Change Detection in Combination with Spatial Models and its Effectiveness on Underwater Scenarios

Dissertation
zur
Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
der Fakultät für Informatik und Elektrotechnik
der Universität Rostock

vorgelegt von
Martin Radolko
geboren am 27.04.1986 in Parchim
aus Rostock
am

25. Juli 2018

https://doi.org/10.18453/rosdok_id00002418

Dieses Werk ist lizenziert unter einer
Creative Commons Namensnennung - Nicht kommerziell - Keine Bearbeitungen 4.0 International Lizenz.

Gutachter:

Prof. Dr.-Ing. Bodo Urban
Universität Rostock, Germany

Prof. Dr.-Ing. Uwe Freiherr von Lukas
Universität Rostock, Germany

Prof. Dr. Arjan Kuijper
Technische Universität Darmstadt, Germany

Datum der Verteidigung:

21.01.2019

Keywords: Underwater Change Detection; Background Subtraction; Blob Tracking; Underwater Image Enhancement; Spatial Coherent Segmentations
Schlagwörter: Unterwasser Veränderungs Detektion; Hintergrund Subtraktion; Blob Tracking; Unterwasser Bildverbesserung; räumlich zusammenhängende Segmentierungen
ACM Classification: • **Computing methodologies** → **Video segmentation**; *Tracking*; *Object detection*

Abstract

In the last years, cheap off-the-shelf consumer cameras became more and more available while the imaging quality of these devices increased drastically. This is not only true for normal in-air cameras but also for underwater devices. In connection with a growing interest in the sea world by science and industry, this led to the need for advanced computer vision algorithms, especially for underwater scenarios, so that the vast amount of acquired data can be automatically processed and important information extracted. However, apart from the normal difficulties of computer vision – the creation of detections that are consistent and coherent in the spatial as well as temporal domain – new problems occur in the underwater world caused by the specialties of the medium water. These difficulties are, among others, blur, color cast, Marine Snow and refraction which all complicate any computer vision task.

Therefore, this thesis proposes a novel change detection approach and combines it with different especially developed spatial models, this allows an accurate and spatially coherent detection of any moving objects with a static camera in arbitrary environments. The spatial models include a novel approach based on the idea behind Ncut but also a derivative of the Markov Random Field model. To deal with the special problems of underwater imaging, different enhancement algorithms were used in combination with the change detection approach to counter the negative effects, e.g. Marine Snow removal or a learning-based deblurring. Furthermore, pre-segmentations based on the optical flow and other special adaptions were added to the change detection algorithm so that it can better handle typical underwater scenarios like a scene crowded by a whole fish swarm. To use these detections and extract the most information out of them, a blob tracking method is introduced that keeps the generality of the change detection approach but can still deal with occlusions or detections errors. Overall, the here presented novel general detection and tracking approach can deliver accurate results in almost all scenarios while dealing especially well with underwater videos.

The different steps of the pipeline were tested and compared on different datasets against many state of the art algorithms and showed competitive results. For the testing on underwater videos a new special dataset had to be created – since this area was grossly neglected so far – and there the proposed change detection could outperform any existing method thanks to the newly developed adaptions.

Zusammenfassung

Während in den letzten Jahren Kameras fortlaufend billiger und verfügbarer wurden, hat sich gleichzeitig die Qualität ihrer Aufnahmen drastisch erhöht. Dies gilt nicht nur für die Standard-Kameras sondern auch für Unterwasserkameras und andere Spezialhardware. Im Zusammenhang mit dem gewachsenen Interesse an der Unterwasserwelt hat dies zu einem stark ansteigenden Bedarf an Bildverarbeitungsalgorithmen geführt welche die Unmengen an akquirierten Daten automatisch verarbeiten und wichtige Informationen extrahieren können. Zusätzlich zu den Standardproblemen – z.b. der Erstellung von konsistenten und zusammenhängenden Detektionen, sowohl räumlich als auch zeitlich – sind in Unterwasser-Szenen durch die Besonderheiten des Mediums Wasser neue Probleme vorhanden. Diese Schwierigkeiten sind unter anderem die vergrößerte Unschärfe, das Vorhandensein eines Farbstichs, die große Anzahl kleiner Schwebeteilchen oder die Lichtbrechung. All diese Effekte müssen bedacht werden und erschweren die Anwendung von komplexen Algorithmen des maschinellen Sehens erheblich.

Deswegen wird in dieser Arbeit ein neuer Ansatz für die Erkennung von Veränderungen in Videos hergeleitet und kombiniert mit verschiedenen räumlichen Modellen. Zusammen ermöglicht die eine zuverlässige und präzise Objekterkennung in fast allen Szenarien, solange die Kamera statisch ist. Diese Verfahren beinhalten unter anderem einen neuen Ansatz basierend auf der Idee des Ncut Verfahrens und einer Abwandlung des Markov Random Field Modells. Weiterhin, um die besonderen Probleme in Unterwasservideos zu untersuchen und deren Effekt auf die Segmentierungen zu minimieren, wurden verschiedene Bildverbesserungsverfahren mit den vorher beschriebenen Ansatz kombiniert, zum Beispiel das Entfernen der Schwebeteilchen oder die Anpassung und Verbesserung der Schärfe basierend auf maschinellem Lernen. Durch Vorsegmentierungen basierend auf dem optischen Fluss und weiteren Ergänzungen wurde die Objekterkennungen auf die spezillen Probleme in Unterwasserszenen angepasst, wie zum Beispiel einen Fischschwarm der den großteil des Bildausschnitts einnimmt.

Um aus diesen Detektionen möglichst viele Informationen zu extrahieren, wurde zuletzt ein Tracking Verfahren vorgestellt. Der vorgestellt Algorithmus arbeitet ausschließlich mit den bereits erkannten Objekten wodurch die Allgemeingültigkeit des gesamten Ansatzes gewahrt bleibt jedoch die Behandlung von Verdeckung und Deketionsfehlern Insgesamt erschwert wird. Zusammengefasst können diese Bildverarbeitungsverfahren Objekte in fast allen Situationen präzise Erkennen und Verfolgen, besonders jedoch in Unterwasservideos dank der speziellen Anpassungen.

Die verschiedenen Schritte des Verfahrens wurden auf öffentlich verfügbaren Datensätzen getestet und mit viele aktuellen Ansätzen verglichen. Für 'in Luft' Aufnahmen waren die Resultate vergleichbar mit gegenwärtigen Methoden während für Unterwasservideos dank der neu entwickelten Anpassungen eine klare Verbesserung erkennbar war.

Contents

Abstract V

Zusammenfassung VII

1 Introduction 1
- 1.1 Purpose and Scientific Problem 2
- 1.2 Structure of the Thesis 4

2 Fundamental Definitions 7

3 Related Work 17
- 3.1 Sonar Imaging 17
- 3.2 Optical Imaging - Single Image Segmentation 19
- 3.3 Video Segmentation 31
  - 3.3.1 General Video Segmentation 32
  - 3.3.2 Underwater Video Segmentation 49
- 3.4 Conclusion 55

4 General Change Detection 59
- 4.1 Gaussian Switch Model 60
- 4.2 Spatial Coherence 65
  - 4.2.1 Higher Order Markov Random Fields and Belief Propagation 66
  - 4.2.2 N^2Cut 76
  - 4.2.3 Temporal Trajectories 83
- 4.3 Results 90
- 4.4 Conclusion 97

5 A Dataset for Underwater Change Detection 105
- 5.1 Existing Change Detection Datasets 105
- 5.2 Difficulties in the Underwater Context 107
- 5.3 Underwater Change Detection dataset 108
- 5.4 Conclusion 109

6 Image Enhancement on Underwater Change Detection 113
- 6.1 Enhancement Methods 113
  - 6.1.1 Marine Snow Removal - MSR 113
  - 6.1.2 Learning-based Deblurring - LbD 114
  - 6.1.3 Adaptive Gray World - AGW 114

