the article for reasons of space. In addition, the developed tracking algorithm is presented in detail in Section 2.2. Moreover, Chapter 5 gives computational details on the implementation of the methods and information on the application in practice. Finally, the discussion in Chapter 6 is thoroughly extended to justify the choices made in our analysis and to point out directions for future research. Furthermore, the results are discussed from a user's perspective.

Automated processing of webcam images for phenological classification

In July 2014, researchers from the *Institut für Statistik* at the *Ludwig-Maximilians-Universität München*, the *Wissenschaftszentrum Weihenstephan für Ernährung, Landnutzung und Umwelt* at the *Technische Universität München, Freising*, and the *Fakultät für Informatik* at the *Technische Universität München* started an interdisciplinary research project.

The idea was to bring together knowledge from the fields of computer science and statistics for answering phenological questions arising with the global climate change. In a first project, which is treated in this thesis, we pursued the question whether season onset dates can be identified in a fully automated way using digital webcam images of natural motifs such as deciduous forests. These methods could later be used on a larger scale: By analyzing images from several hundreds or thousands of webcams with data from various years, knowledge about temporal and spatial variation of such onset dates could be gained.

In March 2016, a manuscript (Bothmann *et al.*, 2016a) was submitted presenting the developed methods which could be successfully applied to two scientific cameras, three open-access cameras with data available at `http://www.foto-webcam.eu` and 13988 cameras from the AMOS database with data available at `http://amos.cse.wustl.edu/`. Based on this manuscript, the second part of this thesis thoroughly describes and illustrates the developed methods and the results. Additionally, Chapter 10 contains an alternative approach for the identification of season onset dates. This approach uses supervised classification methods based on eigenimages of the data.

Contributing manuscripts

The present work is mainly based on the following manuscripts:

- *Bothmann L, Windmann M, Kauermann G (2016b). Realtime classification of fish in underwater sonar videos. Journal of the Royal Statistical Society: Series C (Applied Statistics), 65(4), 565–584. doi: 10.1111/rssc.12139.*

  Part I of this thesis is based on this manuscript. Contributions of the authors are:

  Involved in the project setup and communication within the project: LB MW GK. Developed methods for the localization of hotspots: LB GK. Developed tracking algorithms: LB. Developed methods for feature extraction: LB GK. Implemented the methods: LB. Analyzed the data: LB. Wrote the paper: LB GK.

- *Bothmann L, Menzel A, Menze BH, Schunk C, Kauermann G (2016a). Automated processing of webcam images for phenological classification. PLOS ONE. Under review.*

  Part II of this thesis is based on this manuscript. Contributions of the authors are:

Conceived and designed the experiments: LB AM BM CS GK. Collected and prepared the data: LB AM CS. Developed optimality criteria: LB. Developed sROI method: LB. Developed uROI method: LB BM AM GK. Developed the method for supervised classification based on eigenimages: LB. Implemented the methods: LB. Analyzed the data: LB AM. Wrote the paper: LB AM BM CS GK.

Part I.

Realtime classification of fish in underwater sonar videos

Chapter 1: Introduction

1.1. Motivation and research goals

Motivation

In answer to the global climate change, renewable energy sources become of increasing relevance. The wide use of renewable energy sources such as water power, wind power and solar power shall ensure the energy supply of men and simultaneously decrease the adverse environmental impact. Nevertheless, the use of renewable energy sources leads to new challenges in environmental protection.

There are many fish species which migrate at some stage of their lives and thereby cover large distances. These migrating fish are distinguished in two classes: While *anadromous* fish live most of their lives in seawater and migrate upstream into fresh water to spawn and breed, *catadromous* fish do exactly the opposite. They live most of their lives in fresh water and migrate downstream into the sea to spawn and breed (Northeast Fisheries Science Center, 2015).

The most important example for catadromous fish is the eel. Towards the end of their lives, eels migrate from European rivers to the Sargasso Sea in the Atlantic Ocean. Water power plants are major obstacles for fish migration. Since eels are a threatened species, power plant operators in Germany are obliged to ensure their migration by implementing protection measures.

One of these protection measures are so called *fish passes*. Fish passes can be found at most modern barrages and water power plants. The basic idea is that the flow in the fish pass is weak enough such that the fish are able to pass the barrage or power plant upstream. If the design of the fish pass is successful, this is a good way to help the fish migrating upstream.

Unfortunately, fish passes hardly work downstream and “satisfactory solutions for downstream migration problems” have not been found yet (Larinier and Travade, 2002). Therefore, new approaches have to be developed that help the fish migrating downstream.

This is the point where our project starts: We want to develop a system which allows to observe and investigate the underwater activity of fish in front of a water power plant automatically. With the information at which time how many fish of protected species are in front of the power plant, protection measures for the downstream migration can be put in place. The description and choice of possible protection measures goes beyond the scope of this work. In this work we focus on counting and classifying the fish. This means we want to

1. count how many fish are in front of the power plant in a given period and
2. classify the detected fish into the categories *eel* and *other fish*.

The discrimination into eels and other fish is of interest since the European eel is a threatened species and its protection is thus of paramount importance.

Sonar videos

One approach to observe underwater activity are sonar videos. Sonar videos have the advantage over optical videos that they require no light and that they deliver good videos even in turbid water. The sonar video device used in the present analysis is the so called DIDSON (Dual-frequency IDentification SONar), see `http://www.soundmetrics.com` for details on the sonar camera.

The sonar camera emits sonar waves in different directions and records the respective echoes. From the delay and strength of the recorded echoes, a software generates a two-dimensional gray-scale image per time point, see for an example Figure 1.1 where the four dark shadows represent four trouts swimming in front of the camera. Note that these images represent a top view, i.e., the fish are observed from the top, not from the side. Over the time the DIDSON delivers a stream of these images and thus, a video is recorded. Example videos can be found at `http://bothmann.userweb.mwn.de/dissertation.html`.

Research goals

Based on these sonar videos, a simple strategy for the counting and classification of fish would be to engage at each water power plant a team of biologists who would watch the sonar videos day and night and manually count how many fish of which species could be seen. But obviously, this strategy is neither feasible nor affordable.

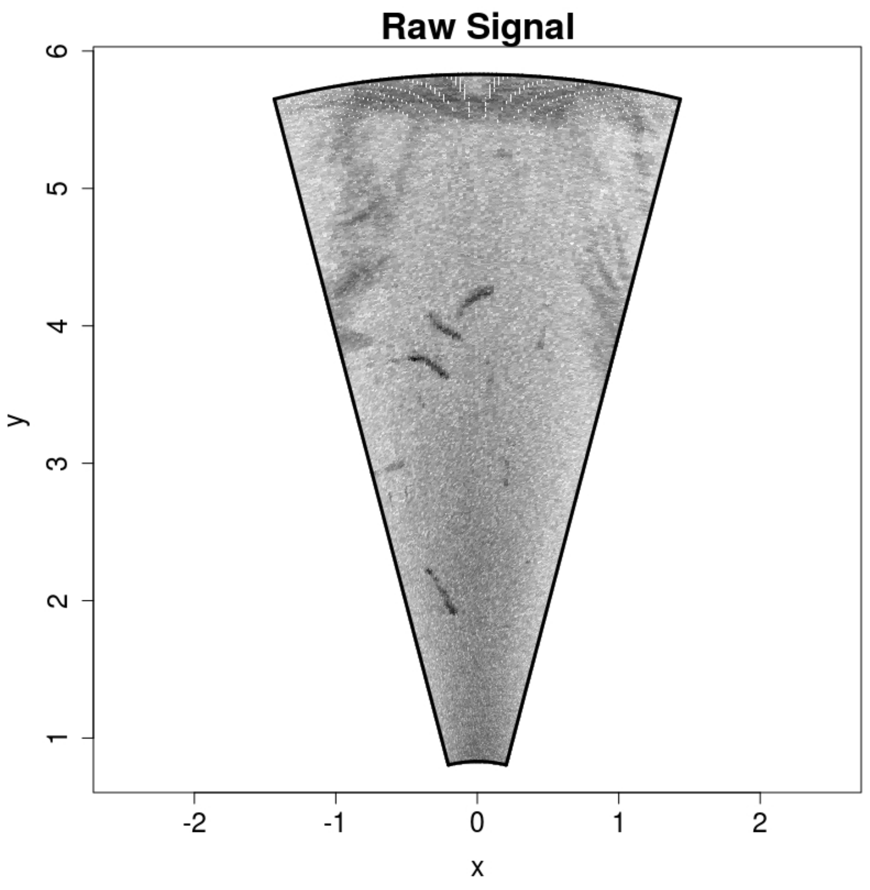


Figure 1.1. Example sonar image showing trouts.

Thus, we want to automate the process and therefore define the following research goals. Given a sonar video of T seconds, we want to develop a software which:

1. delivers a sensible count of fish present in the video sequence,
2. distinguishes between *eels* and *other fish*,
3. runs in realtime, i.e., the computing time shall be less than T, the running time of the video and
4. has a high usability for practitioners not familiar with the statistical software R or even with statistics.

The final system shall run continuously at the water power plant and report fish presence in realtime. Thus, short computing time and high usability are as important as high classification accuracy of the system.

Literature

The analysis of DIDSON sonar images and videos has become a recent research focus in fisheries ecology. Holmes *et al.* (2006) explore the accuracy of the system by visually counting fish on the DIDSON videos and comparing them with simultaneous observer counts in the river. Rakowitz *et al.* (2012) investigate fish behavior towards a surface trawl of a fishing boat. Burwen *et al.* (2010) explore the accuracy of length measurements of sonar images of fish. Langkau *et al.* (2012) pursue the question of identifying fish by optical projection. Crossman *et al.* (2011) use DIDSON videos to monitor presence and activity of white sturgeons in a Canadian river, while Pipal *et al.* (2012) estimate the escapement of small populations of steelhead. Overall, most (if not all) investigations of DIDSON data are based on visual analyses of the image sequences while computer-driven image analysis is only rudimentary developed, see Mueller *et al.* (2010) and Mueller *et al.* (2008) and references given there.

1.2. From sonar videos to fish classification

Data structure of sonar videos

The DIDSON emits sonar waves at different angles between -14.25° and 14.25° in steps of 0.3° or 0.6° resulting in 96 or 48 beams, respectively. At each beam, the sonar response is recorded at 512 pixels on an equidistant grid, this sonar response is visualized in Figure 1.1 as gray intensity. The range of the grid can be tuned; for example, in our data the range is from 0.83 to 5.83 meters measured from the lense of the DIDSON. Figure 1.2 visualizes the structure of such an image. With the information about angles of the beams and range of the grid, it is possible to compute either cartesian coordinates or polar coordinates of each pixel.

For each pixel (i, j) and time point t we observe a signal y_{ijt} standing for the gray intensity. The data array $\boldsymbol{Y} = \{y_{ijt}; i = 1, \ldots, n_1, j = 1, \ldots, n_2, t = 1, \ldots, n_3\}$ serves as our three-dimensional raw data.

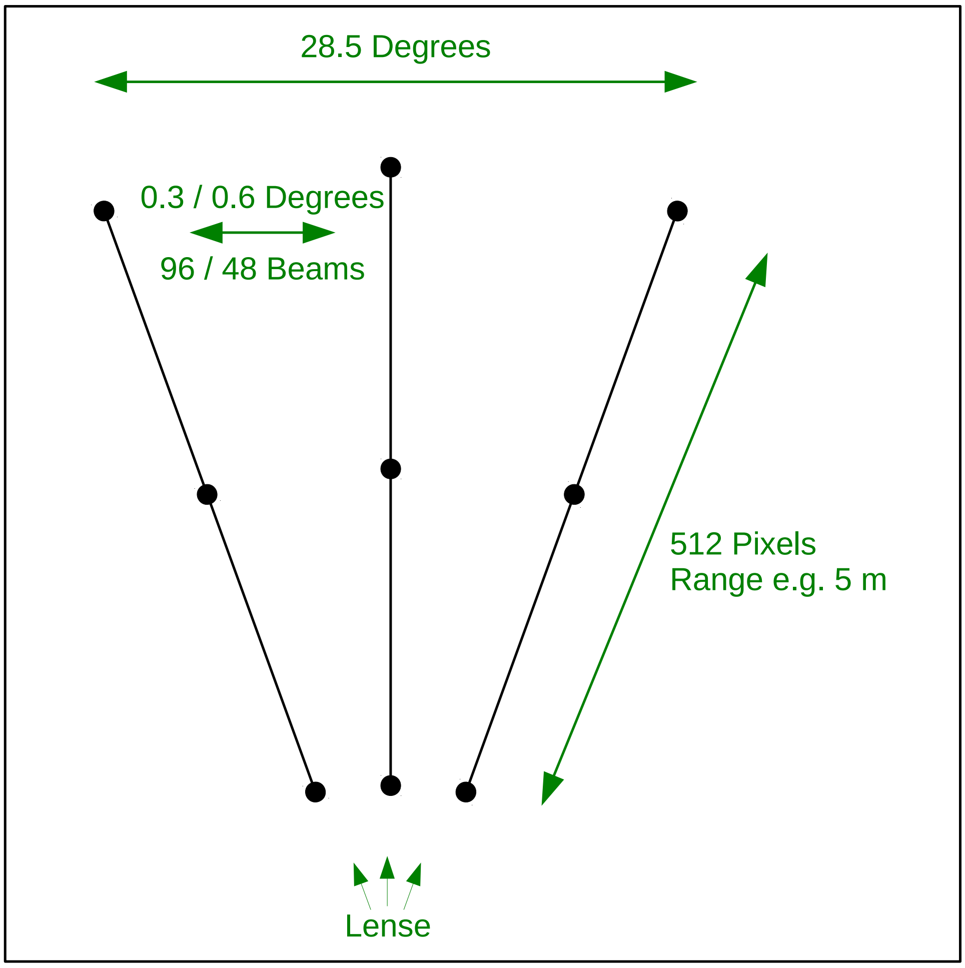


Figure 1.2. Structure of a DIDSON image with descriptions in green.

Analysis steps

The main steps to classify and count fish present in the videos are as follows.

First, we have to find those areas in each image which most likely contain a fish, these areas will be called *hotspots* in the remainder (Section 2.1). Once we have found the hotspots on each single image, we have to connect the hotspots of the same fish over time because we would like to classify each object swimming in the water rather than each hotspot of each object. This procedure is called *tracking* in the remainder (Section 2.2).

Second, we have to extract features from the tracked objects which describe the fish and allow for the discrimination of fish species (Chapter 3). This step results in a data matrix which contains in each row classification variables for each object, as for example given in Table 1.1.

Table 1.1. Data structure of classification data set (example).

ID	Species	Length (in cm)	Width (in cm)	Speed (in m/s)	...
1	?	45.0	6.2	0.5	...
2	?	42.2	5.9	0.4	...
3	?	24.0	5.0	1.7	...
⋮	⋮	⋮	⋮	⋮	...

Once we have generated this data matrix, we can make use of the whole spectrum of statistical classification methods such as discriminant analysis, support vector machines, random forests etc. to classify the tracked objects with respect to the three classes *eel*, *other fish* and *debris* (Chapter 4).

Table 1.2 shows an example output of the analysis: Each object is classified with respect to the three classes *eel*, *other fish* and *debris*, additionally the time points of first and last appearance are given.

Table 1.2. Intended data structure after classification (example).

ID	Species	First appearance (in s)	Last appearance (in s)
1	Eel	15.0	18.3
2	Eel	21.4	29.0
3	Other Fish	24.0	31.0
⋮	⋮	⋮	⋮

Nomenclature: Objects and hotspots

Note the use of the terms *object* and *hotspot*: In our nomenclature, each solid object, as for example an eel, a trout or a branch is called *object*. Each *object* can be seen on several images of the sonar video, i.e., each *object* consists of one or more *hotspots*. The classification shall be on the level of objects, i.e., we have to extract and gather all hotspots corresponding to the same object. In Chapter 3, we will extract features on the level of objects which are based on these sets of hotspots and describe the objects.

Data at hand

We apply the proposed methods to three classes of sonar videos recorded by a DIDSON camera, Table 1.3 summarizes basic properties of these videos.

Table 1.3. Properties of the videos used for the analysis. Example fragments of these videos can be found at http://bothmann.userweb.mwn.de/dissertation.html

Video	Duration	Time resolution	No. of images	Image resolution
Eels	11 min	10 images/sec	6600	512 × 96 pixels
Trouts	13 min	10 images/sec	7800	512 × 96 pixels
Debris	30 min	10 images/sec	18000	512 × 96 pixels

For the eel video, several eels were put in a net cage into the water in front of a sonar camera, the same applies to the trout video. These videos were recorded in the year 2010 in the lake Möhnesee near Dortmund, Germany. A careful visual inspection showed the purity of these videos, i.e., we can be sure that all objects found on the eel video are eels and all objects found on the trout video are trouts.

For the debris video, the camera monitored the river Lippe near Hamm, Germany, upstream a water power plant in the year 2009 during a period of time when a huge amount of leaves was swimming in the water. On this video, no living objects can be seen.

Classification outcome: Eels, trouts and debris

Note that in our data, all fish of the class *other fish* are actually trouts. Since most fish species living in European rivers look like trouts from above, we do not consider this as a drawback of our method and think that the results can be generalized to other fish species as well. Therefore, we will use *trout* synonymously with *other fish* in the remainder.

The third class – debris – is needed for the following reason: Our first goal is to get a sensible count of the number of fish present in the video. The preprocessing steps leading to the classification data set do not distinguish between the types of objects swimming in the water. Additionally to fish there are many things which could swim in an ordinary river, for example leaves, branches but also waste etc. Therefore, we have to divide the found objects into dead and living objects. All dead objects shall be classified into the class *debris*. Our second goal is to get a sensible count of the number of eels. Thus, the living objects are divided into classes *eel* and *other fish*.

Implementation

As mentioned above, short computing time and high usability are as important as high classification accuracy of the system. Therefore, all methods are implemented in the statistical software package R (R Core Team, 2016) and C++ as efficiently as possible (see Section 5.1 for details). The resulting R package `sonar` is available on R-Forge at https://r-forge.r-project.org/projects/sonar/, the documentation of the package is sketched in Appendix B. Additionally, a computer scientist was engaged to develop a user interface for the application of the system at a water power plant (see Section 5.2). Thereby, the user can read in sonar videos and analyze them automatically with a few mouse clicks.

Chapter 2: Preprocessing of the sonar videos

The first step of the analysis is the preprocessing of the sonar videos. On each single image, those pixel clouds have to be located which most likely represent a fish. These relevant pixel clouds are called *hotspots*, Section 2.1 describes in detail how the hotspots are located.

Since the classification shall be carried out on the level of objects rather than on the level of hotspots, we have to gather those hotspots which correspond to the same object. This procedure is called *tracking*, Section 2.2 describes the developed tracking algorithms in detail. (For the use of the terms *object* and *hotspot* see page 13.)

2.1. Localization of hotspots

This section explains how hotspots can be located and cut out from a video. We proceed in four steps:

1. Noise filtering: We filter the white noise from the video with three-dimensional splines using the linear array model proposed by Currie *et al.* (2006).
2. Centering: We center each image of the video around the mean image over time.
3. Thresholding: We threshold the centered image to delete pixels which do not correspond to an object.
4. Extracting the hotspots: We cut out each area of interest, i.e., hotspot, on each image using the flood-fill algorithm proposed by Lee *et al.* (1987).

Figure 2.1 visualizes the goal of this section: We want to delete every pixel from the raw signal which does not correspond to an area of interest. Note that this and the following figures are two-dimensional visualizations of a problem which indeed is three-dimensional. For a better understanding of the analysis we provide short videos in addition to most of the following figures. The videos can be viewed by clicking on the link in the caption of the respective figures or by accessing http://bothmann.userweb.mwn.de/dissertation.html. All steps of the preprocessing are explained in the following for this example video showing a trout moving from left to right.

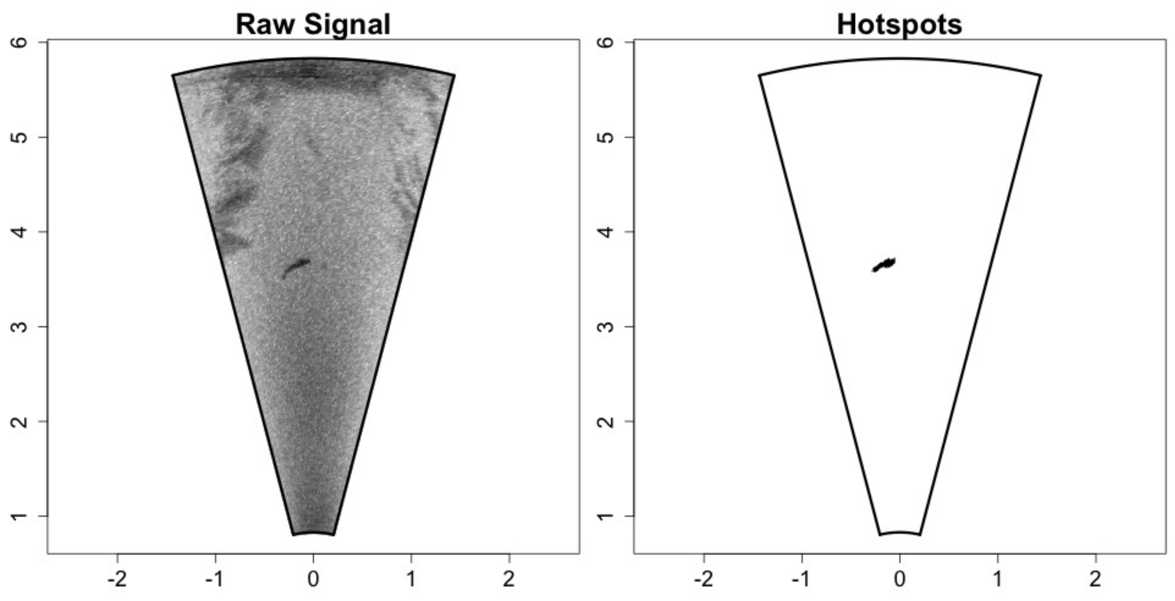


Figure 2.1. From raw signal (left) to hotspots (right). (Link to the video)

An alternative visualization of the goal of this section is given by Figure 2.2. The top row shows three snapshots of a sonar video at three time points: An eel is swimming from right to left. The bottom row shows the result of the preprocessing: A set of hotspots associated to the eel.

Note that in the preprocessing we ignore the fact that the pixels lie in reality on a complex, somehow curvy grid as shown in Figure 1.2. We pretend that the pixels lie on a regular rectangular grid which considerably simplifies the preprocessing yielding good results. In further work the curvy grid of the pixels could be considered, but we feel that the effort would be huge while the benefit would be rather small.

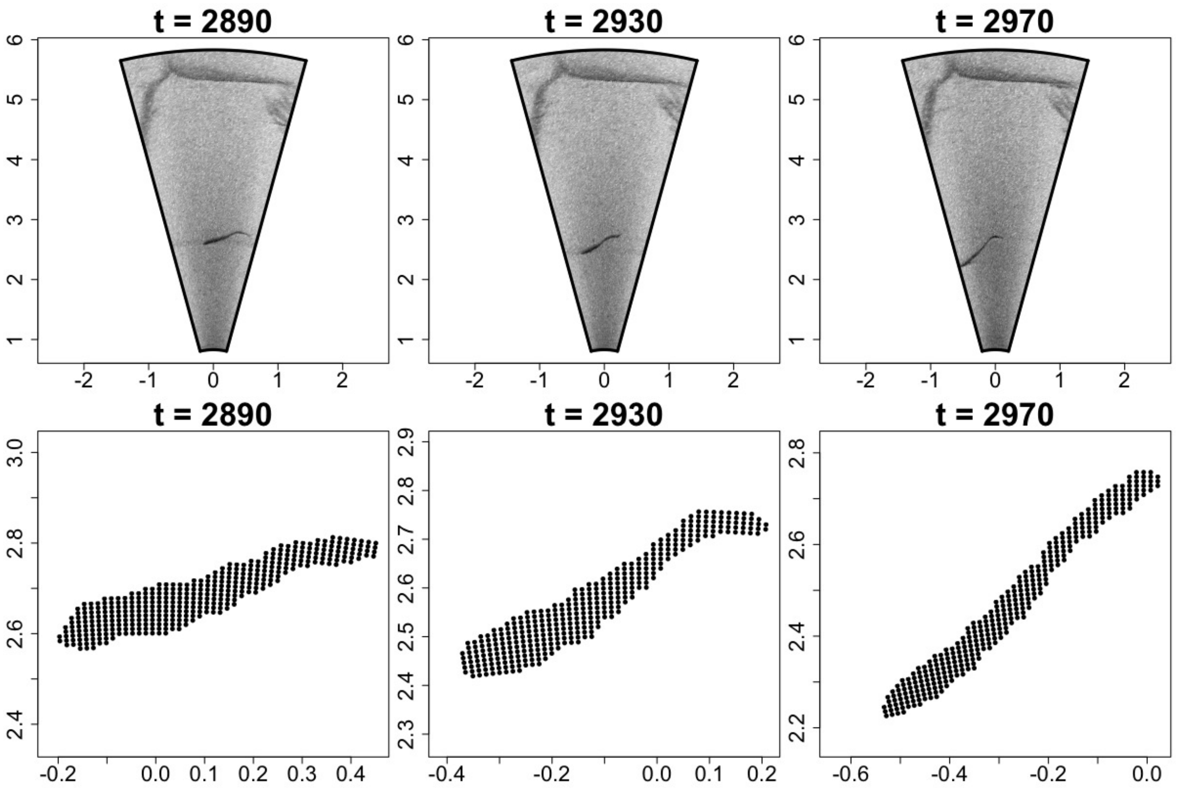


Figure 2.2. Under-water images of an eel swimming from right to left (top row) and detected corresponding hotspots (bottom row) at three time points given in 1/10 of a second.

Noise filtering

The first step of the preprocessing is to reduce the white noise in the video. The data array $\boldsymbol{Y}$ containing the signals y_{ijt} for pixel (i, j) and time point t is three-dimensional. To remove the white noise we want to smooth a three-dimensional surface using the linear array model proposed by Currie *et al.* (2006). For a better understanding, we first present the idea in usual matrix notation. We therefore vectorize the data to $\boldsymbol{y} = (y_{111}, \ldots, y_{n_1 n_2 1}, y_{112}, \ldots, y_{n_1 n_2 n_3})^\top$ with $n_1 \times n_2$ as the dimension of each single image and n_3 as the number of images over time, i.e., the length of the video. In matrix notation, the linear model can be written as follows:

$$\boldsymbol{y} = \boldsymbol{B\theta} + \boldsymbol{\varepsilon}, \qquad \boldsymbol{\varepsilon} \sim \mathsf{N}_n\left(\boldsymbol{0}, \sigma^2 \boldsymbol{I}\right), \tag{2.1}$$

where $n = n_1 \cdot n_2 \cdot n_3$. The design matrix $\boldsymbol{B}$ is built from the Kronecker product $\boldsymbol{B} = \boldsymbol{B}_1 \otimes \boldsymbol{B}_2 \otimes \boldsymbol{B}_3$, with $\boldsymbol{B}_k$ $(k = 1, 2, 3)$ being marginal design matrices of dimension $n_k \times p_k$ built from B-splines to be specified subsequently. In our data, each image is of dimension 512×96. Thus, a choice of $100 \times 25 \times (n_3/2)$ B-Spline basis functions with equidistant knots shows to be reasonable and leads to good results. While (2.1) is conceptually simple, it is numerically unfeasible to estimate $\boldsymbol{\theta}$ when n_1, n_2 and n_3 are large. For example, a rather short video of $T = 10$ seconds with 10 images per second has dimension $512 \times 96 \times 100$, and when using B-splines of dimension $100 \times 25 \times 50$, $\boldsymbol{B}$ is of dimension $4.915.200 \times 125.000$. We therefore rewrite model (2.1) as a linear array model

$$\boldsymbol{Y} = \boldsymbol{B}\Theta + \boldsymbol{E}\ . \tag{2.2}$$

Here, $\boldsymbol{Y}$ contains the data in a three-dimensional array structure. The design matrix $\boldsymbol{B}$ is built from $\boldsymbol{B}_1$, $\boldsymbol{B}_2$ and $\boldsymbol{B}_3$ as above and Θ is a three-dimensional array of dimension $p_1 \times p_2 \times p_3$. The error terms are stored in an array $\boldsymbol{E}$ of same dimension as $\boldsymbol{Y}$, namely $n_1 \times n_2 \times n_3$. Writing the model in this form obviously requests a definition of the product of a matrix and an array. This definition is given in Currie *et al.* (2006) and we refer to Appendix A for the exact statement.

Using model formulation (2.2) instead of (2.1) the parameter estimations $\widehat{\Theta}$ and the predictions $\widehat{\boldsymbol{Y}} = \boldsymbol{B}\widehat{\Theta}$ can be computed in a very fast way with software allowing for array manipulation (e.g. the statistical software R, R Core Team, 2016). For further information about algebraic details and computational advantages of array calculation see Currie *et al.* (2006).

Note again that we ignore the fact that the pixels are not located on a regular equidistant grid in all three dimensions while computing the B-Spline basis functions. Due to the fact that the results are very appealing we consider this simplification as reasonable. Note further that we use unpenalized splines. A penalization could be incorporated easily if necessary, see Currie *et al.* (2006), with the drawback of higher computational effort and time.

Figure 2.3 shows the data before and after this smoothing step.

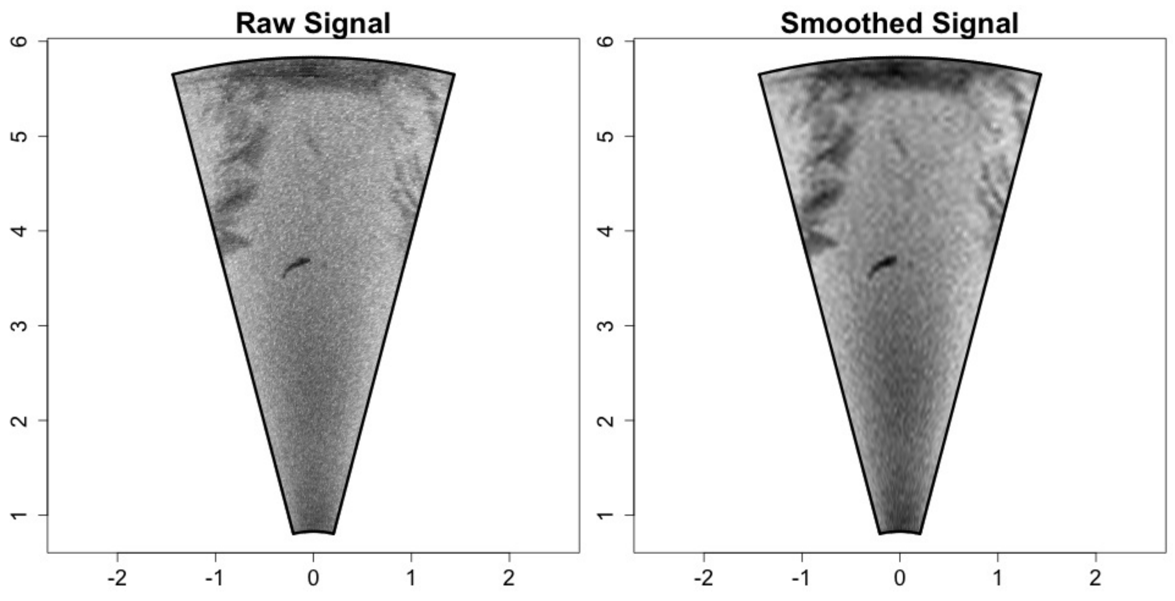


Figure 2.3. Raw signal (left) and smoothed signal (right). (Link to the video)

Centering

Now, the signal is smoother than before the noise filtering, but still the echo of the river bank can be seen in the upper left and right corners. To reduce the signals of such fixed objects in the water, we apply a pixel-wise mean correction as follows: Let $\widehat{y}_{ij1}, \ldots, \widehat{y}_{ijn_3}$ denote the smoothed signal at pixel (i, j) over time. We apply a pixel-wise mean correction and calculate $\tilde{y}_{ijt} = \widehat{y}_{ijt} - \bar{y}_{ij}$, where $\bar{y}_{ij} = (1/n_3) \sum_{t=1}^{n_3} \widehat{y}_{ijt}$ is the mean of the predicted signal at pixel (i, j) over time. The result of this step is shown in Figure 2.4 where the echo of the river bank is no longer present in the video.

Thresholding

Now, the signal of the fish is much larger than everything else. Therefore, we threshold each image to delete those pixel signals which do not correspond to the fish. This means that we set all pixel signals to zero for which $\tilde{y}_{ijt}$ is below a threshold a, and get the signal of the cleaned image after background subtraction z_{ijt} as

$$z_{ijt} = \begin{cases} \widehat{y}_{ijt}, & \text{if} \quad \tilde{y}_{ijt} > a \\ 0, & \text{if} \quad \tilde{y}_{ijt} \leq a \end{cases} \tag{2.3}$$

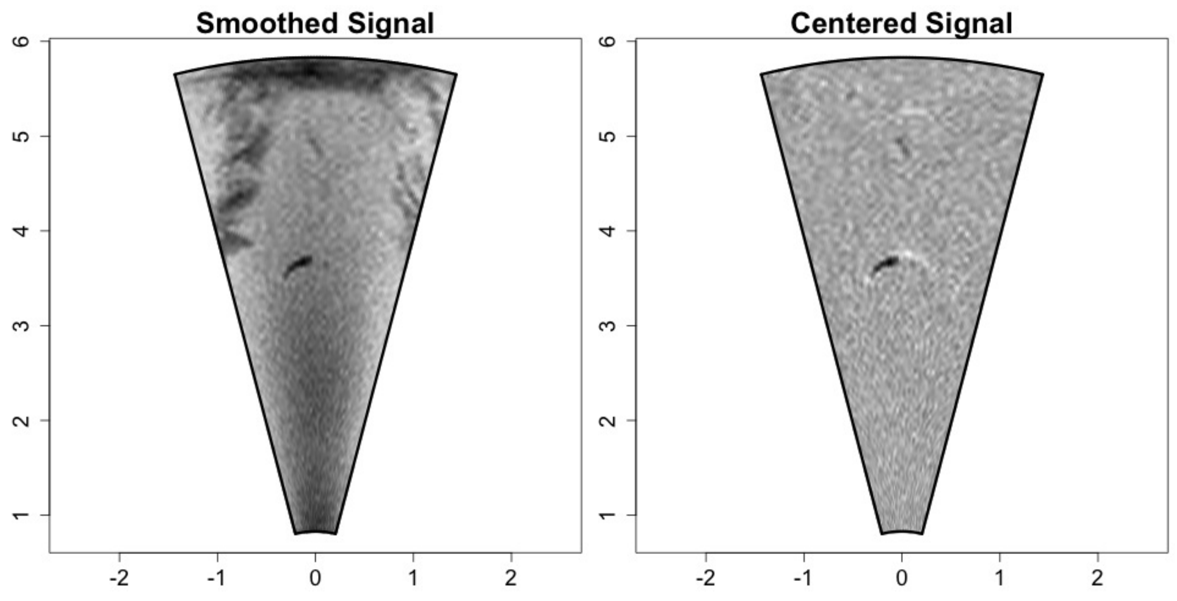


Figure 2.4. Smoothed signal (left) and centered signal (right). (Link to the video)

The threshold a has to be determined for the given sonar camera and location in a calibration process. For our data, a good threshold turned out to be $a = 18$. Figure 2.5 shows the data before and after this thresholding step.

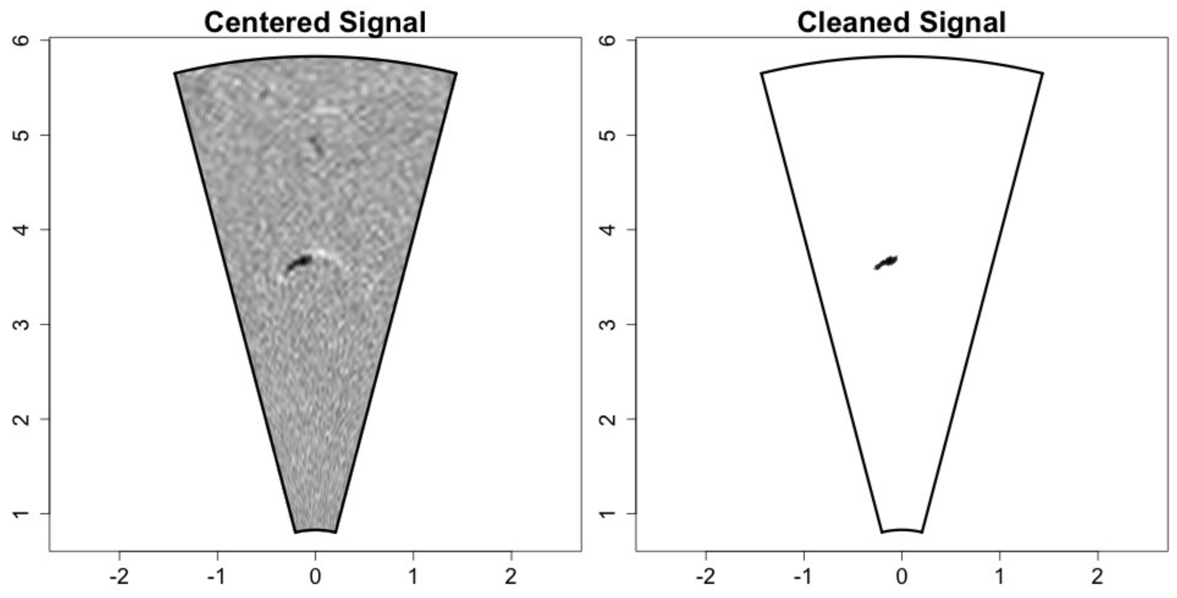


Figure 2.5. Centered signal (left) and cleaned signal (right). (Link to the video)

Figures 2.6 and 2.7 illustrate the previous cleaning steps from another perspective. For three selected pixels (a), (b) and (c) located in different areas of the image (see Figure 2.6), Figure 2.7 displays the temporal behavior of the raw, smoothed, centered and cleaned signal over time. Pixel (a) lies in the upper left area where the river bank is visible, pixel (b) lies in the center where only noise can be seen and pixel (c) lies on a location where the fish passes by between time points 9 and 19. In Figure 2.7 one can see that the centered signals $\tilde{y}_t$ (solid lines) for fixed object pixels (a) and noise pixels (b) are almost zero over time and can therefore be filtered by thresholding. The centered signal of the fish pixel (c) is clearly non-zero over time and thus remains in the cleaned image after the thresholding.

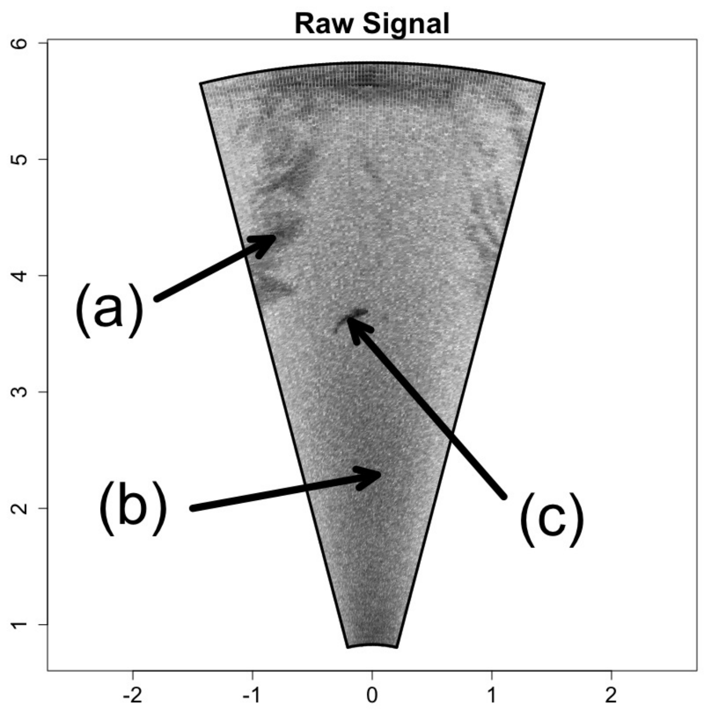


Figure 2.6. Location of the three selected pixels illustrated in Figure 2.7.

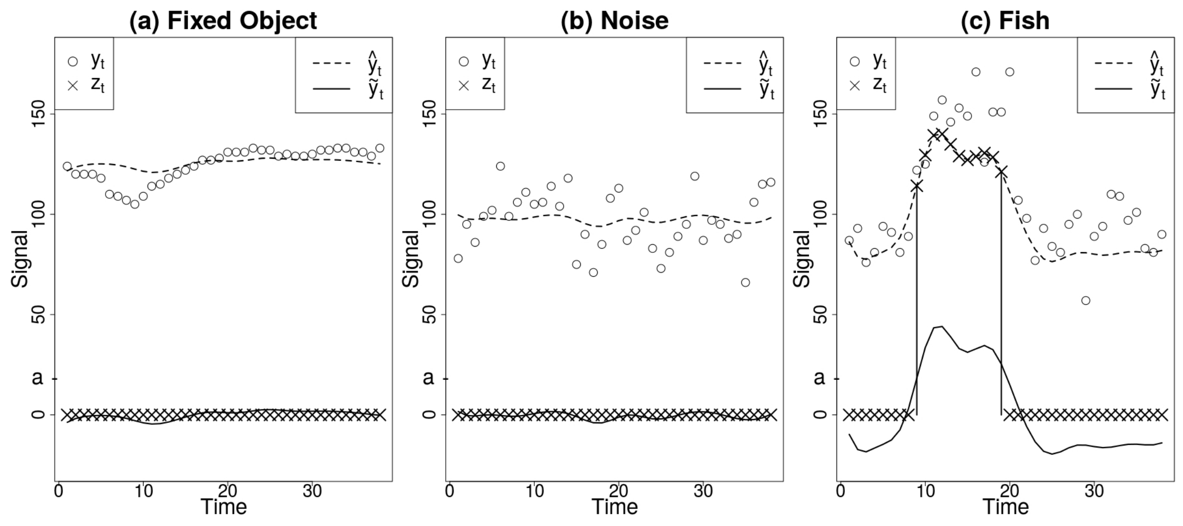


Figure 2.7. Temporal behavior of the signal at three pixels (a) – (c) and visualization of the cleaning process (for exact locations see Figure 2.6). Points (∘) show observed signal y_t while crosses ($\times$) stand for modified signal z_t after cleaning. Dashed lines visualize the smoothed signal $\widehat{y}_t$, solid lines show the centered signal $\tilde{y}_t$.

Extracting the hotspots

All pixels with a positive signal z_{ijt} after the thresholding step are considered to be fish pixels. All fish pixels together yield the hotspots. If only one fish is present in the video, the localization of hotspots is completed.

However, on most videos, several fish can be seen. Therefore, we have to handle the case where more than one fish is present on a single image. For example, after the cleaning process applied to the image shown in Figure 2.8 (left) we get the image shown in Figure 2.8 (right). In order to separate the four fish we assign a unique label to each point cloud that sticks together. With statistical tools, this problem can be solved with classical methods of cluster analysis, but it turns out that the computing takes too long. Therefore, we use the flood-fill algorithm first proposed by Lee *et al.* (1987) to assign the labels.

The labelling of point clouds that stick together has a further advantage: Sometimes, a few spam pixels remain in the cleaned image after the thresholding step. By deleting all pixel clouds which do not exceed a certain size, we can remove these spam pixels. In our data, pixel clouds with less than 50 pixels were deleted.

This step finishes the localization of relevant hotspots in the video. Henceforth we work with a changed data structure: Instead of a three-dimensional array – representing the sonar video – we have a large set of hotspot images as shown in Figure 2.2 (bottom row). These hotspot images are small two-dimensional matrices with entries 1 or 0. Additionally, for each hotspot image we record the localization in the original sonar image.