6.1.4 Automatic Color Equalization - ACE 115
6.2 Evaluation 116
6.3 Conclusion 123

7 Underwater Change Detection 125
7.1 Extension of the GSM - eGSM 125
7.2 Segmentation of Crowded Underwater Scenes 129
7.2.1 Optical Flow as Pre-Segmentation in Crowded Scenes 131
7.3 Spatial Methods 135
7.4 Evaluation 137
7.5 Conclusion 142

8 Underwater Blob Tracking 145
8.1 Matching Function 145
8.2 Optimal Match Finding 152
8.2.1 Special Cases 155
8.3 Results 156
8.3.1 Quantitative Evaluation 159
8.3.2 Conclusion 169

9 Conclusion and Future Work 171

Abbreviations and Symbols 175

References 179

Own Publications 195

Thesis Statements 197

List of Figures

1.1 Change detection example - Wallflower dataset . . . 3

2.1 Digital imaging . . . 8
2.2 Standard pixel neighborhoods . . . 9
2.3 Objects and connected components . . . 10
2.4 Binary classification example . . . 12
2.5 Gaussian distribution . . . 13
2.6 Matching of detections between frames . . . 15

3.1 3D Reconstruction via sonar imaging . . . 18
3.2 Segmentation in sonar images . . . 19
3.3 Detection of a fish swarm via sonar . . . 20
3.4 Detection and tracking via thresholding . . . 21
3.5 Mean shift segmentation . . . 22
3.6 Results of fuzzy C-Means . . . 24
3.7 Example of a neural network . . . 25
3.8 Segmentation of corals with neural networks . . . 26
3.9 Fish detection with convolutional neural networks . . . 27
3.10 Illustration of an underwater stereo system . . . 28
3.11 Markov Random Fields as a spatial model . . . 29
3.12 Lobster segmentation with a Bayesian framework . . . 30
3.13 Segmentation with a moving camera . . . 33
3.14 Background subtraction example . . . 35
3.15 Example: Mixture of Gaussian . . . 36
3.16 Shadow detection with IHLS color space . . . 37
3.17 Markov Random Field with background subtraction . . . 38
3.18 Subspace Learning . . . 40
3.19 Optical flow with convolutional neural networks . . . 44
3.20 Optical flow with feature tracking . . . 45
3.21 Example: Flux Tensor . . . 46
3.22 The idea of graph cut approaches . . . 47
3.23 Example of spatio-temporal Bayesian framework segmentation . . . 50
3.24 Optical flow via focus of expansion . . . 51
3.25 Fish swarm tracking with background subtraction . . . 52
3.26 Foreground probability maps via background modeling . . . 53
3.27 Background subtraction combined with image enhancement . . . 54

4.1 Comparison of updating schemes . . . 63

4.2 Comparison of different color spaces 64
4.3 The Markov Random Field model 66
4.4 Cliques in the Moore neighborhood 67
4.5 Different neighborhood sizes 68
4.6 Markov Random Field model for an image 69
4.7 Example of an acyclic Bayesian network 72
4.8 Conversion: Markov Random Field to factor graph 73
4.9 Effect of the MRF with different neighborhoods 74
4.10 Diagram: parallelization of the N^2Cut 82
4.11 Parallelized N^2Cut - timeflow 84
4.12 Example of a temporal trajectory 85
4.13 Result of a dense optical flow 86
4.14 Segmentations of the proposed algorithms on the Wallflower dataset 99
4.15 Segmentations of different Gaussian models on the Wallflower dataset 100
4.16 Segmentations of non-Gaussian models on the Wallflower dataset 101
4.17 Accuracy comparison on the Wallflower dataset 102

5.1 Examples from the Underwater Change Detection dataset 106
5.2 Underwater degradation effects 110
5.3 Underwater color cast 110
5.4 The Underwater Change Detection dataset 111
5.5 Difficult examples of underwater segmentation 112

6.1 Effect of Marine Snow removal 114
6.2 Effect of Learning-based deblurring 115
6.3 Effect of color restoration techniques 115
6.4 Trained background model in an underwater scene 118
6.5 Comparison of underwater segmentation results 120
6.6 Examples of the optical flow feature tracking 121
6.7 Optical flow feature tracking with and without Marine Snow removal 122

7.1 Background models in underwater videos 130
7.2 Background modeling problems in crowded scenes 132
7.3 Segmentations based on the Flux Tensor 134
7.4 Effect of the Flux Tensor pre-segmentations on the background modeling . . . 135
7.5 Effect of spatial methods 138
7.6 Segmentations on the Underwater Change Detection dataset 141

8.1 Connected components in a binary segmentation 146
8.2 Polar coordinates 148
8.3 Conversion of a detection to polar coordinates 149
8.4 Contour of a polar image 151
8.5 Difficulties in matching detections 153
8.6 Matching of an element 155
8.7 Comparison of tracking approaches 157

8.8 Different detection errors . . . 158
8.9 Occlusion handling . . . 159
8.10 Fragmented detection . . . 160
8.11 Different tracking errors . . . 160
8.12 Merging and splitting of blobs . . . 161
8.13 Seperation problem . . . 164
8.14 Occlusion handling on the S2.L1 Video . . . 165
8.15 Comparison on the TownCenter video . . . 166
8.16 Comparision with the Hungarian method . . . 168

List of Tables

4.1 Comparison of spatial methods 96
4.2 Results on the Wallflower dataset 104

6.1 Background subtraction on the Underwater Change Detection dataset 117
6.2 Evaluation of different image enhancement methods 118
6.3 Results of optical flow feature tracking 120

7.1 Results on the Underwater Change Detection dataset 140
7.2 F1-Scores for each video of the Underwater Change Detection dataset 140

8.1 Results on the S2.L1 Video of the PETS 2009 dataset 163
8.2 Evaluation on the TownCenter video 167
8.3 Run time comparison 169

1 Introduction

Computer Vision is an area that tries to enable machines to conceive their surroundings autonomously with the help of vision sensors, similar to humans who mostly perceive their surroundings with the help of their eyes. At first glance, this task may seem quite simple to many people because the understanding of our surroundings comes so natural to us humans. However, the general problem is extremely difficult and even the best systems with the most advanced algorithms do not come close to the ability of humans in perceiving and understanding universal scenes. In some specific aspects the computer vision systems can already have an advantage in comparison to humans, e.g. they never get tired or can measure distances very exactly, however, they lose when it comes to the interpretation of new scenes with unknown objects and the adaption to them.

To build viable computer vision systems, an understanding of various research areas is necessary, from optics over electrical engineering to machine learning. In all these fields great advancements have been achieved in the last years. Today, high-resolution color cameras are becoming ubiquitous in our society and fast computer vision algorithms are present in every pocket of every human, incorporated in their mobile phones or cameras. In the next years, these developments will continue and further advancements will make self-driving cars or grocery stores without checkouts possible, which all rely heavily on computer vision algorithms. Both of these and many more technologies are at the moment in the experimental stage and will bring great changes to our daily life and society as a whole in the next decades.

In many areas these changes have already happened so that computer vision plays a vital role in millions of economic and social interactions daily; examples are the automated license plate recognition of cars with optical character recognition (OCR) [KBV15] which is used to check if all car owners have paid their tolls but also for security purposes to identify cars that were stolen or are used by criminals. In this ever more importance gaining area of security falls also the face recognition which is already quite advanced and able to compete with the human perception. It is used often for purposes like the border control at airports [OK15] but also has other civil usages, e.g. the correlation of images to specific users in social networks. The last and maybe most significant example for individual lives is in the area of medical imaging where computer vision algorithms are already helping doctors identifying illnesses like cancer more accurate and reliable [DP16].

In the case of medical imaging, the task is often to segment an image derived by a medical imaging device, e.g. an MRT, into diseased/abnormal and healthy tissue to support the physician. These segmentation tasks are a cornerstone of computer vision as they present usually the first step in a whole pipeline of different algorithms and the quality of all later results depends heavily on accurate segmentations. For example, the automated license plate reader previously mentioned will first segment all license plates in a frame and afterwards identify the individual letters on each plate with OCR. Therefore, the development of accurate, fast and universal segmentation algorithms has been a constant and much-discussed problem in the last decades. One of the main difficulties is that there are so many different scenarios in which segmentations are necessary that it is impossible (at the moment) to develop an algorithm that works with a consistent accuracy in all of them (e.g. single image vs video segmentation or segmentation of specific object vs segmentation based on behavior/events) and hence many special algorithms have been developed for specific scenarios. This work will deal mainly with one of these segmentation tasks which is very frequently used in computer vision, namely change detection in videos.

1.1 Purpose and Scientific Problem

The detection of change in videos is a classical segmentation task that derives its importance from the fact that changes in an otherwise static scene correspond to moving objects and that these objects are usually the ones an observer is interested in. For a self-driving car or any other robot, for example, a moving object poses a much greater threat of collision than any static object. To avoid a collision with an immobile object it is always enough to stop and do nothing whereas the collision avoidance with an object in motion often requires evasive maneuvers and the prediction of the path that object follows. For the detection of these objects, different algorithms like background subtraction or the optical flow have been proposed but each of them faces separate problems in certain scenarios and the research for an optimal solution is still ongoing. The optical flow, in general, is better suited for situations where the camera is under movement itself (ego-motion) but has problems with large uniform or slow-moving objects. This work will mainly deal with change detection via background modeling and subtraction which is a fast and general method but requires a static camera so that the background can be modeled. The main disadvantage of this method is the lack of spatial coherence in the results since each pixel is treated completely separately in the original formulation.

Natural images and videos exhibit a great amount of smoothness and this feature is not inherently reflected in the background subtraction results. Because of this, the segmentations look unnatural and it becomes also more difficult to automatically process them further. The smoothness in natural images comes from the fact that the objects there have continuous and smooth borders, hardly ever holes and a certain minimal size (at least if they have any importance to the observer). Therefore, the segmented objects should have these characteristics as well and not have zigzagging borders, holes or consist of only one or two pixels. Methods that try to model these smoothness criteria directly, e.g. with Markov Random Fields (MRF), tend to be very computationally expensive and are, therefore, not suitable for real-time applications. Simpler methods, like morphological filters or a Gaussian smoothing of the video stream, accomplish significant improvements but do not use the information from the video stream and, therefore, cannot achieve the accuracy of more advanced models. In this work, three approaches will be presented and evaluated which address this problem specifically from different viewpoints.

This spatial coherence is also especially important for the further usage of the segmentation results, as it is necessary to have clear and unambiguous detections for the next stages of the computer vision pipeline. Subsequent tracking or classification algorithms, no matter how good these are, will get into serious problems if large objects are detected as several smaller objects or small detections of noise are seen as real objects. An example of the effect of these spatial methods can be seen in Figure 1.1, although the shape of the segmentation is far from ideal even

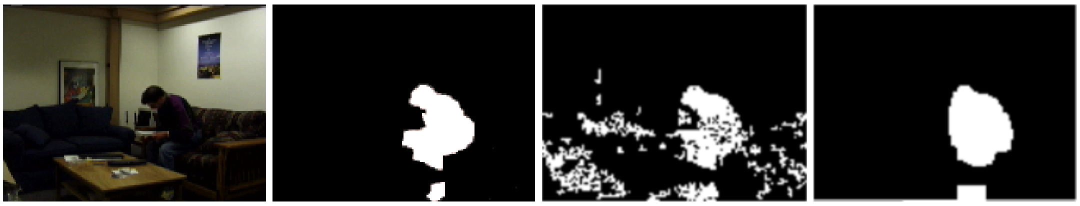


Figure 1.1: The images on the left show a frame from the *Time of Day* video from the Wallflower dataset and the corresponding ground truth data, on the right side are the results of the approach from [SW06], once with only the background subtraction and the other with a MRF model added as a spatial method.

after the spatial method was applied, it becomes clear that there is one large and another small object in the scene. Without the spatial method, the segmentation is quite open for interpretation and it becomes very difficult to further process this result for any algorithm. Also, it should be noted that the methods do not have the ability to know that both detected objects belong to the same human and that there is just an occlusion from the table. The low-level algorithms do not have any concept of a human and a realization of this level of understanding has to happen in higher layers of the computer vision pipeline.

To evaluate the results of these segmentation methods there exist many datasets with a great variety of different scenarios, from low frame rate videos to thermal recordings and from scenes with bad weather conditions to situations with difficult shadows. However, none of these have any underwater footage included and, hence, the specific problems of the underwater world are a blind spot for the change detection community. These problems include the changed physical conditions in water which cause refraction, induce a color cast or let many small organic particles float in the water which can then obstruct the view of the camera. The smallest of the floating particles cannot be seen by a human directly but reflect and scatter the light which causes the images to be blurry. The larger floating particles (Marine Snow) are visible and can be so big that it becomes difficult to differentiate them from moving objects like fish, jellyfish or divers.

These problems are known to underwater photographers and filmmakers for a long time, and thus there have been many image enhancement algorithms proposed to eliminate these degradation effects. Usually, the goal of these algorithms is to improve the human perception of a video or image and not to support computer vision tasks. Nonetheless, as the effect of underwater image enhancement algorithms for a human observer can be quite beneficial, it is sensible to assume that similar favorable effects can be found for segmentation algorithms. Since there does not exist a general change detection dataset for underwater scenarios, a conclusive evaluation of these effects was not possible so far, let alone an adaption or new development of algorithms to positively affect the segmentation process. This adaption would probably be necessary as the original approaches are often quite slow because they were designed for single images and/or offline usages which limits their usability drastically.

Other effects that make the segmentation and tracking more difficult come from the special scenarios that occur in underwater situations. An example are fish that often appear in whole swarms and constantly swim in front and behind each other, this can make it very difficult to identify a single fish. For in-air situations crowded scenes are also possible – although not as frequently – but the main advantage here is that the problem can be mostly mitigated by placing the camera in a higher position and getting a bird's-eye perspective. This will limit the occlusion drastically because humans or cars are not stacked on top of each other and, thereby, the segmentation task becomes a lot easier. The same is not possible for underwater scenarios as fish or divers can and do use all three dimensions of the medium and occlude each other in all of these dimensions as well. Another problem that becomes more difficult in underwater scenarios is shadows. On the one hand, there is a new kind of shadow/light effect that does not exist in-air, caustics. On the other hand, the normal shadows are harder to detect because of the deteriorated color information in underwater scenes and the fact that an attribution of a shadow to a specific object is often very difficult for swimming or floating objects.

Another specific difficulty of fish detection – apart from the swarm behavior of fish – is that they tend to camouflage themselves. Of course, this does not apply to all fish, some are very colorful and clearly stick out from the background, but basically all commercially used fish (especially in the European area) have a grayish color so that they are harder to see in the open sea for predators. This naturally makes it harder for computer vision algorithms to detect them accurately since the difference between the static background and the moving foreground objects becomes very small in an already degraded image. Nonetheless, it is often an essential task to detect these fish, e.g. in aquacultures to monitor the growth and health conditions of them or for biologists to automatically survey the marine life. However, the similarity of these fish can also be a small advantage, in particular in the crowded scenes when a whole swarm is present. As all of the fish feature almost the same color, a foreground model has an easier task in modeling the moving objects. An accurately modeled foreground can be helpful in differentiating between moving objects and the background. Foreground models are not often used for in-air scenarios since there the difference between two cars or two persons is usually quite large which negatively affects the reliability of such a model and makes it overall not as useful.

1.2 Structure of the Thesis

There is already a large base of contributions on change detection and general segmentation methods in videos and therefore, in this work, the focus will be on two specific points. One is the lack of spatial coherence in the results of background subtraction methods for which three different solutions will be proposed, each having their own special advantages and drawbacks. The other topic that will be discussed in depth is the difficult conditions in underwater scenes and how to best deal with them. For this, a special dataset is proposed and published, different image enhancement algorithms are evaluated on this dataset and a special method for crowded scenes is developed.