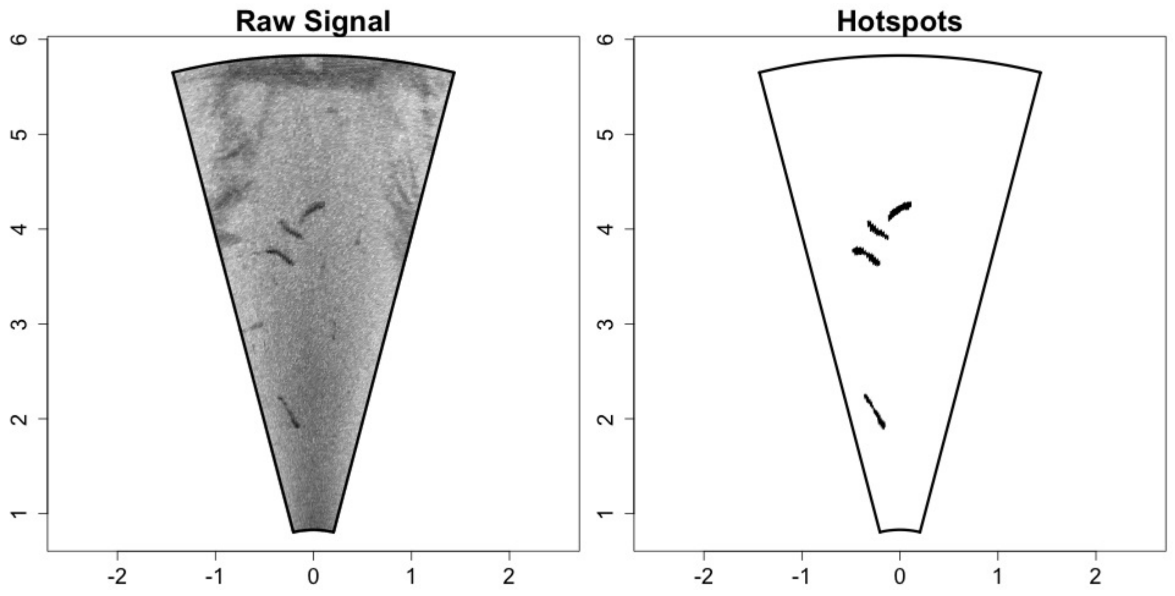


Figure 2.8. Raw signal (left) and hotspots (right) for a case with multiple fish on a single image.

2.2. Tracking of objects

As a result from the preprocessing in Section 2.1, we extracted for each object, i.e., for each eel, trout or piece of debris in the video all hotspots at all time points. Before we can move on to the extraction of features for preparing the classification, we have to take care of another problem: In the final step we want to classify each object, not each hotspot. Therefore, we have to assign the hotspots to their corresponding objects. For now we only have a large set of hotspots and we do not know which hotspot belongs to which object.

Figure 2.9 illustrates the problem and the desired solution: At time t = 1430, four hotspots are detected, half a second later, at time t = 1435, three hotspots are detected. It is easy to match the corresponding hotspots visually and it is obvious that the second fish from the top in the left image disappears on the right image. But, in practice it is not possible to match the hotspots visually. The challenge is now to create an algorithm which tracks the objects over time automatically. This tracking algorithm is based on differences between the centroids of the hotspots and is explained in detail in this section.

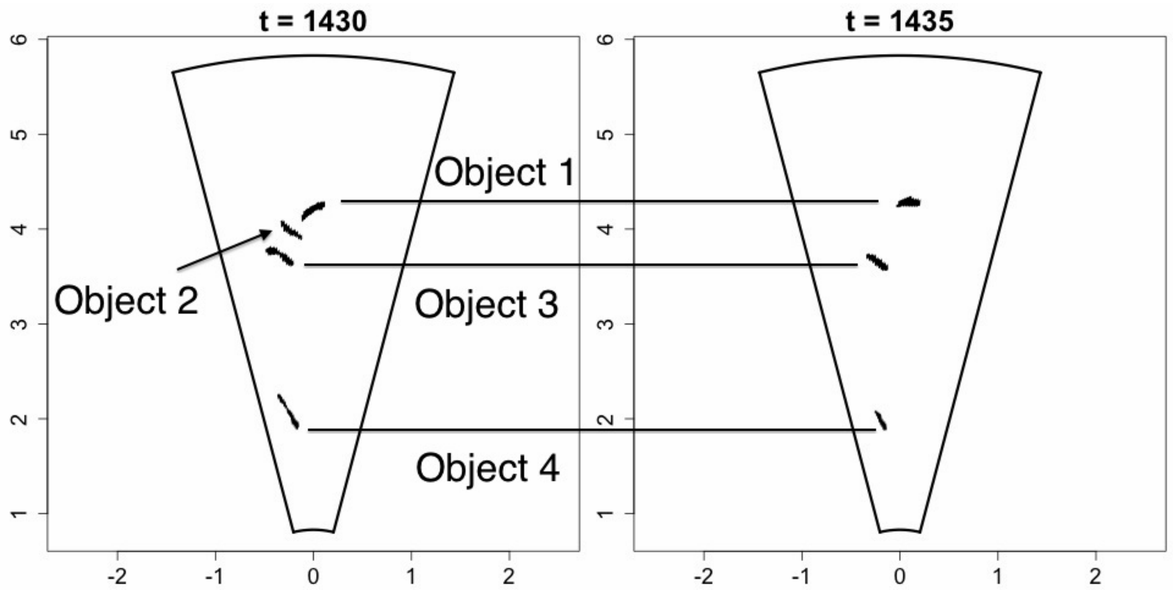
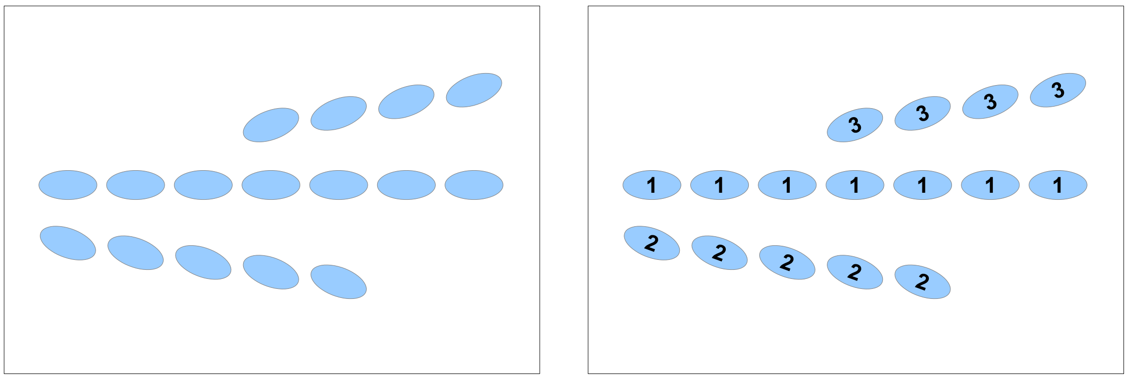


Figure 2.9. Tracking of objects: Example for matching of hotspots at two different time points.

As stated above, assigning the hotspots to their corresponding objects is the main challenge for videos that contain more than one single object. Figure 2.10 provides an alternative illustration of the problem. This kind of plot gives a sketch of the tracking problem and its desired solution over time and is used throughout the section. Each plot is a fictitious overlay of a small number of hotspot images as shown in Figure 2.8 (right), where each blue ellipse stands for one hotspot. All objects are swimming from the left to the right with a maximum of 7 appearances in the illustrations. Figure 2.10 a) shows the situation prior to the tracking: We have located the hotspots for three objects over time but do not know which hotspot corresponds to which object. Figure 2.10 b) shows the desired solution: Each hotspot is assigned to one object and is labelled with a tracking number which is unique for each object.



(a) Tracking problem (b) Tracking solution

Figure 2.10. (a) Detected hotspots over time, (b) Hotspots labelled with tracking numbers of three objects.

For this task, a huge amount of algorithms was proposed in the past years, see for example Yilmaz *et al.* (2006) and Trucco and Plakas (2006) for a broad overview. Thus, the basic ideas used for our tracking method are not new. However, as the application is very special we could not just take one of the existing methods but had to adapt the ideas of these methods for our requirements.

This section describes our proceeding regarding the tracking of the objects, i.e., the matching of the objects to their hotspots. First we describe the developed tracking algorithms in Section 2.2.1. Section 2.2.2 describes a method to evaluate the tracking procedure. Finally, Section 2.2.3 mentions some problems that could not be solved yet.

2.2.1. Tracking algorithms

The basic idea of our tracking procedure is to assign two hotspots to the same object if the distance of their centers of gravity – in the remainder called centroids – does not exceed a certain value. This means that we go through time and compare the centroids of the hotspots at time t with those at time $t-1$ and assign them to the same object if their distance is relatively small, i.e., they are labelled with the same tracking number.

Algorithm 1 describes the basic tracking algorithm. The basic tracking algorithm can handle situations as shown in Figure 2.10. However, there are some special cases where the tracking remains unsatisfying. Below we describe these special cases along with our ideas of a solution through Algorithms 2 – 4.

Algorithm 1 Basic tracking algorithm

```
Input: $hotspots$, the hotspots resulting of the previous step as two-
    dimensional black / white images, i.e., matrices containing 1s and 0s;
    $max.dist$, the maximal distance that two centroids are allowed to have to
    be assigned to the same object (depends preferably on the time between
    two images)
1: Initialize $tracker$, a list that will contain the tracking / object number for
    each hotspot
2: Compute $matcher$, a matrix that matches the IDs of the hotspots to their
    time point $t$ and index within that time point
3: Compute the coordinates of the centroids for each hotspot and store them
    as additional columns in $matcher$
4: Each hotspot at time $t = 1$ gets a new, unique tracking number
5: $t.max \leftarrow$ Number of images
```

Algorithm 1 (continued) Basic tracking algorithm

```
 6: for $t = 2, \dots, t.max$ do
 7:   if There is at least one hotspot at time $t$ then
 8:     if There is at least one hotspot at time $t-1$ then
 9:       $cent.t \leftarrow$ centroids of hotspots at time $t$
10:       $cent.t\_1 \leftarrow$ centroids of hotspots at time $t-1$
11:       Compute distances between all centroids at time $t$ and $t-1$
12:       for $i = 1, \dots,$ number of hotspots at time $t$ do
13:         $flag.newnumber \leftarrow$ true
14:         $ID.i \leftarrow$ ID of $i$-th hotspot at time $t$
15:         for $j = 1, \dots,$ number of hotspots at time $t-1$ do
16:           $ID.j \leftarrow$ ID of $j$-th hotspot at time $t-1$
17:           if Distance of centroids of $ID.i$ and $ID.j \leq maxdist$ then
18:             Hotspot $ID.i$ gets tracking number(s) of hotspot $ID.j$
19:             {NOTE: It is possible that one hotspot gets several tracking
                numbers and that several hotspots can get the same
                tracking number at the same time point. Solutions to these
                problems are provided in the following algorithms.}
20:             $flag.newnumber \leftarrow$ false
21:           end if
22:         end for
23:         if $flag.newnumber =$ true then
24:           {NOTE: Hotspot $ID.i$ does not match to any hotspot at time $t-1$}
25:           Hotspot $ID.i$ gets a new, unique tracking number
26:         end if
27:       end for
28:     else
29:       Each hotspot at time $t$ gets a new, unique tracking number
30:     end if
31:   end if
32: end for
33: $numb.objects \leftarrow$ number of objects computed as number of distinct tracking
    numbers
34: $track.out \leftarrow$ list of $tracker, matcher$ and $numb.objects$
Output: $track.out$
```

Problems of the basic tracking algorithm and solutions

As result of the tracking procedure, each hotspot should be uniquely assigned to one single object, i.e., each hotspot should be labelled with a unique tracking number as shown in Figure 2.10 b). However, in some cases the output of the basic tracking algorithm 1 does not comply with these requirements. There are two main cases which have to be treated separately:

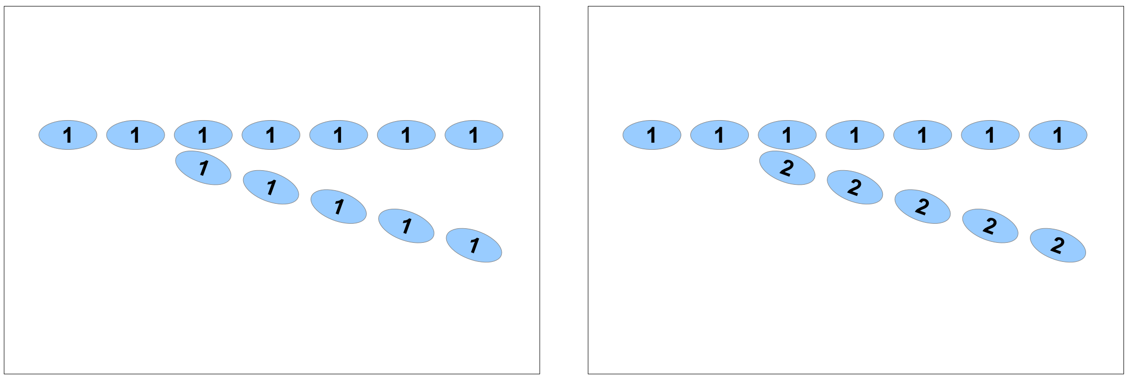
Problem 1: It is possible that two hotspots at the same time point have the same tracking number.
$\rightarrow$ Multiple hotspots per tracking number and time point

Problem 2: It is possible that one hotspot has two different tracking numbers.
$\rightarrow$ Multiple tracking numbers per hotspot

These problems are solved by Algorithms 2 – 4. The order of solving these two problems has no significant impact on the results, i.e., it does not matter if problem 1 or problem 2 is treated first. Note that in practice it can also happen that three or more hotspots at the same time point have the same tracking number or that one hotspot has three or more different tracking numbers. These hotspots are deleted because meaningful features cannot be extracted. In our data, the frequency of these cases is minimal.

Problem 1: Multiple hotspots per tracking number and time point

Using the basic tracking algorithm 1 it is possible that two hotspots at the same time point are labelled with the same tracking number. Figure 2.11 a) illustrates this problem: At time $t = 3$ appears a second object. Due to the fact that the localization of its first appearance is very close to the already existing object, the basic tracking algorithm 1 labels it with the same tracking number. Figure 2.11 b) illustrates the desired solution: The second object gets a new tracking number.



(a) Problem (b) Solution

Figure 2.11. Problem and solution: Multiple hotspots per tracking number and time point.

The basic idea of the procedure is to split the tracks if there is more than one hotspot per time point that is labelled with the same tracking number. For the tracking numbers where the problem of multiple hotspots appears we go through time and split the track beginning at the first appearance of the second object, i.e., this hotspot of the second object gets a new tracking number and the subsequent hotspots of both objects are assigned either to the old or to the new tracking number.

Algorithm 2 describes the procedure.

Algorithm 2 Split tracks with multiple hotspots

```
Input: $track.out$, the output of the basic tracking algorithm 1 (or Algorithm 3
    respectively) containing $tracker$, $matcher$ and $numb.objects$
1: for $i = 1, \ldots, numb.objects$ do
2:   $ids \leftarrow$ IDs of the hotspots with tracking number $i$ (from $tracker$)
3:   if Tracking number $i$ is assigned to $> 1$ hotspots at $> 0$ time points
4:      and Each hotspot of $ids$ is assigned to a unique tracking
5:      number then
6:     {NOTE: The case of multiple tracking numbers is treated in
       Algorithms 3 and 4.}
```

Algorithm 2 (continued) Split tracks with multiple hotspots

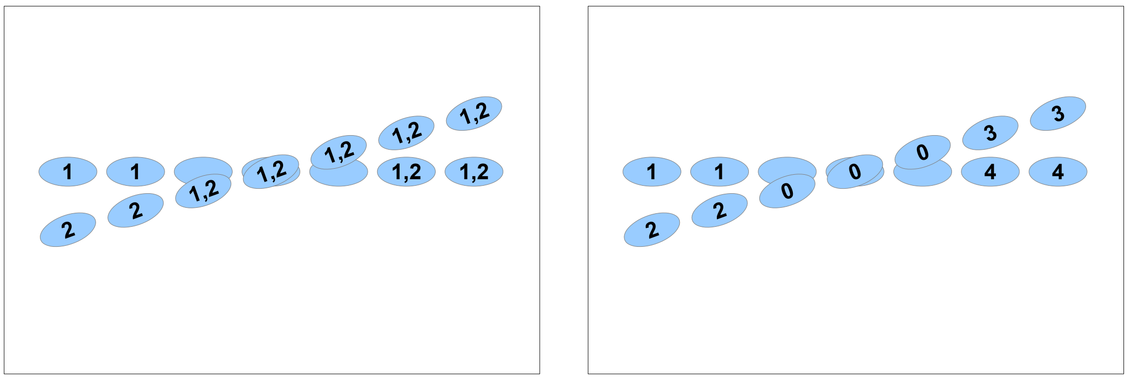
```
 7:       if At each time point $\leq 2$ hotspots are assigned to tracking
 8:       number $i$ then
 9:         {NOTE: Until now, cases with $3$ or more hotspots per tracking
            number and time are deleted}
10:         $t.min \leftarrow$ First time point where two hotspots are assigned to
            tracking number $i$
11:         $numb.objects \leftarrow numb.objects + 1$
12:         if Tracking number $i$ appears the first time at $t.min - 1$ then
13:           The hotspot at time $t.min$ whose centroid is further away from
              the centroid of the hotspot at $t.min - 1$ gets the new tracking
              number $numb.objects$, the other hotspot keeps the tracking
              number $i$
14:         else
15:           Compute a linear prediction from the centroids of the hotspots
              at time $t.min - 2$ and $t.min - 1$ for time $t.min$
16:           The hotspot at time $t.min$ whose centroid is further away from
              the prediction gets the new tracking number $numb.objects$, the
              other hotspot keeps the tracking number $i$
17:         end if
18:         $t.max \leftarrow$ time point of last appearance of tracking number $i$
19:         for $t = t.min + 1, \dots, t.max$ do
20:           if At time $t$ two hotspots are assigned to tracking number $i$ then
21:             Compute a linear prediction from the centroids of the
                hotspots at time $t - 2$ and $t - 1$ for time $t$
22:             The hotspot at time $t$ whose centroid is further away from
                the prediction gets the new tracking number $numb.objects$,
                the other hotspot keeps the tracking number $i$
23:           end if
24:         end for
25:       end if
26:     end if
27: end for
28: $track.out \leftarrow$ list of $tracker$ (updated), $matcher$ and $numb.objects$ (updated)
Output: $track.out$
```

Problem 2: Multiple tracking numbers per hotspot

Using the basic tracking algorithm 1 it is possible that one hotspot is labelled with two different tracking numbers. Here we have to distinguish two cases:

1. Crossing tracks: The tracks of two objects are crossing each other such that at time t there remains just one hotspot, an overlay of the two objects.
2. Close tracks: The tracks of two objects are at time t so close to each other that both hotspots are assigned to both tracking numbers.

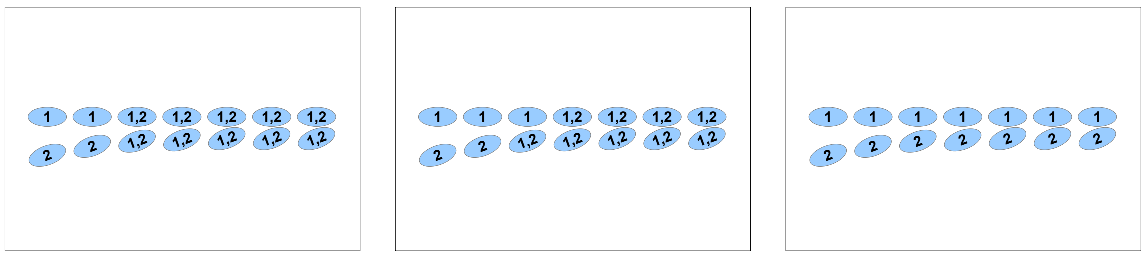
Figure 2.12 a) illustrates the first case: At time $t = 3$ the tracks of the two objects are crossing each other, at time $t = 6$ they are separated again. Due to the fact that in practice it is very difficult to match the hotspots after separation to the hotspots before the crossing, the hotspots get a new tracking number after the separation and are henceforth considered as new objects, see Figure 2.12 b). The hotspots representing the overlay of the two objects are deleted for the further analysis.



(a) Problem (b) Solution

Figure 2.12. Problem and solution: Multiple tracking numbers per hotspot - crossing tracks. The tracking number 0 means that the respective hotspot is deleted.

Figures 2.13 a) and b) illustrate the second case: At time $t = 3$ the hotspots are so close to each other that both hotspots are assigned to both tracking numbers at the following time points. It is possible that – as shown in Figure 2.13 b) – only one of the two hotspots is assigned to both tracking numbers. The solution is the same for both cases: The hotspots are assigned to the corresponding tracking numbers as in Figure 2.13 c).



(a) Problem (b) Problem (c) Solution

Figure 2.13. Problem and solution: Multiple tracking numbers per hotspot - close tracks.

Algorithms 3 and 4 describe the procedure to solve these problems.

Algorithm 3 Revise tracks with multiple tracking numbers per hotspot

```
Input: $track.out$, the output of the basic tracking algorithm 1 (or Algorithm 2
    respectively) containing $tracker$, $matcher$ and $numb.objects$
 1: $i \leftarrow 1$
 2: while $i \leq numb.objects$ do
 3:   {NOTE: The advantage of the while loop over a for loop is that
      $numb.objects$ can increase in this loop but nevertheless the new tracking
      numbers can also be analyzed}
 4:   $ids \leftarrow$ IDs of the hotspots assigned to tracking number $i$
 5:   $n.tracks \leftarrow$ Vector that contains for each hotspot of $ids$ the number of
      tracking numbers the respective hotspot is assigned to
 6:   if max$(n.tracks) = 2$ then
 7:     {NOTE: Until now it is not possible to handle hotspots that are
        assigned to three or more tracking numbers. In our data, this
        does not occur very often and we prefer deleting the whole tracks
        if occuring because these hotspots are not useful for the later
        classification problem.}
 8:     $j \leftarrow$ Tracking number of the second object
 9:     $t.min \leftarrow$ First time point when a hotspot is assigned to tracking
        number $i$ and the second tracking number $j$
10:     $t.max \leftarrow$ Last time point when a hotspot is assigned to tracking
        number $i$ and the second tracking number $j$
11:     $flag.overlap \leftarrow$ false
12:     while $t \in [t.min, t.max]$ do
13:       {NOTE: The advantage is that we can leave the loop after
          assigning new tracking numbers}
14:       Run Algorithm 4
15:     end while
16:   end if
17:   $i \leftarrow i + 1$
18: end while
19: $track.out \leftarrow$ list of $tracker$ (updated), $matcher$ and $numb.objects$ (updated)
Output: $track.out$
```

Algorithm 4 Helper of Algorithm 3, to run only within that algorithm

```
1:  $id.t \leftarrow$ IDs of hotspots detected at time $t$ (from $ids$)
2:  if length($id.t$) = 1 then
3:      {NOTE: Overlapping objects $\rightarrow$ Hotspot is deleted}
4:      Set tracking number of this hotspot to $0$ (use $tracker$)
5:      $flag.overlap \leftarrow$ true
6:  else
7:      if One hotspot $h1$ is only assigned to tracking number $i$ and the other
8:          hotspot $h2$ to both tracking numbers $i$ and $j$ then
9:          {NOTE: Case from Figure 2.13 b) at $t = 3$}
10:         Hotspot $h2$ gets only the other tracking number $j$
11:     end if
12:     if Both hotspots are assigned to tracking numbers $i$ and $j$ then
13:         if $flag.overlap =$ false then
14:             {NOTE: Case from Figure 2.13 a)}
15:             if Only one time point prior to $t$ with tracking number $i$ then
16:                 The hotspot at time $t$ whose centroid is further away from the
                    centroid of the hotspot corresponding to object $i$ at $t - 1$ gets
                    tracking number $j$, the other hotspot keeps tracking number $i$
17:             else
18:                 Compute a linear prediction from the centroids of the hotspots
                    corresponding to object $i$ at times $t - 2$ and $t - 1$ for time $t$
19:                 The hotspot at time $t$ whose centroid is further away from
                    the prediction gets tracking number $j$, the other hotspot keeps
                    tracking number $i$
20:             end if
21:         else
22:             {NOTE: Case from Figure 2.12 a)}
23:             $k \leftarrow numb.objects + 1$
24:             $l \leftarrow numb.objects + 2$
25:             $numb.objects \leftarrow l$
26:             Set tracking number of one hotspot to $k$, the other to $l$ and all
                following to $(k, l)$ {NOTE: will be treated later}
27:             $t \leftarrow t.max$ {NOTE: stop while loop}
28:         end if
29:     end if
30: end if
31: $t \leftarrow t + 1$
```

2.2.2. Evaluation of the tracking procedure

After performing all tracking steps it is of interest to evaluate the tracking procedure and its algorithms also with the aim of error search. Therefore we implemented an algorithm that computes per video:

- the number of tracks
- the length of each track, i.e., the number of hotspots per track
- a flag for the problem of multiple hotspots per tracking number and time point
- a flag for the problem of multiple tracking numbers per hotspot

The basic idea of the algorithm is to go through all tracking numbers and decide whether there is one of the mentioned problems or not. Algorithm 5 describes the procedure.

Note that even after performing all tracking steps not all cases are solved. This is due to the fact that until now we do not solve cases where hotspots are assigned to three or more tracking numbers and where tracking numbers are assigned to three or more hotspots at the same time point.

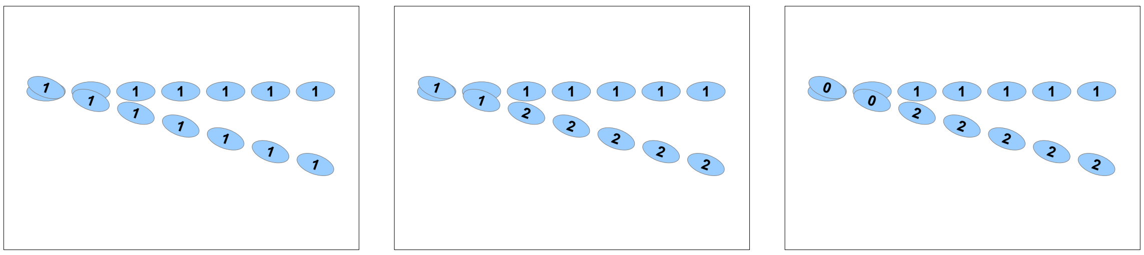
The results of the evaluation for our analysis were the following: There was only one case where a tracking number was assigned to three hotspots at the same time point and only three cases where hotspots had three tracking numbers. Therefore, there was no need to develop solutions for these cases. These cases were deleted before the next step because it turned out that they are not useful for the later classification: In cases where three or more objects overlap or where three or more hotspots are so close that they are labelled with the same tracking number, meaningful features cannot be derived. For the results of the number and length of tracks identified for our data, see Table 4.1 on page 57 where each object stands for one track and the average number of hotspots per object reflects the average track length.

Algorithm 5 Evaluation of the tracking procedure

```
Input: $track.out$, the output of the tracking algorithms containing $tracker$,
    $matcher$ and $numb.objects$
 1: Initialize $track.eval$, an output matrix with $numb.objects$ rows and $4$ columns
 2: Initialize $object.ids$, an output list with $numb.objects$ elements
 3: for $i = 1, \dots, numb.objects$ do
 4:   $ids \leftarrow$ Vector that contains the IDs of hotspots with tracking number $i$
 5:   $n.tracks \leftarrow$ Vector that contains for each ID of $ids$ the number of
      objects the hotspot is assigned to
 6:   if max$(n.tracks) > 1$ then
 7:     {NOTE: Multiple tracking numbers per hotspot}
 8:     $problem \leftarrow$ true
 9:     $prob.num \leftarrow 1$
10:   else
11:     if There is a time $t$ where $i$ has more than one hotspot then
12:       {NOTE: Multiple hotspots per tracking number and time}
13:       $problem \leftarrow$ true
14:       $prob.num \leftarrow 2$
15:     end if
16:   end if
17:   $count.hotspots \leftarrow$ length$(ids)$
18:   {NOTE: Number of hotspots that are assigned to object $i$}
19:   $i$-th row of $track.eval \leftarrow (i, count.hotspots, problem, prob.num)$
20:   $i$-th element of $object.ids \leftarrow$ Matrix with two columns: $ids$ and
      corresponding time points
21: end for
Output: $track.eval, object.ids$
```

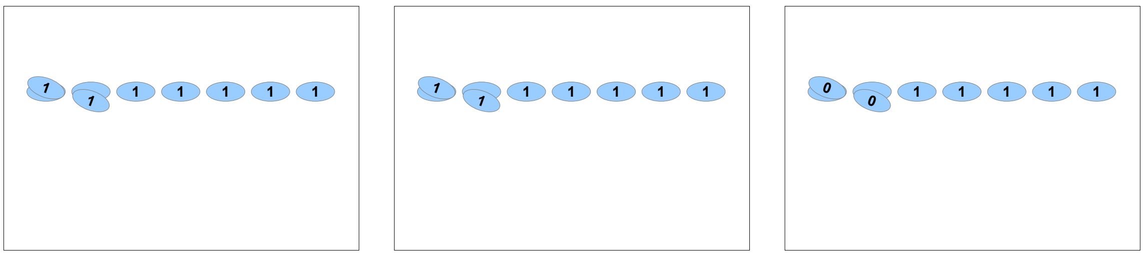
2.2.3. Open challenges

There are some problems that can occur within the tracking procedure which are not solved by the algorithms described in this section. These problems are shown together with the current and the desired solution in Figures 2.14, 2.15 and 2.16. Due to the fact that these problems rarely occur in our data, we did not develop solutions for these problems. Depending on the application for which the tracking procedure shall be used, one could think about implementing solutions to these problems.



(a) Problem (b) Current solution (c) Better solution

Figure 2.14. Problem: Two objects start overlapped to one hotspot, later they are separated.



(a) Problem (b) Current solution (c) Better solution

Figure 2.15. Problem: Two objects start overlapped to one hotspot, one object disappears prior to the separation of the two objects.

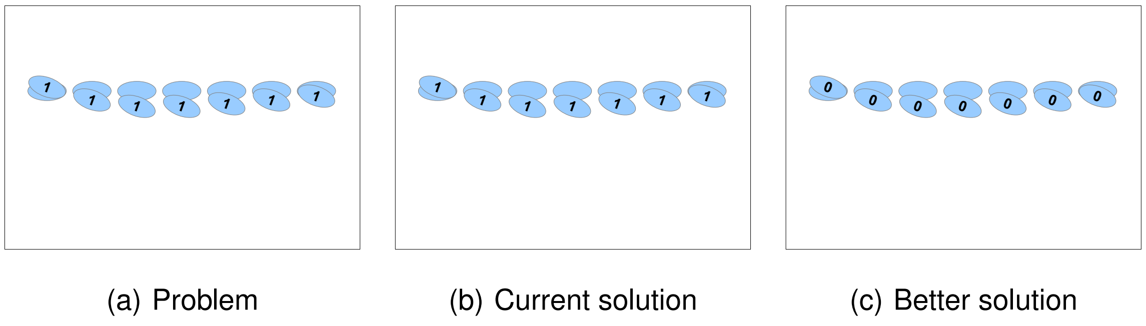


Figure 2.16. Problem: Two objects overlap at all time points.

Chapter 3: Feature extraction

In Chapter 2 we preprocessed the sonar video so far that for each object we have a set of tracked hotspots, i.e., small images containing the relevant pixel clouds. Based on these hotspots we want to extract features which allow to discriminate the objects and classify them into the three classes *eel*, *other fish* and *debris*.

We proceed as follows: Until now, each object consists of a certain number of hotspots. We first compute variables on hotspot level which form the basis for the construction of variables on object level, see Section 3.1. In the *Baseline* variables, these variables are simply summarized on object level to capture the rough size of the objects, see Section 3.2.1. In the *Shape* variables, we make use of methods from the field of functional data analysis to represent the shape of the objects, see Section 3.2.2. In the *Motio*n variables, we derive variables which describe motion features such as swimming direction and velocity of the objects, see Section 3.2.3. Finally, in Section 3.3, we state further ideas for discriminating variables which seemed to be useful but did not improve the classification results further.

3.1. Hotspot level

For each hotspot we compute the following variables:

1. *Centroid*: The cartesian coordinates of the center of gravity of the hotspot.
2. *Area of the hotspot*: Due to the special geometry of the DIDSON image, we cannot just count the pixels of a hotspot to determine the area of this hotspot. Pixels that are further away from the lense stand for a larger area because the distance between two pixels increases with increasing distance from the lense, see Figure 1.2 on page 11 for an illustration. Instead of just counting the pixels, we compute for each pixel the area one pixel represents. By summing up the areas of all pixels of the hotspot we get the area of the hotspot.

Watch hands

The next step is to capture the silhouette of the fish which is done by constructing a circular silhouette function in angle $\alpha \in [0, 2\pi)$. For angle α we compute $x(\alpha)$, the distance from the centroid of the hotspot to the contour of the hotspot, which we call "watch hand" function subsequently. (For different representations of the shape of the hotspots see the discussion on page 70.)

To do so, we first need to orientate each fish with respect to its swimming direction so that the watch hands of a fish are standardized, see Figure 3.1. Hence, the value for $\alpha = 0$, namely $x(0)$, always stands for the distance between centroid and head of the fish and $x(\pi)$ always stands for the distance between centroid and tail of the fish. This orientation of the silhouette is done as follows: First, the main body axis of the fish is determined using a linear regression of the y- on the x-coordinates of all fish pixels, i.e., the y-coordinate of a pixel is considered as response variable while the x-coordinate is the explanatory variable. (For the definition of y- and x-axis see for example Figure 1.1.) Second, the location of the head of the fish is determined using all hotspots of this fish. Looking at the centroids of the same fish at all time points, we find its swimming direction. Given that the fish is swimming forwards, we thereby have the information where the head is. We will make use of the so constructed silhouette functions throughout this chapter. We additionally define the following variables:

3. *Watch hands*: For a grid of angles $0 = \alpha_1 < \cdots < \alpha_M < 2\pi$ we calculate the watch hands $x(\alpha_1), \ldots, x(\alpha_M)$.
4. *Length*: Length of the fish, i.e., sum of the watch hands $x(0) + x(\pi)$.
5. *Width*: Width of the fish, i.e., sum of the watch hands $x(\pi/2) + x(3\pi/2)$.
6. *Aspect ratio*: Aspect ratio of the fish, i.e., ratio of *Length* and *Width*.
7. *Product*: Product of *Length* and *Width*. This mirrors the area of the fish but measured as a rectangle.

Data structure

The resulting data matrix of constructed variables on hotspot level is denoted by $\boldsymbol{X}_{hot}$ having dimension $n \times (M + 7)$ where $n = \sum_{i=1}^{N} n_i$ with n_i as number of hotspots for the i-th object, N as number of objects and $M = 72$ watch hands are used in our analysis.

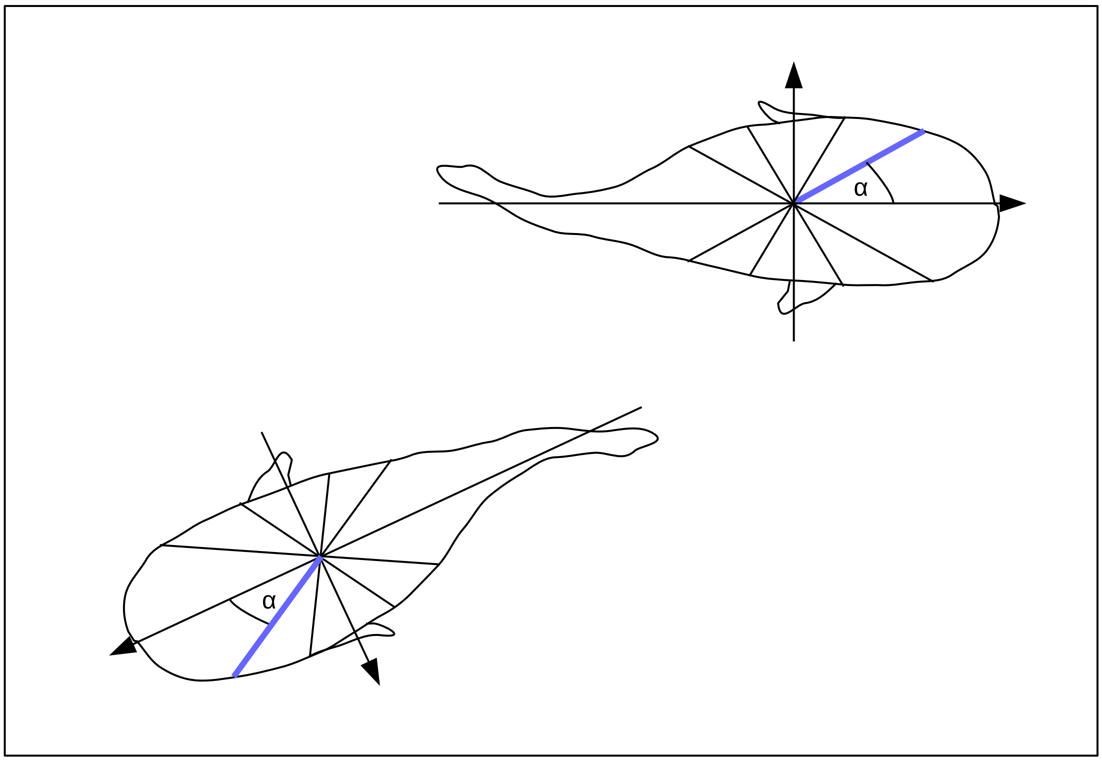


Figure 3.1. Illustration of the watch hands. A watch hand is the distance from the centroid to the fish contour. Recall that we see the fish from the top.

3.2. Object level

Each hotspot image is now represented by the quantities above leading to $M + 7$ variables (M for the watch hands, two for the cartesian coordinates of the centroid and one for the area of the hotspot, length, width, aspect ratio and product, respectively). As mentioned above, a sequence of hotspots are images of the same object. Since we are interested in the classification of objects we combine hotspot variables to object variables, leading to a classification data set consisting of N rows, one row per object. To do so, each n_i rows of $\boldsymbol{X}_{hot}$ corresponding to all hotspots of the i-th object are aggregated to one row. The remainder of this section presents different ways of aggregating the information on object level and thereby defines sets of classification variables.

3.2.1. Baseline variables

To extract information about the average size of an object we compute the *mean* of the *Length*, *Width*, *Aspect ratio*, *Product* and *Area* for each object. The resulting matrix of classification variables is denoted by $Z_{Baseline}$ having dimension $N \times 5$.

While the mean of these variables is a natural choice, it would also be possible to compute other functions to combine the hotspot variables on object level, for example the minimum, maximum, median or other quantiles. However, our analyses showed that computing the mean is optimal in terms of classification accuracy.

3.2.2. Shape variables

We already extracted the information about the size of the object in the *Baseline* variables and are now primarily interested in the shape of the object. To extract information about the shape of an object we further exploit the watch hand functions defined above. Let therefore $x_{ij}(\alpha)$ denote the watch hand function of the j-th hotspot of object i. The watch hand functions are recorded at a finite grid of angles $\alpha_1, \dots, \alpha_M$. Note that fish of the same species can be of different sizes while the shape is the same. In the *Shape variables* we are not interested in the size of the object but only in the shape. We therefore standardize the watch hand functions such that the sum equals one and denote the standardized watch hand functions with $x_{ij}^s(\alpha)$, i.e., $x_{ij}^s(\alpha) = \frac{x_{ij}(\alpha)}{\sum_{t=1}^M x_{ij}(\alpha_t)}$.

Model

The watch hand functions are noisy due to the blurred image, so that we decompose them stochastically as follows. Let

$$x_{ij}^s(\alpha) = y_{ij}(\alpha) + \varepsilon_{ij}(\alpha), \tag{3.1}$$

where $y_{ij}(\alpha)$ is a smooth silhouette function and $\varepsilon_{ij}(\alpha)$ is an error term. This error term captures both, the measurement error of the sonar camera as well as the error due to the fact that the watch hands are calculated based on pixeled images only.

The next step is to decompose the smooth silhouette function $y_{ij}(\alpha)$ further by setting

$$y_{ij}(\alpha) = m(\alpha|g(i)) + \tilde{\varepsilon}_{ij}(\alpha) \tag{3.2}$$

where $g(i)$ stands for the class of object i, that is $g(i) \in G = \{eel, trout, debris\}$. Equation (3.2) means that $y_{ij}(\alpha)$ is composed of a class-specific mean function $m(\alpha|g(i))$ of the object's class $g(i)$ and its deviation $\tilde{\varepsilon}_{ij}(\alpha)$. Note that $m(\alpha|g)$ gives the mean silhouette of object class g (eel, trout or debris).

The final step is to look at the error term $\tilde{\varepsilon}_{ij}(\alpha)$, which can be explained as silhouette or shape variation, corrected for the mean shape of the class. We approximate $\tilde{\varepsilon}_{ij}(\alpha)$ using a Karhunen-Loève approach (see e.g. Dony, 2001) and set

$$\tilde{\varepsilon}_{ij}(\alpha) = \underbrace{\sum_{k=1}^{K} z_{ijk} v_k(\alpha|g(i))}_{f_{ij}(\alpha|g(i))} + \tilde{\tilde{\varepsilon}}_{ij}(\alpha) \tag{3.3}$$

where K gives the degree of approximation and z_{ijk} are random variables with zero mean and decreasing order of variance in $k = 1, \ldots, K$. Functions $v_k(\alpha|g)$ are class-specific functional silhouette components which reflect the possible shapes of objects in class g. The remaining error term $\tilde{\tilde{\varepsilon}}_{ij}(\alpha)$ is considered to be unstructured and not further explored.

Class-specific estimation of model components

For the construction of classification variables we need to estimate the components $m(\alpha|g)$ and $v_k(\alpha|g), k = 1, \ldots, K$ in (3.2) and (3.3) separately for each class $g \in G$. This means, we estimate one model for eels, one model for trouts and one model for debris. To do so we first smooth the standardized watch hand functions $x^s_{ij}(\alpha)$ for all hotspots using a Fourier basis which provides us with cyclic estimated silhouette functions $\widehat{y}_{ij}(\alpha)$. This is illustrated in Figure 3.2 for a set of standardized watch hand functions $x^s_{ij}(\alpha)$. We use a Fourier basis with 65 basis functions to represent the $M = 72$ discrete measures of the watch hands. Note that this high number does not force very smooth watch hand functions, as can be seen in Figure 3.2. Essentially, we transform the discrete measures of the watch hands into functional objects, see for example Ramsay and Silverman (2005) for details on this approach.

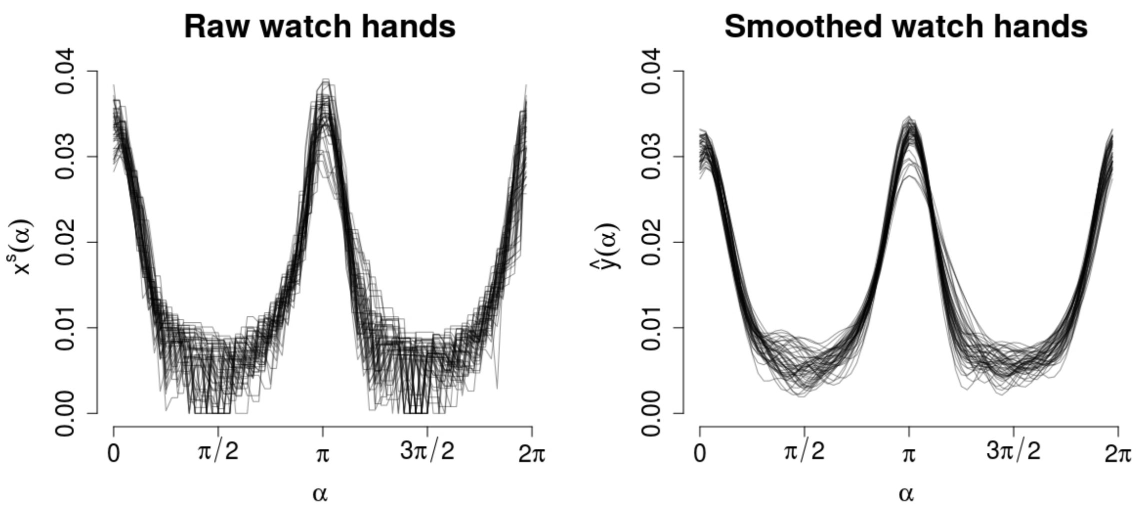


Figure 3.2. Raw standardized watch hand functions (left plot) and smoothed watch hand functions (right plot) for a randomly chosen eel. One line stands for one hotspot. Horizontal axis: Angle α, vertical axis: Length of the corresponding watch hand, see Figure 3.1 for an explanation.