Before discussing these methods, the thesis begins with a state of the art review on change detection and segmentation algorithms for videos in chapter 3. Since the approaches for (single) image segmentation, in-air video segmentation and underwater video segmentation are very closely related and often overlap, it is difficult to differentiate perfectly between them. In this work an algorithm is put into the category of video segmentation if it uses the temporal progression of the video, e.g. a simple thresholding would not qualify since it is applied on each frame separately (previous or following frames do not influence the algorithm) and therefore counts as image segmentation even if it is used on videos. However, since image segmentation methods often are the foundation of video segmentation methods – or at least are present in several substeps of the segmentation process – the first part of the state of the art deals with image segmentation methods. To limit the overview to a reasonable length only the methods that were used in underwater scenarios are investigated. Subsequently, the focus is put on video segmentation and primarily in-air scenarios are investigated where plenty of literature and several datasets are available. The last part then deals with the quite rare occasions where video segmentation algorithms were applied in the underwater context.

The chapter 4, Change Detection, introduces a novel background modeling and subtraction algorithm which was previously published in [RG15]. Based on this, three different spatial methods are presented to enhance the completely pixel-wise segmentations derived from the background subtraction. The first idea uses MRF to model the spatial connections between the pixels and the resulting optimization problem is solved approximately with Belief Propagation. In contrast to other similar approaches, higher order MRFs are used to better model the spatial relationships in large complex frames and Otsu's method was integrated into the Belief Propagation optimization for a better adaption to edges in the frame (previously published in [RG15]). The second approach uses the idea behind NCut, which is originally a single image segmentation method, and adapts it for videos so that it becomes a fast way to adapt the segmentation results of the background subtraction to the edges in the frame. This method was previously published in [Rad+15]. The last approach is a spatiotemporal method that combines several consecutive segmentations by using dense optical flows to eliminate single outliers and make the segmentations themselves spatiotemporal more consistent (published in [RF16]). Eventually, all these methods are evaluated and compared with many already existing methods on the Wallflower dataset to point out the strengths and weaknesses of each of them.

In the next chapters the difficult situation for change detection algorithms in underwater scenarios is examined and some ways to better handle these difficulties are proposed. At first, in 5, a Dataset for Underwater Change Detection, a new dataset is presented because until now no common change detection dataset contains any underwater videos. It consists of five videos which depict different difficulties present in many underwater videos and was previously published in [Rad+16]. In Chapter 6, Effect of Image Enhancement on Underwater Change Detection, four different image enhancement methods are tested in combination with different change detection methods to see if they can have a constantly positive impact on the segmentation quality. Afterwards, in the chapter 7, Underwater Change Detection, the GSM background subtraction introduced earlier is adapted to better deal with the special degradation effects in the medium water and the swarm behavior of fish. For this, the GSM is combined with the Mixture of Gaussian approach and important parameters are adapted automatically so that it can deal better with complex scenes. Also, a foreground model is integrated and a special method to deal with very crowded scenes that is based on the optical flow of the scene. These changes are already published in [RFL17b] and are at the end evaluated on the new underwater dataset in combination with the different spatial methods introduced earlier.

Underwater Blob Tracking is the 8th and last chapter where the previous results will be used to extract important higher-level information from the videos, e.g. the path of a single fish or how fast the fish swims. It is based on the methods described in [RFL18]. To keep the approach as general as possible it is solely based on the found foreground blobs from the change detection which makes the spatial coherency of these results very important. First, a complex similarity measure is defined, the Connected Component Similarity Measure (CCSM), to evaluate how likely it is that two foreground blobs in two different frames represent the same object. The CCSM uses many different properties of the blobs, e.g. position, shape or velocity, and has also a way to prevent outliers in the most important parameters. Based on this, the foreground detections of different frames can be matched, however, the special matching process introduced here does not only look at single blobs but also takes into account possible unions or splits of these blobs. This allows a better handling of occlusions and segmentation inaccuracies. The evaluation is done on two in-air videos with humans as foreground objects since for these videos ground truth data and many previous results are available. On the underwater videos, only a comparison with a reimplemented Hungarian method was possible.

Summarized, this work will address the following problems and questions:

1. Development of a novel and lightweight background modeling approach for foreground-background detection
   - Handling of special difficulties like shadows or global changes
   - How does it compare to State of the Art in-air methods?
   - Adaption to the challenges of underwater footage, especially crowded scenes and image degradation
2. Search for better approaches to increase the spatial coherence in binary segmentations
   - Development of different approaches, comparison in run-time and accuracy improvements
   - How is the performance on underwater videos? How is their interaction with image enhancement methods?
3. Creation of an underwater dataset for foreground-background segmentation
   - Important: representation of all common problems in the underwater context
   - Are some algorithms better suited for the segmentation underwater? (especially background subtraction versus optical flow)
   - Can image enhancement methods mitigate the common problems in underwater videos?
4. Creation of a novel very general tracking approach that relies solely on binary segmentations

To all of these problems solutions will be proposed in the following chapters.

2 Fundamental Definitions

At first, some brief definitions will be given which are essential for understanding the following parts. The readers that are already familiar with topics like Computer Vision or Change Detection can skip this chapter.

This work will deal mostly with digital videos which, in this case, can be viewed as a stack of single images. Then again, each (digital) image is a two-dimensional grid of pixels.

Pixel

The word pixel comes from ”picture element” and is the smallest component in a digital image or computer screen. Each pixel is assigned to a specific color, which is usually encoded with the three RGB values. RGB stands for red, green and blue, these colors are used because out of them all other colors the human is able to perceive can be mixed. The intensity of each color is commonly given by an integer value that ranges between 0 and 255. For the pixel v this means

$$v = (r, g, b) \in \{0, 1, 2, \ldots, 255\}^3. \tag{2.1}$$

Each of these three values is called a channel (e.g. red-channel); each channel represents specific information on a distinct domain. There are pixels with only one channel which then only encode information about the brightness (grayscale).

In general, a pixel in a digital image is not limited to one or three channels but can represent much more and universal information. By adding more channels it could also give information about radio waves, the infrared spectrum or any other data source. However, the human perception is limited and therefore the devices that are used for displaying pixels (monitors, printers ...) are also limited to the visible spectrum. If more or different information shall be displayed they have first to be mapped for each pixel to the RGB range.

Digital Image

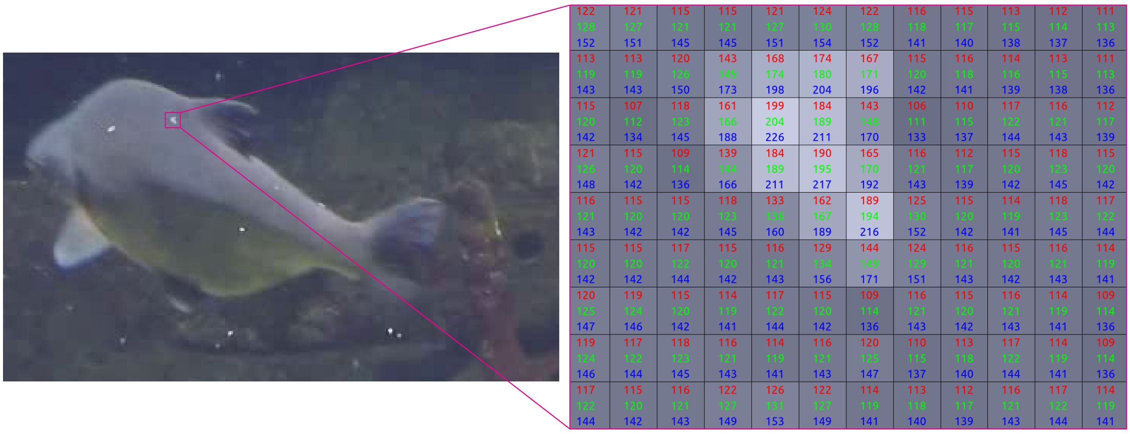
A digital image (hereinafter simply referred to as image) is a two-dimensional grid of pixels. Let I be an image of width m and height n, then

$$I = (\psi_{i,j}) \quad \text{with} \;\; 1 \leq i \leq m \;\; \text{and} \;\; 1 \leq j \leq n \tag{2.2}$$

and

$$I(i, j) = \psi_{i,j} \tag{2.3}$$

is the pixel at position (i, j). If not stated otherwise, a pixel in an image in this work will always have the three RBG channels and a specific channel can be addressed by giving the channel