The next step is to estimate a class-specific mean function which is done by simple averaging over all smoothed watch hand functions from all objects of a given class, i.e.,

$$\widehat{m}(\alpha|g) = \frac{1}{n_g} \sum_{i \in I_g} \sum_{j=1}^{n_i} \widehat{y}_{ij}(\alpha) \quad \text{for} \quad g \in G, \tag{3.4}$$

where I_g denotes the index set of all objects of class g and $n_g = \sum_{i \in I_g} n_i$ is the total number of hotspots of these objects. The estimated class-specific mean functions $\widehat{m}(\alpha|g)$ for the three classes eel, trout and debris are shown in Figure 3.3.

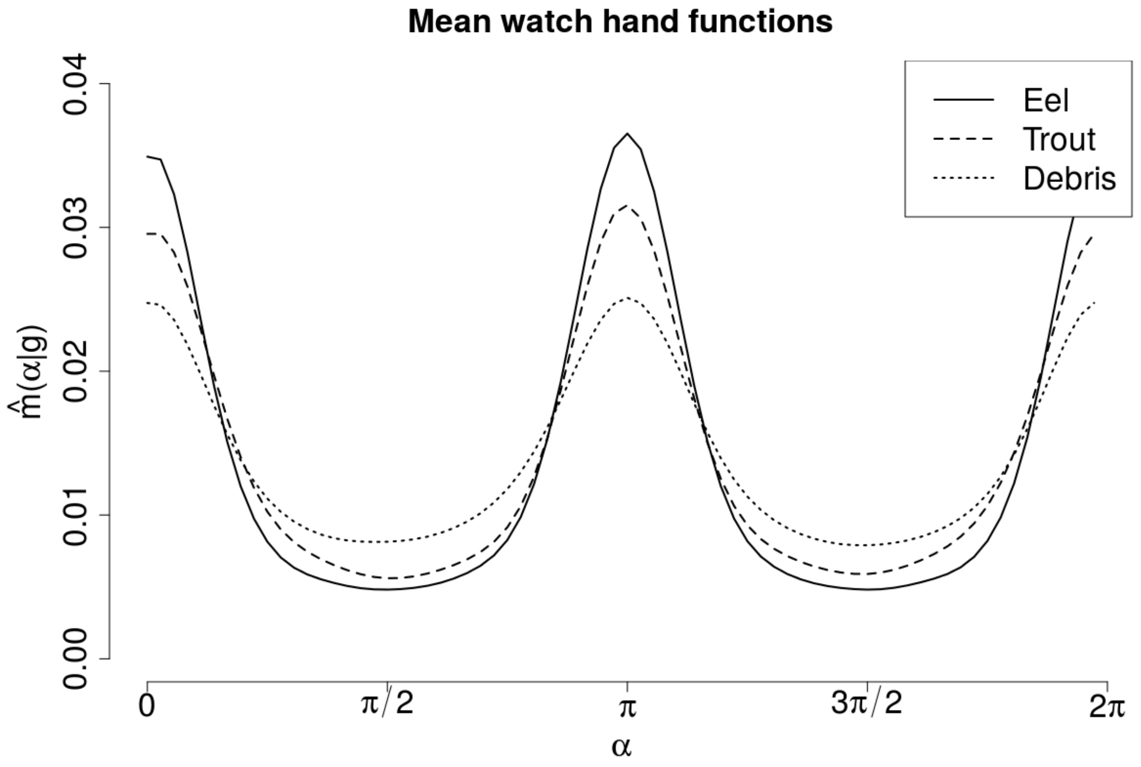


Figure 3.3. Class-specific mean watch hand functions $\widehat{m}(\alpha|g)$ for the three classes eel, trout and debris, computed as given in Equation (3.4).

The fitted residuals $\hat{\hat{\varepsilon}}_{ij}(\alpha) = \widehat{y}_{ij}(\alpha) - \widehat{m}(\alpha|g)$ are directly available and stored in an $n \times M$ matrix where $n = \sum_{i=1}^{N} n_i$ is the total number of watch hand functions. Now we can estimate the class-specific silhouette components $v_k(\alpha|g)$ for each class assuming and fitting smooth, circular and orthonormal eigenfunctions via functional principal component analysis (fPCA). As above, we use 65 Fourier basis functions with an additional smoothing parameter of $\lambda = 10$, which showed to be optimal in terms of classification accuracy, forcing smooth principal components, see Ramsay and Silverman (2005) for details on this estimation approach. We make use of $K = 3 < M$ eigenfunctions covering 57.1% of the overall variability in the data for eels, 46.6% for trouts and 61.2% for debris, respectively. These estimations are based on the R-packages `fda` (Ramsay *et al.*, 2013) and `fda.usc` (Febrero-Bande and Oviedo de la Fuente, 2012).

The estimated class-specific silhouette components $\hat{v}_k(\alpha|g)$ are shown in Figure 3.4, where an alternative way of visualization is used: In Figures 3.2 and 3.3, the polar coordinates – i.e., the angle α on the horizontal axis and the length of the watch hand on the vertical axis – are directly plotted. In contrast, in Figure 3.4, the polar coordinates are retransformed into cartesian coordinates – i.e., this visualization represents a top view as shown throughout Chapter 2. The dashed lines represent the class-specific mean watch hand functions $\widehat{m}(\alpha|g)$ as shown in Figure 3.3, the solid lines represent the sum of $\widehat{m}(\alpha|g)$ and a multiple of the respective estimated class-specific silhouette components, i.e., $\widehat{m}(\alpha|g) + a_{k,g}\hat{v}_k(\alpha|g)$, for $g \in G$ and $k = 1, 2, 3$, where as factor $a_{k,g}$ the 95%-quantile of the scores on the respective component was chosen to achieve reasonable plots.

We see differences between the classes: The aspect ratio of eels is larger than for trouts and debris and the aspect ratio of trouts is larger than for debris, this can best be seen in Figure 3.3, but also in the solid lines of Figure 3.4. The first silhouette component of all three classes adjusts the aspect ratio, see the first column of Figure 3.4. The second and third silhouette components of eels and trouts show typical swimming movements of fish whereas the second and third silhouette components for debris seem to reflect rather random shapes. The amount of variance explained by the debris components suggests that the first component is much more important than the other components for this class which is not the case, to this extent, for eels and trouts. This reflects the fact that the shape of dead objects does not fundamentally change over time whereas the shape of living objects changes due to movement activity.

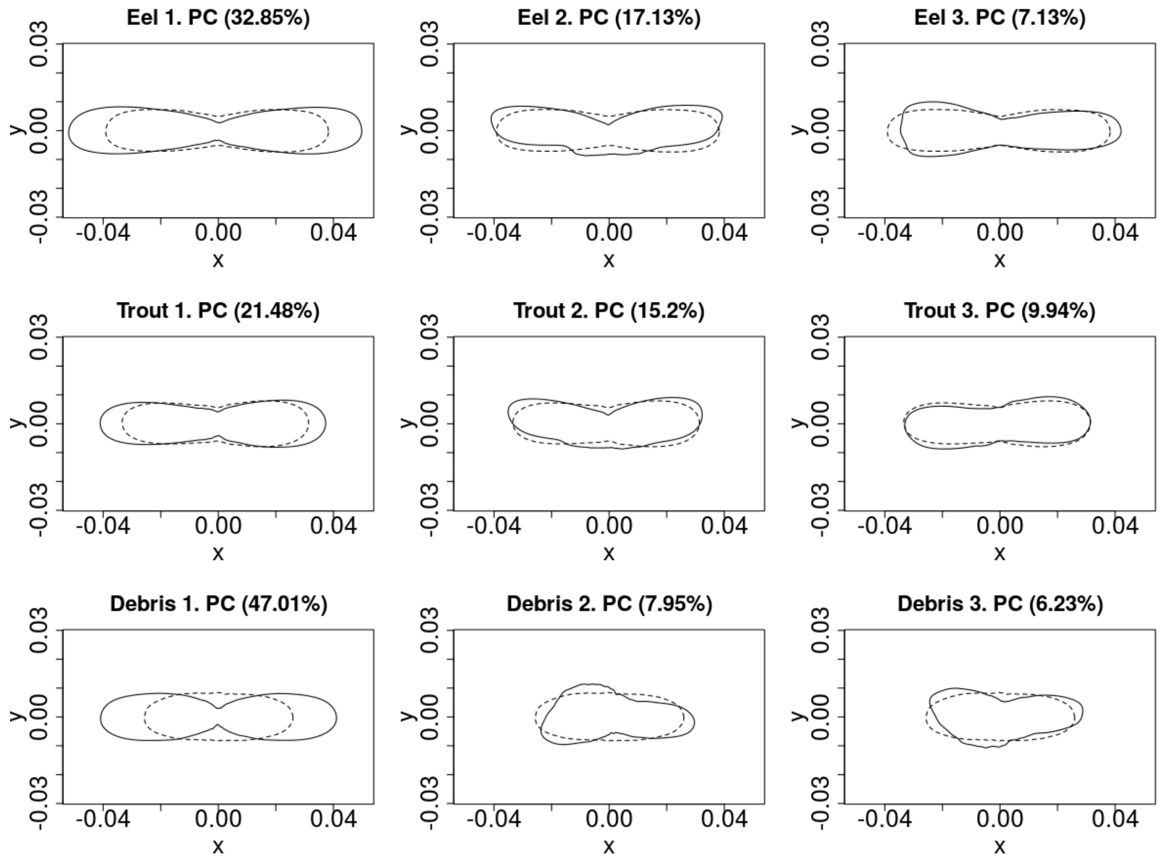


Figure 3.4. Effect of the first three class-specific silhouette components of eels, trouts and debris on the respective class average shape. The dashed lines give the class-specific mean watch hand functions $\widehat{m}(\alpha|g)$ while the solid lines show the sum of $\widehat{m}(\alpha|g)$ and a multiple of the respective class-specific silhouette component $\hat{v}_k(\alpha|g)$. The numbers in brackets give the amount of variance explained by the respective principal component. In the rows from top to bottom: eels, trouts, debris. In the columns from left to right: First, second and third principal component.

Prediction for a new object

How can we use these class-specific model estimations to construct classification variables? We assume that the dataset is divided into training and test cases and that estimates $\widehat{m}(\alpha|g)$ and $\hat{v}_k(\alpha|g)$ for all $g \in G$ are based on the training part only. For a new object i of the test cases with unknown class $g(i)$ we first compute the smooth silhouette functions $\widehat{y}_{ij}(\alpha)$ using a Fourier basis as described above for all hotspots $j = 1, \ldots, n_i$ of the new object i.

Assuming the new object to come from class g we obtain class-specific fitted residuals

$$\hat{\hat{\varepsilon}}_{ij}(\alpha|g) = \widehat{y}_{ij}(\alpha) - \widehat{m}(\alpha|g) \quad \text{for} \quad g \in G.$$

Next we calculate predictions for the residual component in (3.3) through

$$\begin{aligned}\hat{\hat{\hat{\varepsilon}}}_{ij}(\alpha|g) &= \hat{\hat{\varepsilon}}_{ij}(\alpha|g) - \sum_{k=1}^{K} \hat{z}_{ijk}\hat{v}_k(\alpha|g) \\ &= \widehat{y}_{ij}(\alpha) - \widehat{m}(\alpha|g) - \sum_{k=1}^{K} \hat{z}_{ijk}\hat{v}_k(\alpha|g) \quad \text{for} \quad g \in G,\end{aligned}$$

where $\hat{z}_{ijk} = \hat{\boldsymbol{v}}_k^\top(\cdot|g)\hat{\hat{\boldsymbol{\varepsilon}}}_{ij}(\cdot|g)$ and $\hat{\hat{\boldsymbol{\varepsilon}}}_{ij}(\cdot|g) = (\hat{\hat{\varepsilon}}_{ij}(\alpha_1|g), \ldots, \hat{\hat{\varepsilon}}_{ij}(\alpha_M|g))^\top$ and analogous definition for $\hat{\boldsymbol{v}}_k(\cdot|g)$. Note that $\hat{z}_{ijk}$ apparently depends on the assumed class g which is suppressed in the notation for simplicity.

Classification variables

The idea for the construction of classification variables is the following: If a given hotspots belongs to an object from class g, the prediction of the shape of this hotspot through a model with estimations based on class g should be better than the predictions through models with estimations based on the other classes. Hence, the residuals $\hat{\hat{\varepsilon}}_{ij}(\alpha|g)$ and $\hat{\hat{\hat{\varepsilon}}}_{ij}(\alpha|g)$ should be rather small for the true class and higher for the other classes.

Therefore, for each hotspot j of object i we compute the L_2-norms of the residual components

$$\begin{aligned}d^1_{ij}(g) &= ||\hat{\hat{\varepsilon}}_{ij}(\alpha|g)||_2 \quad \text{and} \\ d^2_{ij}(g) &= ||\hat{\hat{\hat{\varepsilon}}}_{ij}(\alpha|g)||_2 \quad \text{for} \quad g \in G, j = 1, \ldots, n_i.\end{aligned}$$

For object i with a number of n_i hotspots, we average $d^1_{ij}(g)$ and $d^2_{ij}(g)$ on object level resulting in six classification variables

$$d^1_i(g) = \frac{1}{n_i}\sum_{j=1}^{n_i} d^1_{ij}(g) \quad \text{and} \qquad (3.5)$$

$$d^2_i(g) = \frac{1}{n_i}\sum_{j=1}^{n_i} d^2_{ij}(g) \quad \text{for} \quad g \in G. \qquad (3.6)$$

The variables corresponding to Equation (3.5) are stored in the matrix $\boldsymbol{Z}_{ShapeMean}$ and the variables corresponding to Equation (3.6) are stored in the matrix $\boldsymbol{Z}_{ShapeVariation}$, each having dimension $N \times 3$, i.e., one row per object and one column per class.

At this point we would like to argue why we use only $K = 3$ eigenfunctions: The more eigenfunctions we use, the better becomes the approximation of the residual shape $\hat{\varepsilon}_{ij}(\alpha|g)$. Hence, the residual component $\hat{\hat{\varepsilon}}_{ij}(\alpha|g)$ becomes small for each of the three classes and the differences between the three class approximations vanish resulting in a poor classification power of the so constructed variables $d_i^2(g)$. Thus, with the aim of classification, the number of eigenfunctions should not be too large. Our analyses showed that a number of $K = 3$ eigenfunctions provides the best classification rate.

3.2.3. Motion variables

The features proposed so far are invariant with respect to the permutation of hotspots, i.e., we do not account for the time chronology of the hotspots. Therefore, we propose to additionally exploit the path of an object's centroid. Figure 3.5 shows the path of the centroids for the example trout used for the presentation of the preprocessing methods in Chapter 2.

We may construct features based on such paths: First, we compute the *distances* between temporally adjacent centroids of a given object. Second, we compute the *angles* between the line connecting temporally adjacent centroids and the y-axis. For both *distances* and *angles*, we get $n_i - 1$ values for an object i with n_i hotspots. From these values, we construct four classification variables:

1. *Mean velocity*: Mean of the *distances* per object
2. *Mean moving direction*: Mean of the *angles* per object
3. *Variability of velocity*: Standard deviation of the *distances* per object
4. *Variability of moving direction*: Standard deviation of the *angles* per object

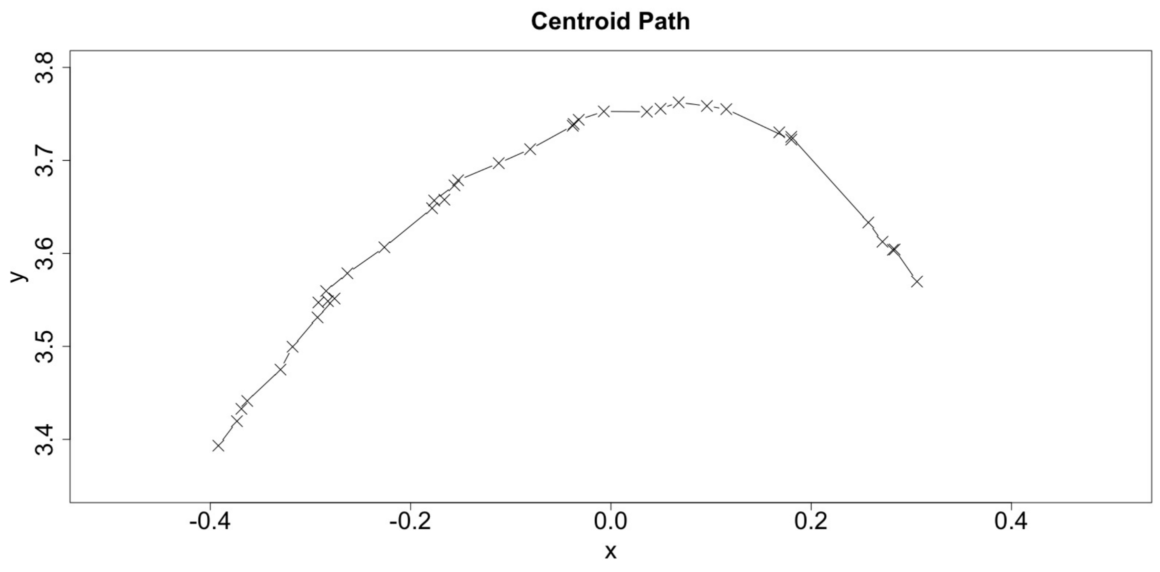


Figure 3.5. Path of a trout's centroid moving in an arc from left to right. Crosses stand for centroids of hotspots over time.

The idea of using these variables is as follows: Since debris can only swim with the current we can expect that the *mean velocity* and the *mean moving direction* as proxies for speed and swimming direction are very specific for this class. Additionally, the variability in speed and swimming direction should be very small and thus the standard deviations of *distances* and *angles* should be small. Moreover, one can see in the example videos that the movement of eels is more monotone than the movement of trouts. Thus, the variability in speed and swimming direction should be larger for trouts than for eels and therefore these variables are promising for the discrimination of dead vs. living objects as well as for the discrimination of eels vs. trouts. The resulting variable matrix is denoted by $\boldsymbol{Z}_{Motion}$ and has dimension $N \times 4$.

3.3. Summary and further ideas

To summarize, in the previous steps we generated four different feature matrices $\boldsymbol{Z}_{Baseline}$, $\boldsymbol{Z}_{ShapeMean}$, $\boldsymbol{Z}_{ShapeVariation}$ and $\boldsymbol{Z}_{Motion}$. These matrices share the property that they have N rows, i.e., they contain one row vector of variables for each object. In total we constructed $5+3+3+4 = 15$ variables which will be used for classification.

We delevoped further ideas which took as a starting point the mean watch hand functions shown in Figure 3.3. These ideas, which will be sketched in the following, lead to further classification variables. However, our analyses showed that for the data at hand, the classification accuracy cannot be improved by including these variables. Therefore we do not include them in the results in Chapter 4. Nevertheless we think that these variables could be useful for other data sets because they capture the class information in a different manner than the above variables. Thus, we shortly describe them in the following.

Mean functions

We look again at the smoothed watch hand functions $\widehat{y}_{ij}(\alpha)$ from Section 3.2.2. The idea for the construction of the *Shape mean* variables in Equation (3.5) is basically the same as the following idea: Compute the mean smoothed watch hand function of a given object and compare it with the three class means using an L_2-norm. This leads to three classification variables similar to the *Shape mean* variables.

Variance and autocorrelation functions

From that point of view, a natural extension comes to our mind: We can compute a variance function instead of the mean function for each object and compare it with the three class variance functions. Figure 3.6 (left) shows the mean variance functions for the three classes. They apparently vary strongly between the classes and can be interpreted as follows: Eels show the highest variation in the watch hand functions, most of all for the watch hands with angles around 0 and π representing the head and tail of the fish. This is reasonable because the aspect ratio of eels is larger than for the other classes and hence the distance between head and tail can vary stronger due to swimming movements. (Remind that we are working with the standardized watch hand functions here, i.e., no information about the absolute size of the objects is present.) The variation of the watch hand functions for debris is very small which goes well with the fact that this class only contains dead objects. For dead objects we cannot expect much variation in the shape. Nevertheless, the variation is not zero because debris is swimming with the current and thus for example rotating while passing by the sonar camera.

Analogously we can compute the autocorrelation functions to lag 1 for each object and compare it with the three class autocorrelation functions to lag 1. Figure 3.6 (right) shows the mean autocorrelation functions to lag 1 for the three classes. They apparently vary strongly between the classes, too. The advantage of using the autocorrelation functions is: Both mean and variance functions do not consider the time chronology of the hotspot functions. This means, permutation of time points does not change these classification variables which implies a loss of information. For the autocorrelation functions however, the order in time matters. The autocorrelation functions shown in Figure 3.6 (right) have a reasonable interpretation: The high autocorrelations for eels mean that the shape varies rather slowly over time whereas the shape of trouts seems to vary rather fast. On the other hand, the autocorrelation for debris is nearly zero which means that there is hardly information in the time chronology of debris. For dead objects, this has to be expected, because only living objects should show temporally recurring movement patterns.

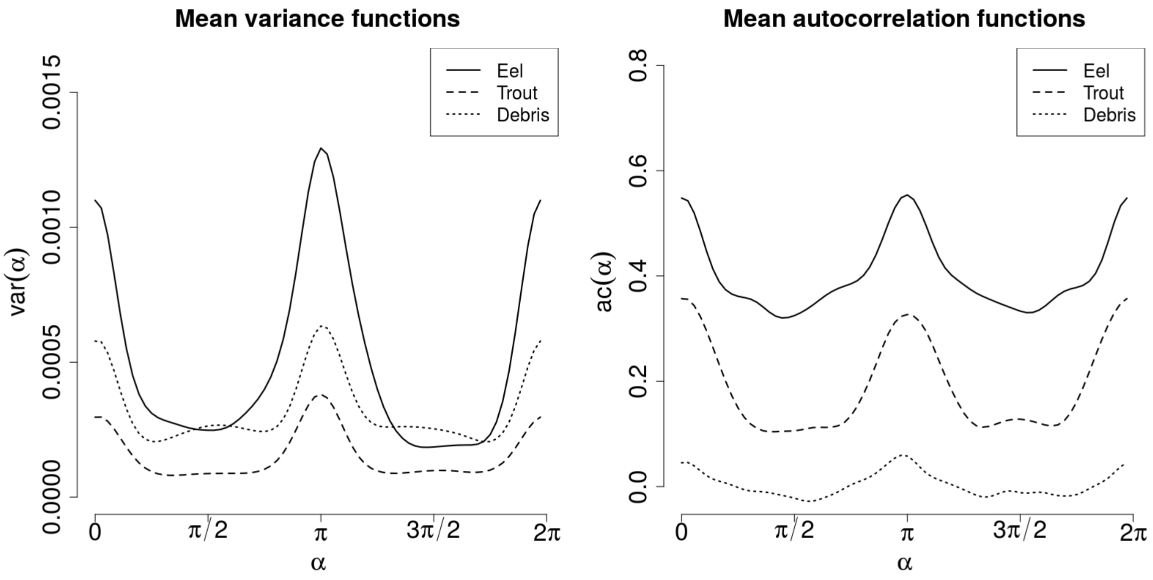


Figure 3.6. Mean variance (left) and autocorrelation (right) functions of the watch hand functions per class.

Derivatives of the watch hand functions

Now that we are looking at the watch hands from a functional data point of view, we can additionally inspect the derivatives of the watch hand functions. Figure 3.7 shows the mean function of the first derivatives of the smoothed watch hand functions for each class.

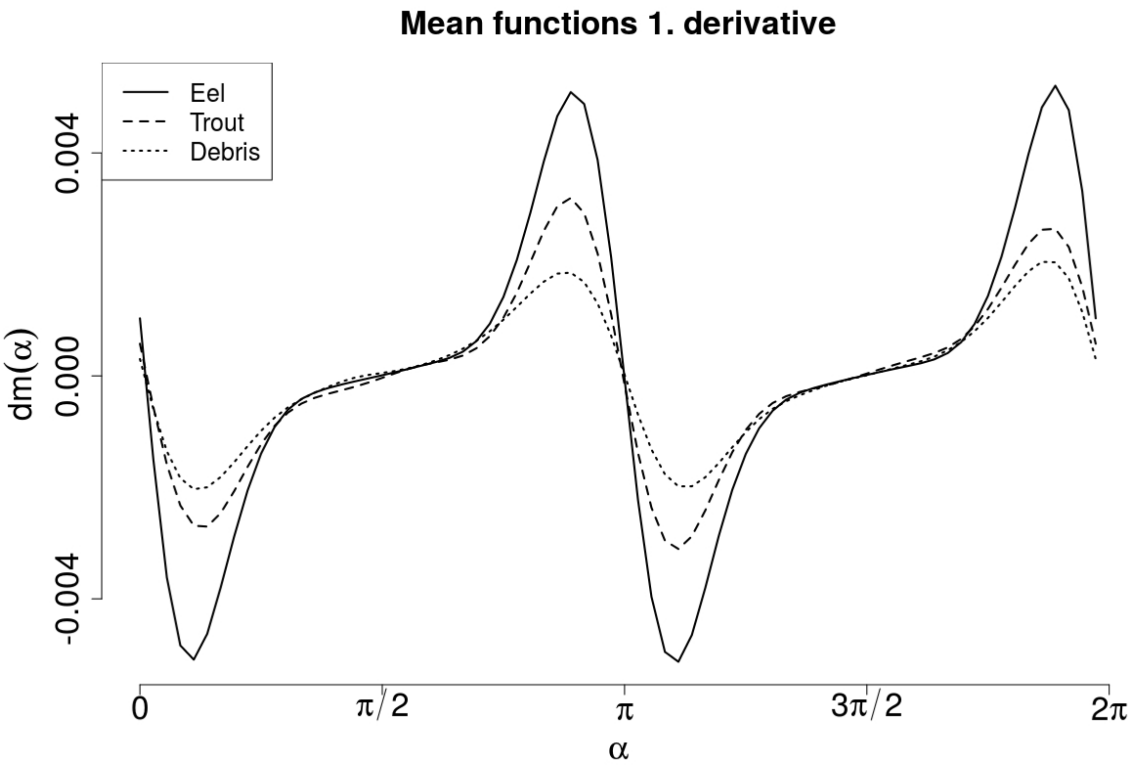


Figure 3.7. Mean functions of the first derivative of the watch hand functions per class.

Similarly to the ideas given above, we can compute the variance and autocorrelation functions of the first derivatives. Figure 3.8 shows the mean variance and the mean autocorrelation functions of each class.

Using mean, variance and autocorrelation function we can construct further nine variables computing L_2-distances between mean, variance and autocorrelation functions of the considered object and the means of this functions for the three classes.

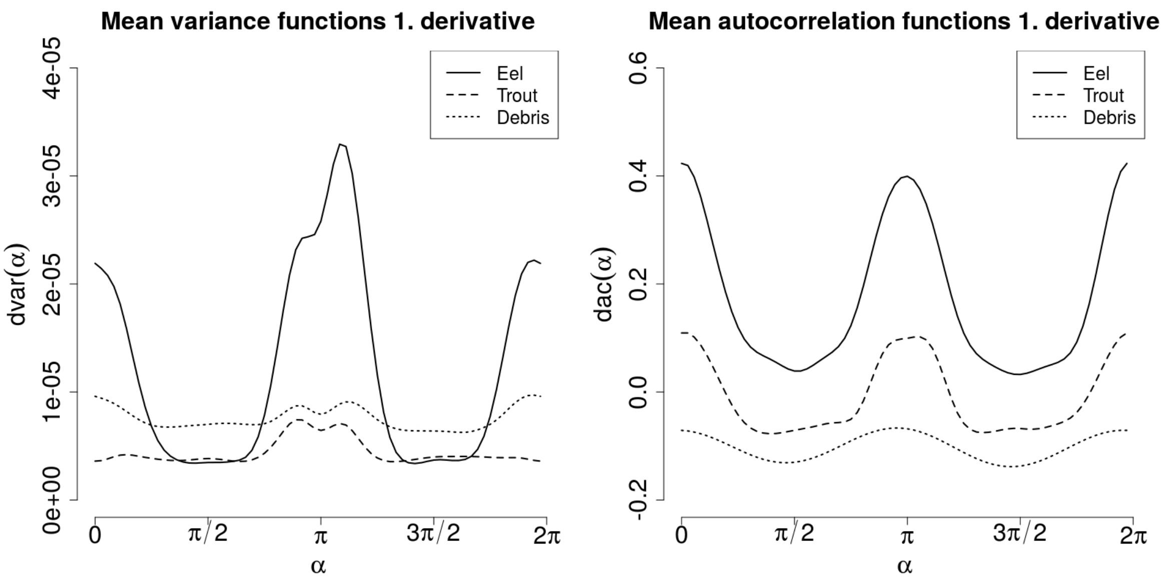


Figure 3.8. Mean variance (left) and autocorrelation (right) functions of the first derivative of the watch hand functions per class.

However, although these ideas seem quite convincing, for our data they could not improve the classification accuracy. Apparently, the information captured by these variables is already included in the other variables.

Chapter 4: Classification and results

In Chapter 3 we finally completed the preprocessing of the videos by extracting features for each object. With these features we can now classify the objects detected in the sonar videos, decide which combination of features classifies best and evaluate the classification accuracy of our approach.

Figure 4.1 visualizes the work flow of the steps necessary to conduct the classification and to derive the misclassification rates. First, on the given data basis of sonar videos all the preprocessing steps of Chapter 2 are carried out. Second, the data set of detected objects is splitted into training and test data. Third, the features of the detected objects are computed as explained in Chapter 3: For the *Shape mean* and *Shape variation* variables, the model components of Equations (3.2) and (3.3) are estimated using the training data only. With these estimations, the features for training and test data are computed. (For the *Baseline* and *Motion* variables, no model estimations are necessary, these variables can be directly computed.) Based on the features extracted from the training data, a classification rule is learned with classification methods stated below. Then, all objects of the test data are classified with the learned classification rule with respect to the three classes *eel*, *trout* and *debris*. Finally, the misclassification rate in the test data is computed to evaluate the classification procedure.

The classification steps (learning and testing) are carried out several times with different settings and each setting is evaluated by subsampling bootstrap with 1000 iterations. On the one hand, the features used for the classification are varied, on the other hand, the classification method is varied. The aim is to find the best combination of classification variables and classification method which can be used for the application of the system in practice. Details on the different settings are given in Section 4.1

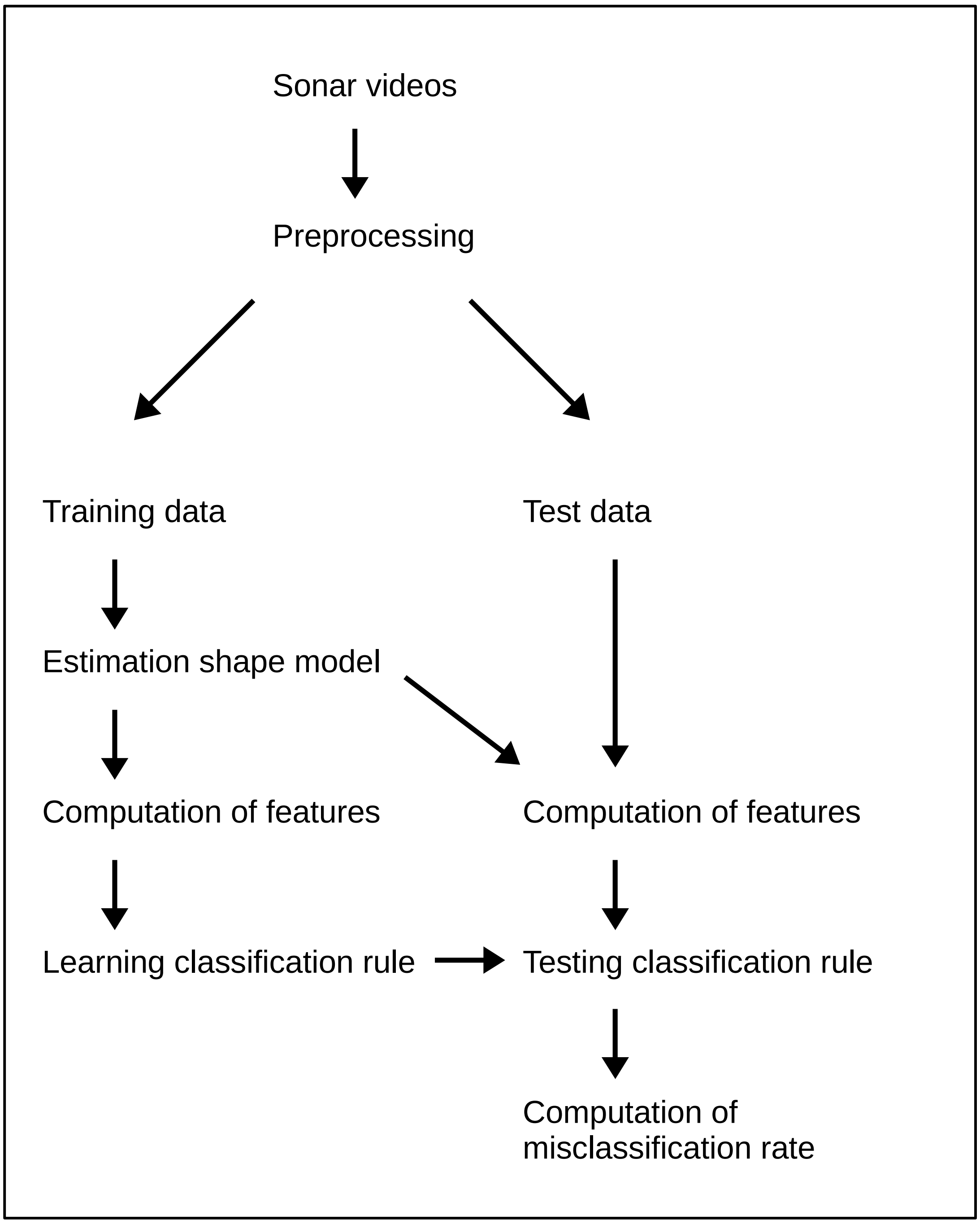


Figure 4.1. Work flow for the classification procedure and the computation of misclassification rates.

4.1. Data and methods

We apply the proposed methods to three sonar videos introduced in Section 1.2 on page 13. After the preprocessing steps, we deleted all objects which could only be seen for a short time or which were rather small. The reason is that for such objects meaningful features cannot be extracted. An object was deleted

- if the object stayed in the focus of the camera less than half a second, i.e., less than five images, or
- if the mean length of the object's hotspots did not exceed 10 cm.

After this step, 134 eels, 414 trouts and 166 pieces of debris, i.e., 714 objects remained in the final data set. Table 4.1 summarizes basic properties of the videos.

Table 4.1. Properties of the videos used for the analysis.

Video	Duration	No. of images	No. of objects	Average no. of hotspots per object
Eels	11 min	6600	134	25.4
Trouts	13 min	7800	414	18.5
Debris	30 min	18000	166	10.8

We perform a subsampling bootstrap with $B = 1000$ iterations for the estimation of the misclassification rates of different models and classification methods as described in Figure 4.1. One bootstrap iteration starts with the splitting into training and test data, the preprocessing has to be conducted only once. In each iteration $b = 1, \ldots, B$ we draw randomly (without replacement) 80% of the $N = 714$ objects and consider them as training data. The remaining objects are considered as test data. On the training data we compute all the variables proposed in Chapter 3, i.e., we generate matrices $\boldsymbol{Z}_{Baseline}$, $\boldsymbol{Z}_{ShapeMean}$, $\boldsymbol{Z}_{ShapeVariation}$ and $\boldsymbol{Z}_{Motion}$. With those variables we compute the respective variables of the test data. Combining these design matrices we generate 15 different models for the classification. Table 4.2 lists the different components included in the classification models.

Table 4.2. Composition of the 15 considered models. 1 indicates that the respective variables are included in the model, 0 indicates that the respective variables are not included in the model.

Model	Baseline	Shape mean	Shape variation	Motion
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1
11	1	1	0	1
12	0	0	1	1
13	1	0	1	1
14	0	1	1	1
15	1	1	1	1

For each model we use as classification methods linear discriminant analysis (lda), quadratic discriminant analysis (qda), multinomial logit model (mnl), trees (tree, see for example Breiman *et al.*, 1984), support vector machines (svm, see for example Cortes and Vapnik, 1995) and random forests (rf, see for example Breiman, 2001).

4.2. Results

Figure 4.2 shows the mean misclassification rates resulting from the subsampling bootstrap for all combinations of the 15 models and the 6 classification methods.

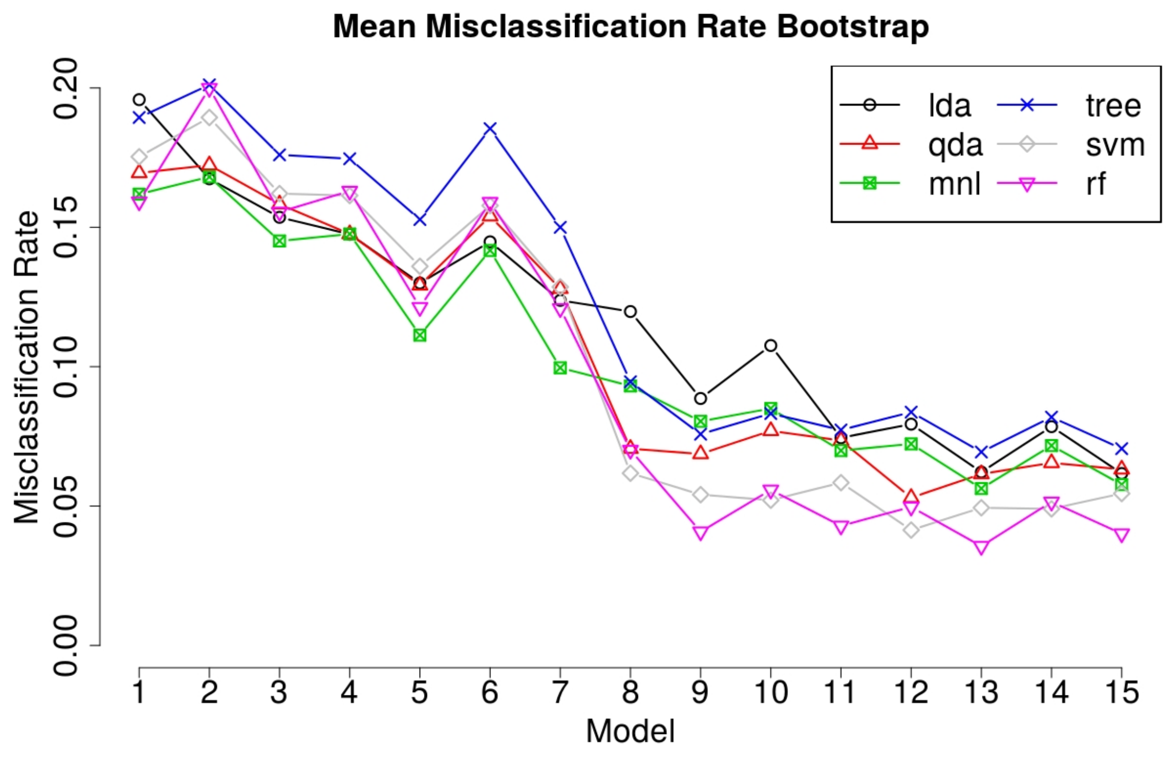


Figure 4.2. Mean misclassification rates of subsampling bootstrap with 1000 iterations for different models and classification methods.

Figure 4.2 suggests that including the *Motion* variables (models 8 to 15) clearly improves the classification. For these models support vector machines and random forests outperform the competing classification methods. The lowest misclassification rate is obtained using random forests with model 13, i.e., *Baseline*, *Shape variation* and *Motion* variables where only 3.6% of the objects are misclassified on average. Closest competitor to model 13 is model 15. To finally decide which combination of model and method predicts best, Figure 4.3 shows the estimated distributions of the misclassification rates for model 13 and 15 for all considered classification methods. These results support the prior impression in favoring model 13 classified with random forests. Note that even though this is the best combination of classification variables and classification method, the misclassification rates of other combinations are close which means that there are no significant differences between the best models. For a different set of sonar videos, a different combination could be optimal. Nevertheless, for our data model 13 classified with random forests is the optimal choice in terms of out-of-sample classification accuracy.

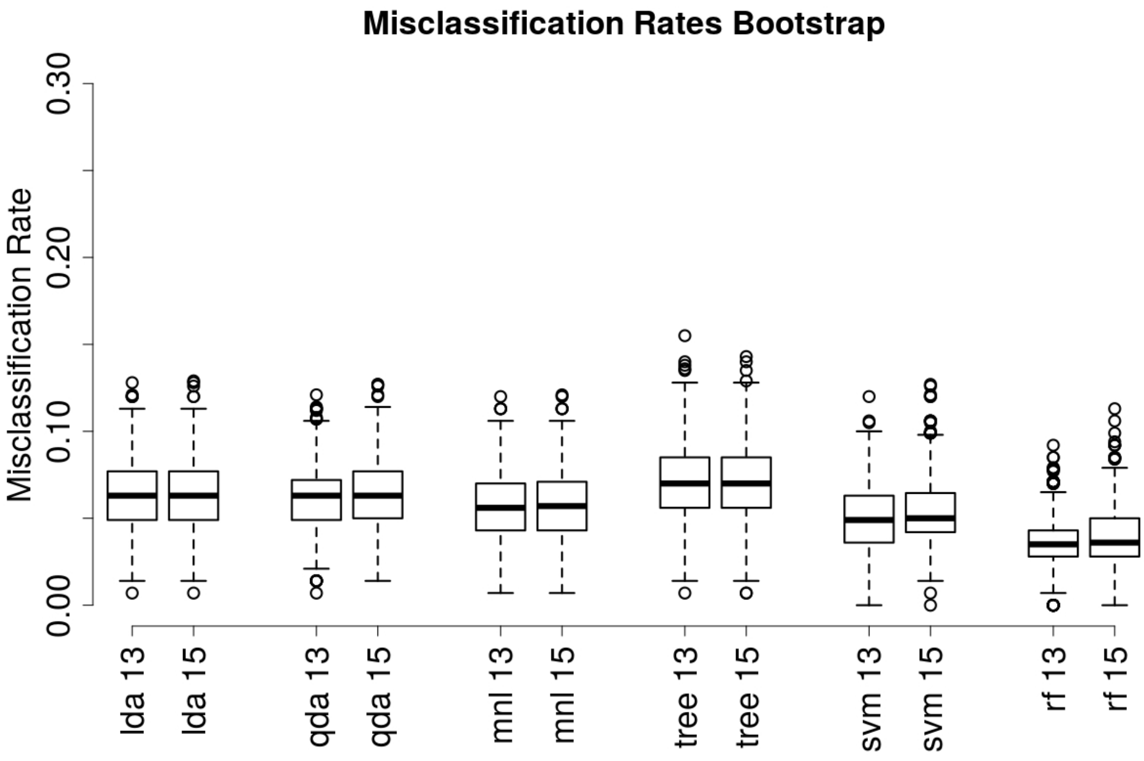


Figure 4.3. Distribution of the misclassification rates estimated by subsampling bootstrap with 1000 iterations. Comparison of models 13 and 15 with all considered classification methods.