122 128 152	121 127 151	115 121 145	115 121 145	121 127 151	124 130 154	122 128 152	116 118 141	115 117 140	113 115 138	112 114 137	111 113 136
113 119 143	113 119 143	120 126 150	143 149 173	168 174 198	174 180 204	167 171 196	115 120 142	116 118 141	114 116 139	113 115 138	111 113 136
115 120 142	107 112 134	118 123 145	161 166 188	199 204 226	184 189 211	143 148 170	106 111 133	110 115 137	117 122 144	116 121 143	112 117 139
121 126 148	115 120 142	109 114 136	139 144 166	184 189 211	190 195 217	165 170 192	116 121 143	112 117 139	115 120 142	118 123 145	115 120 142
116 121 143	115 120 142	115 120 142	118 123 145	133 138 160	162 167 189	189 194 216	125 130 152	115 120 142	114 119 141	118 123 145	117 122 144
115 120 142	115 120 142	117 122 144	115 120 142	116 121 143	129 134 156	144 149 171	124 129 151	116 121 143	115 120 142	116 121 143	114 119 141
120 125 147	119 124 146	115 120 142	114 119 141	117 122 144	115 120 142	109 114 136	116 121 143	115 120 142	116 121 143	114 119 141	109 114 136
119 124 146	117 122 144	118 123 145	116 121 143	114 119 141	116 121 143	120 125 147	110 115 137	113 118 140	117 122 144	114 119 141	109 114 136
117 122 144	115 120 142	116 121 143	122 127 149	126 131 153	122 127 149	114 119 141	113 118 140	112 117 139	116 121 143	117 122 144	114 119 141

Figure 2.1: On the left side is the underwater image of a fish and a small part of this is zoomed in and displayed on the right side. There the separate pixels can be distinguished and also the RGB values are shown for each of them.

$c \in \{R, G, B\}$ after the pixel position ($I(i, j, c)$). A good visualization of an image with its pixels can be seen in Figure 2.1.

There are two kinds of images, the first one is created with cameras and then the image is a reproduction of the reality (of the electromagnetic waves that reach the camera). Normally this is in relation to the visible electromagnetic spectrum, so a reproduction of how we humans perceive our surroundings, but there are also cameras that work in other spectra. The second type is artificial images, they can be created on a computer and do not have to have any connection to the reality. An example is Computer games which are a steady stream of artificial images.

This work will solely work with the first type, taken with cameras, as these images are measurements of the reality and therefore contain unknown information about our surroundings which can be extracted and used. At the same time, the analysis of artificial images is moot since all information that can possibly be extracted from them is or was already present somewhere. For example, when an image that depicts humans is created, the number and positions of the humans is known at the time of the creation and therefore a complicated algorithm that detects humans is in most cases not required.

Video

A video is composed of several images of the same size - often many thousands per video - that are ordered equidistantly on a timeline and played/shown one after another. Therefore, a video has an additional time domain to the already existing spatial domain of every single image. This allows the capturing of changes over time in a scene which is often very beneficial and opens completely new ways to obtain meaningful data. An example would be the detection of all moving objects in a scene which cannot be done on single images alone. Additionally, sound is also often present in videos as a kind of fourth dimension. In this work, however, it is not used as a source of information and only the previously mentioned time and spatial dimensions are used.

The new time domain not only enhances the usefulness of a video dramatically (in contrast to an image), but also the amount of memory needed is much higher and increases approximately linear with the length of the video. Thus, an efficient compression is necessary to work comfortably with videos. Usually, in addition to an already efficient single image compression, only the changes from one image to the next are saved since there are only minor changes present most of the time (e.g. the background stays unchanged). The images in a video are also often called frames, which comes historically from the fact that an image on a strip of photographic film looked similar to a framed pictured.

The standard symbol for a video will be a capital V, similar to the I for an image. Each video is a combination of T Images

$$V = (I_t) \qquad \text{for } 1 \leq t \leq T \tag{2.4}$$

and the pixel at position (i, j) and time t can be denoted as

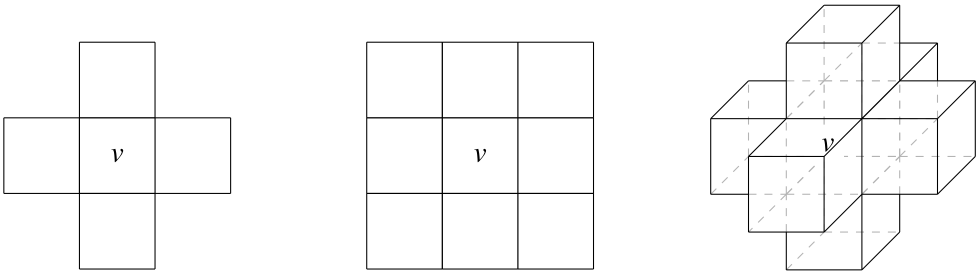
$$V(i, j, t) = I_t(i, j) = \psi_{i,j} \in I_t. \tag{2.5}$$

A specific channel of that pixel can be addressed with $V(i, j, t, c)$.

Neighborhood

The neighborhood of a pixel v in an image I is a set of pixels from that image that are spatially close to the pixel v. Depending on the metric to measure the spatial distance different neighborhoods can be created, the most common ones are the von Neumann and the Moore neighborhood, the first is based on the L_1 distance (Manhattan distance) and the second on the L_∞ distance (Chebyshev distance). For images, the neighborhoods are always two-dimensional but for videos they can become three-dimensional since pixels in the next or previous frames can become part of the neighborhood as well, an example is depicted in Figure 2.2.

Neighborhoods are important for many image processing and computer vision tasks since they can give a spatial component to otherwise pixel-wise methods. For example, thresholding the intensity differences between neighbors can be used for a simple edge detection.



von Neumann neighborhood

Moore neighborhood

3D von Neumann neighborhood

Figure 2.2: Shown are three standard neighborhoods for digital image and video processing. The pixel v is always in the center and its neighborhood consists of all pixel around it that are shown here.

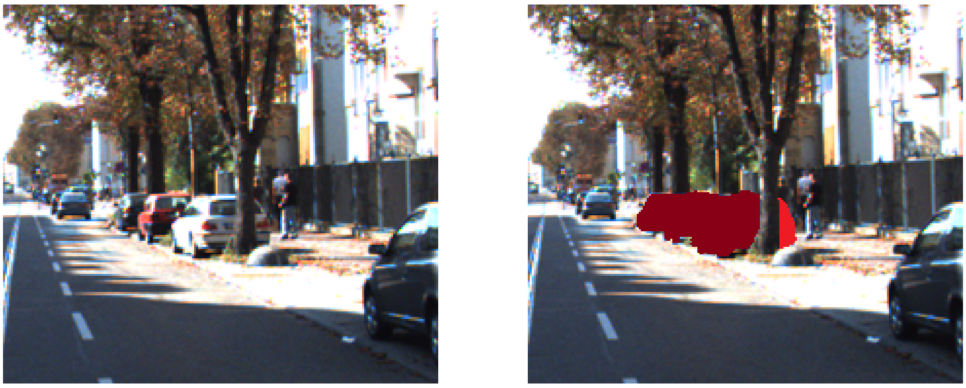


Figure 2.3: On the left is the original frame from the KITTI Vision Benchmark Suite (Optical Flow Evaluation) [MHG18] and on the right the same frame is shown with two connected components marked (e.g. the result of a car detection algorithm). It can be seen that the grey car is separated into two connected components because of the tree and that the red connected component comprises of three different objects because they all overlap.