To finally evaluate the classification accuracy of our procedure, we take a closer look into the misclassification pattern of model 13 using random forests. Table 4.3 shows the misclassification pattern resulting from a 5-fold cross validation.

Table 4.3. Misclassification pattern for model 13 with random forests and 5-fold cross validation, absolute values and row percentages are shown.

	Prediction			
Truth	Eel	Trout	Debris	Total
Eel	122 91.0%	11 8.2%	1 0.7%	134 100%
Trout	5 1.2%	408 98.6%	1 0.2%	414 100%
Debris	0 0.0%	6 3.6%	160 96.4%	166 100%
Total	127	425	162	714

Misclassification mainly appears between the two fish classes: The misclassification rate between eel and trout is 2.9% while the misclassification rate between fish and debris is just 1.1%. The overall misclassification rate is 3.4%. Hence, even though classification of fish carries some error, our routine performs promising to detect fish and living objects in general in the videos.

We finally state that model 13, containing *Baseline*, *Shape variation* and *Motion* variables is the best predicting model. Note that these three components in total only consist of twelve variables, i.e., we compressed the information in the data very efficiently. This compression of information was achieved through a careful construction of the classification variables: The *Baseline* variables capture the rough size and dimensions of the objects. The *Shape variation* variables reflect class-specific possible shapes which go beyond the mean shape of the objects. Finally, the *Motion* variables capture the class-specific movement patterns of eels, trouts and debris.

The *Shape mean* variables do not improve the classification accuracy further which means that the information of these three variables is apparently contained in the other twelve variables. There is a reasonable explanation for this result: As shown in Chapter 3 in Figure 3.4, the mean shape of all three classes is approximately an ellipse where the difference between the classes lies mainly in the aspect ratios of these ellipses. But, the aspect ratio of the objects is already included in the *Baseline* variables and the aspect ratio is a feature which does not change with the standardization of the watch hand functions. (Remind that for the *Baseline* variables, the unstandardized watch hand functions are used whereas for the *Shape* variables the standardized watch hand functions are used.)

As an interim conclusion, we can state that we have reached the first two of our four research goals defined in Chapter 1 on page 8: We have developed a system which is able to deliver a sensible count of fish present in the video and to distinguish between eels and other fish. Note again that although the category *other fish* is restricted to trouts in our analysis, the results can be generalized to other fish species as well because most fish species found in European rivers look like trouts from above.

Chapter 5: Implementation and application in practice

While Chapter 4 has shown that the first two research goals, the accurate counting and classification of fish (see page 8), were reached by our approach, the present chapter is concerned with the other two research goals regarding computational issues: The system has to run in realtime and to ensure high usability in order to be suitable for the usage at water power plants. *Realtime* means in this context that the computing time shall be less than the running time of the video. Additional details on the implementation in R and on the application of the system in practice are given here.

5.1. Implementation and computing time

Implementation

As one part of this thesis, the developed methods for preprocessing, feature extraction and classification were implemented in R (R Core Team, 2016) with one exception: Only the flood-fill algorithm, used for the extraction of hotspots in Chapter 2, is outsourced into C++ to ensure low computing times. Apart from the base packages in R, the following packages were used:

Preprocessing:

- `hexView` (Murrell, 2014) for directly reading in the sonar videos (.ddf).

Feature extraction:

- `fda` (Ramsay *et al.*, 2013) for most of the functional data analysis methods used in Section 3.2.2 and
- `fda.usc` (Febrero-Bande and Oviedo de la Fuente, 2012) for selected functional data analysis methods used in Section 3.2.2, above all the derivatives of the watch hand functions.

Classification:

- `e1071` (Meyer *et al.*, 2014) for support vector machines,

- `MASS` (Venables and Ripley, 2002) for linear and quadratic discriminant analysis,
- `nnet` (Venables and Ripley, 2002) for the multinomial logit model,
- `randomForest` (Liaw and Wiener, 2002) for random forests and
- `tree` (Ripley, 2015) for trees.

R package

The entire implementation is written as user-friendly and generic as possible: First, some parameters have to be specified such as the threshold a used for the cleaning of the images or the number of cores if parallelization is desired. Then, the analysis runs and displays – additionally to the final classification table – various plots and videos which might be useful for the user. Of course, the user can specify which output shall be generated. The resulting R package `sonar` is available on R-Forge at `https://r-forge.r-project.org/projects/sonar/`, see Appendix B for an excerpt of the documentation.

Computing time

With the proposed methods for preprocessing, feature extraction and classification we are able to process whole sonar videos and to count objects with respect to the three categories *eel*, *trout* and *debris*. The three videos used for our analysis have a total running time of 54 minutes. In order to avoid stack overflow, we split each video in parts of one minute, i.e., 600 images. Once the classification rule is learned, an entire analysis of all videos, i.e., preprocessing of the videos with localization of hotspots and tracking, feature extraction and classification of detected objects lasts about 36 minutes using a MacBook Pro Intel Core i7 2 GHz, 8 GB RAM from the year 2011. Note that we can easily parallelize the analysis and thus reduce the computing time further by using multiple cores. For example, using two cores on the same device the computing time is about 22 minutes and thus less than half of the videos' running time. This means that we also achieved our third goal – running in realtime.

5.2. Application in practice

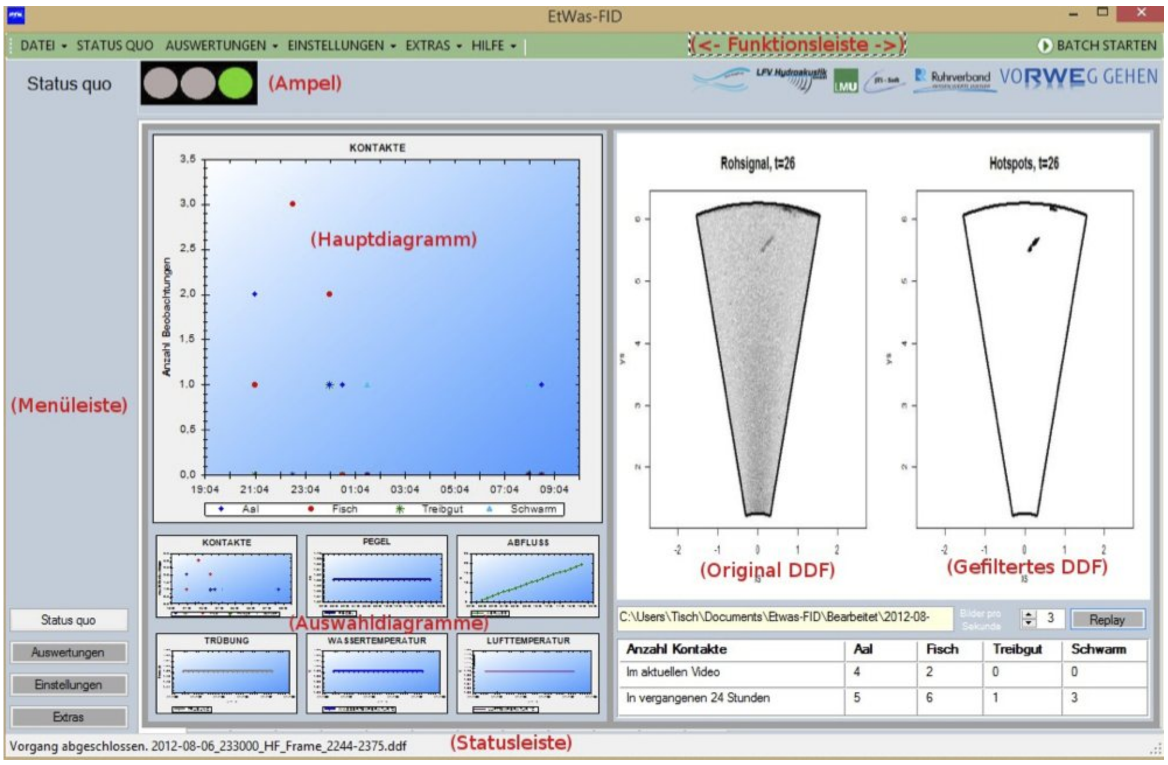
User interface

The final purpose of our project was to provide a software which can be used for fish counting and classification at a water power plant during operation. And of course, no matter how well implemented or how fast an R package is – the usability in practice suffers because few practitioners are trained in using R. Since this task is not part of our statistical work, we worked together with a computer scientist (Joachim Tischbierek from jTi-Soft, Gütersloh, Germany) who implemented a user interface for our R package. With this user interface, the staff at the water power plant is able to use our system autonomously and without additional training in R. The user can read in sonar videos and analyze them automatically with a few mouse clicks. As alternative, the online modus can be selected – then a continuous stream of sonar videos is analyzed. In any case, as output a table as shown in Table 1.2 on page 12 is generated. Supplemented by descriptive analyses, the amount and species of fish in front of the turbine can be easily assessed. Figure 5.1 shows a screenshot of the user interface.

Application steps

Calibration of the system: If our program shall be used at a water power plant, the system has to be calibrated first. This means, the tuning parameters of the preprocessing steps such as the number of B-spline basis functions, the threshold a etc. have to be optimized for the given location. Then, the best combination of the extracted features and the best classification method have to be found and with this optimal combination, the classification rule has to be learned. The easiest way to achieve this is to put fish of known species in the water, to record a video for each species and then to derive the optimal classification rule as explained in Chapter 4 and visualized in Figure 4.1. It is necessary to learn a rule for each new location because rivers and thus fish behavior vary strongly. For example the direction and strength of the current can be very different for different rivers and this information is important for the classification, most of all for the discrimination of dead and living objects.

For our example data, the *Baseline*, *Shape variation* and *Motion* variables were selected and random forests were used as classification method. For



Anzahl Kontakte	Aal	Fisch	Treibgut	Schwarm
Im aktuellen Video	4	2	0	0
In vergangenen 24 Stunden	5	6	1	3

Vorgang abgeschlossen. 2012-08-06_233000_HF_Frame_2244-2375.ddf

Figure 5.1. Screenshot of the user interface, provided and approved for publication courtesy of jTi-Soft, Gütersloh, Germany

a new site, the analysis presented in Chapter 4 can be replicated and other sets of variables and another classification method could be chosen.

Ongoing operation: After calibration, the system at the water power plant works as follows: First, one minute of the sonar video is recorded. While the next minute of video is recorded, the present video is analyzed by the system, i.e., the preprocessing steps with the optimized tuning parameters are performed, the selected features are extracted and finally the objects are classified with the learned classification rule. This takes on average 30 seconds (this time varies with the amount of fish in front of the camera) which means that the system detects a fish not later than 1.5 minutes after its first appearance. For practical purposes, this can be considered as *realtime* because fish in front of a turbine stay there for some time even up to several hours. Thus, measures for the protection of the fish can be taken in time.

To sum up, we reached also our third and fourth research goal: We provide a software which is easy to use at the water power plant and fast enough for the planned purpose.

Chapter 6: Discussion and outlook

6.1. Summary and research goals

Summary

In this project, we were faced with a very complex problem of applied statistics: Given a sequence of underwater sonar images, i.e., a sonar video, we had to develop a software which allows to detect, count and classify fish of different fish species and which is fast and user-friendly enough for the application at a water power plant.

Therefore, we first preprocessed the videos in order to locate pixel clouds on each image which most likely represent a fish and called them hotspots. Then, we tracked those hotspots which belong to the same object over time. From these sets of tracked hotspots, we extracted features which allow to discriminate living objects from dead objects and eels from other fish. In the results section we showed the high classification accuracy of the system using statistical classification methods. A careful and thorough implementation in the statistical software package R ensures low computing times and high usability for practitioners.

Research goals

In the introduction on page 8 we defined four main research goals of the project which were achieved as follows:

1. *The software should deliver a sensible count of fish present in the video:* After several steps of preprocessing and feature extraction, we can finally divide the found objects into living and dead objects with a very high accuracy leading to a sensible count of fish.

   However, although the classification into living and dead objects proved to be very accurate, we cannot be sure that we found all objects present in the video. It could happen that an object is overlooked by the preprocessing and thus cannot be classified into living or dead. Until now we have no satisfying tool to check whether the amount of overlooked objects is high or not. All we can do ad hoc is viewing the videos and counting the number of objects manually. Then we can compare this number with the number resulting from our automated analysis. Doing so, we see that for the data treated in this thesis, the amount of overlooked objects is minimal. However, this is no guarantee that in other data the performance will be equally satisfying. Another limitation is that a fish which leaves the focus of the sonar camera and later returns into the focus will be counted twice. To the best of our knowledge, there is no possibility to solve this problem.

2. *The software should distinguish between eels and other fish:* The results in Chapter 4 show that with the proposed features, the classification accuracy is very high. Due to the fact that we can additionally interpret the features well and explain why these features should discriminate eels and other fish, we are very confident that the performance will be comparable for sonar videos at other locations.

3. *The software should run in realtime:* This goal was achieved. The implementation was optimized with regard to computing time and error handling. Thus, the software is suitable for the application at a water power plant.

   We would like to emphasize that the realization of such low computing times is one of the substantial achievements of this project: At each step of the analysis, low computational complexity was a key factor when choosing the methods. Furthermore, the concrete implementation in R and C++ pursued the goal of ensuring low computing times on the one hand and robust error handling on the other hand. The second point is of great importance because when applying the system in practice, many and perhaps unexpected exceptional cases have to be dealt with without resulting in a system crash. A pilot study at a water power plant showed the robustness of our system with other data as well.

4. *The software should have a high usability for practitioners not familiar with R:* Thanks to the cooperating computer scientist, we can also provide a user-interface for our R package `sonar` (https://r-forge.r-project.org/projects/sonar/). Thus, no knowledge of R and not even of statistics are necessary to use our software in practice.

6.2. Methodological alternatives

At several points of our analysis, alternative approaches would have been possible. We discuss these alternatives in the order of the analysis steps and justify our own choices.

Orientation of hotspots (Section 3.1, Page 40)

For the orientation of the hotspots we proceed in two steps:

- We find the main body axis using a linear regression of the y- on the x-coordinates of all pixels of the hotspot. This approach could fail in the case where a fish swims from the top to the bottom of the image (or from bottom to top), because the slope would be infinity then. But this does not happen in the given videos. Additionally, if a fish would swim from top to bottom there would be another problem: The sonar camera emits sonar waves from bottom to top resulting in a poor image quality when a fish swims exactly with or against the direction of the waves. An alternative to linear regression is principal component analysis (PCA). The result would be more or less the same in most cases but the problem with a slope of infinity does not exist. The reason why we decided in favor of linear regression and against PCA to find the main body axis was: Consider a case where the hotspot is more or less circular. Using a PCA, the angle of the main body axis with the x-axis could be anything between 0 and π, potentially only depending on a few pixels. In contrast, the default of the regression line is an angle of 0. Hence, the linear regression is more stable in these cases.
- We find the head of a fish using all hotspots of this fish at all time points: Looking at the centroids at all time points we extract the information if the fish swims from right to left or from left to right. Assuming that the fish is swimming forwards, we know which side is the head and which side is the tail. This approach could fail if a fish changes its swimming direction or is swimming backwards. A visual inspection of the analyzed data showed that this does not occur in our data.

Representation of the shape of the hotspots (Section 3.1, Page 40)

As representation of the shape of the hotspots, we use the watch-hand function which is also known as radius-vector function in the literature, see for example Stoyan and Stoyan (1994). Alternative shape representations would have been possible. We discarded simpler representations such as the cross-section or support functions because they are constructed for symmetric or convex shapes, respectively. Competitors to the radius-vector function were the contour parametric or contour complex function and the tangent-angle function, see Stoyan and Stoyan (1994) and Kindratenko (2003). These functions do not require star-shaped figures and are hence alternatives for the radius-vector function which is constructed for star-shaped figures. After careful consideration, we decided us in favor of the radius-vector approach and against the alternatives because

- it is simple both to compute (complexity of implementation and computing time) and to explain it to the practitioners,
- the *Baseline* variables are computed straightforward from the radius-vector function. For the alternatives, the calculations would have been far more complicated (complexity of implementation and computing time),
- it suits very well with our approach for the orientation of the figures. Because once the angle of the main body axis is determined, all we have to do is to translate the original angles of the radius-vector function by this constant angle.

However, the radius-vector functions require star-shaped figures and not all hotspots are star-shaped. We handle two exceptions as follows:

- Non-star-shaped figures: In the case where more than one contour point is available for a given angle, we choose the furthermost point from the centroid to define the length of the respective watch hand. The idea was here that we do not want to underestimate the length of the fish, especially for eels, which tend to not being star-shaped. If we chose the nearest point to the centroid we would run the risk of “not seeing” the tail of the fish resulting in an underestimation of the length. Choosing the furthermost point from the centroid results in a slight smoothing for non-star-shaped figures. However, this smoothing is not very strong and, as implicated by the results, no major problem. If there would be a large amount of extremely non-star-shaped figures in another application of our method, one would have to think about how to incorporate the more complex shape functions mentioned above.

- U-shaped figures: In the case where the centroid is outside the fish, we set all watch hands to zero which do not point into the direction of the fish. We think that this procedure is sufficient because (a) there are very few cases with such an extreme shape (less than 0.6% for eels and less than 0.02% for trouts), (b) in these cases, the centroid is not far away from the fish, and (c) it happens almost exclusively for eels and not for trouts, resulting in a special feature of eels which does not affect the discrimination power.

Selection of variables (Section 3.2 and Table 4.2)

With regard to the set of *Shape variation* variables, we summarize the information in three variables (note that the distances from the mean curves $d^1_{ij}(g)$ do not result from the PCA). It would be an alternative to use the scores of the PCA directly: We could compute more principal components, say 20 or 30 for each class, and would thus have between 60 and 90 variables in total. Then, in the following step we would have to apply a variable selection procedure to filter the important variables. However, this would not be in line with the idea of generating four sparse sets of variables:

As the original purpose was the application at a water power plant, the stability of the system is of paramount importance. This means, not only the classification accuracy in the given data is important but also the classification accuracy in new unknown data. In contrast to many other statistical applications the uncertainty of how these unknown new data will look is rather high: Even if the camera and the fish species are comparable, many other characteristics of the river such as flow velocity, turbidity and behavior of the fish could differ strongly from our data. In order to develop a system which promises to classify satisfactory even under rather unknown conditions, we feel that it is favorable to put some thought in the selection of the variables we use for classification, in contrast to leave everything up to a black box algorithm. All our variables are motivated substantially (*Baseline* - rough size of the object, *Shape mean* - standardized average shape, *Shape variation* - deviation from this standardized average shape, *Motion* - moving characteristics). With only four sets of variables, a formal variable selection procedure is not necessary in the following. All we have to do is to test each combination of the four sets of variables to get the best combination. We are hence confident to classify even in different and rather unknown conditions because the variables we constructed should always differentiate well between the considered classes.

Additionally to the good classification performance we have a good interpretation of our procedure and the results which is also favorably when explaining the methods and results to biologists and engineers. In contrast, using many scores followed by an algorithm-driven variable selection would rather be a black box.

Number K of eigenfunctions (Section 3.2.2, Page 49)

We use $K = 3$ because we got the best classification with this number of eigenfunctions compared to $2, 4, 5$ or more eigenfunctions. However, in some cases, omitting the low-variance eigenfunctions could mean to throw away relevant information on between-group variation (see Jolliffe, 2002, pp. 200-201 for an illustration). Thus, we cannot be sure that we use the best set of eigenfunctions in terms of classification accuracy. Nevertheless, the results in Section 4.2 show that this selection yields a good and satisfactory classification. It would be an interesting challenge for future work to select the set of eigenfunctions which is optimal for classification. However, we think this is not trivial as each subset of indices $\{1, ..., T\}$ would be a candidate, and this even for each class since the optimal subset of indices can differ between classes.

6.3. Possible extensions

Methodological developments

With the proposed *Shape* variables in Section 3.2.2, we do not exploit the information that several hotspots belong to the same object. We average the distances $d^1_{ij}(g)$ and $d^2_{ij}(g)$ on object level, but for the computation of these distances each hotspot is treated separately. Actually, we know even more: Not only that the hotspots are clustered on object level but we have additionally the information about the chronological order. It seems plausible that the results could be improved by exploiting this information. Recent work in the context of functional data analysis created the possibility to treat the watch hand functions as *longitudinal functional data*, see the review of Morris (2015) for a broad overview of the development in this field over the past years. Especially, the concept of *longitudinal functional principal component analysis*, developed by Greven *et al.* (2010), could be very useful here. It would be an interesting work to adopt these concepts to further increase the information contained in the *Shape* variables.

Another way to account for the order information would be to smooth the watch hand functions of one object over time. Either a suitable penalisation for the differences of the basis function weights over time could be applied. Alternatively, a two-dimensional basis for angle α and time t could be defined. Both approaches would be additional steps in the preprocessing of the watch hand functions. Afterwards the same methods as proposed in Chapter 3 could be used for these preprocessed functions.

The *Motion* variables seem to carry strong information about class memberships. These variables are based on the track of an object's centroid. An alternative to pull out information from these tracks could be to understand the track of a given object as function in time and two-dimensional space. Then, methods of functional data analysis could be adopted to define informative classification variables.

Though the tracking procedure works satisfactory for the given data, we cannot be sure that our tracking procedure is optimal. There are many other tracking methods proposed in the literature, see for example Yilmaz *et al.* (2006) for a broad overview. It would be interesting to adopt some of these methods for the context of sonar videos and to compare the performance with our approach.

R package

At the moment, the developed R package `sonar` is rather specialized and therefore may not have the chance to be used by a large part of the R community. As future work it would be promising to extend the package in order to reach a larger group of users. Among the new features should be the following:

- Possibility to directly read in further video formats such as .avi and .mp4, for example, thus including color videos (the sonar videos consist only of one color channel). To the best of our knowledge, until now there does not exist an R package with this functionality.

- Possibility to additionally analyze three-dimensional videos. This could be interesting for example for the analysis of functional magnetic resonance imaging (fMRI) data.
- Possibility to carry out the preprocessing steps such as smoothing, identification and tracking of hotspots for all video formats.
- Possibility to compute the features proposed here and to define own features.

Alternatively, the package could be more oriented towards the needs of practitioners working with sonar videos. For example, Mueller *et al.* (2008) is heading for the automatic classification of sonar images and identifying eels using R, but there, several additional software packages are needed to carry out the preprocessing. Langkau *et al.* (2012) uses acoustic shadows to identify fish species, also working with a series of preprocessing programs and the final classification is carried out in SPSS. Thus, there could be a significant demand for an R package unifying and simplifying the processing of sonar videos.

Possible applications

The DIDSON sonar camera is not the only sonar recording device used for scientific questions. Another device is for example the Simrad EK15 (see http://www.simrad.com/ek15 for details), which is much cheaper. Our project partners are interested in the question to what extent the results achieved with the DIDSON sonar camera can be reproduced with the EK15. Thus, comparable results would be very welcome in the community working with sonar images and videos.

In our analysis, we focus on fish of medium and large size with a minimal length of 10 cm, smaller objects were deleted prior to the classification because meaningful features cannot be computed for such small fish. But very small fish often appear in shoal of fish. Depending on the monitored waters, it could be interesting to incorporate an additional class *shoal*. Therefore, the preprocessing would have to be extended because currently the focus is on separating individual fish. When aiming on the detection of shoals, the focus would pass over to connecting clusters of small fish.

Finally, an interesting field of application is the study of so called *tail-beat patterns*, see for example Mueller *et al.* (2010) who investigate tail-beat patterns based on DIDSON sonar videos. Tail-beat patterns of fish can on the one hand be used for species identification. On the other hand they can be used for bioenergetics studies, see Mueller *et al.* (2010). As in other applications, data have to be preprocessed, noise has to be reduced and objects have to be tracked. Thus, the methods proposed in this thesis could be beneficiary in that field as well.

Related own work

In a recent master's thesis, Maier (2013) applied and expanded the ideas and algorithms given in this thesis on a different setting of sonar videos. There, the DIDSON camera was rotated such that the fish in the videos can be seen from the side rather than from the top. The aim was then to distinguish the two fish species *barbel* and *bream* which look rather similar in comparison to the here treated case of *eels* versus *trouts*. Classification variables were constructed using the rough size of the hotspots and coefficients resulting from a Fourier approximation of the silhouettes. In addition, distances between examined silhouettes and prototype silhouettes of barbel and bream were used as classification variables. Apart from some difficulties regarding the preprocessing of the videos, the classification results were appealing confirming the broader applicability of the shown methods and our approach.

6.4. User's perspective

Last but not least we would like to assess our work from a practitioner's point of view.

The research goals of our project were defined in interdisciplinary work and therefore reflect the practitioner's requirements. As thoroughly described in Section 6.1, all goals were achieved. An additionally interesting point from the user's perspective might be that the developed approach is not only a black box. Although the used methods are complex, the steps of the approach can also be explained without formulas and the results have meaningful interpretations.

In case that similar conditions as in our project (i.e., detect eels vs. other fish) are given, our system is ready for operation under real-world conditions. To set up the system at a new location, a few parameters have to be calibrated for the given water and the classification rule has to be learned as explained on page 65. This is considered to be between one and two weeks' work for a person trained in statistics and familiar with R. Compared to the high costs of a DIDSON, the expenses for the calibration are neglectable. After the calibration, the system can be easily used by the staff of the water power plant.

In case that other conditions are given, the system would have to be expanded and adapted and the present system could be used as a starting point. Except for eels, the classification of other fish species cannot be guaranteed at the moment. This is mainly due to the fact that we see the fish from a top view. In Maier (2013) we showed how resembling fish species such as barbel and bream can be distinguished by a different setup of the camera. Also for shoals of fish, a separate solution would have to be developed.

From a user's perspective, the high costs of the presented approach are certainly of interest: The DIDSON is with a price of around 100.000 € rather expensive if such a system should be implemented at all water power plants of a country or of a specific power company. Depending on the size of the river, multiple cameras could be necessary to cover the area in front of a power plant. To reduce these costs, it would be desirable to repeat our analysis with cheaper cameras such as the EK15 with a price of around 5.000 €.

Further issues might also be important from a practitioner's point of view: We cannot be sure that we detect all fish swimming in the monitored river for two reasons. First, our system could overlook fish. Second, it is possible that fish pass above or below the field of view of the camera and thus are not recorded. Construction measures or the use of multiple cameras could circumvent this problem at the price of additional costs.

Furthermore, the experimental setup was not optimal. As described in Chapter 1 on page 13, the videos of eels and trouts originate from a water without current while the video showing debris originates from a river with current. Thus, parts of the high classification accuracy of living versus dead objects could be explained by the different settings. However, we think that this is no major drawback and that our results can be generalized for two reasons: First, the constructed variables discriminate living and dead objects above all due to the *Motion* variables. This would also be the case in the same water. Second, in a further pilot study we analyzed data of fish and debris recorded at a water power plant near Hamm, Germany, with satisfying results regarding the classification accuracy.

Finally, a user would have to decide about protection measures which could be taken when fish are detected in front of a water power plant. An obvious option would be to shut down the turbines and to let the fish pass. This comes along with a loss of generated energy and thus with a loss of profit for the power companies which adds to the high costs of the sonar cameras. However, protection measures lie beyond the scope of this thesis.

All in all we are confident that the present analysis is a significant step for the automatic detection of fish at water power plants. We hope that our approach can be a contribution for the environmental compatibility of water power plants and for the protection of endangered fish species.

Part II.

Automated processing of webcam images for phenological classification

Chapter 7: Introduction

7.1. Motivation and research goals

Motivation

Along with the global climate change, there is an increasing interest for its effect on the nature: How affects the global warming the flora and fauna on the earth? A science sector called *phenology* deals with the description and study of annually recurring events in nature. For example, the time of the first appearance of migratory birds and the dates of leaf unfolding in spring and leaf coloring and leaf fall in autumn are objects of phenological studies.

Thereby, phenology has been recognized to be a key factor in the description of ecosystem processes. The dates of the start and end of the growing season define the length of the vegetation season, then triggering biogeochemical fluxes between atmosphere and biosphere, such as carbon sequestration (Piao *et al.*, 2008; Richardson *et al.*, 2013), ecosystem respiration (Piao *et al.*, 2008; Migliavacca *et al.*, 2011), and biomass production (Keenan *et al.*, 2012). Moreover, the seasonality of vegetation activity determines vegetation cover and biodiversity, but also controls the vegetation feedbacks to the climate system, in terms of oxygen production, evapotranspiration, BVOC (biogenic volatile organic compound) emission, and other surface layer changes (Schwartz, 1992; Menzel, 2002; Peñuelas *et al.*, 2009; Richardson *et al.*, 2013).

To estimate the effect of the global warming, it is of great interest how the dates of the start and end of the growing season vary over the years. Additionally, the spatial variation of these dates can give insights in phenological processes.

Phenological analyses – state of the art

Ground observations. Today, most phenological information is either collected by volunteers or semi-professionals in phenological networks run by national meteorological services or citizen science organisations (Menzel, 2013). At this local scale that is also covered by ground-based webcams, onset dates of leaf unfolding or leaf coloring are recorded for selected sites and species.

Satellite remote sensing. At the regional and even national and continental scale, start and end of season dates can also be derived from annual time series of vegetation indices from various remote sensing products, mainly AVHRR (advanced very-high-resolution radiometer), MODIS (moderate-resolution imaging spectroradiometer), and MERIS (medium-resolution imaging spectrometer) instruments. The longest available record is the Normalized Difference Vegetation Index (NDVI) with data from 1982 onwards (Jeong *et al.*, 2011; Jeganathan *et al.*, 2014). However, remote sensing only provides a coarser spatial and temporal resolution, the so-called integrated land surface phenology (LSP, see Liang and Schwartz, 2009). For example, using the so-called maximum-value composite procedure (Holben, 1986) with a MODIS instrument, the resulting resolution is $250m \times 250m$ per pixel and one observation every 16 days. Unfortunately, linking LSP with species- and site-specific ground observations has turned out to be quite complicated or even impossible (White *et al.*, 2009; Fu *et al.*, 2014; Rodriguez-Galiano *et al.*, 2015).

Webcams. To close the gap between LSP and species- and site-specific ground observations, close surface remote sensing in terms of daily digital camera images has been proposed (Richardson *et al.*, 2007), see also Graham *et al.* (2010), Zhao *et al.* (2012), Henneken *et al.* (2013) and Alberton *et al.* (2014). A major intention is to automatically capture seasonal changes on a fine spatial resolution, which could be at the end also on species level. Secondly, one wants to relate the findings with satellite remote sensing systems, since the percentage of greenness (%greenness) derived from the red, green and blue information mirrors the temporal behavior of vegetation indices by remote sensing, see e.g. Hufkens *et al.* (2012). Still, single scientific cameras will not yield spatially dense information. Therefore, a dense webcam network is required which then demands for advanced and automatic image processing in order to handle these high resolution spatio-temporal data. This thesis deals with the development of such image processing methods.

Use of webcam images for the identification of phenological patterns

Usually, webcams are used for the identification of phenological patterns as follows: A webcam takes every day one or more images from the same natural motive showing for example trees or grassland sites (Julitta *et al.*, 2014; Alberton *et al.*, 2014). Based on these images, the dates of (1) the start of the growing season (SOS), (2) the time point of maximum %greenness (MAX) and (3) the end of the growing season with start (EOS1) and end (EOS2) of leaf coloring in autumn shall be determined. Therefore, the dimension of the data is reduced by first defining regions of interest (ROIs) on the image. A ROI is a region which is subjectively relevant, for example pixels representing crowns or canopies of deciduous trees because they show high seasonal variation necessary for phenological classification. In these ROIs, a %greenness time series is computed: For each time point t, the value pg_t of %greenness inside the ROI is defined as

$$pg_t = \frac{1}{n} \sum_{i \in I_R} \frac{g_{it}}{r_{it} + g_{it} + b_{it}}, \tag{7.1}$$

where r_{it}, g_{it} and b_{it} are the red, green and blue values at pixel i and time point t, respectively, I_R denotes the index set of all pixels inside the ROI and $n = |I_R|$ is the number of pixels inside the ROI. Finally, the dates of SOS, MAX, EOS1 and EOS2 are determined by a search for structural changes in the %greenness time series, see for example Ahrends *et al.* (2008), Henneken *et al.* (2013) and Menzel *et al.* (2015).

Abbreviations

In the remainder we will use the term *DOY* (day of year) to refer to the dates of images, i.e., all days of the year are numbered from January 1 to December 31 as DOY 1 to DOY 365 (or DOY 366 in leap years). Table 7.1 summarizes the most important abbreviations used in this part.

Table 7.1. Definition of frequently used abbreviations

Abbreviation	Definition
DOY	Day of year
%greenness	Percentage of greenness in an image / ROI
SOS	Start of the growing season
MAX	Time point of maximum %greenness
EOS1	End of the growing season - start of leaf coloring in autumn
EOS2	End of the growing season - end of leaf coloring in autumn
ROI	Region of interest on an image
eROI	Expert-based region of interest
sROI	Semi-supervised region of interest
uROI	Unsupervised region of interest
SVD	Singular value decomposition

Research goals

While the above identification procedure is suitable for determining the dates of SOS, MAX, EOS1 and EOS2, it is associated with a considerable amount of manual work, because the ROIs have to be defined manually by an expert. We call these regions “expert-based regions of interest” (**eROI**) in the remainder to differentiate this standard approach from our new approaches for the definition of ROIs. Our aim is to use webcam images on a larger scale and to analyze the data of several hundreds or thousands of webcams at the same time. For this purpose, the processing of the webcam images has to be automated and an efficient implementation is needed.

This leads to the definition of the following research goals:

1. Develop a method for the automated definition of ROIs.
2. Develop a reasonable optimality criterion to evaluate the amount of information contained in the resulting ROIs. The automated ROIs have to be at least as informative as the eROIs.
3. Develop a procedure for determining the dates of SOS, MAX, EOS1 and EOS2 for given image data. To some extent, existing methods for structural change point detection in time series from the literature can be applied for this point.

4. Implement the methods so efficiently that the analysis can be scaled up to several hundreds or thousands of webcams. For this purpose, the required manual work has to be reduced to a minimum.

The third research goal is also referred to as "phenological classification" in the remainder, because from another point of view, the days of the year are classified as "before SOS", "between SOS and MAX" and so on. Thus, the phenological dates can be matched with the classical terms for seasons as shown in Table 7.2.

Table 7.2. Definition of season names and season numbers used in this thesis.

Season name	Definition	Season number
Spring	Days between SOS and MAX	1
Summer	Days between MAX and EOS1	2
Autumn	Days between EOS1 and EOS2	3
Winter	Days between EOS2 and SOS	4

7.2. From webcam data to phenological classification

Data structure of webcam images

In contrast to the images of the sonar videos presented in Part I, the images of a webcam are color images recorded on an equidistant rectangular grid. For each pixel (i, j), color channel c and time point t we observe a signal y_{ijct} standing for the red $(c = r)$, green $(c = g)$ and blue $(c = b)$ intensity with values in $[0, 1]$. The data array $\boldsymbol{Y} = \{y_{ijct}; i = 1, \dots, n_1, j = 1, \dots, n_2, c \in \{r, g, b\}, t = 1, \dots, T\}$ serves as our four-dimensional raw data where n_1 stands for the number of pixels in horizontal direction, n_2 for the number of pixels in vertical direction and T for the number of recorded images for a given webcam.

For example, from the scientific camera described later in Section 7.3 we have $T = 1441$ images with $n_1 = 1276$ and $n_2 = 960$ pixels for the year 2014. In total, this sums up to approximately 5.3 billion observed pixel values. This large number of observations shows that we are already dealing with a big data problem by analyzing the data produced by one single webcam in one single year. But, as mentioned above, our aim is to use webcam images on a larger scale and to analyze the data of several hundreds or thousands of webcams. Thus, it is necessary to develop efficient tools which allow for fully automated analyses. At the same time the computational complexity has to be as low as possible to ensure the applicability of the methods in practice.

Analysis steps

Optimality criteria. As first step, we define two optimality criteria in Section 8.1 to meet the second research goal. We begin with this step because we will make use of the optimality criteria in the automated definition of ROIs as well as in the comparison of the resulting ROIs. Both optimality criteria assign a scalar value to a given ROI which is high when the %greenness time series inside the ROI captures much information about phenological variation. Thus, the optimality criteria can be used to compare ROIs and to decide which approach delivers the most informative ROI.

Automated definition of ROIs. Regarding the first research goal, we propose two data-driven approaches for the definition of ROIs as alternatives to the eROIs in Sections 8.3 and 8.4:

- First, we propose a **semi-supervised** approach: We select a very small number of pixels which clearly show phenological features like deciduous trees. This is in principle done as in the eROI approach, but here we just select a few number of pixels (6×6) rather than delineating the structures of interest at fine resolution as done in the eROI approach. Once these pixels of interest are defined, we let them grow to a data-driven region of interest by adding pixels with high seasonal correlation. We call this region “semi-supervised region of interest” (**sROI**) in the remainder.

  This strategy is an improvement of the standard approach because instead of defining large expert-based ROIs we just set a pinprick on the image which requires less manual effort from the expert and may even be applied by non-experts.

  Additionally, we provide a fully automated and purely data-driven version of the sROI approach: Instead of defining a single expert-chosen pinprick, multiple random pinpricks can be placed on the image.

Then, based on the two optimality criteria, the best sROI in terms of phenological information can be selected.

- As second strategy we propose an **unsupervised** approach: We carry out a singular value decomposition (SVD) of the images in order to reduce the dimension of the data while maintaining the information about phenological variation. A subsequent cluster analysis groups the pixels with regard to their variation over time. Finally, we identify the interesting cluster of pixels which show the highest phenological variation by comparing the optimality criteria for the resulting %greenness time series. The resulting cluster of pixels is called “unsupervised region of interest” (**uROI**) in the remainder.

Phenological classification. Both approaches lead to ROIs for the images. Regarding the third research goal, determining the dates of SOS, MAX, EOS1 and EOS2, these ROIs can be used to derive and analyze %greenness time series as commonly done in the phenological community. This means, structural changes in these time series can be searched for and thus dates of SOS, MAX, EOS1 and EOS2 are identified. We show that with our more advanced definitions of ROIs we obtain %greenness time series which carry at least as much phenological information as those based on the standard eROI approach. The advantage of our approach is that considerably less manual effort is required. Chapter 8 presents our new approaches and Chapter 9 compares the performance of our approaches with the state-of-the-art approach of expert-based eROIs for images resulting from a scientific webcam. Additionally, we show how the methods can be applied to publicly available open-access webcams.

Thus, once the ROIs are defined, we use existing methods for the search for the dates of SOS, MAX, EOS1 and EOS2 in Chapters 8 and 9. In Chapter 10 we additionally propose a completely different approach for determining these dates: After the SVD, each image is represented via scores on a small number of eigenimages, these scores can be used for a supervised classification. Therefore, each image of a training year has to be classified manually into four categories *spring*, *summer*, *autumn* and *winter*. Then, images of a test year can be classified with respect to these four categories based on the scores of the SVD using standard statistical classification methods. By summarizing the predicted categories on day level, each day is uniquely classified as *spring*, *summer*, *autumn* or *winter* day. As final step, we further process these predictions to derive unique dates of season changes. Thereby we propose a different approach for the third research goal, the determination of the dates of SOS, MAX, EOS1 and EOS2.

As additional benefit of this supervised approach, the days of the year can be classified with respect to other sets of categories. One example is the identification of snow days: By learning a classification rule to discriminate snow days from non-snow days, all snow days from a test year can be identified. This is for example useful when monitoring remote forest areas because sudden and unexpected onsets of winter can be identified.

Implementation. All methods are implemented in the statistical software package R (R Core Team, 2016). The resulting R package `phenofun` is available on R-Forge at https://r-forge.r-project.org/projects/phenofun/, the documentation of the package is sketched in Appendix C. In Section 9.2 we show how the methods can be applied to a large database of several thousands of webcams. We thereby show that also the fourth research goal of an efficient implementation is achieved.

7.3. Scientific and open-access webcams

We demonstrate our routines in detail with five webcams, two scientific webcams and three publicly available open-access webcams. Additionally, we show that the methods can be used on a larger scale as well by applying them to a database of several thousands of webcams.

Scientific webcams

Kranzberg Forest. The study site Kranzberg Forest is located near Freising/Germany at $48°25'08''$ N and $11°39'41''$ E (485 m.a.s.l.). During the year 2014, a scientific MOBOTIX M12D-Sec-DNight camera recorded four images per day with the same field of view. Two images were taken in the morning around 9.45 am and two images were taken on midday around 1.45 pm. Due to technical reasons, some images had to be deleted, resulting in a total of 1441 images. For the year 2013, 1000 images were recorded starting at DOY 105 (April 15). The image resolution is 1276×960 pixels, Figure 7.1 (top row) shows two example images from DOY 121 (May 1) and DOY 365 (December 31) for the year 2014.

Brandwiese. For the study site Brandwiese, located at $47°26'44''$ N and $11°6'30''$ E (900 m.a.s.l.), a camera of the same type recorded seven images per day around midday between 1.45 pm and 2.15 pm. For the year 2012, 1413 images were recorded starting at DOY 122 (May 1), for the year 2013, only 725 images were recorded between January 1 and December 31 due to technical problems. Figure 7.1 (bottom row) shows two example images from DOY 140 (May 19) and DOY 348 (December 13) for the year 2012.

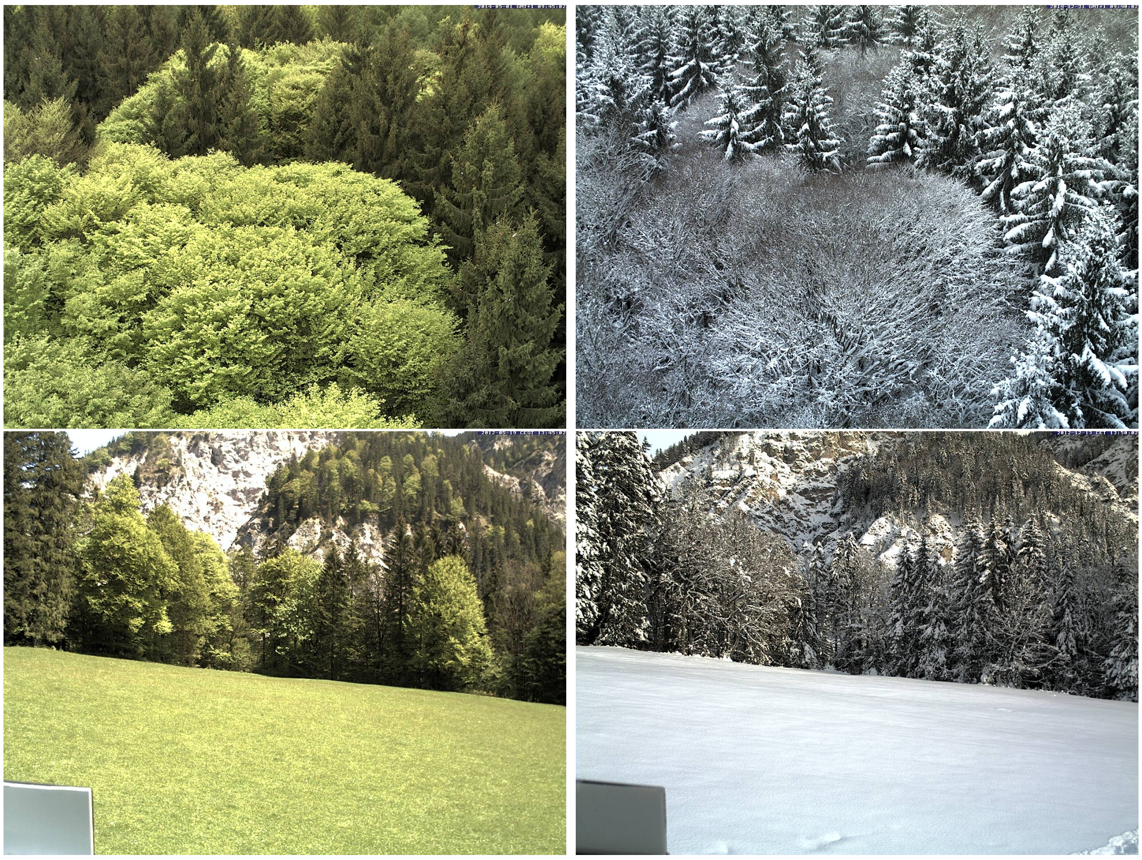


Figure 7.1. Example images of Kranzberg forest on DOY 121 (top left) and DOY 365 (top right) and of Brandwiese on DOY 140 (bottom left) and DOY 348 (bottom right).

Open-access webcams – foto-webcam.eu

The first open-access webcam data were taken from the website http://www.foto-webcam.eu. This website hosts images from a large variety of webcams which are mainly located in the Alps. The original purpose of this collection was not to enable phenological analyses but many of the webcams are well suited for this purpose as well. Here, we use three sites: First, data from Marquartstein Süd, near the lake Chiemsee in South Germany from the year 2013; second, data from Kolm-Saigurn, Austria, from the year 2014; and third, data from Studentenstadt in the north of Munich, Germany, from the year 2013. Example images are shown in Figure 7.2. The webcams were selected to cover a wide range of possible views, i.e., from rich phenological information (Marquartstein and Kolm-Saigurn, top and middle image) to poor phenological information looking at buildings and streets in a big city (Studentenstadt, bottom image). All images have a resolution of 1200×675 pixels where the top 50 pixel rows of each image were discarded before the analysis to delete the time stamp and location information. At each site we use one image per day taken around midday for each day of the respective year.

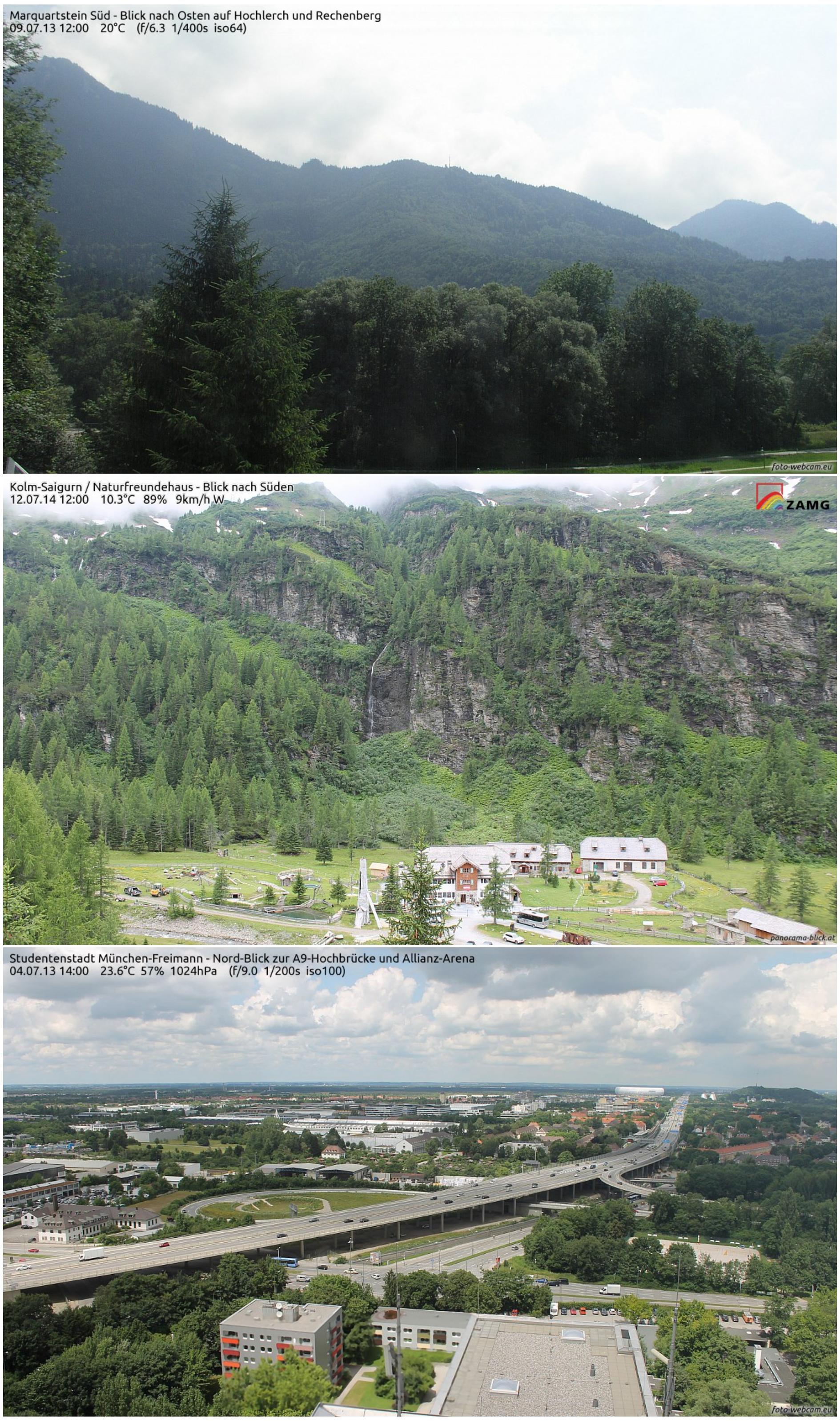

Marquartstein Süd - Blick nach Osten auf Hochlerch und Rechenberg
09.07.13 12:00 20°C (f/6.3 1/400s iso64)
foto-webcam.eu
Kolm-Saigurn / Naturfreundehaus - Blick nach Süden
12.07.14 12:00 10.3°C 89% 9km/h W
ZAMG
panorama-blick.at
Studentenstadt München-Freimann - Nord-Blick zur A9-Hochbrücke und Allianz-Arena
04.07.13 14:00 23.6°C 57% 1024hPa (f/9.0 1/200s iso100)
foto-webcam.eu

Figure 7.2. Example images of Marquartstein on DOY 190 (top image), Kolm-Saigurn Sonnblickbasis on DOY 193 (middle image) and Studentenstadt München-Freimann on DOY 185 (bottom image). Images are available at http://www.foto-webcam.eu.

Open-access webcams – AMOS

AMOS - the archive of many outdoor scenes “is a collection of long-term timelapse imagery from publicly accessible outdoor webcams around the world” (Jacobs *et al.*, 2007) and can be accessed through http://amos.cse.wustl.edu/. It contains image data from almost 30000 webcams of which the majority is located in the United States. A map with the locations of all cameras can be found at http://amos.cse.wustl.edu/browse_map. Figure 7.3 shows example images for camera ID 8221 from Iowa, USA (top image), camera ID 441 from New York, USA (middle image) and camera ID 22231 from Lake Louise, Canada (bottom image). The first two cameras show deciduous trees and grassland in large parts of the images. Here, the challenge for the algorithm is not to mix up pixels of deciduous trees and grassland. The third camera shows no deciduous trees at all. Here, seasonality has to be derived from the grassland only.

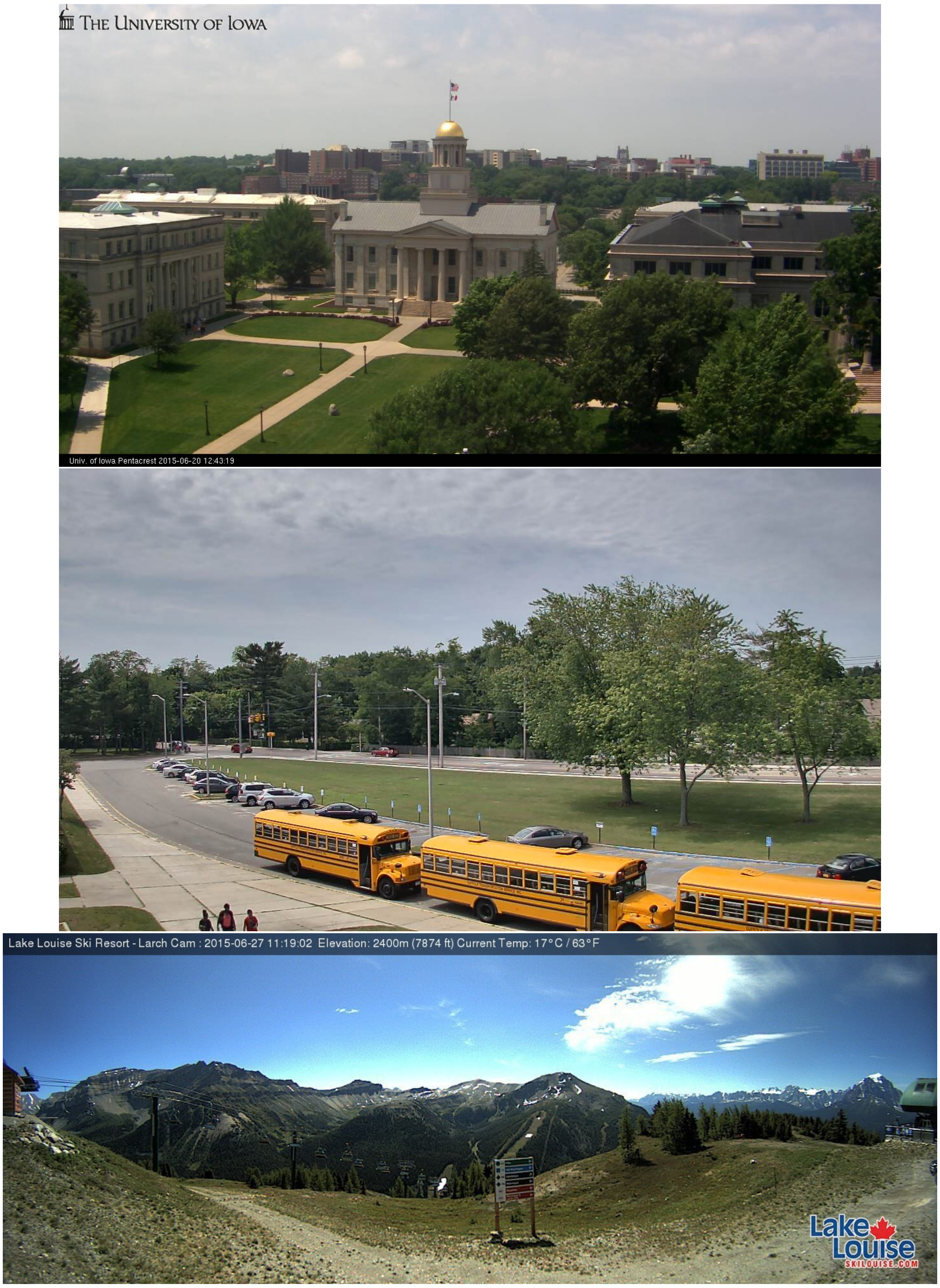

THE UNIVERSITY OF IOWA
Univ. of Iowa Pentacrest 2015-06-20 12:43:19
Lake Louise Ski Resort - Larch Cam : 2015-06-27 11:19:02 Elevation: 2400m (7874 ft) Current Temp: 17°C / 63°F
Lake Louise
SKILOUISE.COM

Figure 7.3. Example images for three selected webcams from AMOS. Images are available at http://amos.cse.wustl.edu/.

Chapter 8: Automated definition of regions of interest

In this chapter we propose two new approaches for the definition of regions of interest (ROIs), namely the “semi-supervised regions of interest” (sROI, Section 8.3) and the “unsupervised regions of interest” (uROI, Section 8.4) which are automated alternatives to the standard approach of “expert-based regions of interest” (eROI, Section 8.2). As first step, we present in Section 8.1 two optimality criteria which can be used to compare ROIs with respect to their amount of phenological information. The optimality criteria will also be used in the definition of sROIs. In Chapter 9, we apply and compare the approaches for the given webcam data.

8.1. Optimality criteria

We propose two optimality criteria which can be used to assess the amount of phenological information in a ROI. Both optimality criteria have in common that they assign a scalar value to a given ROI which is high when the %greenness time series inside that ROI captures much information about phenological variation.

For a given ROI, the %greenness time series pg_t inside the ROI is computed as given in Equation (7.1). If more than one image is taken at day t, the values are additionally averaged per day resulting in

$$pg_t = \frac{1}{n \cdot n_t} \sum_{s \in I_t} \sum_{i \in I_R} \frac{g_{is}}{r_{is} + g_{is} + b_{is}}, \quad \text{for} \quad t = 1, \ldots, T \tag{8.1}$$

where r_{is}, g_{is} and b_{is} are the red, green and blue values at pixel i and image s, respectively, I_R denotes the index set of all pixels inside the ROI, I_t denotes the index set of all images at day t, $n = |I_R|$ is the number of pixels inside the ROI, $n_t = |I_t|$ is the number of images at day t. Note that in this section the index t refers to a day of the year while in the previous chapter the index t referred to a specific image. This shall allow for multiple images per day. Figure 8.1 shows as an example the %greenness time series for the eROI of the Kranzberg data.

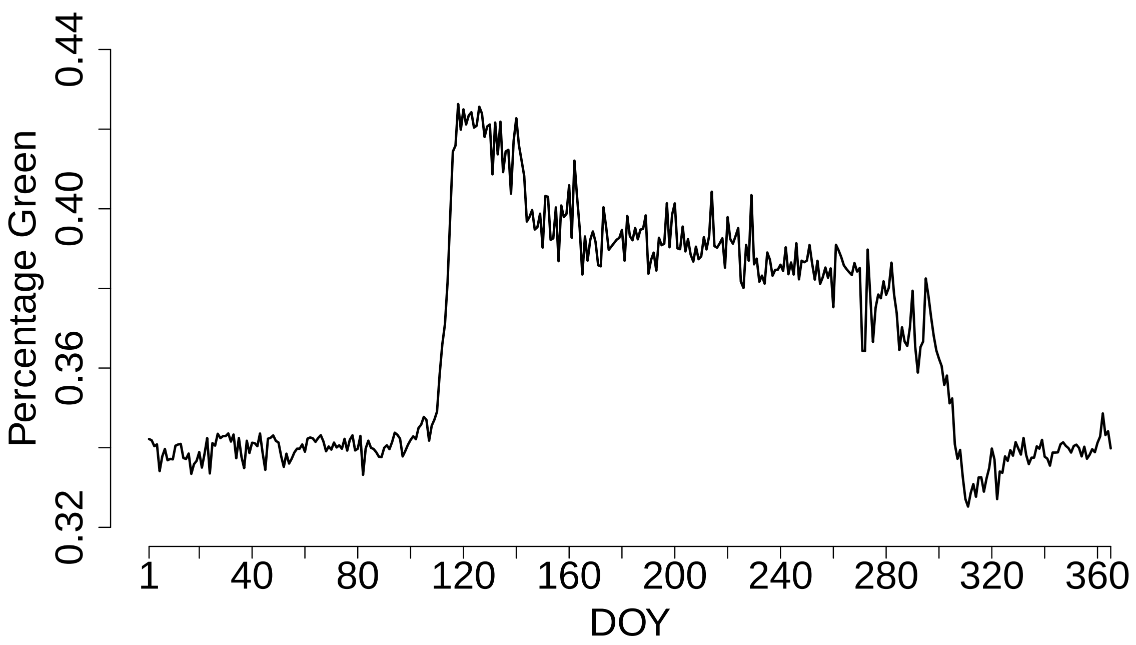


Figure 8.1. %greenness pg_t in eROI over the year 2014, resulting time series from Kranzberg data. For the used eROI see Figure 8.3 (top left).

Optimality criterion 1

The first optimality criterion (OC1) is based on methods from the field of structural change point analysis. Since we are searching for the pixels with the highest phenological variation over time, we know that the corresponding %greenness time series has to show a significant structural change in spring as can be seen for example around DOY 120 in Figure 8.1. Before and after this change, the %greenness time series has to be more or less linear until autumn. We use this knowledge to define an optimality criterion as follows.

We only consider the first 240 days, i.e., all days until the end of August, and thereby delete the autumn and beginning of winter because at this point we are only interested in the structural change in spring. First, we compute the %greenness time series pg_t of the pixels inside the given ROI as shown in Equation (8.1).

Then we search for a structural change in this %greenness time series with usual methods for structural change point detection, for example described in Hansen (1992), Andrews (1993) and Zeileis *et al.* (2003) and implemented in the R package `strucchange` by Zeileis *et al.* (2002). All time points from DOY 30 to DOY 210 are subsequently considered as potential change point. The reason for not considering DOYs from 1 to 29 and from 211 to 240 is that at some point the sample size would be too small to fit the corresponding models. Additionally we know that the spring onset in Germany will neither be in January nor in August.

For each potential change point $cp = 30, \ldots, 210$, a linear model LM_{cp} with changing intercept and slope at time cp is fitted:

$$pg_t = (\beta_0 + \beta_1 \cdot t) \cdot I(t \leq cp) + (\beta_3 + \beta_4 \cdot t) \cdot I(t > cp) + \varepsilon_t, \quad t = 1, \ldots, T, \qquad (8.2)$$

where $I(\cdot)$ is an indicator function, T is the number of time points, i.e., here $T = 240$, and ε_t is the error term for which usual assumptions apply.

Additionally, a linear model LM_0 without a change point is fitted:

$$pg_t = \beta_0 + \beta_1 \cdot t + \varepsilon_t, \quad t = 1, \ldots, T. \qquad (8.3)$$

Then, for each potential change point cp, an F-statistic is computed to compare the model with change point, LM_{cp}, with the model without change point, LM_0:

$$F_{cp} = \frac{(RSS_0 - RSS_{cp})/k}{RSS_{cp}/(T - 2k)}, \quad \text{for} \quad cp = 30, \ldots, 210, \qquad (8.4)$$

where RSS_0 and RSS_{cp} are the residual sums of squares for the models LM_0 and LM_{cp}, respectively, and k is the number of estimated parameters in model LM_0, i.e., here $k = 2$. See also Zeileis *et al.* (2002) for details.

Finally, the change point cp_{opt} with the largest F-statistic is considered to be the point of structural change. The reason is that for this change point, the signal-to-noise ratio in the resulting %greenness time series is maximal with respect to the fitted models. The value of the corresponding F-statistic $F_{cp_{opt}}$ is the value of the optimality criterion OC1 for this time series and corresponding ROI.

Optimality criterion 2

The second optimality criterion (OC2) is closer to the expected seasonal pattern of the %greenness time series. From long-term phenological experience we know that the %greenness time series of a deciduous tree behaves as follows: In winter, the time series stays constant on a rather low level. In spring, there is a steep increase followed by a small decrease and then, in summer, the time series remains constant again but on a higher level than in winter. Finally, in autumn, the time series further decreases until the level of winter is reached.

Although we know this rough functional pattern, we do not know the concrete dates of the change points. But we can exploit our structural knowledge: We define a large set of template time series which reflect the known seasonal patterns but differ in the time points of start of spring a and end of autumn b. Figure 8.2 shows nine of these template time series for varying values of a and b. In principle also other templates such as typical %greenness time series of grassland systems, agricultural cereals or infrastructure could be used.

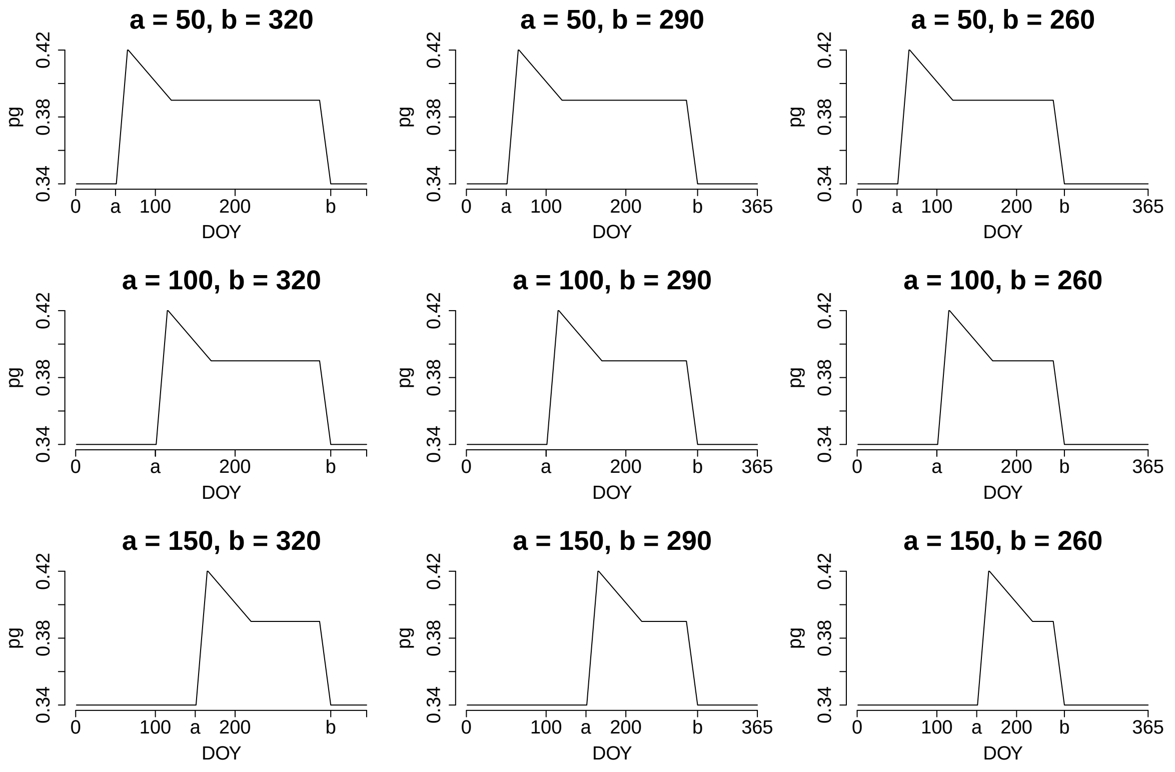


Figure 8.2. Template %greenness time series for varying values of start of spring a and end of autumn b.

For a given ROI, we can now compute the correlation of the resulting %greenness time series with each of the template time series, where the values a and b vary in all possible combinations between $a \in \{50, 51, \ldots 149, 150\}$ and $b \in \{265, 266, \ldots 364, 365\}$. Then, the value of the highest correlation is considered as optimality criterion OC2.

This approach has the advantage that additionally to the definition of an optimality criterion, we obtain rough estimates of the SOS and EOS2 dates via the values a and b of the best combination.

8.2. Expert-based ROI approach

The standard approach for filtering the relevant pixels from the data is the following: The analyzing researcher defines one or more ROIs per hand by simply drawing rectangles, polygons or free shapes around regions which are subjectively relevant, usually around crowns or canopies of deciduous trees. Deciduous trees are of special interest because they show high seasonal variation necessary for phenological classification, whereas evergreen trees and grassland display less; other abiotic items, such as bare soil, stone and sky should not be affected.

This means, a ROI is generated that defines which pixels are discarded and which pixels remain in the data for further analyses. Figure 8.3 (top left) shows the "expert-based regions of interest" (eROI) for the scientific webcam as example. These eROIs were defined with a considerable amount of manual work, see Stratopoulos (2015) for details on this process which aimed at identifying individual specimen-specific parts of the upper sun crown. Such an approach is state-of-the-art and yields satisfying results, since it allows the identification of single trees even in an homogeneous stand of one species, see for example Henneken *et al.* (2013), Alberton *et al.* (2014) and Menzel *et al.* (2015). Nevertheless, this approach requires an expert to define the eROIs manually. Therefore, it does not allow to be scaled up to an analysis of hundreds or thousands of webcams where the focus would be on the phenology of general vegetation types (e.g. deciduous trees, evergreen vegetation).

8.3. Semi-supervised ROI approach

The choice of eROIs can be subjective and the results may be improved by a more sophisticated and data-driven approach which we call "semi-supervised regions of interest" (sROI). The term semi-supervised refers to the fact that we start with a few handpicked pixels. Then, the region of interest grows data-driven to capture most of the relevant information in the images. The sROI approach works as follows.

First, we select a very small ROI in the middle of the crown of a deciduous tree, for example 6×6 pixels. This means that we point with a needle on the picture, where the needle head is on a clearly identifiable deciduous tree. This pinprick is visualized in Figure 8.3 (top right). Second, for the selected few pinprick pixels, we compute the %greenness time series $pg_{t,pp}$ according to Equation (8.1), visualized in Figure 8.4.

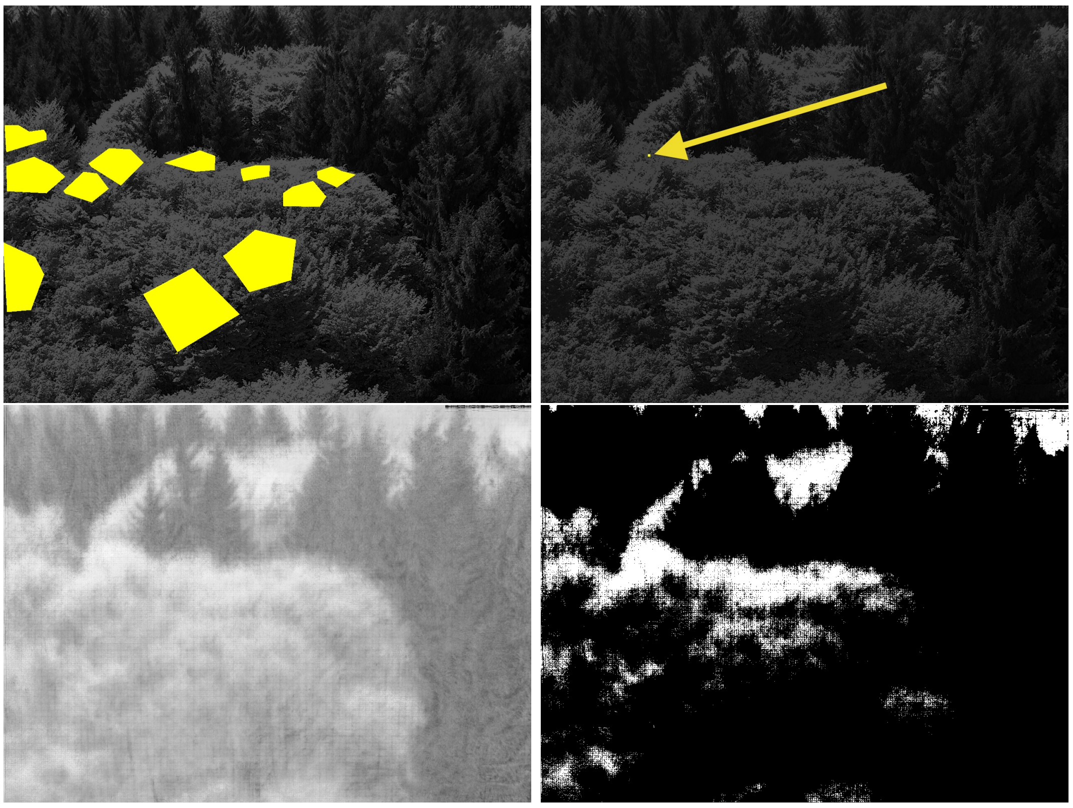


Figure 8.3. Expert-based eROI (top left), pinprick (top right), correlation image (bottom left, brighter colors indicate higher correlations) and resulting sROI (bottom right, white pixels are selected for the sROI) for Kranzberg data 2014.

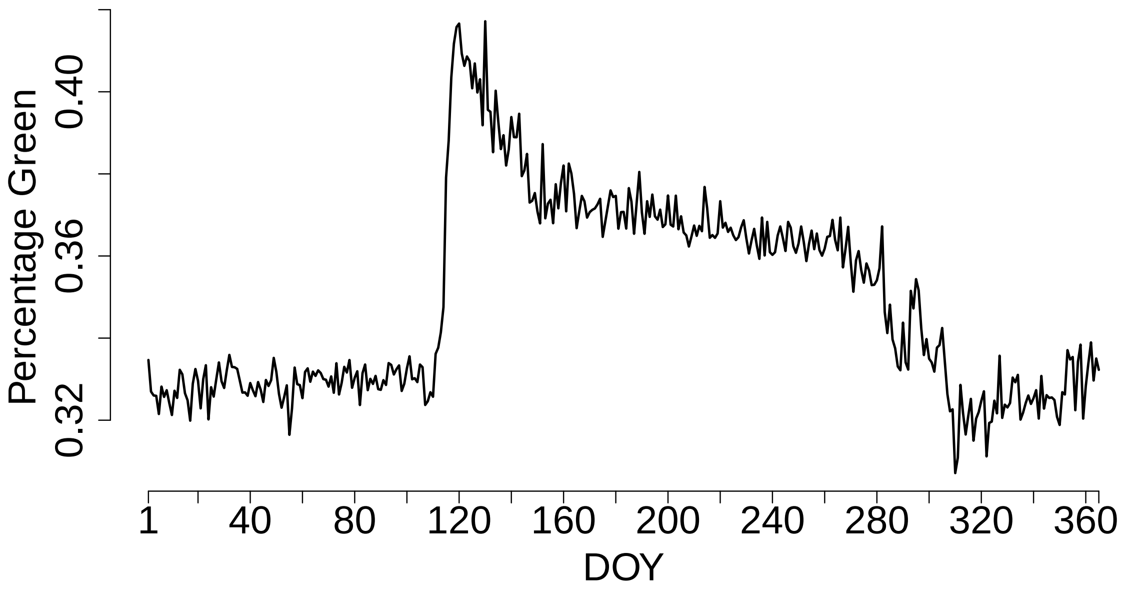


Figure 8.4. %greenness $pg_{t,pp}$ in pinprick over the year 2014, resulting time series from Kranzberg data. For the used pinprick see Figure 8.3 (top right).

As a third step, we compute for each single pixel $i = 1, \ldots, N$, where $N = n_1 \cdot n_2$ is the total number of pixels per image, the %greenness time series $pg_{t,i}$ over the year and correlate it with the pinprick pixels' %greenness time series $pg_{t,pp}$. This results in N correlations $\rho_i = cor(pg_{t,i}, pg_{t,pp})$ which can be visualized in a correlation image as shown in Figure 8.3 (bottom left). The brighter the level of gray is at a pixel i, the higher is ρ_i, the pixel's correlation with the pinprick. As final step, we threshold this correlation image and discard all pixels which have small correlations, in the example correlations of less than 0.65 are neglected. The remaining pixels constitute the sROI, visualized in Figure 8.3 (bottom right).

Automation of sROI approach

In the example given above, the pinprick and the correlation threshold were chosen manually to obtain a visually optimal ROI. Since we are heading towards an automated approach, we propose the following extension: Instead of setting a single pinprick expert-based in the crown of a deciduous tree, we distribute a number of q pinpricks randomly over the image. Additionally, the threshold for the correlation image is not chosen subjectively as above but we compute candidate ROIs for a grid of r correlation thresholds between 0 and 1. This leads to a number of $M = qr$ candidate ROIs, i.e., one for each combination of pinprick and correlation threshold.

Now, the crucial question is: How can we select the best ROI from these M candidate ROIs? We can make use of the two optimality criteria presented above:

We compute for each candidate ROI $m = 1, \dots, M$ the optimality criterion OC1 or OC2. The ROI with the highest value of the optimality criterion is considered to be the optimal ROI and hence the corresponding combination of random pinprick and correlation threshold is considered to be optimal. For practical purposes, one of the optimality criteria OC1 and OC2 has to be chosen beforehand because they might vote for different optimal ROIs. Our analyses showed that using OC2 is preferable since the resulting ROIs are visually more convincing.

Thus, there are two options when using the sROI procedure: A semi-automated and an automated version. For the semi-automated version, a single prinprick has to be set by an expert. With this pinprick, the sROI procedure computes an optimal ROI by optimizing the correlation threshold. For the fully automated version, a number of q pinpricks are randomly distributed over the image. Then, for each pinprick the best threshold is found and finally the best combination of pinprick and correlation threshold is identified.

8.4. Unsupervised ROI approach

We propose an alternative approach for the automated definition of ROIs, called “unsupervised regions of interest” (uROI).

We assume the following data structure of the color images: We consider each color channel of each pixel as observation unit and each time point $t = 1, \dots, T$ as variable. Each image has $N = n_1 \cdot n_2$ pixels with three color channels red, green and blue. We store each image in a vector of length $3N$ and denote it as $\boldsymbol{x}_t = (r_{1t}, \dots, r_{Nt}, g_{1t}, \dots, g_{Nt}, b_{1t}, \dots, b_{Nt})^\top$ for $t = 1, \dots, T$ where r_{it}, g_{it} and b_{it} denote the intensity of the red, green and blue color channel of pixel i at time t, respectively. Altogether we get a data matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$ of dimension $3N \times T$ which is centered prior to the following steps such that the mean of the pixel values is 0 for each image $\boldsymbol{x}_t, t = 1, \ldots, T$.

On $\boldsymbol{X}$ we perform a singular value decomposition (SVD). This means, $\boldsymbol{X}$ is decomposed into $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$, where the columns of $\boldsymbol{U}$ are the eigenvectors of $\boldsymbol{X}\boldsymbol{X}^\top$, the columns of $\boldsymbol{V}$ are the eigenvectors of $\boldsymbol{X}^\top\boldsymbol{X}$ and $\boldsymbol{D}$ contains the square roots of the non-zero eigenvalues of $\boldsymbol{X}\boldsymbol{X}^\top$ or $\boldsymbol{X}^\top\boldsymbol{X}$, see for example Golub and Reinsch (1970). In order to achieve a dimension reduction, we make use of a truncated version of the SVD and only compute the first $p \ll T$ left singular vectors, i.e., the matrix $\boldsymbol{U}$ is of dimension $3N \times p$. The columns of $\boldsymbol{U}$ may also be called eigenimages, in Chapter 10 we will explain how to interpret these eigenimages. We rearrange rows and columns of $\boldsymbol{U}$ such that each pixel is described by $3p$ variables, i.e., p variables per color channel, and denote it as $\boldsymbol{U}^*$ with dimension $N \times 3p$. Now that each pixel is described by a small set of only $3p$ variables (in contrast to $3T$ variables prior to the SVD), we can find similar pixels by performing cluster analyses on the rows of $\boldsymbol{U}^*$. While hierarchical clustering is computationally unfeasible due to the large number of pixels, k-means procedures lead to good results with low computational effort. As a result, the image is segmented into k clusters and each cluster of pixels defines a candidate ROI.

Figure 8.5 shows as an example the result of the cluster analysis for $p = 12$ and $k = 3$ for the Kranzberg data from 2014. Comparing the resulting $k = 3$ clusters with the example image in Figure 7.1 (top left), we can see that cluster 3 (black) is a candidate for the final uROI since it reflects the deciduous trees. Clusters 1 and 2 refer to evergreen Norway spruce specimens of which cluster 1 trees in the foreground clearly reveal spring seasonality due to May shoots as well as edge effects by the deciduous European beech trees.

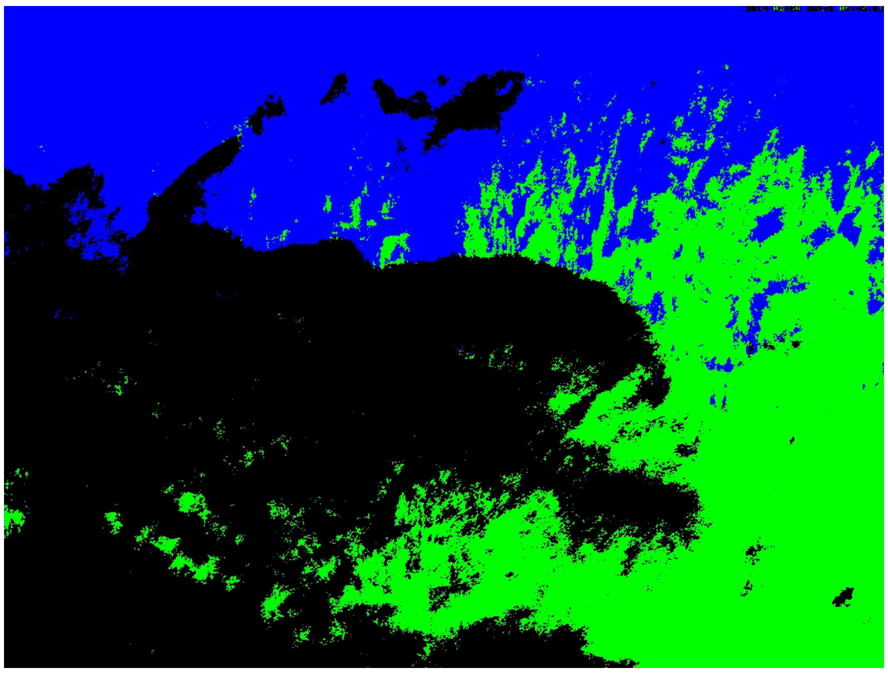


Figure 8.5. Resulting three clusters from uROI method: cluster 1 in green, cluster 2 in blue and cluster 3 in black, to compare with Figure 7.1 (top row). Kranzberg data 2014.

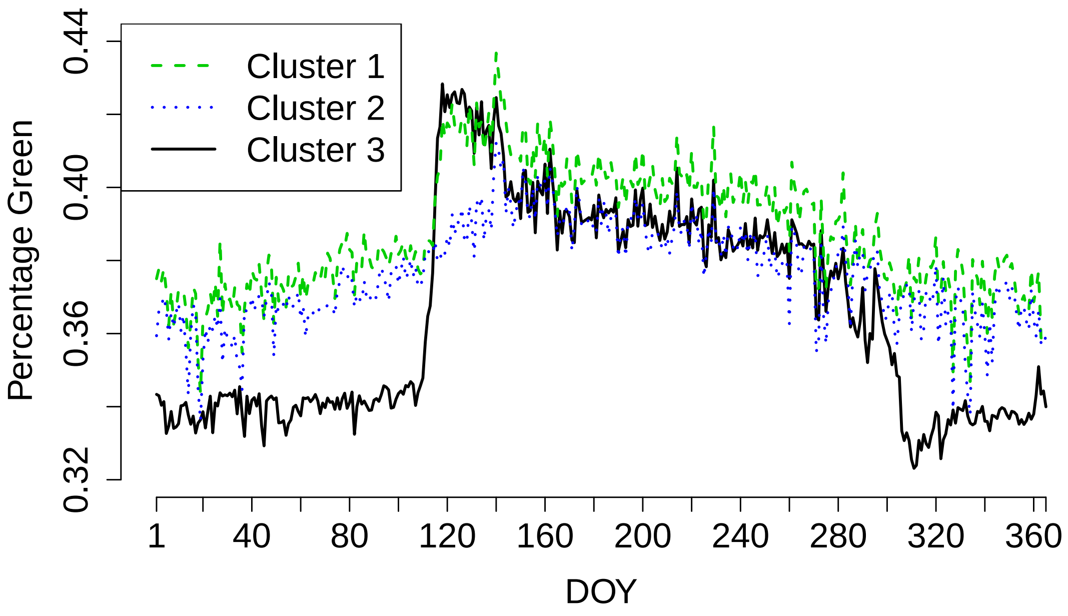


Figure 8.6. Resulting %greenness time series for clusters from uROI approach shown in Figure 8.5. Kranzberg data 2014.

Automation of uROI approach

However, as we seek for a fully automated procedure, we have to select the final uROI data-driven and automatically: Therefore, we compute again the optimality criteria OC1 and OC2 for the %greenness time series for each cluster, the time series are shown in Figure 8.6. In our example, the OC1 for the three clusters are 665, 299 and 1863, respectively, and the OC2 for the three clusters are 0.89, 0.81 and 0.96, respectively. Thus, both optimality criteria clearly select cluster 3 as final uROI, as expected by comparing the clusters with the original images.

Additionally, the optimal number of left singular vectors p and clusters k has to be determined automatically: Therefore, we carry out the SVD and the k-means clustering for all combinations of a grid of n_p values for p and of a grid of n_k values for k. Then, we compute the optimality criterion for each of the k clusters for all $n_p \cdot n_k$ combinations of p and k. The cluster with the largest value of the optimality criterion is the final uROI and the corresponding values for p and k are considered as optimal.

Implementation

The computational burden of the uROI approach is rather small, although an SVD of a very large matrix has to be carried out. Using the statistical software R (R Core Team, 2016), the singular vectors can be computed very efficiently with the package `irlba` by Baglama and Reichel (2015).

The developed R package `phenofun` contains all relevant functions to use the presented sROI and uROI methods. It is available on R-Forge at https://r-forge.r-project.org/projects/phenofun/, see Appendix C for an excerpt of the documentation.

Chapter 9: Resulting ROIs and phenological change points

In this chapter we compare the eROI, sROI and uROI approaches for the definition of ROIs with the goal to identify the SOS, MAX, EOS1 and EOS2 dates. These phenological onset dates are derived from the %greenness time series of eROI, sROI and uROI. Change points in these time series are modelled using the Bayesian multiple change point approach proposed by Henneken *et al.* (2013), based on Dose and Menzel (2004) and Schleip *et al.* (2006), and e.g. applied in Pope *et al.* (2013) and Menzel *et al.* (2015). First, we use the images from the scientific camera at Kranzberg forest, second we use the images from the open-access webcam data from http://www.foto-webcam.eu and then we challenge the approach and analyze images from several thousands of webcams from AMOS (http://amos.cse.wustl.edu/).

9.1. Scientific webcam

Figure 9.1 shows an overlay of the ROIs resulting from the eROI, sROI and uROI approaches. For the sROI approach, one expert pinprick is used, both sROI and uROI use optimality criterion 2. All approaches exclude the evergreen trees but differ in the amount and selection of pixels corresponding to deciduous trees. In the following our aim is to decide which of these ROIs is optimal with respect to phenological information.

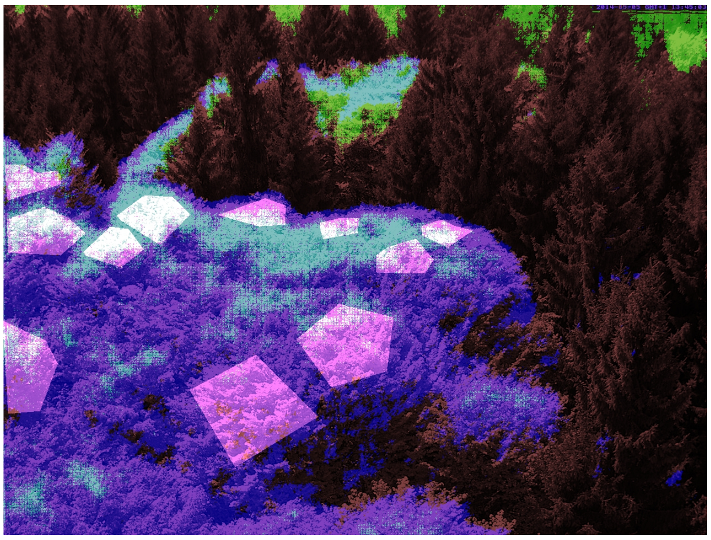


Figure 9.1. Overlay of the ROIs resulting from the eROI (pink polygons), sROI (green) and uROI (purple, overlay with sROI light blue) approaches for the images from Kranzberg data 2014.

The resulting %greenness time series for eROI, sROI and uROI for the data of Kranzberg 2014 are shown in Figure 9.2, see also Figure 9.3 for a comparison of the LOESS estimators. It seems that the time series resulting from the uROI and eROI approach are the most informatives because the decisive spring amplitude is slightly larger than for the sROI approach. However, the error variability is slightly smaller for the sROI approach. Thus, we need an objective measure to decide which time series is the most informative and therefore use the optimality criteria OC1 and OC2. The resulting values for OC1 and OC2 are shown in Table 9.1. It can be seen that the amount of information contained in the three time series is approximately the same, but the sROI time series carries the most information, followed by uROI and eROI.