Connected Component

A Component C is a simple subset of pixels from an image, for example foreground pixels that were detected by background subtraction,

$$C = \{v \mid v \in I\}. \tag{2.6}$$

Two pixels a and b from C are called connected if a path between them can be constructed in the following way: From a a neighboring pixel $c \in C$ is chosen. Then from c another neighboring pixel (from C) is chosen and so on. If the pixel b can be reached by repeating this process, a and b are connected. A component C is called a connected component if all pixels are connected to one another and the set C consists of the maximal number of pixels, that means no other (foreground) pixel can be added that is connected to all pixels of C. The neighborhood chosen has a strong influence on this property, in the following always the 2D von Neumann neighborhood is used to determine connected components.

Object

An object O in an image is also a set of pixels. This set, however, corresponds to a real object (car, tree, human, fish …) in the scene and comprises of all pixels that represent it in the image. The object O can be a connected component but does not have to be. A connected component can consist of several objects and vice versa a single object can include several connected components. An example of this can be seen in Figure 2.3. Furthermore, in a video the same object can be visible for many frames and, therefore, it will be defined over time, so $O(t)$ is an object at time t.

Temporal Change

Temporal change cannot occur in an image since it only depicts a moment in time but it is an important property of videos. A single pixel of a video, $V(i,j,t)$, will exhibit changes of its intensity over time. If these changes are analyzed for all pixels in the video while, optimally, also taking the neighborhood and global relations into account a great quantity of information can be extracted from the video. For example, if most of the pixels suddenly drop in their intensity it can be deduced that probably the light was turned off or (for outdoor scenes) a cloud blocked the sunlight. Similarly, when the pixels go back to their previous intensities it can be inferred that the light was turned on again or the cloud moved away.

Optical Flow

The Optical Flow combines the Temporal Change analysis with a spatial component. There are dense Optical Flows (DOF) which compute the flow for each pixel of the video and sparse Optical Flows which only consider specific points (e.g. edges and maxima). Both assume that pixels not just simply change their intensities but move through the scene over time. This assumption is true for moving objects in a scene; the pixels of that object do not change their intensities, they just change their location. This means that

$$V(i,j,t) \approx V(i+f_i, j+f_j, t+f_t) \tag{2.7}$$

should be true for all pixels of that object as long as it keeps its velocity and direction. Then the vector (f_i, f_j, f_t) is the Optical Flow for that object and its velocity and direction of movement can be deduced. However, humans cannot be seen as one single object since different body parts move with different velocities and directions all the time. Furthermore, the Optical Flow is not apt to handle events that actually change the intensity of pixels - like turning off a light.

The calculation of the Optical Flow is also an example where the neighborhood becomes important. If the equation 2.7 is only evaluated for one single pixel it will be extremely prone to errors. It is very probable that a nearby pixel has a similar intensity and then the unambiguous assignment of pixels over time becomes almost impossible. Therefore, a whole neighborhood is usually evaluated at once and the Optical Flow assigned to that area. The larger the neighborhood the more robust the algorithm becomes, but with the size of the neighborhood also the problems at the border of objects increase and smaller objects will be missed completely.

Foreground and Background in a Scene

In a segmentation of a video or image into foreground and background, the scene is separated into two disjoint parts that add up to the whole scene. That means each pixel has to be classified either as foreground or as background. The foreground part of the scene contains all objects that are important at the moment (to the user, for the next computer vision task, ...) and in the background are the parts that are not relevant and can be ignored for the time being. This detection of foreground objects can be useful in itself, e.g. a surveillance camera that detects movement during the night can alert the security. However, most often it is used as the foundation for other classification and segmentation algorithms.

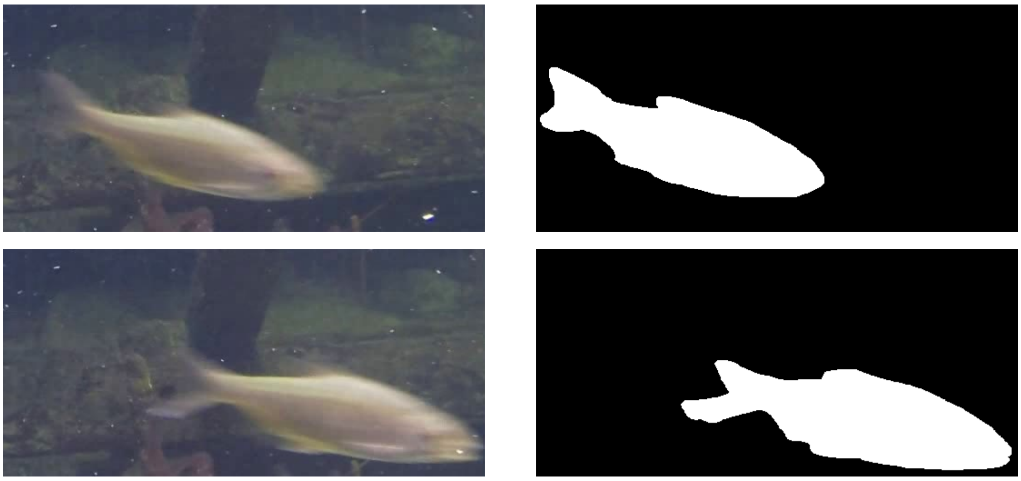


Figure 2.4: On the left are two frames from a video taken with a fixed camera, the background is static and only the fish is moving. A result of a change detection algorithm is shown on the right side. All pixels that were found to belong to moving objects are shown in white and pixels that show the static background in black.

What is considered to be important (foreground) is of course extremely dependent on the specific task and scene. For example, for a self-driving car street signs would be considered as foreground but the sky rather as background since it contains no important information. This thesis will work on the specific task of change detection, where all moving objects in a scene are seen as important and should be classified as foreground and the rest is considered background.

The Change Detection Problem

The task of detecting change in a video is based on the assumption that there is a static component in the video, the background. This static component includes all objects of the scene that are immobile (houses, trees, parking cars and so on). Everything that is under motion does not belong to that static component and should be detected by a change detection algorithm. Since this task relates to motion it is only natural to perform it on videos; doing it on single images is hard to impossible. The two main methods to conduct this classification are either: modeling the background of the scene and comparing it with the current frame of the video (background subtraction) or are based on the optical flow in the video. An example of a detection is shown in Figure 2.4 where the classification is pixel-wise and binary, each pixel belongs either to the static background component or to a moving foreground object.

Gaussian

The modeling of the background - which is the static, non-changing part of the scene - is often done by using Gaussians (normal distributions) which are continuous probability distributions. If many independent measurements are conducted the errors often have a distribution that is almost normal [Lyo14]. In this case, each capturing of a frame is one measurement and since the

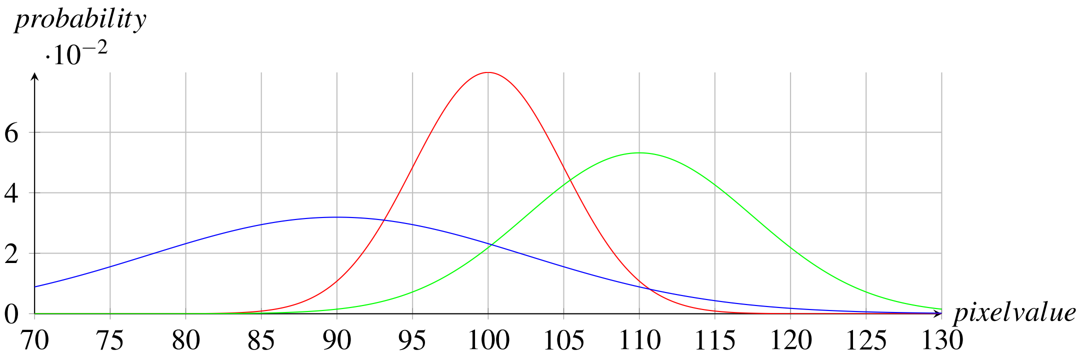


Figure 2.5: Three Gaussians that model the color of a pixel. The certainty of the blue value is the lowest and therefore it has a high probability to attain values in a broad range around the mean of 90.

static component of the scene is constant the measurements should have a normal distribution around this value. Therefore, Gaussians are adequate for this modeling process and every normal distribution is defined by two values, its mean μ and variance σ. In the case of background modeling, μ would be the mean intensity over the last frames of the video. With this, the probability for the value x to appear is

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. \tag{2.8}$$

A visualization of this can be seen in Figure 2.5 where three different Gaussian distributions are displayed, each modeling one of the RGB values of a specific pixel.