Table 9.1. Values of optimality criteria OC1 and OC2 for different ROI approaches for images from Kranzberg data 2014.

	eROI	sROI	uROI
OC1	1848	1944	1863
OC2	0.967	0.972	0.961

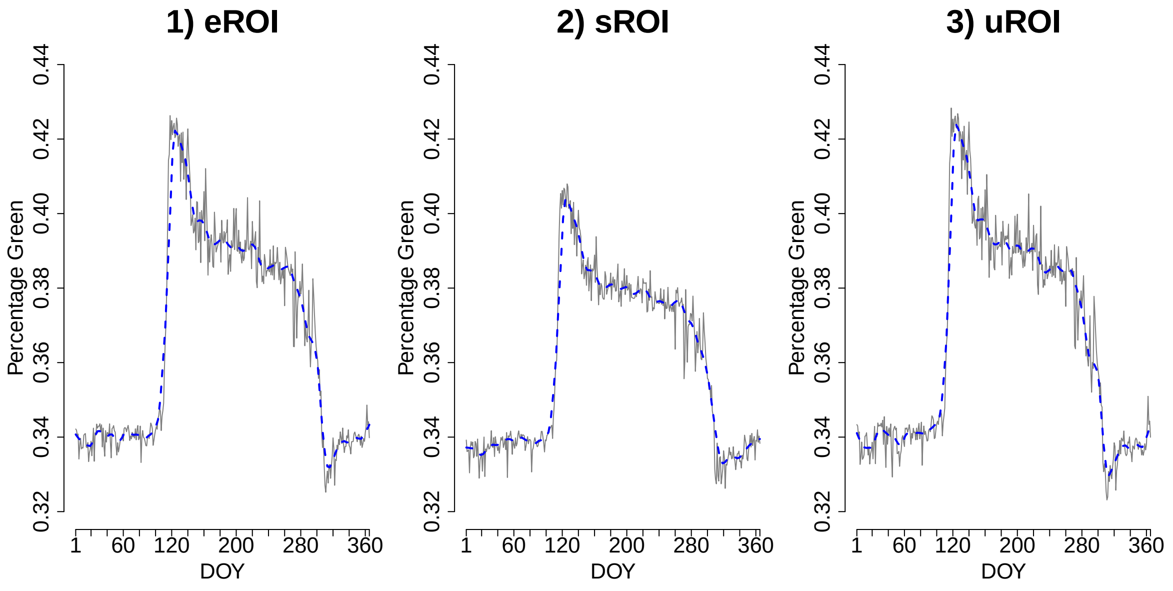


Figure 9.2. %greenness time series from Kranzberg data 2014 for three different approaches. Blue dashed lines show LOESS estimators for smoothed time series.

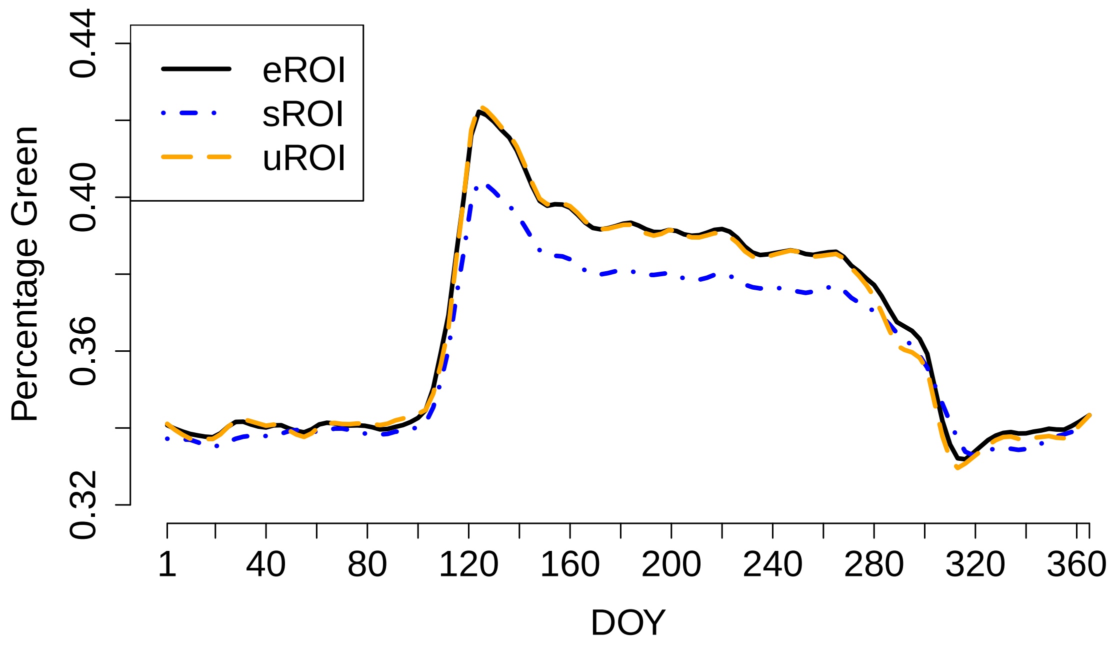


Figure 9.3. Comparison of LOESS estimators for smoothed %greenness time series for the images from Kranzberg data 2014.

For the identification of the SOS, MAX, EOS1 and EOS2 dates we use the approach of Henneken *et al.* (2013) and Menzel *et al.* (2015). There, the dates of change points in the univariate %greenness time series values are found via a Bayesian multiple change point analysis. The total number of change points is not specified by the researcher but determined automatically by the method.

In all three cases eROI, sROI and uROI, four change points turned out to be optimal to describe the annual course of %greenness comprising SOS, MAX, EOS1 and EOS2. Table 10.1 shows the results.

Table 9.2. Estimated phenological change points (DOYs) for different ROI approaches for images from Kranzberg data 2014.

	eROI	sROI	uROI	Date
SOS	105.0	105.0	105.1	April 15
MAX	119.0	119.0	118.9	April 29
EOS1	303.0	303.0	302.8	October 30
EOS2	308.0	308.0	307.7	November 4

Thus, the information about phenological variation resulting from eROI, sROI and uROI are almost identical, both regarding the optimality criteria OC1 and OC2 and regarding the estimated phenological change points. Therefore, we conclude that the sROI and uROI approaches outperform the approach of expert-based eROIs because additionally to the degree of information contained in the %greenness time series, they have the great advantage that they are fully automated.

9.2. Open-access webcams – foto-webcam.eu

For the open-access data from http://www.foto-webcam.eu, where expert-based ROIs were not available, we apply the uROI approach as follows: For each webcam site we compute clusters as described above for different settings, namely for $p \in P = \{12, 24\}$ singular vectors and $k \in K = \{4, 5, 6, 7, 8, 9, 10\}$ clusters for the k-means procedure. This results in $|P| \cdot |K| = 14$ partitions of the image and yields in total $\sum_{p \in P} \sum_{k \in K} k = 98$ clusters. For each of the 98 clusters we compare the OC2 of the %greenness time series defined above and thereby identify one optimal cluster for each webcam site.

Figure 9.4 shows the resulting %greenness time series for the optimal cluster of each webcam site. Most interestingly the resulting %greenness time series of the optimal cluster in each picture display different shapes. Whereas in case of Studentenstadt 2013 and Kolm-Saigurn 2014 a typical deciduous type is achieved, Marquartstein 2013 exhibits after the spring increase an early summer depletion and a second increase from DOY 200 again. We guess that this might be linked to quite dark / underexposed pictures in mid of July 2013. Additionally, careful checking of all pictures reveals a felling of a prominent deciduous tree in the foreground in front of the spruce (cluster 4).

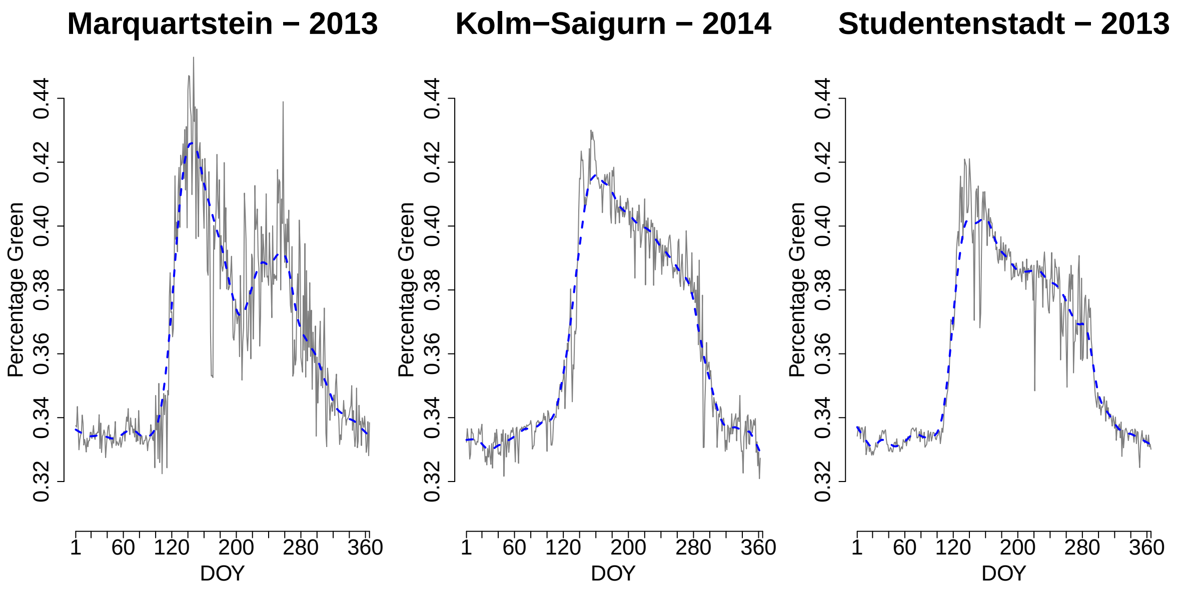


Figure 9.4. Resulting %greenness time series for sites Marquartstein (left), Kolm-Saigurn (center) and Studentenstadt (right) obtained by the uROI approach.

In Figure 9.5 - Figure 9.7 we show the optimal cluster for each webcam site along with all other clusters from the respective partition by plotting an overlay of a selected original image and the respective cluster in yellow. The clusters are ordered with respect to the information criterion OC2.

Figure 9.5 shows the clusters for the site Marquartstein. It can be seen that besides two sky clusters (10, 9) four clusters differentiated background vegetation, mainly mixed forests, in elevational belts (8, 7, 6, 5). The foreground vegetation was separated into a single dominating spruce (4) and deciduous vegetation (3), partly also assigned to (2) due to different winter aspects. The optimal cluster (1) consists of the upper crown part of the deciduous forest edge and a tree at the left side of the picture.

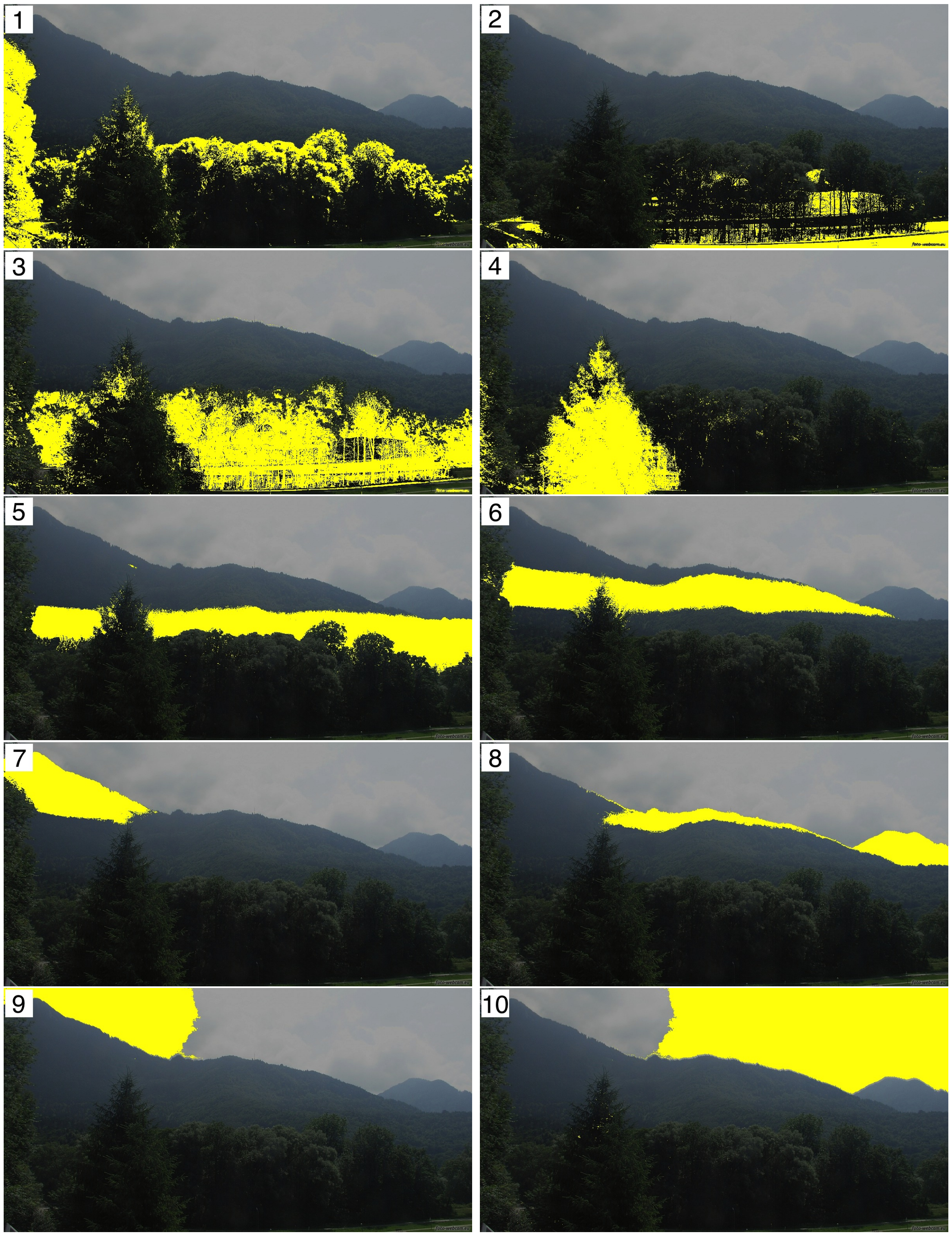


Figure 9.5. Resulting $k = 10$ clusters from uROI approach for the site Marquartstein, ordered with respect to OC2. Cluster 1 has the highest OC2.

Figure 9.6 shows the clusters for the site Kolm-Saigurn Sonnblickbasis. It displays clusters of meadows in the valley (3), roofs of buildings (10), wall structures and rocks (8) as well as water fall / rocks (7). All vegetation is divided into various clusters from the background (9, likely mugo pine) to deciduous broadleaf vegetation (2, likely alder) in the valley. The optimal cluster (1), however, are deciduous European larch (Larix decidua) turning bright green needles in spring into yellow in autumn.

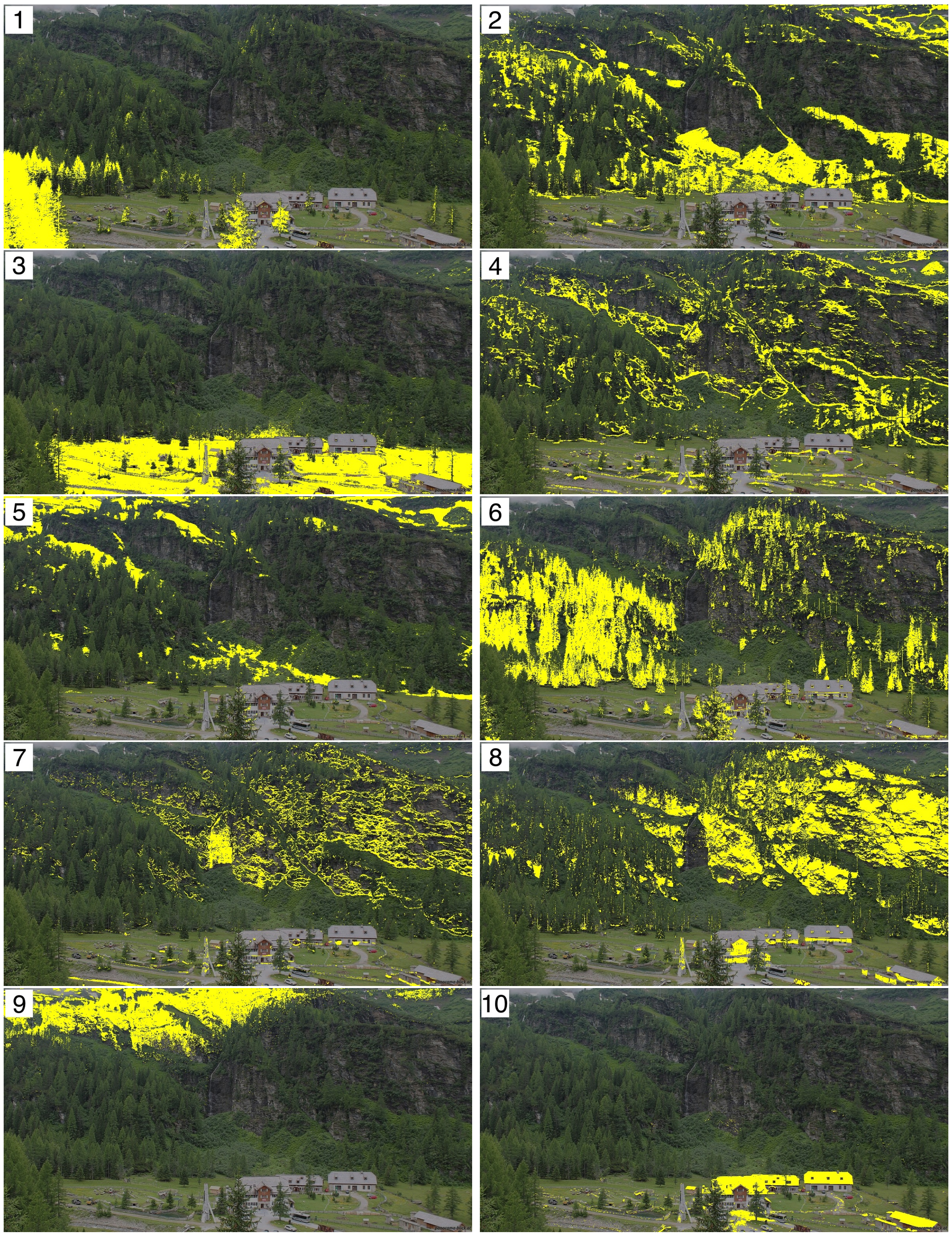


Figure 9.6. Resulting $k = 10$ clusters from uROI approach for the site Kolm-Saigurn Sonnblickbasis, ordered with respect to OC2. Cluster 1 has the highest OC2.

Figure 9.7 shows the clusters for the site Studentenstadt-Nord, which is a mixed urban landscape in the northern area of Munich. The clusters comprise sky (9) as well as infrastructure components such as flat roofs / parking space (8), facade and contour lines (7, 6) and streets (5). Cluster comprising predominantly vegetation are background vegetation (4), grass (3), deciduous vegetation including facades and seasonal shadows from the autobahn (2) as well as the optimal cluster (1) with pure deciduous vegetation components.

These results for images from open-access webcams emphasize the broad applicability of the presented methods. Additionally to the definition of an optimal ROI, the uROI method delivers a complete partition of the images. Thereby, vegetation is separated from infrastructure components and sky (Studentenstadt), and within the vegetation, different species can be discriminated (Kolm-Saigurn), or different elevations of similar vegetation (Marquartstein). Most importantly, the fully automated implementation allows to use the proposed methods on a larger scale of several hundreds or thousands of webcams as well. Thereby, the analysis is not restricted to scientific cameras, especially set up for phenological analyses, but the huge amount of publicly available, open-access webcam data can be exploited. This is demonstrated for webcams of the AMOS database in the following.

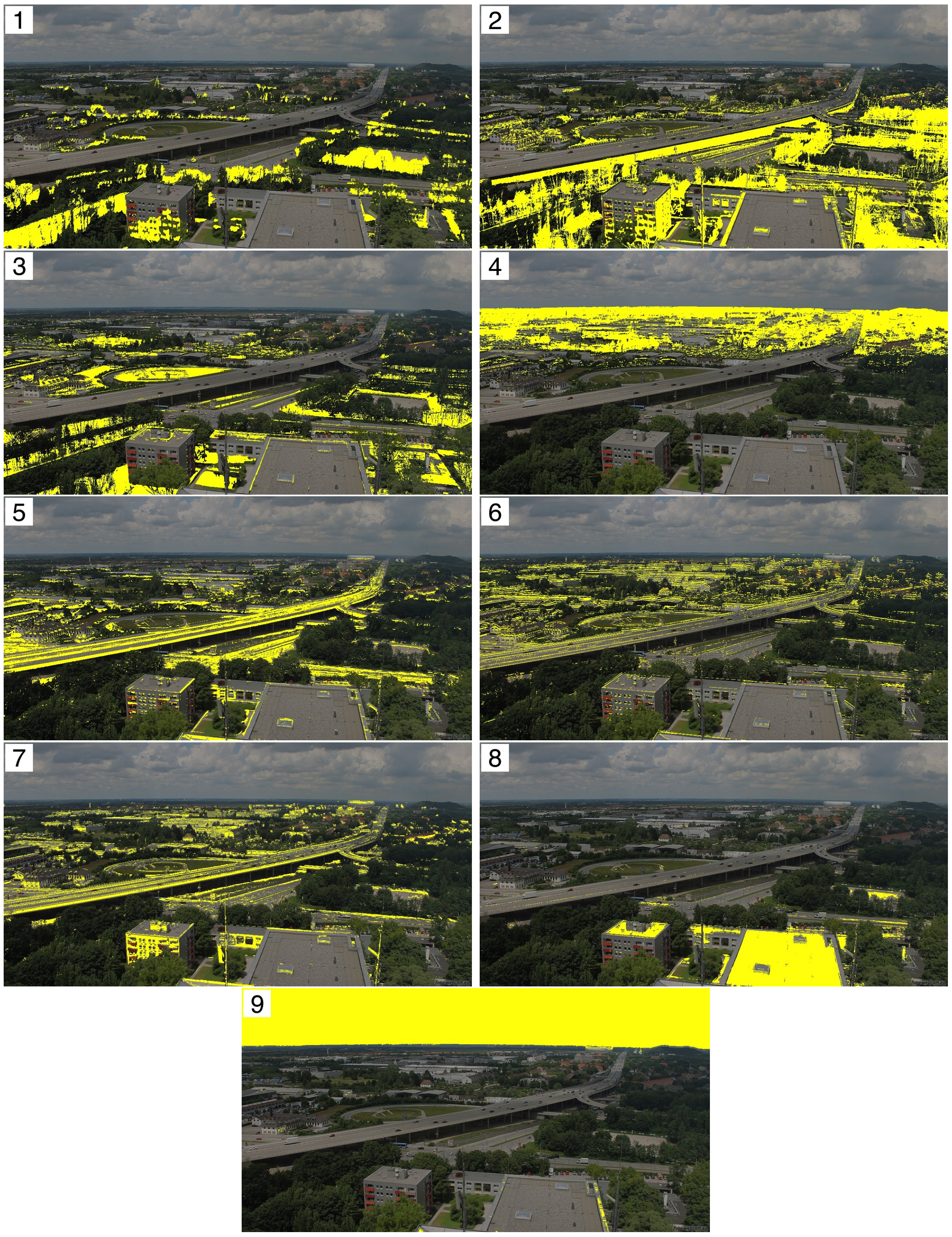


Figure 9.7. Resulting $k = 9$ clusters from uROI approach for the site Studentenstadt, ordered with respect to OC2. Cluster 1 has the highest OC2.

9.3. Open-access webcams – AMOS

Finally we challenge our methods and apply the uROI approach to a large database of webcams from AMOS. We automatically analyze all webcams which offer images for the year 2015. For most of these webcams, between one and four images per hour are available. We reduce the computational complexity by selecting images of a fixed hour during day time. This is challenging because the timestamp of the images reflects GMT (Greenwich Mean Time) instead of the local time. We restrict the analyses to images with GMT between 5pm and 6pm. This should deliver useful images for webcams located in the United States as well as for webcams located in Europe. We apply the uROI method to all thereby selected webcams. As number of left singular vectors we choose $p = 12$ and as candidates for the number of clusters we define the grid $K = \{4, 5, 6, 7, 8, 9, 10\}$. Finally, we use OC2 to find the best of the resulting clusters.

To automatically perform the analyses, we proceed as follows. The website offers a python script at http://amos.cse.wustl.edu/dataset which allows to download the data for a given webcam, year and month. We embed this python script in an R script and are thereby able to start the download directly with R. The format of the images is `yyyymmdd_hhmmss.jpg`. We restrict the analyses to images where the timestamp starts with 17 for the given hour. We distribute the computations on 16 cores on a server with four Intel Xeon E5-4620 CPUs with 2.20GHz and 8 cores each and 528 GB RAM in total. This results in a computation time of 23 days. Full results can be downloaded at http://bothmann.userweb.mwn.de/dissertation.html. For each webcam all ROIs from the best setting are shown together with the OC2 values and the resulting %greenness time series.

In total, for 13988 webcams a uROI analysis was started. For 13095 of these the entire analysis could be carried out without errors, i.e., ROIs and %greenness time series were derived. 9299 webcams offer enough data to compute OC2, for most of the other cameras there are large gaps in the year where no images are recorded. Figures 9.8 and 9.9 show example images, final uROIs and corresponding %greenness time series for three selected webcams. In all three cases, the uROI method finds ROIs with high phenological information. For ID 8221 (Figure 9.8, top) and ID 441 (Figure 9.8, bottom), the deciduous trees are identified and the resulting %greenness time series show clear structural changes in spring and autumn.

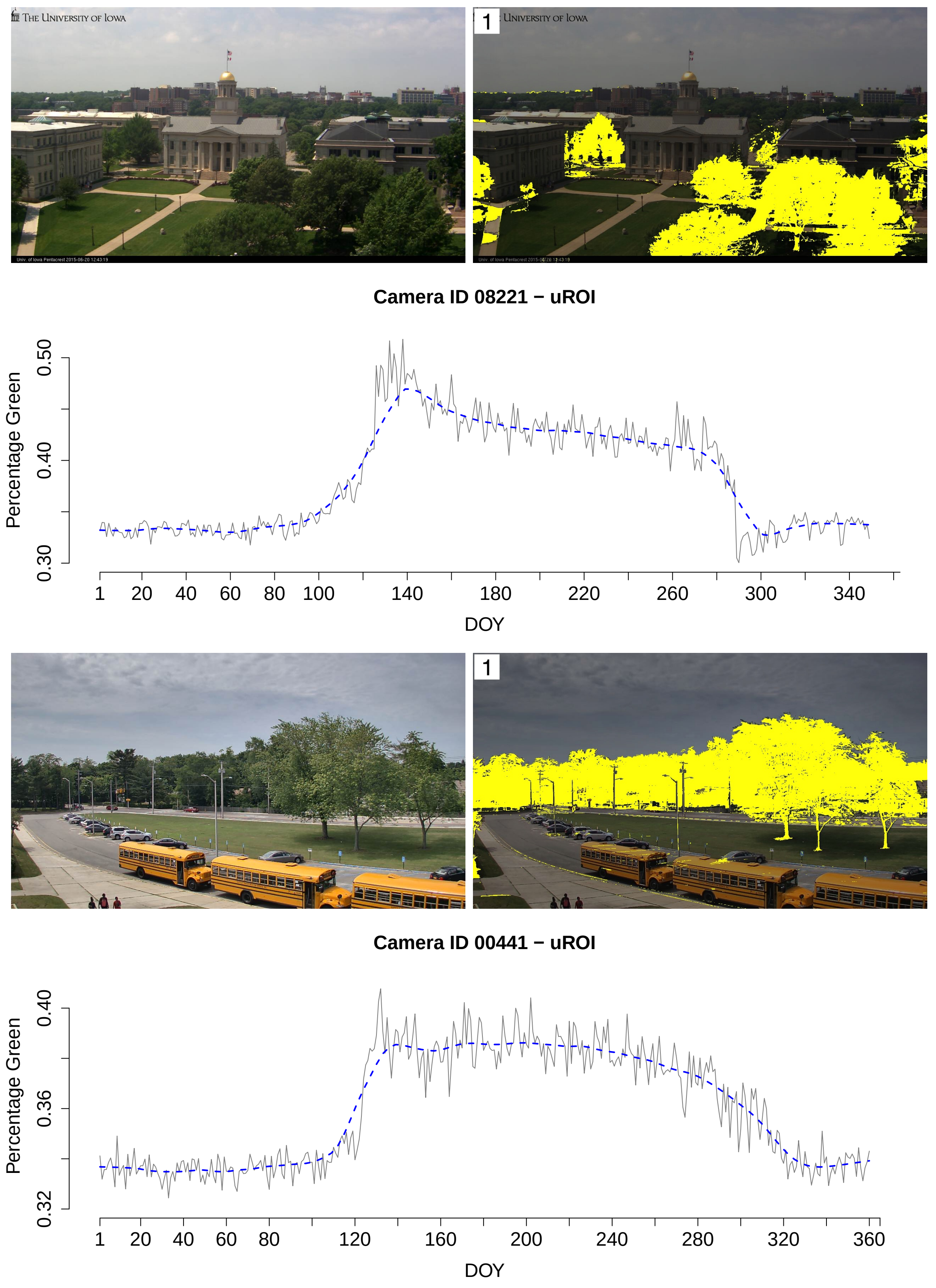


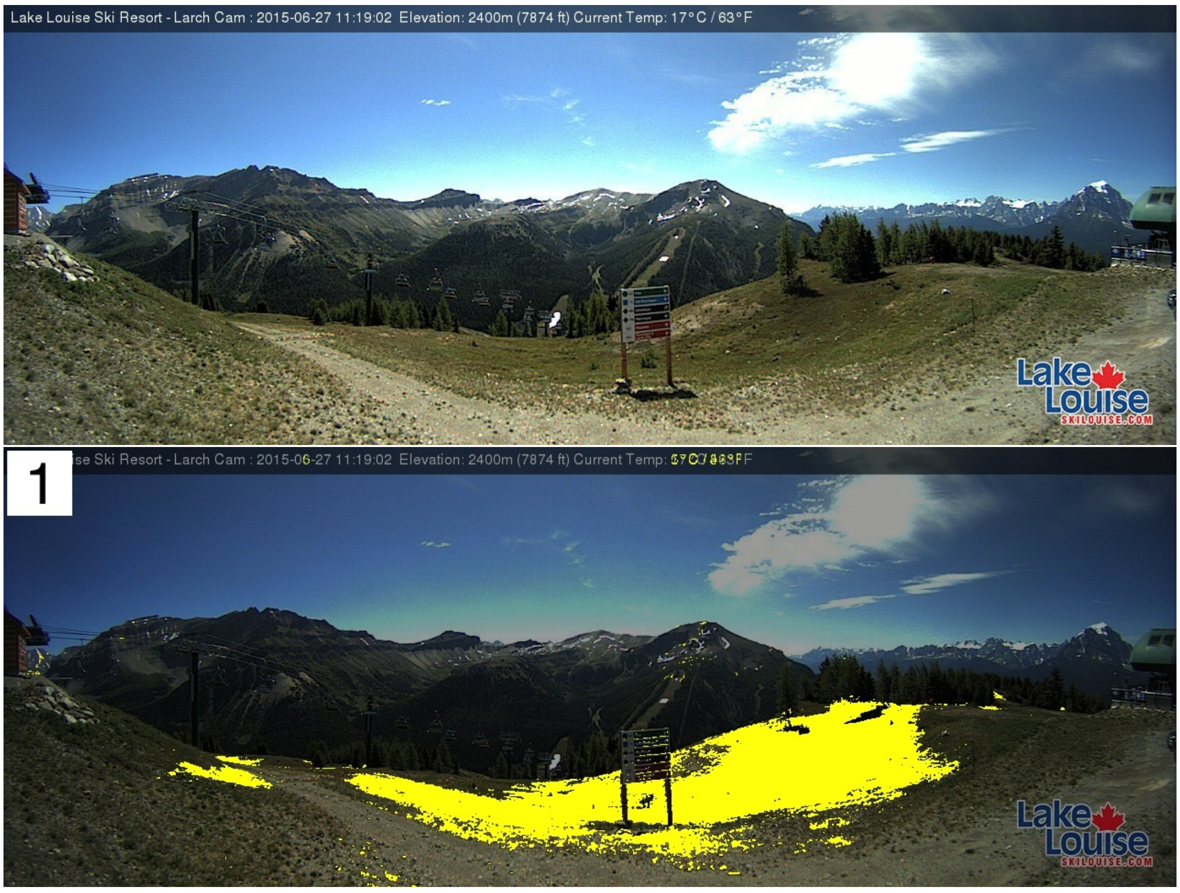
Figure 9.8. Example image, uROI and %greenness time series in uROI for cameras with ID 8221 (top) and ID 441 (bottom).

For ID 22231 (Figure 9.9), the grassland in the foreground is defined as uROI since no deciduous trees are captured by the camera. The resulting %greenness time series shows a clear seasonal pattern as well.

All in all, the results are satisfying but nevertheless a blind automatic proceeding is problematic due to the following issues:

- For some webcams, the field of view shifts significantly over the year. This means that some webcams are for example directed towards the north on some days and towards the south on other days. For these cases, meaningful uROIs can obviously not be extracted.
- In other cases, the field of view shifts only slightly. For these cases a registration of the images as preprocessing step would be beneficial.
- As mentioned above, we use images where the timestamp starts with 17 for the given hour. Due to the fact that the timestamp of the images does not refer to the time at the webcam location but to GMT, we get images at night for a variety of locations. Of course, the analysis does not make sense for images at night. Therefore, in a further analysis, the location of the webcams should be extracted from the AMOS website in order to determine the time zone.
- For some webcams, the image quality is rather poor and meaningful uROIs cannot be derived. Of course, the quality of the results depends on the image quality.

We conclude that the presented results for the AMOS data demonstrate that our new methods are suitable for the analysis on a large scale because the computing time is reasonable and the results are overall satisfying. Even if the above mentioned problems have to be solved in further research, this first analysis seems to be a very promising start.


Lake Louise Ski Resort - Larch Cam : 2015-06-27 11:19:02 Elevation: 2400m (7874 ft) Current Temp: 17°C / 63°F
Lake Louise
SKILOUISE.COM
1

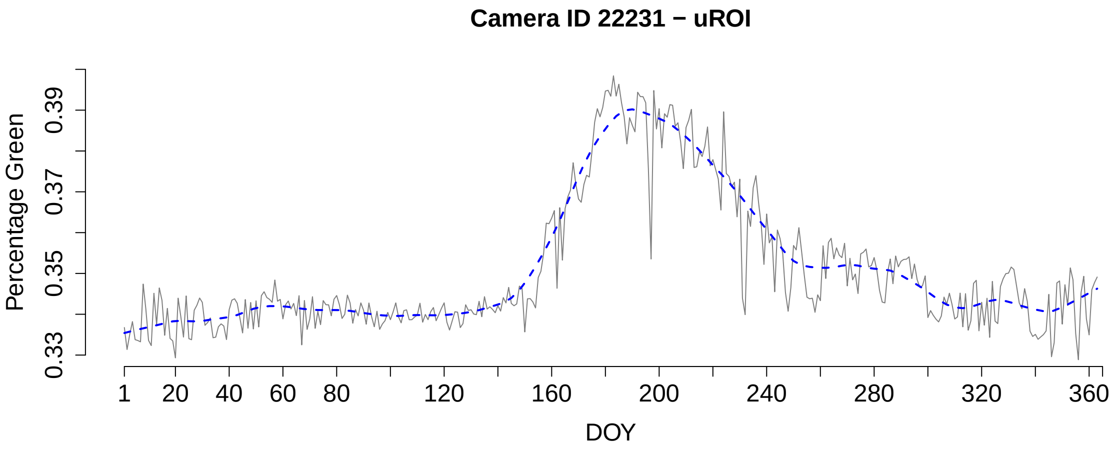


Figure 9.9. Example image, uROI and %greenness time series in uROI for camera with ID 22231.

Chapter 10: Supervised classification of webcam images

In this chapter we present an approach for phenological classification which is completely different to the approach presented in Chapter 9. There are two ways of describing phenological changes:

1. Either unique dates of season changes are defined and then all days between these change point dates are considered as *spring*, *summer*, *autumn* or *winter*, respectively (see Chapter 9).
2. Alternatively, all days of the year are separately classified as *spring*, *summer*, *autumn* or *winter* resulting in a discrete time series of seasons. Then, unique dates of season changes are defined by further processing this time series.

In this chapter, we pursue the second approach. Section 10.1 describes the methods. Section 10.2 presents the results of applying the approach to the scientific webcam data of Kranzberg and Brandwiese.

10.1. Methods

Eigenimages

Recall a singular value decomposition (SVD) of $\boldsymbol{X}$, where $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_T)$, the matrix containing the vectorized images, is of dimension $3N \times T$ and centered prior to the following steps. Through the SVD, $\boldsymbol{X}$ is decomposed into $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$. The left singular vectors of $\boldsymbol{X}$, stored in $\boldsymbol{U}$, can also be considered as “eigenimages”. Since we make use of a truncated version of the SVD and only compute the first $p \ll T$ left singular vectors, the matrix $\boldsymbol{U}$ is of dimension $3N \times p$. Each column $\boldsymbol{u}_j, j = 1, \dots, p,$ of $\boldsymbol{U}$ is a vector of length $3N$ and is hence of the same dimension as the vectorized images $\boldsymbol{x}_t, t = 1, \dots, T$. We can rearrange the dimensions of $\boldsymbol{U}$ such that we get a four-dimensional array $\boldsymbol{A}$ of dimension $n_1 \times n_2 \times 3 \times p$. Thus, we generate p eigenimages with three color channels and $n_1 \times n_2$ pixels.

The uROI approach described in Section 8.4 was also based on an SVD of $\boldsymbol{X}$. There, the matrix $\boldsymbol{U}$ was used for a clustering of pixels yielding ROIs. Then, the %greenness time series inside the ROIs were used for phenological classification. Here we directly use the outcome of the SVD for phenological classification as described below. This means that no ROIs and no %greenness time series are generated and that all three color channels are directly used for classification.

What is the interpretation of eigenimages? As in a usual principal component analysis, the first eigenimage explains the largest part of the variance. But since this is a rather theoretical interpretation, we visualize the effect of the eigenimage on the mean image in order to derive an illustrative interpretation. Figure 10.1 shows the mean image (top left) of all images of Kranzberg 2014 and the first eigenimage (top right). For the mean image, the average of all values over the year is computed separately for each color channel and each pixel. Additionally, the sum (bottom left) and difference (bottom right) of the mean image and a suitable multiple of the first eigenimage are shown. It can be seen that a low score on the first eigenimage means that the deciduous trees carry leaves (bottom right) while a high score means that no leaves are present (bottom left).

In this example, the 5%- and the 95%-quantile of the scores of the images on the first eigenimage where chosen as "suitable multiple". But these are only two possible values. If we use a grid of values between the minimal and the maximal score as weights and stick the resulting sums of mean image and weighted first eigenimage together, we create an "eigenvideo" where we can watch the leaves growing. The eigenvideo for the first eigenimage of the Kranzberg data can be accessed at `http://bothmann.userweb.mwn.de/dissertation/videos/kranzberg_eigenvideo1.mp4`.

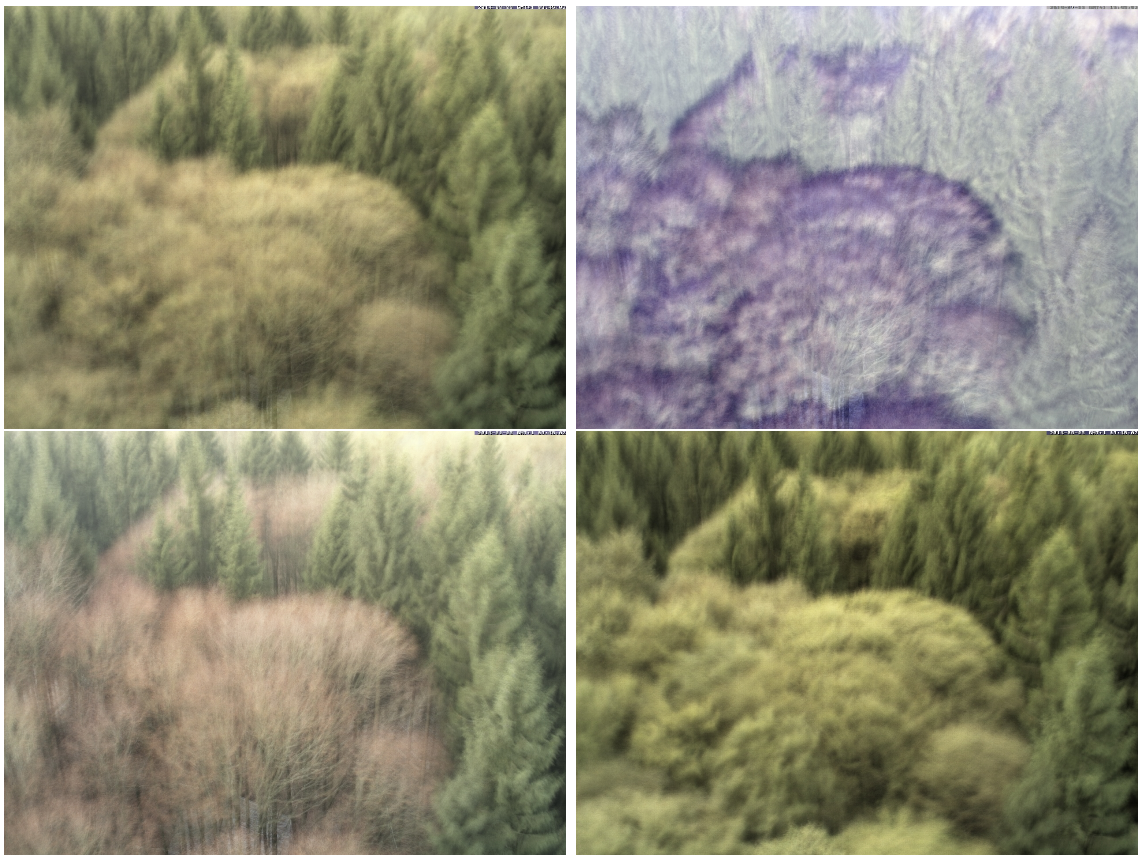


Figure 10.1. Mean image (top left), first eigenimage (top right), sum (bottom left) and difference (bottom right) of mean image and suitable multiple of first eigenimage for Kranzberg data 2014.

This interpretation goes along with the score time series corresponding to the first eigenimage. To compute the score s_{tj} of an image at time point t on the j-th eigenimage, the vectorized image $\boldsymbol{x}_t$ is projected on the vectorized eigenimage $\boldsymbol{u}_j$, i.e., $s_{tj} = \boldsymbol{x}_t^\top \boldsymbol{u}_j$. Figure 10.2 shows the score time series resulting from all images from Kranzberg 2014 projected on the first eigenimage where scores are averaged per DOY. It can be seen that the first eigenimage captures the gross seasonal variation: At the beginning of spring around DOY 110, the scores decrease significantly and stay on a low value until the beginning of autumn around DOY 300. In this period of the year, the deciduous trees carry leaves which corresponds to low scores on the first eigenimage as suggested by Figure 10.1 (bottom right). Note that this time series is generated on all three color channels, not only on %greenness values.