Evaluation

To evaluate the accuracy of binary segmentations/classifications there exists many different scores which are deduced from the four basic numbers:

- TP *(true positives)* - a correct detection, e.g. classification as foreground
- TN *(true negatives)* - a correct rejection, e.g. classification as background
- FP *(false positives)* - a false alarm, e.g. classification of background as foreground
- FN *(false negative)* - a missing detection, e.g. classification of foreground as background

To get a useful evaluation of different segmentations from these numbers directly is difficult because they just count the different classification types and do not take into account the specific scene. For example, in a video with very little foreground a method that would just mark everything as background has a very high rate of correct classifications (TP+TN) and may therefore seem good at first glance but it does not convey any information at all to the user.

To tackle this problem different coefficients have been proposed that take the total number of foreground or background into account,

$$\begin{aligned} &\text{Sensitivity:} && \frac{TP}{TP+FN}\,, \\ &\text{Specifity:} && \frac{TN}{TN+FP}\,, \\ &\text{Precision:} && \frac{TP}{TP+FP}\,. \end{aligned} \tag{2.9}$$

The Sensitivity is the ratio of correct detections to the total amount of foreground and signifies how likely the method is to detect a foreground object. Specificity is the ratio of correct rejection to the total amount of background and signifies how likely the method is to detect background correctly. The last number, Precision, is the ratio of correct detections to total detections and shows how likely it is that one detection made by the algorithm is true. These numbers are all between 0 and 1 and a higher value indicates a better segmentation.

However, each of these numbers only describes one very specific aspect of the segmentation accuracy and cannot be taken alone for evaluation. In the previous example – an algorithm which classifies everything as background – the specificity would be one, but the Precision and Sensitivity would both be zero. Therefore, measures have been proposed that aim to give a comprehensive overview of the accuracy and contained information of a segmentation in only a single number. This is a wide field and many proposals have been made [Pow11] but here the focus will be put on two numbers, the F1-Score and Matthews Correlation Coefficient (MCC), because they will also be used later in this work.

F1-Score is defined as the harmonic mean of Precision and Sensitivity and focusses on the detection accuracy. It is a number between 0 and 1 where 1 can only be reached if no detections have been missed and no false detections have been made. It ignores completely the TN value. That means for the F1-Score it is irrelevant if five correct detections and one false alarm have been made on 100 samples or on 100.000 samples. It is defined as

$$\text{F1-Score} = \frac{2}{\frac{1}{Sensitivity} + \frac{1}{Precision}} = 2 \cdot \frac{Precision \cdot Sensitiviy}{Precision + Sensitivity} = \frac{2 \cdot TP}{2 \cdot TP + FN + FP}. \tag{2.10}$$

The second measure is introduced in [Mat75] and based on the *phi coefficient* in statistics. It is a value between -1 and 1 where a measure of 0 would indicate a truly random segmentation, 1 a perfect segmentation and -1 a completely inverted segmentation. The MCC can be calculated as follows

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}. \tag{2.11}$$

Both measures are adequate for evaluating binary segmentations, but the F1-Score is traditionally more often used. In Table 6.1 both values are given for some underwater segmentations and it can be seen that they are closely related.

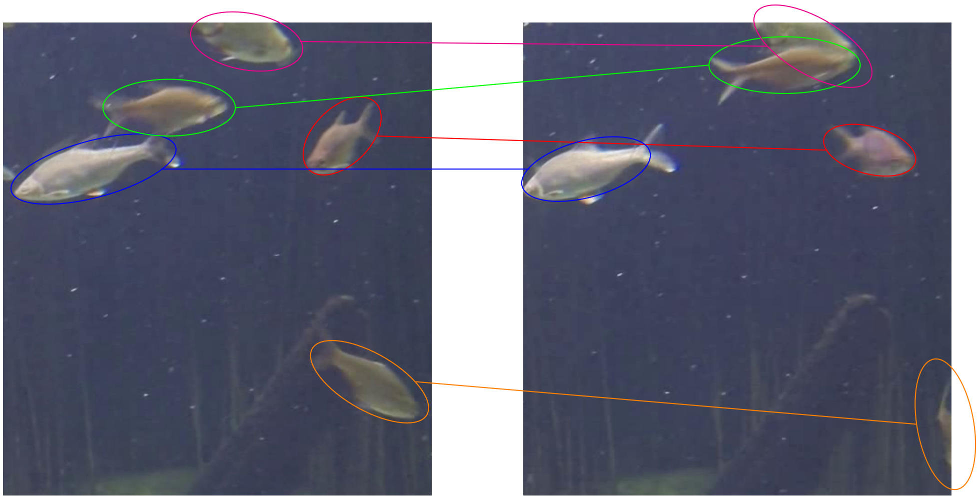


Figure 2.6: Matching of fish between two frames of a video like it is necessary for a tracking algorithm.

If several videos are in a dataset the best way to evaluate the overall accuracy is to compute the F1-Score or MCC value for each video and then give the average number over all videos. This is preferable to adding up the values of TP, TN, FP and FN and then computing the measure. The reason for this is that the videos can contain different amounts of foreground and background and if a video contains very few foreground pixels, each correctly classified pixel has a big impact on the accuracy measure for that video. However, if the numbers of correct detections are added up over all the videos then this impact will be marginalized because other videos may have huge amounts of foreground. As a result, the correct detection of a few small foreground objects in one video would be unimportant for the accuracy measure and only the videos with large amounts of foreground would matter.

Tracking Problem

To track one object in a scene over time it is necessary to associate detections in several frames to one specific object. Humans usually do these by assigning different properties – e.g. the men with the black jacket, green pants and big glasses – and then can find this object again even after many frames and drastic changes in the scene. Computer Vision algorithm work similarly but use different properties like shape, position or size since they usually do not have any concept of *glasses* or *jacket*. With these properties, a matching of all detected objects in a frame has to be done with all detections in the next frame (see Figure 2.6). A matching between two frames that are far apart (with respect to time) is very difficult since properties like position or size usually change drastically over time. Therefore, the aim of a good tracking algorithm is to make correct matchings in consecutive frames and not confuse or lose objects. In modern algorithms, these errors are usually rare but since these errors propagate themselves even a few can lead to poor results, e.g. once swapped tracks stay swapped forever or one missed detection might cause a chain reaction so that many objects are matched falsely. Furthermore, the evaluation of these results is difficult since they not only depend on the tracking method but also heavily on the type and accuracy of the detection and the specific task that is assigned (e.g. only detect and humans vs detect and track all moving objects).

Image Enhancement

Image enhancement or editing is the process of altering the image (digital or analog) after the actual capturing with the camera. It can be done manually with tools like Photoshop to change single images (e.g. for magazines) or automated by a computer (e.g. the red-eye removal in many cameras). Since in this work large videos are processed only the automated enhancement of digital images/frames is feasible and discussed. Especially in unusual and difficult situations (e.g. foggy weather or underwater) image enhancement techniques like deblurring or color correction can help to make important objects more distinct in images. However, it must be noted that no new information is added during this process, just the present information is remapped so that certain aspects are better visible.