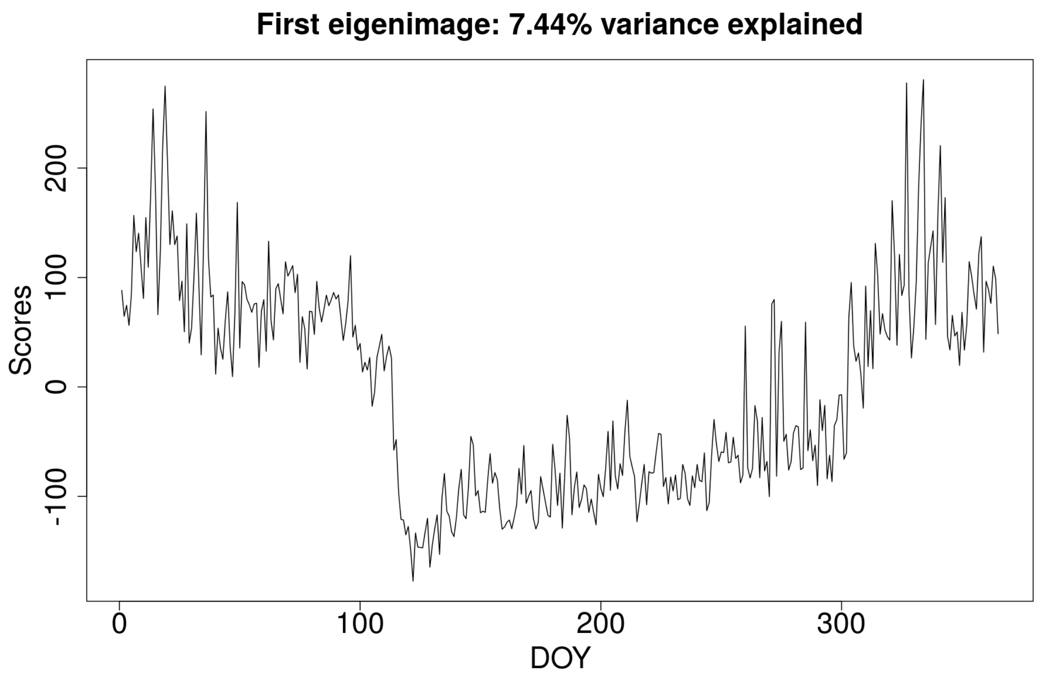


Figure 10.2. Score time series of the images on the first eigenimage for Kranzberg data 2014. The number in the header reflects the amount of variance explained by this eigenimage.

Eigenimages and scores for Brandwiese data

Figure 10.3 shows the sum (left column) and difference (right column) of the mean image over the year and a suitable multiple of the first, second and third eigenimage for the Brandwiese data from 2012. The first eigenimage covers illumination effects: A high score on the first eigenimage means that the image is rather bright or in other words – the sun is shining (top row). It can be seen that a high score on the second eigenimage means that the site is covered by snow while a low score means that no snow is present (middle row). The third eigenimage covers seasonal variation: A low score on the third eigenimage means that the deciduous trees carry leaves while a high score means that no leaves are present (bottom row). Links to the corresponding eigenvideos can be found in the caption of Figure 10.3.

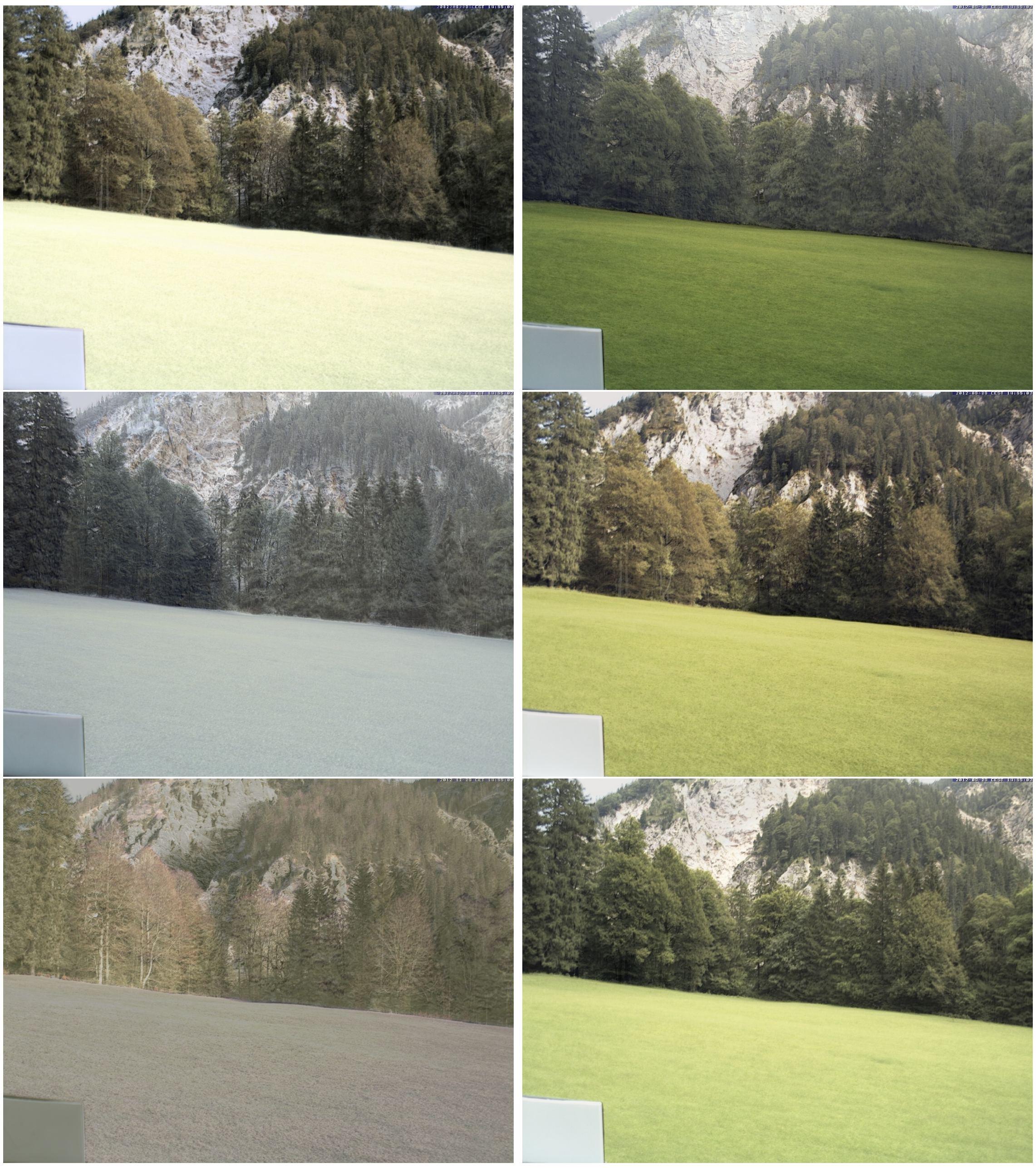


Figure 10.3. Sum (left column) and difference (right column) of mean image and a multiple of first (top row), second (middle row) and third (bottom row) eigenimage for Brandwiese data. Differences in the degree of illumination (top row), presence of snow (middle row) and amount of leaves (bottom row) can be seen. Links to the corresponding first, second and third eigenvideo.

This goes along with the score time series corresponding to the three eigenimages shown in Figure 10.4. The first score time series shows no specific pattern, which means that in every season there can be brighter and darker days. The second score time series has particularly high scores around DOY 300 and later from DOY 340 onwards. These are exactly the days where the site was covered by snow. And finally, the third score time series reflects the gross seasonal variation: Here, the scores decrease in spring and increase in autumn which corresponds to the interpretation of the eigenimage in Figure 10.3, where a low score means that the deciduous trees carry leaves.

This means that especially the scores on the third eigenimage are promising for the classification of images with respect to seasons and thereby for the detection of phenological change points SOS, MAX, EOS1 and EOS2. One could think of applying structural change point methods on the score time series to detect season onset dates. However, in the approach presented in this chapter, no methods for the detection of structural change points are used. Among others this is due to the fact that we cannot know beforehand which eigenimage will carry information about seasonal variation. For Kranzberg data it is the first eigenimage, for Brandwiese data it is the third eigenimage. We therefore use all p eigenimages to set up a supervised classification as explained in the following.

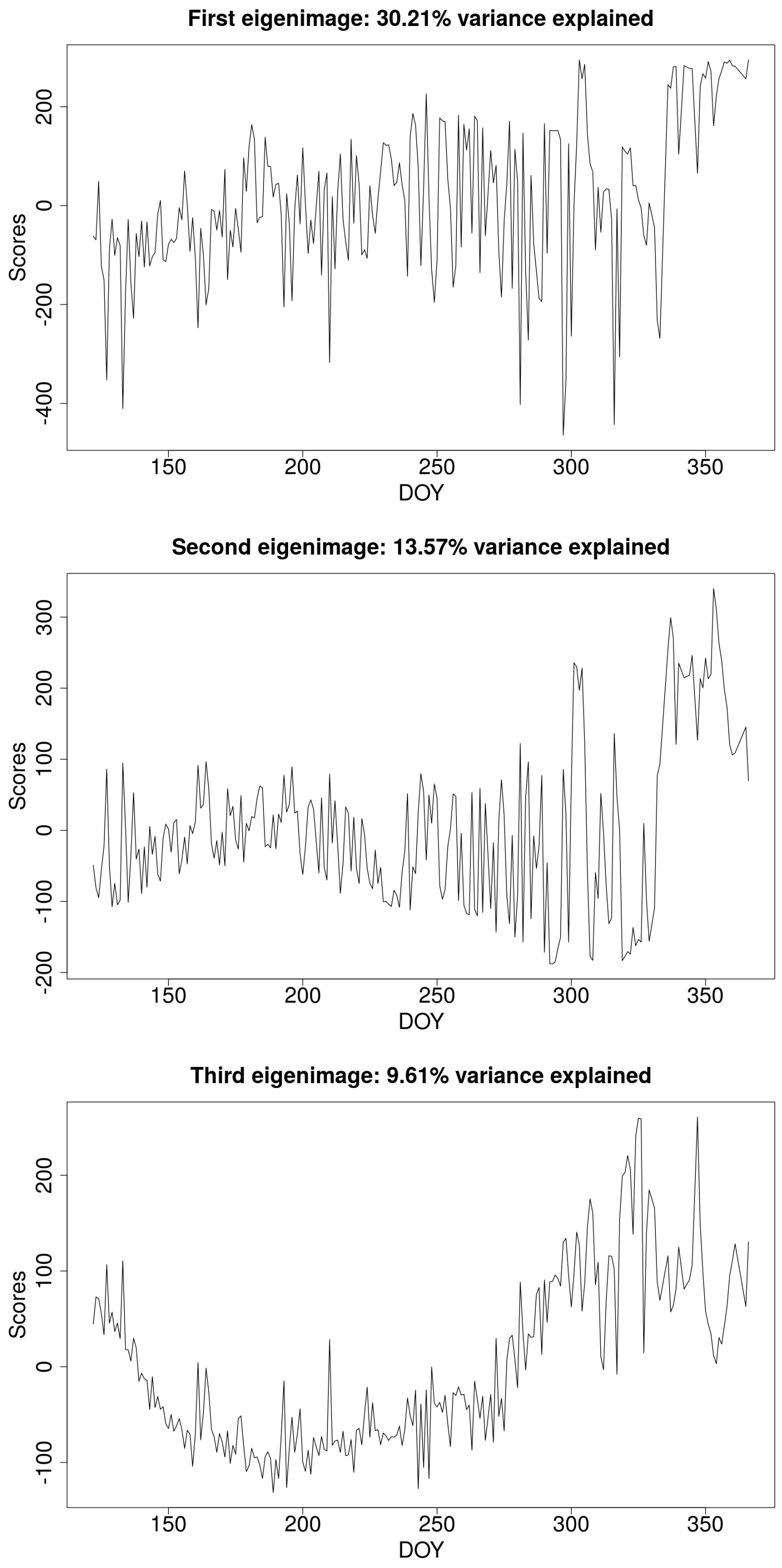


Figure 10.4. Scores of the images on the first, second and third eigenimage for Brandwiese data 2012. The number in the header reflects the amount of variance explained by each eigenimage.

Definition of score-based classification variables

The good interpretability of the eigenimages encourages us to compute scores on all eigenimages and to later use these scores for a supervised classification as follows: We project each vectorized image $\boldsymbol{x}_t, t = 1, \dots, T,$ on each vectorized eigenimage $\boldsymbol{u}_j, j = 1, \dots, p$, where $\boldsymbol{u}_j$ is the j-th column of $\boldsymbol{U}$. This results in p scores per given image, i.e., $s_{tj} = \boldsymbol{x}_t^\top \boldsymbol{u}_j$. This can be done for all T images and all p eigenimages, resulting in a matrix of scores $\boldsymbol{S} = \boldsymbol{X}^\top \boldsymbol{U}$ of dimension $T \times p$.

This procedure can be seen as a dimension reduction technique: Each original image is now represented by p scores and the corresponding p eigenimages instead of $n_1 \cdot n_2 \cdot 3 = 3N$ pixel color channels.

Supervised classification

We can use the score matrix $\boldsymbol{S}$ to set up a supervised classification procedure: For a training year, all days are manually and visually classified with respect to the season categories *spring*, *summer*, *autumn* or *winter*. Thereby, each image is assigned a “true” season for the training year, i.e., for each row of $\boldsymbol{S}$, the season is known. Then, a classification rule can be learned with standard classification methods, such as discriminant analysis, which use the columns of $\boldsymbol{S}$ as discriminating variables.

This classification rule can be used along with the eigenimages of the training year to classify the images of a new test year: First, each image $\boldsymbol{x}_t^*, t = 1, \dots, T^*$ of the test year is projected on each eigenimage of the training year, $\boldsymbol{u}_j, j = 1, \dots, p$, i.e., $s_{tj}^* = \boldsymbol{x}_t^{*\top} \boldsymbol{u}_j$ resulting in a score matrix $\boldsymbol{S}^* = \boldsymbol{X}^{*\top} \boldsymbol{U}$, where $\boldsymbol{X}^* = (\boldsymbol{x}_1^*, \dots, \boldsymbol{x}_{T^*}^*)$ is the matrix of all vectorized images of the test year. Then the classification rule is used to predict the season for each image using the columns of the score matrix $\boldsymbol{S}^*$. Finally, the predicted seasons are averaged per DOY to define a unique season for each DOY in the case of multiple images per DOY: For example using discriminant analysis, the posterior class probabilities can be averaged per DOY and then the season with the highest average posterior class probability is the predicted class.

Likely there will be some misclassified DOYs in practice, even if the classification accuracy is overall high. To determine unique DOYs of season changes, the following rule is applied to revise the predicted seasons: The onset date of a new season is defined as the first DOY at which for the first time three consecutive DOYs are classified as this new season. Thereby, the phenological change points SOS, MAX, EOS1 and EOS2 can be estimated.

To sum up, to detect season onset dates, an SVD is used as in the uROI procedure. However, here no ROIs are defined and no %greenness time series are generated but the SVD outcome is directly used for the classification. No subsequent structural change point detection is necessary but the manual classification of days of a training year into the four season classes. No unique change points are directly obtained by this approach but a broader applicability is given, as exemplarily shown below for the Brandwiese data.

10.2. Results

Kranzberg – classification of seasons

Figure 10.5 shows a scatterplot of the scores on the first two eigenimages of the Kranzberg data 2014 where colors refer to the true classes. The data of 2014 will be used as training data in the remainder whereas the data of 2013 will be used as test data. Already the first two eigenimages seem to differentiate well between the classes. Taking into account for example a number of $p = 12$ components promises a good classification accuracy.

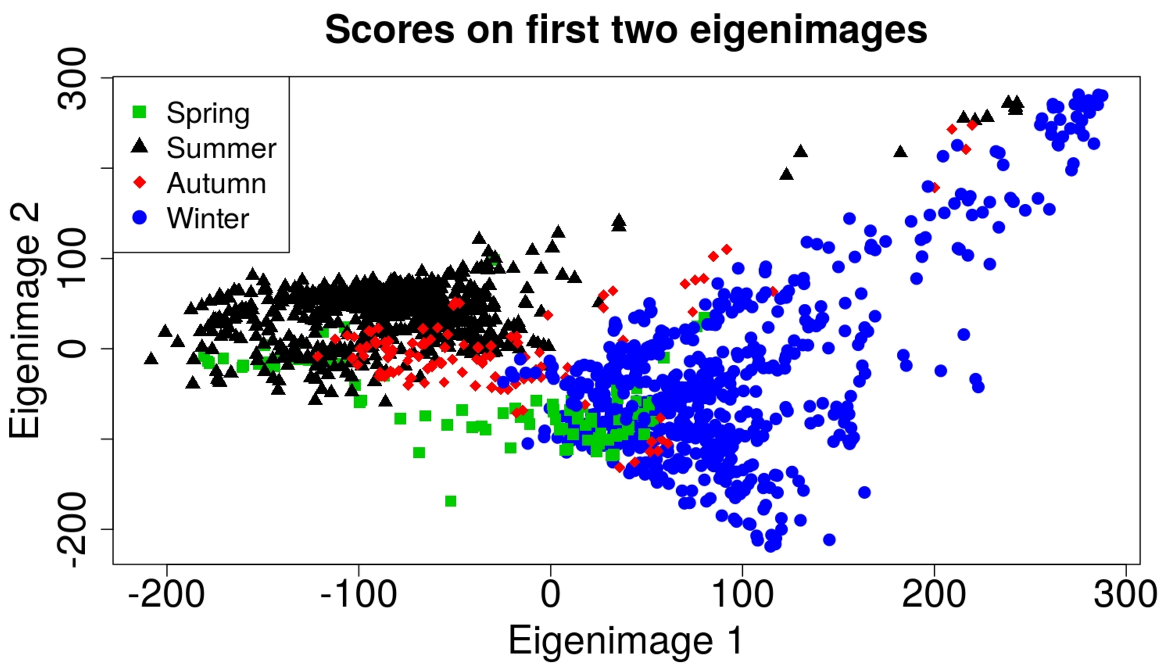


Figure 10.5. Scores of the images on the first two eigenimages for Kranzberg data 2014. Each point corresponds to one image.

To estimate the classification accuracy we perform a leave-one-out cross-validation (LOO-CV) with a quadratic discriminant analysis: The images of each DOY are considered successively as test data and the images of all other DOYs as training data. We predict posterior class probabilities for each image of the test data and gather these on DOY level to get a predicted class for each DOY. Figure 10.6 shows true and predicted classes per DOY: Only 1.4% of the DOYs are misclassified.

Additionally we classify DOYs of the test year 2013. Therefore we first compute the scores of each image of the test year on the eigenimages of the training year. With these scores we classify each DOY of the test year with the classification rule learned on the training year. Figure 10.7 (top) shows predicted classes for each DOY of the test year where images were only available starting from DOY 105 due to technical reasons. About 90% of the DOYs seem to be correctly classified. Nevertheless the results are not entirely satisfying because they do not indicate unique change point DOYs, especially for the transition from summer to autumn.

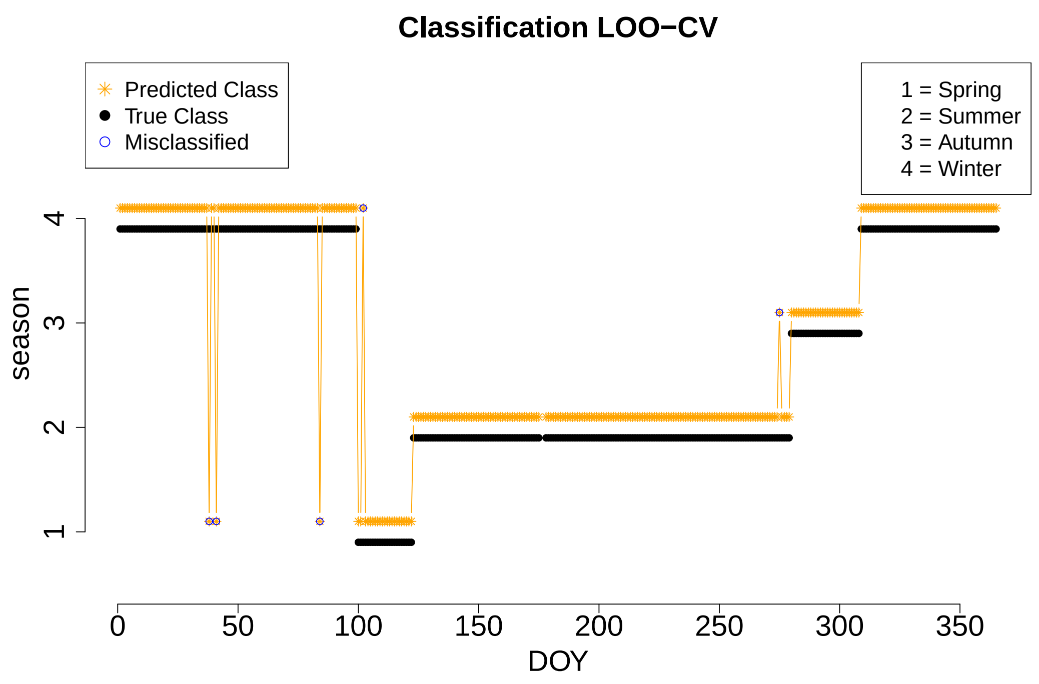


Figure 10.6. True seasons and LOO-CV-predicted seasons per DOY, Kranzberg data 2014.

Therefore, we apply the additional step introduced in Section 10.1: The onset date of a new season is defined as the earliest DOY at which for the first time three consecutive DOYs are classified as this new season. The resulting change point DOYs are compared with the true change point DOYs in Table 10.1 and visualized in Figure 10.7 (bottom). The resulting change points are estimated close to the true change points SOS, MAX, EOS1 and EOS2.

Table 10.1. True and predicted phenological change points for images from Kranzberg data 2013.

	True DOY	Predicted DOY	True date	Predicted date
SOS	116	117	April 26	April 27
MAX	133	133	May 13	May 13
EOS1	279	282	October 6	October 9
EOS2	299	296	October 26	October 23

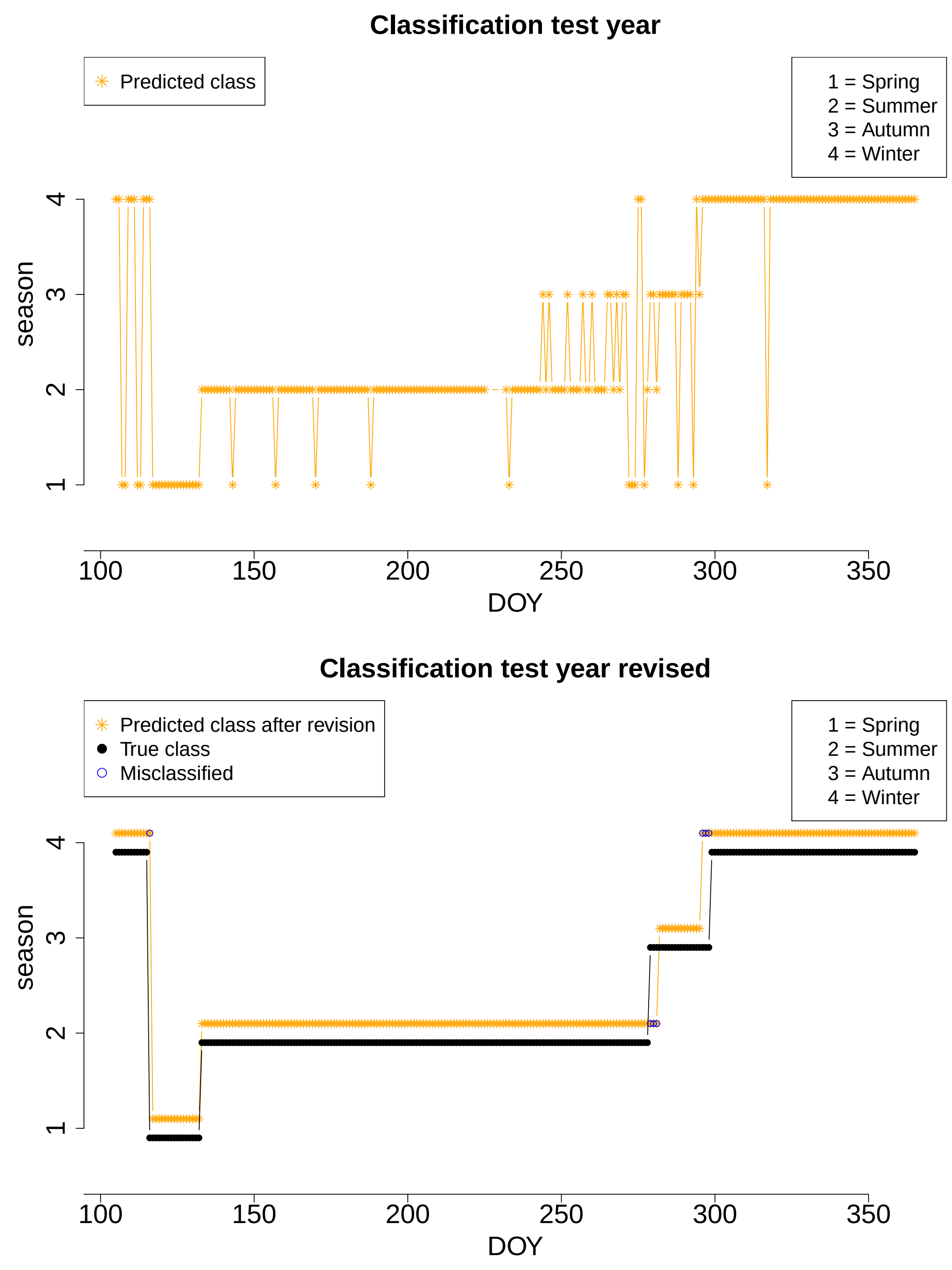


Figure 10.7. True seasons and predicted seasons per DOY for test year 2013 (top). True seasons and predicted seasons per DOY for test year 2013 after revision (bottom), Kranzberg data.

Brandwiese – classification of seasons

We carry out the same analysis for the Brandwiese data. As shown in Figure 10.3, the first eigenimage reflected illumination effects here. Thus Figure 10.8 shows a scatterplot of the scores on the second and third eigenimage of Brandwiese data 2012 reflecting the presence of snow and the amount of leaves on the deciduous trees, respectively. The data of 2012 will be used as training data in the remainder whereas the data of 2013 will be used as test data. Similar to the results for the Kranzberg data, these two eigenimages alone seem to separate the seasons from each other.

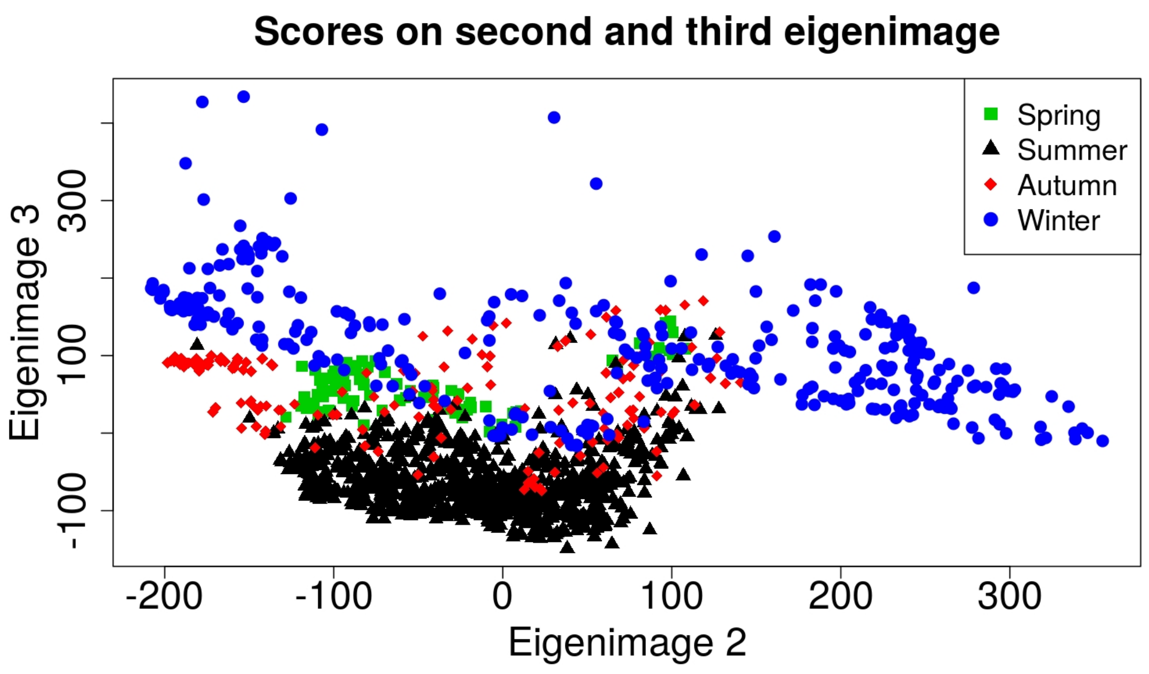


Figure 10.8. Scores of the images on the second and third eigenimage for Brandwiese data 2012. Each point corresponds to one image.

Similarly as above, we use the scores on the first $p = 12$ eigenimages for a supervised classification. Figure 10.10 (top) shows true season classes along with LOO-CV predicted classes where images were only available from DOY 122 onwards. Only four days are misclassified, this corresponds to a classification error of 1.7%.

Figure 10.10 (bottom) shows the classification for the test year 2013. The classification did not really work: No DOY is classified as spring day at all and around DOY 300, no clear class could be identified. This can be explained by the poor image quality in 2013: At some point, humidity entered the camera, this resulted in very foggy images with other colors than in the training year. Figure 10.9 shows two example images on DOY 88 (March 29) and DOY 303 (October 30). It is not surprising that clear seasons cannot be derived from this data.

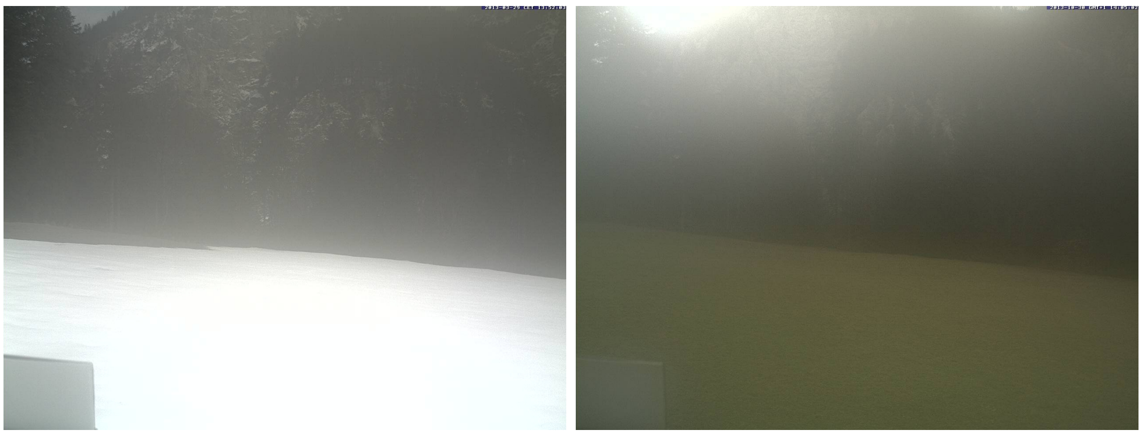
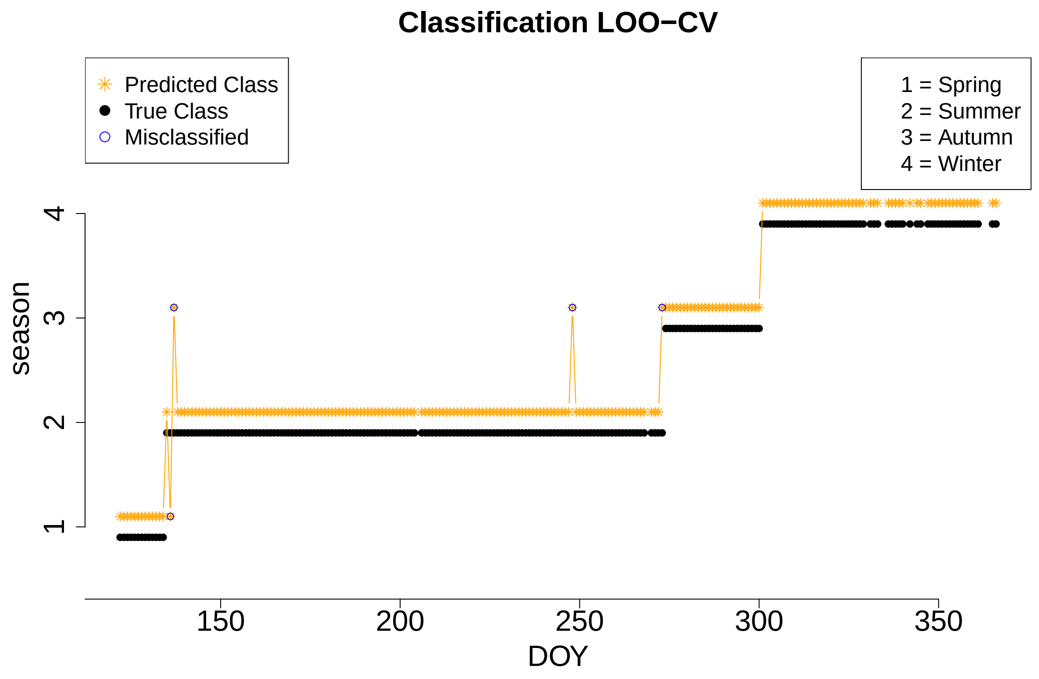


Figure 10.9. Example images of Brandwiese on DOY 88 (left) and DOY 303 (right) from year 2013.



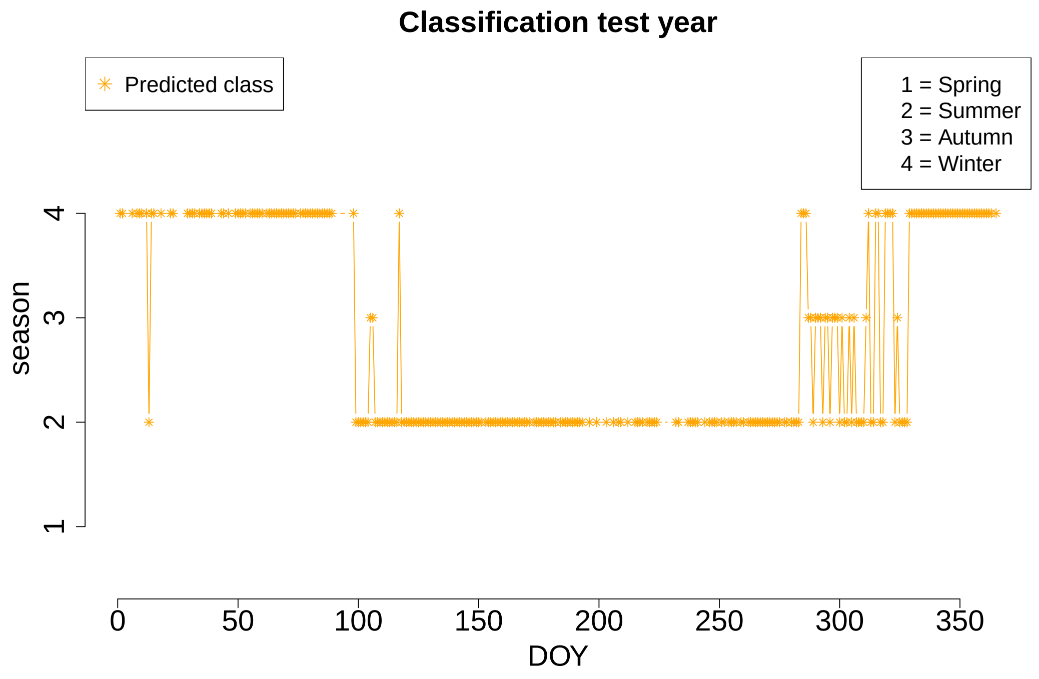


Figure 10.10. True seasons and LOO-CV-predicted seasons per DOY for training year 2012 (top). Predicted seasons per DOY for test year 2013 (bottom), Brandwiese data.

Brandwiese – classification of snow days

Another application of the presented method is the automatic detection of snow days. Instead of classifiying the days with respect to the four seasons, we can classify them as *day with snow* or *day without snow*. Figure 10.11 shows a scatterplot of the scores on the first two eigenimages where points are colored with respect to the two classes *day with snow* and *day without snow*. It can be seen that these two eigenimages alone separate the two classes almost perfectly.

This application of our method could be very useful for the detection of unexpected frost days, as for example dealt with in Menzel *et al.* (2015).

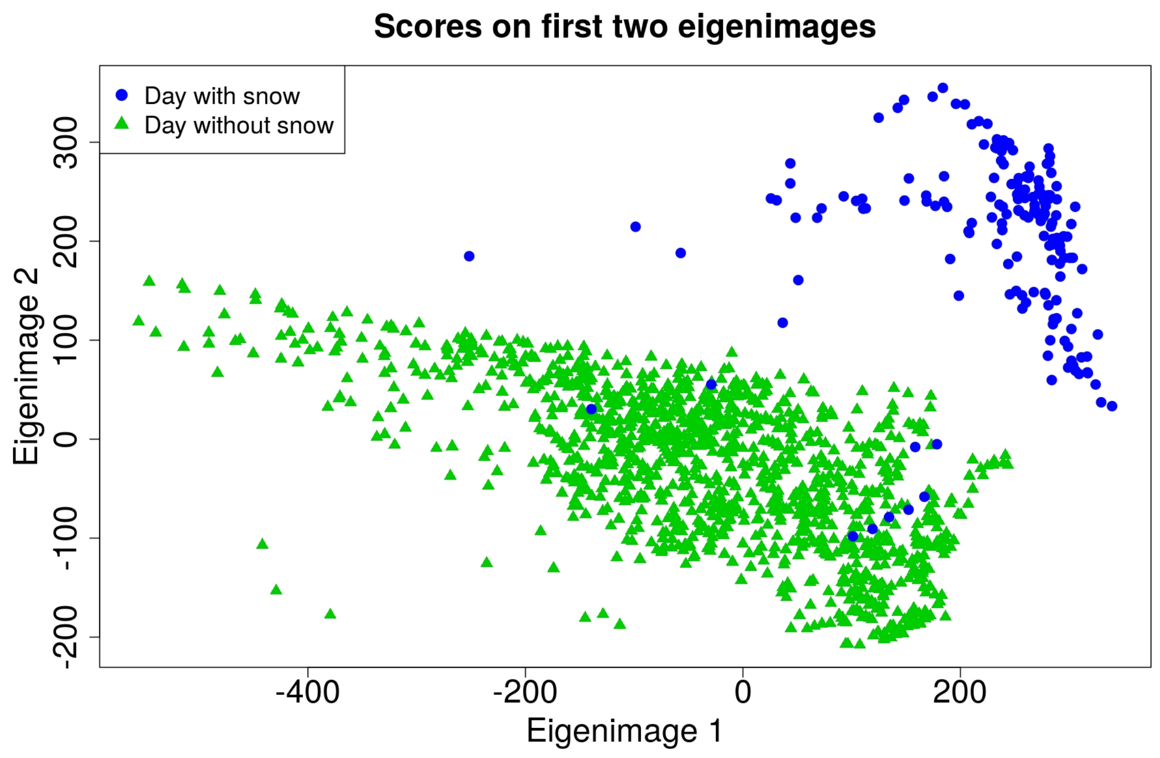
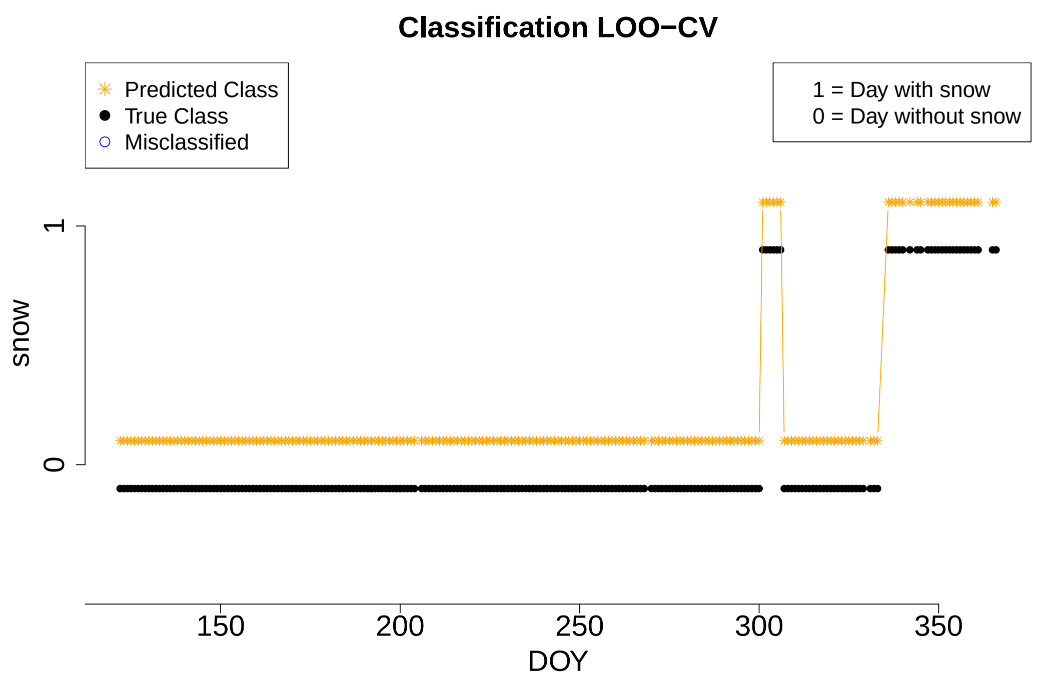


Figure 10.11. Scores of the images on the first and second eigenimage for Brandwiese data 2012. Each point corresponds to one image.

And indeed, the LOO-CV classifies the days perfectly. As can be seen in Figure 10.12 (top), each day is correctly classified as either *day with snow* or *day without snow*. Regardless the poor image quality in the test year 2013 (recall Figure 10.9), even the classification in the test year is satisfying with a misclassification rate of 3.7%, see Figure 10.12 (bottom). This is due to the fact that almost half of the image is dominated by grassland in the lower part so that on a snow day half of the image is white. Even with foggy images due to humidity in the camera, this striking feature of the images can be identified by our method. A visual check of the misclassified days revealed that either the image was so dark that barely anything could be seen or that the snow started melting such that only parts of the grassland were covered with snow.



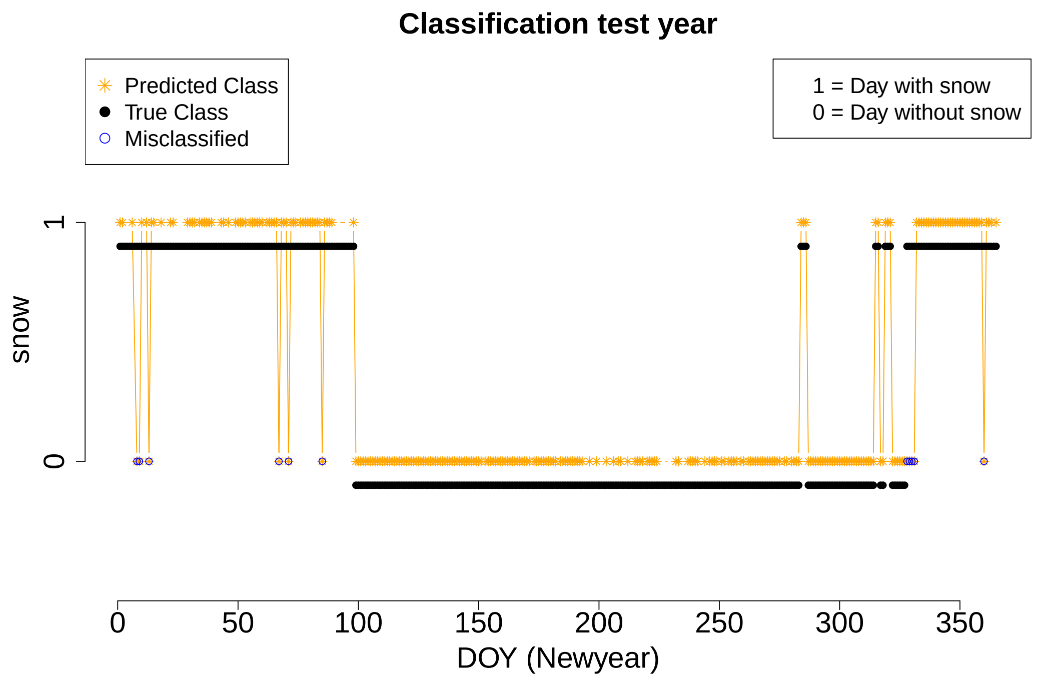


Figure 10.12. True classes and LOO-CV-predicted classes per DOY for training year 2012 (top). True and predicted classes per DOY for test year 2013 (bottom), Brandwiese data.

Chapter 11: Discussion and outlook

11.1. Summary and research goals

Summary

In this project, we developed an efficient solution for a big data problem from applied statistics: Given a set of digital webcam images showing the same motive over the course of a year, we developed a method to identify those pixels of the image which are most important in the sense of phenological information. In the end, onset dates of beginning *spring*, *summer*, *autumn* and *winter* should be determined from the image data. The developed methods should allow to scale up the analyses to several hundreds or thousands of webcams, i.e., the methods should run fully automated and with reasonable computational effort.

Therefore, we first developed two fully automated methods for the definition of regions of interest (ROIs). From these ROIs, time series of percentage greenness (%greenness) were derived and methods for the identification of structural changes delivered the desired onset dates. In an alternative approach, eigenimages were computed and a subsequent supervised classification yielded the onset dates. We showed that our new methods for the definition of ROIs led to %greenness time series which are at least as informative as those resulting from an expert-based definition of ROIs. All methods yielded similar onset dates. Moreover, one of our methods comes along with a partition of the image motive which can be interpreted with respect to different vegetation types, infrastructure and sky pixels. A careful implementation and the development of the R package `phenofun` allows to use the methods on a larger scale as well which was demonstrated on a database of 13988 webcams.

Research goals

In the introduction on page 84 we defined four main research goals of the project which were achieved as follows:

1. *Develop a method for the automated definition of ROIs:* In Chapter 8, we proposed two new methods for the automated definition of ROIs. More

precisely, we proposed two alternatives for the expert-based definition of regions of interest (eROI), namely the semi-supervised regions of interest (sROIs) and the unsupervised regions of interest (uROIs). All ROI approaches lead to ROIs for the image data. Based on these ROIs, %greenness time series can be computed and analyzed. Structural changes in these time series indicate phenological change points. We found that the time series based on the sROI and uROI approaches are as informative as those based on the expert-based ROI approach with respect to two optimality criteria for the example of a scientific webcam in an experimental forest. See the second research goal for details on these optimality criteria.

The uROI method was also successfully applied to open-access webcam data yielding phenologically interesting partitions of the images while clearly separating infrastructure, rocks, buildings and different vegetation types. However, the most “deciduous” vegetation type was either the upper crown part, as also shown for the scientific webcam, or deciduous larch. Quite promising is the fact that different vegetation types such as alpine meadow and urban grass could be clearly separated. Additionally, no manual action is necessary to define the ROIs. Hence, we conclude that our new sROI and uROI methods for the definition of ROIs are favorable, especially when working with a large amount of webcams at different locations.

2. *Develop a reasonable optimality criterion to evaluate the amount of information contained in the resulting ROIs:* In Section 8.1 we presented two optimality criteria for the comparison of the ROI approaches. Both criteria evaluate the phenological information contained in the %greenness time series derived from the pixels inside the ROIs. The first approach tries to find the striking structural change in spring and compares the signal-to-noise ratios subsequently. The second approach fits an optimal template %greenness time series and computes the correlation of %greenness inside the ROI with this template time series. Our analyses showed that the second approach is preferable since the resulting ROIs are visually more convincing.

3. *Develop a procedure for determining the dates of SOS, MAX, EOS1 and EOS2 for given image data:* The different approaches for the definition of ROIs were used to extract the annual course of deciduous vegetation

greenness in Chapter 9. Therefore, a method from the literature called *Bayesian multiple change point analysis* was used to detect change points in the %greenness time series. This led to almost identical results in terms of start, peak and end of season dates for eROI, sROI and uROI. The quality of this Bayesian multiple change point analysis was not investigated in this thesis. This means that other methods for the identification of change points could lead to similar or even better results. However, for the comparison of the different ROI approaches we felt that it is sufficient to use the established method from the literature.

In Chapter 10 we proposed a method for deriving eigenimages from the image data followed by a supervised classification. Thereby, each image can be classified with respect to the four seasons *spring*, *summer*, *autumn* and *winter*. It is even possible to classify the images of a new year considering the eigenimages and classification rule of a training year. The definition of eigenimages is based on a singular value decomposition of the vectorized image data, the resulting eigenimages of the data set have a meaningful interpretation. For example, a high score on a specific eigenimage implies that the deciduous trees on the image are carrying leaves while a high score on another eigenimage implies that the site is covered by snow. The scores of the original images on these eigenimages serve as classification variables for the supervised classification using standard statistical classification methods.

The added value of this approach is that images cannot only be classified with respect to seasons. For example, the images can be separated into images where the site is covered with snow and images where no snow is present. The classification accuracy was even higher than for the classification of seasons because snow on main parts of the image is more striking than leaves on the deciduous trees.

4. *Implement the methods so efficiently that the analysis can be scaled up to several hundreds or thousands of webcams:* All methods are implemented in the statistical software package R and computing times and storage needed are kept as low as possible. For the large AMOS database, the analysis of 13988 webcams lasted about 23 days on a server with 528 GB RAM where the analyses were parallelized to 16 cores. The resulting R package `phenofun` containing all relevant functions to use the presented

methods is available on R-Forge at https://r-forge.r-project.org/projects/phenofun/, see Appendix C for an excerpt of the documentation.

Strengths and limitations

Automated definition of ROIs. One strength of the sROI approach is that ROIs are defined mainly data-driven while the researcher can simultaneously control the definition by setting the pinprick on the favored region of the image. Additionally, the resulting %greenness time series is slightly more informative than with the expert-based ROI approach.

The first strength of the sROI approach could at the same time considered to be a weakness: Through the fact that the pinprick has to be set manually and at a reasonable position, the approach is not optimal when heading for big data applications and analyses where several hundreds or thousands of webcams shall be analyzed simultaneously. However, with the option of setting multiple pinpricks at random locations and selecting the best ROI with the proposed optimality criteria, we also provide a fully automated version of the sROI approach. Nevertheless, this option comes along with an increase in computing time because the analysis has to be carried out for each pinprick. Depending on the given images, a rather large number of random pinpricks could be necessary.

This potential limitation of the sROI approach is not present in the uROI approach: The uROI approach is fully automated and purely data-driven, no manual action is required to define the ROIs and the computing time for this approach is rather small. This property is in particular of great importance for the mentioned big data applications. Moreover, the resulting %greenness time series are actually as informative as with the eROI approach. This finding has to be emphasized since the definition of expert-based ROIs is accompanied by a large amount of manual work while the sROI and uROI approaches require no manual action at all.

However, there is also a limitation: Although the approaches work promising for the data sets considered here, we cannot blindly guarantee that results are equally satisfying for other data sets. It happens that there are more than only one resulting cluster which is important for the identification of phenological patterns, as shown for example for the data from Marquartstein, see Figure

9.5, and it could happen that there is no cluster which is pure enough for the identification of phenological patterns.

Nowadays, there is a large variety of open-access collections of webcams as for example in the two sources used in this thesis: http://www.foto-webcam.eu/ and the "Archive of Many Outdoor Scenes" (AMOS, http://amos.cse.wustl.edu/) with almost 30000 webcams located around the world. With our new and automated approaches it is now possible to exploit these databases for phenological purposes without additional costs for installing and maintaining cameras. This is a fundamental result of our work because by this means many existing webcam databases can be directly used for phenological research.

Supervised classification approach. A strength of the supervised classification approach based on scores on the eigenimages is that each image is assigned to a unique class or season, respectively. This information can be useful for example when a sudden onset of winter happens at an unexpected time in year. Then, the respective days are classified as winter but the surrounding days are not affected.

The drawback of this approach is that a considerable amount of work is required from the researcher: All images or at least all days of the training year have to be classified manually with respect to the defined classes. For big data applications with a large amount of different webcams, this may not be feasible because this manual work has to be carried out once for each webcam. Furthermore, the procedure has to be trained again when the focus of the camera shifts or when the vegetation changes substantially, for example after trees were cut down or have grown significantly.

11.2. Possible extensions

This work is considered as a starting point for a larger project connecting phenologists, computer scientists and statisticians. As next step it is planned to refine the first AMOS analysis and to solve the problems mentioned in Section 9.3. Thereby, season onset dates should be derived for each webcam location and year yielding high-resolution spatio-temporal data. Then, it could be investigated how the season onset dates change over the years, and spatial patterns of these changes could be analyzed. Furthermore, the question should be addressed if these changes are related to the global climate change.

From a user's perspective it would certainly be of interest to implement the Bayesian multiple change point analysis in the R package `phenofun`. Then, not only %greenness time series inside the final ROI but also season onset dates could be derived automatically and with a single R script. Furthermore, the question of uncertainty of the estimated season onset dates should be addressed.

There are some methodological challenges in this area. First, a warping approach could be developed which can be applied to the %greenness time series. Until now, only discrete dates of SOS, MAX, EOS1 and EOS2 can be compared. With a warping approach one would aim at making continuous statements on how the phenology at a particular location in a particular year behaves in comparison to other locations and other years. For example, it is imaginable that the phenology is delayed at some point in the year but then catches up this delay until the next phenological change point. By only assessing a low number of landmarks over the year, this behavior could be overlooked.

Second, a recent methodological development in Happ and Greven (2015) allows us to investigate climate data such as daily temperature, humidity and solar radiation functions simultaneously with the webcam images in order to increase the amount of information contained in the classification variables of the supervised classification method.

Part III.

Summary and contributions of this thesis

Contributions of this thesis

Statistics is a science which gains information from data and describes and quantifies this information. Statistics is thereby applied in various areas to allow for better decision making based on empirical analyses. This thesis contributes to statistical research by applying, combining and extending state-of-the-art statistical methods in two complex interdisciplinary projects from ecological research fields. Efficient methods for automatically analyzing large amounts of image and video data were developed and implemented.

First project: Realtime classification of fish in underwater sonar videos

The first project was motivated by the protection of migrating fish species. With the increasing number of water power plants, the migration of fish is impeded. To implement protection measures for endangered fish species, information about number, species and time of arrival of fish at a water power plant is required. In this project, underwater sonar videos were used to obtain this information as follows.

An approach was developed which allows to automatically count the fish and thereby discriminate eels from other fish species. This is important because the eel is a particularly endangered species. The approach was implemented in a software which is fast enough to monitor the fish in realtime. A user interface ensures that the software can be used by non-statisticians at a water power plant.

Various statistical methods are combined to reach the goal of counting and classifying the fish. As first step, a given sonar video is smoothed using the linear array model. After further preprocessing steps including centering, thresholding and identification of connected pixel clouds, areas of interest – so-called hotspots – are defined. These hotspots most likely represent fish pixels and are processed in further steps: In a tracking procedure, hotspots corresponding to the same object (or fish) are connected. Then, features such as length, width or velocity are derived from these connected hotspots and shall allow to discriminate the fish species. In this step, among others, methods from the field of functional data analysis are applied.

Finally, the objects are classified with respect to the three classes *eels*, *other fish* and *debris* – where the class *debris* is needed to filter all dead objects swimming in the water – using established statistical classification methods such as discriminant analysis, support vector machines or random forests. This results in a high classification accuracy of the system.

Second project: Automated processing of webcam images for phenological classification

The second project was motivated in connection with the global climate change. While nowadays almost nobody seriously doubts the global warming, the effects of the global climate change on the flora and fauna are still under research. Among others, phenologists are interested in season onset dates and how they vary over the years and places on earth. In this project, digital webcam images were used to obtain information about season onset dates as follows.

With the goal of identifying season onset dates, a procedure was developed which allows to automatically process webcam images. The procedure was applied to two scientific cameras and three open-access cameras with publicly available data. Moreover, a number of 13988 cameras from the publicly available AMOS database was analyzed. A careful implementation of the methods in the statistical software package R ensures the scalability, i.e., data from several hundreds or thousands of webcams can be analyzed in rather short time.

In principle, the analysis consists of two steps: First, a region of interest (ROI) is defined on the image which shall only contain pixels which are relevant for the identification of season onset dates, for example deciduous trees. Second, from the time series of percentage greenness inside this ROI, the onset dates are identified using methods from structural change point analysis. While for the second task existing methods from the time series literature can be used, we proposed two alternative approaches for the automated definition of ROIs. These automated ROIs shall replace the current state-of-the-art method of expert-based ROIs, where a considerable amount of manual work is necessary.

As an alternative approach for the identification of season onset dates, we proposed a supervised classification procedure based on eigenimages of the image data. Thereby, each image is classified with respect to the four seasons *spring*, *summer*, *autumn* and *winter* based on scores of the images on the eigenimages of a training year. Further processing of the time series of predicted seasons leads to unique season onset dates.

We conclude that with the proposed new methods for the automated processing of webcam images, we paved the way for analyzing webcam data on a large scale. Especially with the fully automated and purely data-driven definition of regions of interest, we are now able to analyze a large amount of webcam data with low manual effort, as shown for the AMOS data. We hope that our work can be a relevant contribution for the application and analysis of webcam images for phenological questions.

Analogies of the two projects

Apart from the fact that both projects originate from ecological questions, there are close methodological analogies as well.

Obviously, both projects tackle imaging problems: In the fish project, videos produced by an underwater sonar camera are analyzed, in the phenology project, sets of images from several webcams are analyzed.

In both cases, the preprocessing of the data is a challenge requiring efficient solutions due to the huge data volume. Since not the whole images but only parts are of interest, the first step of the analyses is a search for the areas of interest in both cases. In the fish project, these areas are called hotspots and capture a single fish. The hotspots move through the image as the fish move and change over time. In the phenology project, these areas are called regions of interest or ROIs. The ROIs are fixed over time and capture phenologically interesting parts of the image such as deciduous trees. As a result, the data structures are changed by the preprocessing.

In the following steps, based on the areas of interest, features are extracted to prepare for the classification: In the fish project, the sets of hotspots corresponding to the same fish are measured and variables are derived which shall distinguish eels from other fish species and debris. In the phenology project, time series of percentage greenness inside the ROIs are used for the identification of season onset dates. As described in the respective chapters, this can be seen as a classification problem as well.

Finally, both projects originate from cooperations with researchers from ecological areas, fisheries biology in the first project and phenology in the second project. This means that there was a need for efficient implementations in order to allow the project partners and their communities to make use of the developed methods. In this context, different requirements had to be met: In the fish project, the analyses had to run in realtime to ensure the applicability of the system at water power plants during operation. In the phenology project, computing time was a less urgent issue but the methods should allow to scale up to several hundreds or thousands of webcams. Thus the computing time was an important point here as well. To sum up, in both cases the choice of methods and the concrete implementation had to take into account the practical needs of the users. Therefore the R packages `sonar` and `phenofun` were developed to make the presented methods publicly available.

Data science – the future of statistics?

Nowadays, more and more data are collected. This comes along with an increased need for methods which allow to analyze and to extract information from these huge data volumes. In order to be a modern science, a task of statistical research should be to develop new statistical methods for solving big data problems. Ideally this should be motivated by underlying practical research questions. Yet the developed methods should be general enough to be applicable to other problems as well.

Common research in statistics often deals with extending small details in existing methods in a very advanced and sophisticated way. In the end, the benefit of the new method is demonstrated on a well known data set which has been analyzed by generations of other statisticians before. Thereby, common statistical research risks to edge away from application which is dangerous since statistics cannot exist without application.

At the same time, efforts for the development of big data methods are being made by several other science sectors. Quantitative branches of applied sciences such as social sciences and econometrics on the one hand, and computer sciences on the other hand are developing their own ways to analyze large data volumes. Thereby, statistics should not miss the boat in order to play a key role in the analysis of the growing data volumes of our days.

In our opinion, the question is not whether statistics or computer sciences or other sciences engaged with data analysis find the "better" methods. In contrast, we think that the future of research on the analysis of big data volumes lies in “data science”: The interdisciplinary connection of the expertises of statisticians, computer scientists and quantitative applied researchers from different fields.

This thesis is a contribution to this branch of statistical research. It was shown that imaging problems can be solved with statistical techniques as well. Therefore, to yield the desired solutions, modern statistical methods were used, combined and extended. The focus of the methodological developments was on meeting the requirements of the given applied problems in order to provide relevant and useful support for the researchers from fisheries biology and phenology.

The technological developments of our days open up vast and fascinating challenges and opportunities in data analysis. We hope that this thesis can be a contribution to the further development of statistics as a modern science.

Appendix A: Linear array model

The definition of the product of a matrix and an array is given in Currie *et al.* (2006):

Definition 1 *The H-transform of the d-dimensional array $\boldsymbol{A}$ of size $c_1 \times c_2 \times \ldots \times c_d$ by the matrix $\boldsymbol{X}$ of size $r \times c_1$ is denoted $H(\boldsymbol{X}, \boldsymbol{A})$ and defined as follows: let $\boldsymbol{A}^*$ be the $c_1 \times c_2c_3 \ldots c_d$ matrix that is obtained by flattening dimensions 2-d of $\boldsymbol{A}$; form the matrix product $\boldsymbol{X}\boldsymbol{A}^*$ of size $r \times c_2c_3 \ldots c_d$; then $H(\boldsymbol{X}, \boldsymbol{A})$ is the d-dimensional array of size $r \times c_2 \times \ldots \times c_d$ that is obtained from $\boldsymbol{X}\boldsymbol{A}^*$ by reinstating dimensions 2-d of $\boldsymbol{A}$.*

Definition 2 *The rotation of the d-dimensional array $\boldsymbol{A}$ of size $c_1 \times c_2 \times \ldots \times c_d$ is the d-dimensional array $R(\boldsymbol{A})$ of size $c_2 \times c_3 \times \ldots \times c_d \times c_1$ that is obtained by permuting the indices of $\boldsymbol{A}$.*

Definition 3 *The rotated H-transform of the array $\boldsymbol{A}$ by the matrix $\boldsymbol{X}$ is given by*

$$\rho(\boldsymbol{X}, \boldsymbol{A}) = R\big\{H(\boldsymbol{X}, \boldsymbol{A})\big\}. \tag{A.1}$$

Thus, the least-squares estimate of the three-dimensional array $\boldsymbol{\Theta}$ of coefficients of the linear array model (2.2) can be written as

$$\begin{aligned}
\widehat{\boldsymbol{\Theta}} &= \Big[(\boldsymbol{B}_3^\top \boldsymbol{B}_3)^{-1}\, \boldsymbol{B}_3^\top \otimes (\boldsymbol{B}_2^\top \boldsymbol{B}_2)^{-1}\, \boldsymbol{B}_2^\top \otimes (\boldsymbol{B}_1^\top \boldsymbol{B}_1)^{-1}\, \boldsymbol{B}_1^\top\Big]\, \boldsymbol{Y} && \text{(A.2)}\\
&= \rho\Big[(\boldsymbol{B}_3^\top \boldsymbol{B}_3)^{-1}\, \boldsymbol{B}_3^\top,\ \rho\Big[(\boldsymbol{B}_2^\top \boldsymbol{B}_2)^{-1}\, \boldsymbol{B}_2^\top,\ \rho\Big[(\boldsymbol{B}_1^\top \boldsymbol{B}_1)^{-1}\, \boldsymbol{B}_1^\top, \boldsymbol{Y}\Big]\Big]\Big]\,. && \text{(A.3)}
\end{aligned}$$

The smoothed data array $\widehat{\boldsymbol{Y}}$ can then be written as

$$\widehat{\boldsymbol{Y}} = \boldsymbol{B}\widehat{\boldsymbol{\Theta}} = \rho\left[\boldsymbol{B}_3, \rho\left[\boldsymbol{B}_2, \rho\left[\boldsymbol{B}_1, \widehat{\boldsymbol{\Theta}}\right]\right]\right]. \tag{A.4}$$

Appendix B: R package sonar

Excerpt of the documentation of the R package `sonar`. For the full documentation and download of the package see https://r-forge.r-project.org/projects/sonar/. Executable example code cannot be provided due to missing rights for publication of the sonar videos used in this thesis.

Package "sonar"

Title Realtime classification of fish in underwater sonar videos
Version 0.1.1
Author Ludwig Bothmann [aut, cre], Michael Bothmann [ctb]
Maintainer Ludwig Bothmann <ludwig.bothmann@stat.uni-muenchen.de>
Description Realtime classification of fish in underwater sonar videos.
Depends R (>= 2.6.0)
Imports hexView, splines, pixmap, cluster, MASS, tree, e1071, parallel, nnet
License GPL (>=2)
LazyData true
RoxygenNote 5.0.1

`analyze_ddf` *Analyze complete .ddf file*

Description

This function analyzes a given sonar video (.ddf file), i.e., carries out the entire preprocessing with identification of hotspots, tracking, extraction of features and finally classifies the identified objects.

Usage

```
analyze_ddf(ddf.file, win.start = 0.83, win.length = 5, vers = 3,
  max.frame, y.lim = 512, a = 18, cut = 50, m.d.cs = 0.2, folder.output,
  n.angle = 3, do.plots = FALSE, n.cores = 1, frames.pack = 600,
  df.t = NULL, save.preprocess = FALSE, timestamp = "", min.length = 10,
  file.classrule, type.classrule, save.movie = TRUE, each = 5,
  win = FALSE)
```

Arguments

`ddf.file`	File name of .ddf file to be analyzed
`win.start`	Start of the sonar window (distance from the lense of the camera), possibly extracted before via `get.version`
`win.length`	Length of the sonar window, possibly extracted before via `get.version`
`vers`	Version of the .ddf file, possibly extracted before via `get.version`
`max.frame`	Total number of frames (images) of the given video, possibly extracted before via `get.version`
`y.lim`	Number of pixels to analyze in each beam, counted from the camera
`a`	Threshold for the centered data
`cut`	Minimal size of clusters in number of pixels. Smaller clusters are deleted before the tracking and hence not considered as hotspots.
`m.d.cs`	Maximal distance for which two hotspots can be assigned to the same tracking number
`folder.output`	Folder for resulting plots and output
`n.angle`	Number of watch hands per quarter, i.e., total number of watch hands is `4 x n.angle`
`do.plots`	`TRUE`: Plots of preprocessing are saved, default is `FALSE`
`n.cores`	Number of cores if parallelization is needed, default is `1`
`frames.pack`	Number of frames to be analyzed simultaneously, possibly the total number of frames `max.frame` of the given video. If `frames.pack` is less than `max.frame`, the video is splitted in parts of `frames.pack` frames and each part is analyzed separately.
`df.t`	Number of degrees of freedom for the splines for first smoothing step. Default `NULL` results in `c(100, 25, round(n.frames/2))` where `n.frames` is the total number of frames of the analyzed video, i.e., `length(frames.pack)`.
`save.preprocess`	`TRUE`: Results of preprocessing are saved, default is `FALSE`
`timestamp`	Time stamp of .ddf file (used for file names of results)
`min.length`	Minimal length of tracks. Tracks with less hotspots are deleted.
`file.classrule`	File name of classification rule, `.RData` file containing the rules
`type.classrule`	Type of classification rule, i.e., lda, qda...
`save.movie`	`TRUE`: Video of raw signal and cleaned signal is created, default is `TRUE`

`each`	Interval of images for the video, for example default `5` means that each `5`th image is saved in the video
`win`	Specify `TRUE` if you are running under windows, then parallelization is not possible

Value

As output, a table is generated which lists the detected objects with predicted class and computed features at `folder.output`.

Appendix C: R package `phenofun`

Excerpt of the documentation of the R package `phenofun`. For the full documentation and download of the package see `https://r-forge.r-project.org/projects/phenofun/`. Executable example code can be found at `http://bothmann.userweb.mwn.de/dissertation.html`.

Package “phenofun”

Title Automated processing of webcam images for phenological classification
Version 0.1.2
Author Ludwig Bothmann [aut, cre], Michael Matiu [ctb]
Maintainer Ludwig Bothmann `<ludwig.bothmann@stat.uni-muenchen.de>`
Description
Automated processing of webcam images for phenological classification. uROI method, sROI method and supervised classification method are fully implemented.
Depends R (>= 3.2.1)
Imports EBImage, irlba, strucchange, abind, MASS
License GPL (>=2)
LazyData true
RoxygenNote 5.0.1

`amos_uroi_wrap` *Wrapper function for analyzing data from AMOS*

Description

This function allows to analyze webcam data from AMOS (`http://amos.cse.wustl.edu/`) with the uROI method in a fully automated way. Only the parameters of camera, year, month and hour of images to be analyzed have to be specified. Example code can be downloaded at `http://bothmann.userweb.mwn.de/dissertation.html`.

Usage

```
amos_uroi_wrap(camera, year_analysis, months_analysis, hour_analysis,
  do_mean_greenness = TRUE, do_strucchange = TRUE, testmode = FALSE,
  only_download = FALSE, folder_results = getwd(),
  name_of_analysis = substr(as.character(Sys.time()), 1, 10),
  folder_data = getwd(), n_pc_vec = 12, k_vec = 4:10, nstart = 2,
  save_results = TRUE, save_masks = TRUE, masks_type = ".jpg",
  a_vec = seq(1, 150, by = 1), b_vec = seq(1, 100, by = 1), ...)
```

Arguments

`camera`	Name of the camera (five-digit number, see http://amos.cse.wustl.edu/)
`year_analysis`	Year(s) to be analyzed as vector
`months_analysis`	Month(s) to be analyzed as vector
`hour_analysis`	Hour(s) to be analyzed as vector
`do_mean_greenness`	If `TRUE` (default), time series of percentage greenness inside the ROIs are computed
`do_strucchange`	If `TRUE` (default), points of structural changes are searched for and OC values are computed
`testmode`	If `TRUE`, only 10 images for testing are analyzed, default is `FALSE`
`only_download`	If `TRUE`, images are downloaded but not analyzed, default is `FALSE`
`folder_results`	Folder path where the results will be saved, default is the current working directory
`name_of_analysis`	Name of subfolder for the results of the analysis, default is the date as `yyyy-mm-dd`
`folder_data`	Folder where the data will be saved, default is the current working directory
`n_pc_vec`	Vector of numbers of eigenimages
`k_vec`	Vector of numbers of clusters for k-means
`nstart`	Number of iterations for k-means
`save_results`	If `TRUE` (default), results of clustering are saved
`save_masks`	If `TRUE` (default), masks are saved
`masks_type`	File name extension of masks, default is `.jpg`
`a_vec`	Possible spring DOYs
`b_vec`	Possible autumn DOYs (counted backwards from 31.12.)

...	Further arguments

Value

As output, all resulting masks and percentage greenness time series inside the masks are saved in subfolders of `folder_results`. Additionally, overlays of all masks of the best setting with a background image and the background image itself are saved in `folder_results`, ordered with respect to optimality criterion OC2.

Bibliography

Ahrends HE, Brügger R, Stöckli R, Schenk J, Michna P, Jeanneret F, Wanner H, Eugster W (2008). "Quantitative phenological observations of a mixed beech forest in northern Switzerland with digital photography." *Journal of Geophysical Research – Biogeosciences*, **113**(G4). doi: 10.1029/2007JG000650. G04004.

Alberton B, Almeida J, Helm R, da Silva Torres R, Menzel A, Morellato LPC (2014). "Using phenological cameras to track the green up in a cerrado savanna and its on-the-ground-validation." *Ecological Informatics*, **19**, 62–70. doi: 10.1016/j.ecoinf.2013.12.011.

Andrews DWK (1993). "Tests for parameter instability and structural change with unknown change point." *Econometrica*, **61**(4), 821–856. doi: 10.2307/2951764.

Baglama J, Reichel L (2015). *irlba: Fast truncated SVD, PCA and symmetric eigendecomposition for large dense and sparse matrices*. R package version 2.0.0, URL http://CRAN.R-project.org/package=irlba.

Bothmann L, Menzel A, Menze BH, Schunk C, Kauermann G (2016a). "Automated processing of webcam images for phenological classification." *PLOS ONE*. Under review.

Bothmann L, Windmann M, Kauermann G (2016b). "Realtime classification of fish in underwater sonar videos." *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **65**(4), 565–584. doi: 10.1111/rssc.12139.

Breiman L (2001). "Random forests." *Machine Learning*, **45**(1), 5–32. doi: 10.1023/A:1010933404324.

Breiman L, Friedman J, Stone CJ, Olshen RA (1984). *Classification and regression trees*. Wadsworth Statistics/Probability. ISBN 978-0412048418.

Burwen DL, Fleischman SJ, Miller J (2010). "Accuracy and precision of salmon length estimates taken from DIDSON sonar images." *Transactions of the American Fisheries Society*, **139**(5), 1306–1314. doi: 10.1577/T09-173.1.

Cortes C, Vapnik V (1995). “Support-vector networks.” *Machine Learning*, **20**(3), 273–297. doi: 10.1007/BF00994018.

Crossman JA, Martel G, Johnson PN, Bray K (2011). “The use of dual-frequency identification sonar (DIDSON) to document white sturgeon activity in the Columbia River, Canada.” *Journal of Applied Ichthyology*, **27**, 53–57. doi: 10.1111/j.1439-0426.2011.01832.x.

Currie ID, Durban M, Eilers PHC (2006). “Generalized linear array models with applications to multidimensional smoothing.” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **68**(2), 259–280. doi: 10.1111/j.1467-9868.2006.00543.x.

Dony RD (2001). “Karhunen-Loève transform.” In “The transform and data compression handbook,” CRC Press LLC. ISBN 978-0-8493-3692-8.

Dose V, Menzel A (2004). “Bayesian analysis of climate change impacts in phenology.” *Global Change Biology*, **10**, 259–272. doi: 10.1111/j.1529-8817.2003.00731.x.

Febrero-Bande M, Oviedo de la Fuente M (2012). “Statistical computing in functional data analysis: The R package fda.usc.” *Journal of Statistical Software*, **51**(4), 1–28. doi: 10.18637/jss.v051.i04.

Fu Y, Piao S, Op de Beeck M, Cong N, Menzel A, Janssens IA (2014). “Recent spring phenology shifts in western Central Europe based on multi-scale observations.” *Global Ecology and Biogeography*, **23**(11), 1255–1263. doi: 10.1111/geb.12210.

Golub GH, Reinsch C (1970). “Singular value decomposition and least squares solutions.” *Numerische Mathematik*, **14**(5), 403–420. doi: 10.1007/BF02163027.

Graham E, Riordan E, Yuen E, Estrin D, Rundel P (2010). “Public internet-connected cameras used as a cross-continental ground-based plant phenology monitoring system.” *Global Change Biology*, **16**(11), 3014–3023. doi: 10.1111/j.1365-2486.2010.02164.x.

Greven S, Crainiceanu C, Caffo B, Reich D (2010). “Longitudinal functional principal component analysis.” *Electronic Journal of Statistics*, **4**, 1022–1054. doi: 10.1214/10-EJS575.

Hansen BE (1992). “Tests for parameter instability in regressions with I(1) processes.” *Journal of Business & Economic Statistics*, **10**(3), 321–335. doi: 10.2307/1391545.

Happ C, Greven S (2015). *Multivariate functional principal component analysis for data observed on different (dimensional) domains*. URL http://arxiv.org/abs/1509.02029.

Henneken R, Dose V, Schleip C, Menzel A (2013). “Detecting plant seasonality from webcams using Bayesian multiple change point analysis.” *Agricultural and Forest Meteorology*, **168**, 177–185. doi: 10.1016/j.agrformet.2012.09.001.

Holben BN (1986). “Characteristics of maximum-value composite images from temporal AVHRR data.” *International Journal of Remote Sensing*, **7**(11), 1417–1434. doi: 10.1080/01431168608948945.

Holmes JA, Cronkite GMW, Enzenhofer HJ, Mulligan TJ (2006). “Accuracy and precision of fish-count data from a “dual-frequency identification sonar” (DIDSON) imaging system.” *ICES Journal of Marine Science*, **63**, 543–555. doi: 10.1016/j.icesjms.2005.08.015.

Hufkens K, Friedl M, Sonnentag O, Braswell BH, Millimian T, Richardson A (2012). “Linking near-surface and satellite remote sensing measurements of deciduous broadleaf forest phenology.” *Remote Sensing of Environment*, **117**, 307–321. doi: 10.1016/j.rse.2011.10.006.

Jacobs N, Roman N, Pless R (2007). “Consistent temporal variations in many outdoor scenes.” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi: 10.1109/CVPR.2007.383258.

Jeganathan C, Dash J, Atkinson PM (2014). “Remotely sensed trends in the phenology of northern high latitude terrestrial vegetation, controlling for land cover change and vegetation type.” *Remote Sensing of Environment*, **143**, 154–170. doi: 10.1016/j.rse.2013.11.020.

Jeong S, Ho C, Gim H, Brown ME (2011). “Phenology shifts at start vs. end of growing season in temperate vegetation over the Northern Hemisphere for the period 1982–2008.” *Global Change Biology*, **17**(7), 2385–2399. doi: 10.1111/j.1365-2486.2011.02397.x.

Jolliffe IT (2002). *Principal component analysis*. Springer New York, 2nd edition. ISBN 978-0-387-95442-4. doi: 10.1007/b98835.

Julitta T, Cremonese E, Migliavacca M, Colombo R, Galvagno M, Siniscalco C, Rossini M, Fava F, Cogliati S, Morra di Cella U, Menzel A (2014). “Using digital camera images to analyse snowmelt and phenology of a subalpine grassland.” *Agricultural and Forest Meteorology*, **198–199**, 116–125. doi: 10.1016/j.agrformet.2014.08.007.

Keenan TF, Baker I, Barr A, Ciais P, Davis K, Dietze M, Dragoni D, Gough CM, Grant R, Hollinger D, Hufkens K, Poulter B, McCaughey H, Raczka B, Ryu Y, Schaefer K, Tian H, Verbeeck H, Zhao M, Richardson AD (2012). “Terrestrial biosphere model performance for inter-annual variability of land-atmosphere CO2 exchange.” *Global Change Biology*, **18**(6), 1971–1987. doi: 10.1111/j.1365-2486.2012.02678.x.

Kindratenko VV (2003). “On using functions to describe the shape.” *Journal of Mathematical Imaging and Vision*, **18**(3), 225–245. doi: 10.1023/A:1022843426320.

Langkau MC, Balk H, Schmidt MB, Borcherding J (2012). “Can acoustic shadows identify fish species? A novel application of imaging sonar data.” *Fisheries Management and Ecology*, **19**(4), 313–322. doi: 10.1111/j.1365-2400.2011.00843.x.

Larinier M, Travade F (2002). “Downstream migration: Problems and facilities.” *Bulletin Français de la Pêche et de la Pisciculture*, **364**(Supplément), 181–207. doi: 10.1051/kmae/2002102.

Lee ET, Pan YJ, Chu P (1987). “An algorithm for region filling using two-dimensional grammars.” *International Journal of Intelligent Systems*, **2**(3), 255–263. doi: 10.1002/int.4550020302.

Liang L, Schwartz MD (2009). “Landscape phenology: An integrative approach to seasonal vegetation dynamics.” *Landscape Ecology*, **24**(4), 465–472. doi: 10.1007/s10980-009-9328-x.

Liaw A, Wiener M (2002). “Classification and regression by randomForest.” *R News*, **2**(3), 18–22. URL `http://CRAN.R-project.org/doc/Rnews/`.

Maier C (2013). *Klassifikation von Fischen in Sonarfilmen (DIDSON)*. Master's thesis, Institut für Statistik, LMU München.

Menzel A (2002). "Phenology: Its importance to the global change community." *Climatic Change*, **54**(4), 379–385. doi: 10.1023/A:1016125215496.

Menzel A (2013). "Phenological data, networks, and research: Europe." In MD Schwartz (ed.), "Phenology: An Integrative Environmental Science," pp. 53–65. Springer. ISBN 978-94-007-6925-0. doi: 10.1007/978-94-007-6925-0.

Menzel A, Helm R, Zang C (2015). "Patterns of late spring frost leaf damage and recovery in a European beech (Fagus sylvatica L.) stand in south-eastern Germany based on repeated digital photographs." *Frontiers in Plant Science*, **6**(110), 1–13. doi: 10.3389/fpls.2015.00110.

Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F (2014). *e1071: Misc functions of the Department of Statistics (e1071), TU Wien*. R package version 1.6-4, URL http://CRAN.R-project.org/package=e1071.

Migliavacca M, Reichstein M, Richardson AD, Colombo R, Sutton MA, Lasslop G, Tomelleri E, Wohlfahrt G, Carvalhais N, Cescatti A, Mahecha MD, Montagnani L, Papale D, Zaehle S, Arain A, Arneth A, Black TA, Carrara A, Dore S, Gianelle D, Helfter C, Hollinger D, Kutsch WL, Lafleur PM, Nouvellon Y, Rebmann C, Da Rocha HR, Rodeghiero M, Roupsard O, Sebastia MT, Seufert G, Soussana JF, Van der Molen MK (2011). "Semiempirical modeling of abiotic and biotic factors controlling ecosystem respiration across eddy covariance sites." *Global Change Biology*, **17**(1), 390–409. doi: 10.1111/j.1365-2486.2010.02243.x.

Morris JS (2015). "Functional regression." *Annual Review of Statistics and Its Application*, **2**(1), 321–359. doi: 10.1146/annurev-statistics-010814-020413.

Mueller AM, Burwen DL, Boswell KM, Mulligan T (2010). "Tail-beat patterns in dual-frequency identification sonar echograms and their potential use for species identification and bioenergetics studies." *Transactions of the American Fisheries Society*, **139**(3), 900–910. doi: 10.1577/T09-089.1.

Mueller AM, Mulligan T, Withler PK (2008). "Classifying sonar images: Can a computer-driven process identify eels?" *North American Journal of Fisheries Management*, **28**(6), 1876–1886. doi: 10.1577/M08-033.1.

Murrell P (2014). *hexView: Viewing binary files*. R package version 0.3-3, URL http://CRAN.R-project.org/package=hexView.

Northeast Fisheries Science Center (2015). "What is an anadromous fish?" URL http://www.nefsc.noaa.gov/faq/fishfaq1a.html. Accessed: 2015-09-14.

Peñuelas J, Rutishauser T, Filella I (2009). "Phenology feedbacks on climate change." *Science*, **324**(5929), 887–888. doi: 10.1126/science.1173004.

Piao S, Ciais P, Friedlingstein P, Peylin P, Reichstein M, Luyssaert S, Margolis H, Fang J, Barr A, Chen A, Grelle A, Hollinger DY, Laurila T, Lindroth A, Richardson AD, Vesala T (2008). "Net carbon dioxide losses of northern ecosystems in response to autumn warming." *Nature*, **451**(7174), 49–52. doi: 10.1038/nature06444.

Pipal KA, Notch JJ, Hayes SA, Adams PB (2012). "Estimating escapement for a low-abundance steelhead population using dual-frequency identification sonar (DIDSON)." *North American Journal of Fisheries Management*, **32**(5), 880–893. doi: 10.1080/02755947.2012.697096.

Pope K, Dose V, Da Silva D, Brown P, Leslie C, Dejong T (2013). "Detecting nonlinear response of spring phenology to climate change by Bayesian analysis." *Global Change Biology*, **19**(5), 1518–1525. doi: 10.1111/gcb.12130.

R Core Team (2016). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Rakowitz G, Tuser M, Riha M, Juza T, Balk H, Kubecka J (2012). "Use of high-frequency imaging sonar (DIDSON) to observe fish behaviour towards a surface trawl." *Fisheries Research*, **123–124**, 37–48. doi: 10.1016/j.fishres.2011.11.018.

Ramsay JO, Silverman BW (2005). *Functional data analysis*. Springer New York. ISBN 978-0-387-40080-8. doi: 10.1007/b98888.

Ramsay JO, Wickham H, Graves S, Hooker G (2013). *fda: Functional data analysis*. R package version 2.4.0, URL http://CRAN.R-project.org/package=fda.

Richardson AD, Jenkins JP, Braswell BH, Hollinger DY, Ollinger SV, Smith ML (2007). "Use of digital webcam images to track spring green-up in a deciduous broadleaf forest." *Oecologia*, **152**(2), 323–334. doi: 10.1007/s00442-006-0657-z.

Richardson AD, Keenan TF, Migliavacca M, Ryu Y, Sonnentag O, Toomey M (2013). "Climate change, phenology, and phenological control of vegetation feedbacks to the climate system." *Agricultural and Forest Meteorology*, **169**, 156–173. doi: 10.1016/j.agrformet.2012.09.012.

Ripley B (2015). *tree: Classification and regression trees*. R package version 1.0-36, URL http://CRAN.R-project.org/package=tree.

Rodriguez-Galiano VF, Dash J, Atkinson PM (2015). "Intercomparison of satellite sensor land surface phenology and ground phenology in Europe." *Geophysical Research Letters*, **42**(7), 2253–2260. doi: 10.1002/2015GL063586.

Schleip C, Menzel A, Estrella N, Dose V (2006). "The use of Bayesian analysis to detect recent changes in phenological events throughout the year." *Agricultural and Forest Meteorology*, **141**(2–4), 179–191. doi: 10.1016/j.agrformet.2006.09.013.

Schwartz MD (1992). "Phenology and springtime surface-layer change." *Monthly Weather Review*, **120**, 2570–2578.

Stoyan D, Stoyan H (1994). *Fractals, random shapes and point fields. Methods of geometrical statistics.* John Wiley & Sons. doi: 10.1002/zamm.19950750815.

Stratopoulos LMF (2015). *Use of close range remote sensing techniques for the tracking of phenological events and the detection of drought stress of common beech (Fagus sylvatica [L.]) in Upper Bavaria*. Master's thesis, Chair of Ecoclimatology, TU München.

Trucco E, Plakas K (2006). "Video tracking: A concise survey." *IEEE Journal of Oceanic Engineering*, **31**(2), 520–529. doi: 10.1109/JOE.2004.839933.

Venables WN, Ripley BD (2002). *Modern applied statistics with S*. Springer, New York, 4th edition. ISBN 0-387-95457-0, URL http://www.stats.ox.ac.uk/pub/MASS4.

White MA, de Beurs KM, Didan K, Inouye DW, Richardson AD, Jensen OP, O'Keefe J, Zhang G, Nemani RR, van Leeuwen WJD, Brown JF, de Wit A, Schaepman M, Lin X, Dettinger M, Bailey AS, Kimball J, Schwartz MD, Baldocchi DD, Lee JT, Lauenroth WK (2009). "Intercomparison, interpretation, and assessment of spring phenology in North America estimated from remote sensing for 1982-2006." *Global Change Biology*, **15**(10), 2335–2359. doi: 10.1111/j.1365-2486.2009.01910.x.

Yilmaz A, Javed O, Shah M (2006). "Object tracking: A survey." *ACM Computing Surveys*, **38**(4), Article 13. doi: 10.1145/1177352.1177355.

Zeileis A, Kleiber C, Kraemer W, Hornik K (2003). "Testing and dating of structural changes in practice." *Computational Statistics & Data Analysis*, **44**(1), 109–123. doi: 10.1016/S0167-9473(03)00030-6.

Zeileis A, Leisch F, Hornik K, Kleiber C (2002). "strucchange: An R package for testing for structural change in linear regression models." *Journal of Statistical Software*, **7**(2), 1–38. doi: 10.18637/jss.v007.i02.

Zhao J, Zhang Y, Tan Z, Song Q, Liang N, Yu L, Zhao J (2012). "Using digital cameras for comparative phenological monitoring in an evergreen broad-leaved forest and a seasonal rain forest." *Ecological Informatics*, **10**, 65–72. doi: 10.1016/j.ecoinf.2012.03.001.

Eidesstattliche Versicherung

(Gemäß Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. 5)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

München, den 12. Juli 2016

Ludwig Bothmann