3 Related Work

The segmentation problem in-air and underwater images/videos are similar and, therefore, the general approaches in both fields are also alike. However, the underwater scenes have some distinct properties that make some approaches more viable than others. For optical videos, these special properties include color cast, blur, caustics, marine snow, haze or refraction. In this chapter, a general overview of techniques that have been used in the underwater context for the segmentation of videos will be given.

The chapter will start with a short detour to sonar images and their segmentation because they are an important branch of underwater imaging and have some specific benefits. However, they cannot provide enough accuracy overall – e.g. for the identification of single fish and their movements – and that is why the focus will be mainly on optical approaches in this work. The next part of the chapter will address the underwater single image segmentation and gives a brief overview of the current approaches. Although the aim of this thesis is the segmentation of videos, the case of a single image cannot be neglected completely since often very similar algorithms or combinations of them are applied for videos. The last part then deals with current approaches for video segmentation and is divided into two sections, one general video segmentation part and one for underwater videos. The general overview of video segmentation algorithms is necessary because the two topics are very closely related to each other and in-air methods can – of course – also be applied on underwater videos. Furthermore, a comparison of different methods is only possible for in-air scenarios since no common underwater datasets are available until now. This thesis main work will follow a similar structure where first a general segmentation method for in-air videos is presented with three different ways to add spatial coherency to the pixel-wise segmentations. Afterwards, this approach is adapted to the specialties of underwater scenes by refining the modeling process as well as adding underwater image enhancement methods.

3.1 Sonar Imaging

Sonar devices are used for a very long time already and for various purposes in the underwater context, from fish school detection for fishers to the detection of submarines for military purposes. They have some distinct advantages over optical sensors, above all the high range but also the simple determination of depth information through the runtime of the signal. Nonetheless, sonar images are inferior regarding the resolution and signal-to-noise ratio, Furthermore, they can only detect the intensity of one specific wavelength and not of a whole band of wavelengths like optical sensors.

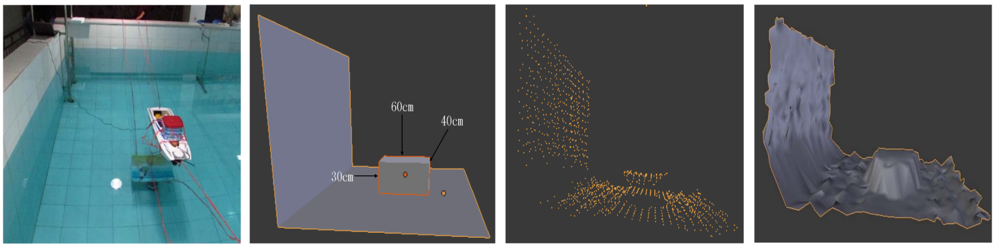


Figure 3.1: Results of [Guo13]. From left to right: pool and boat with sonar device, ground truth data, point cloud after outlier removal and the reconstructed surface. ©2013 IEEE

3D Surface Reconstruction

In [TCM04] Tao et al. focused on real-time capability rather than accuracy and fitted a superquadric surface over sparse 3D points to reconstruct surfaces. To deal with outliers they started with a RANSAC algorithm (Random Sample Consensus) to generate a reliable seed and then added points gradually until no more points were left that fit the model.

The approach of [Guo13] was very similar but only a single beam sonar was used. These devices are cheap and readily available but only deliver 2D slices which then have to be fused to create a whole model of the surface. Again, the outliers were a big challenge for the method but this time an outlier removal filter was applied instead of the RANSAC algorithm. The surface was modeled by a simple rectangular plane where the corner points were matched with points of the point cloud. This plane was then continuously subdivided and the new points on the plane were always shifted on top of already existing points of the point cloud. The algorithm was tested in a small pool, an image of this and some results can be seen in Figure 3.1.

Object Segmentation

For the identification of objects usually more advanced sonar devices are used as a higher resolution is necessary. Also, often side-scan sonar images are taken, these images cover a larger area and are therefore preferable to normal sonar scans. However, these devices also have particular problems with shadows, obstruction, and noise due to the viewing angle and the larger distance to the objects of interest. An unsupervised learning algorithm is proposed for the segmentation of these images in [ZWH16]. Local binary patterns (LBP) were used as well as Haar-Like features to identify features in the sonar images and then cluster them with the K-Means algorithm to segment the image. In this way, a real-time segmentation without any parameters could be achieved. However, more experiments are necessary to verify the quality of the segmentation results as only two images were used for testing.

Huo et al. proposed in [Huo+16] a segmentation method for side-scan sonar images that starts with a special non-local mean-based speckle filtering to reduce the heavy noise of these images. Afterwards, a coarse first segmentation is computed via K-Means clustering which is refined by using a region scale fitting active contour model. Furthermore, the Canny operator was used for the detection of edges to ensure that the active contours coincide with the desired edges. The algorithm was tested on two different datasets of sonar images and on standard black and white

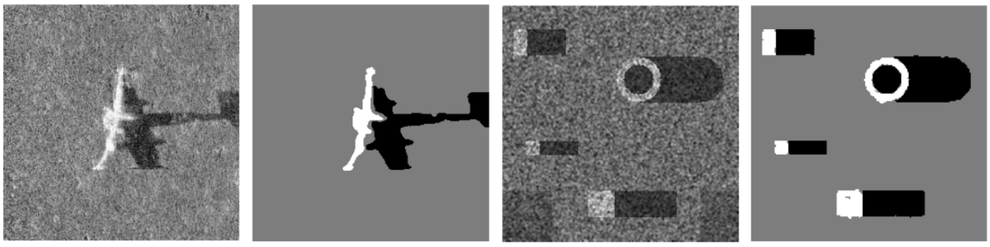


Figure 3.2: Two noisy sonar images are depicted and next to them the segmentation result of [Huo+16]. ©2016 IEEE

images, in almost all scenarios it could beat two previous methods. Some results can be seen in Figure 3.2. There exists a multitude of other approaches for the segmentation of objects in sonar images, among others MRF [Son+16] or statistical background modeling [Hag+16].

Fish Segmentation

The segmentation of fish with sonar is extremely difficult and often it is only used to detect whole schools of fish in the open sea and/or coarsely measure their size [Iid+04]. A refinement of this was proposed by Otaki et al. in [Ota+11] by using a sonar which responded to acoustic tags on the fishing net to increase the accuracy and range of the sonar and thereby helping the fisher to detect fish schools. A fish observatory platform using sonar imaging was proposed in [WB14] but the accuracy of the sonar alone would not have been enough to classify the fish or measure their size. Therefore, the sonar system was used as a first step to generate a coarse segmentation and estimate the number of fish, this detection then activated a stereo camera system to start a more detailed observation. The segmentation of the sonar images was done by a simple background subtraction using Kalman Filters. An example can be seen in Figure 3.3.

The classification of fish solely based on sonar data was attempted by Matsuo et al. in [Mat+07]. A broadband sonar was used and additionally the different reflection properties of fish species in different angles to the sonar system were exploited. However, the conditions were very artificial as the fish were anesthetized or suspended during the imaging. A classification like this under realistic conditions seems very far-fetched at the moment. Overall, sonar imaging is preferable to get coarse information of large areas underwater but if specifics and details are necessary optical imaging is superior. Also, the frame rate is usually quite low due to the lag of the signal and this can be very problematic for tracking or similar tasks [Tru+00].

3.2 Optical Imaging - Single Image Segmentation

Optical underwater imaging can deliver a higher information density and accuracy than sonar imaging but is limited to close range applications because water absorbs and scatters visible light strongly – much more than air. Sonar devices are optimal for the coarse 3D reconstruction of surfaces or the location of large objects or whole swarms of fish, but if accurate information is necessary, e.g. the location of single fish or small objects on the ground, there is no alternative to optical imaging even in the underwater context.

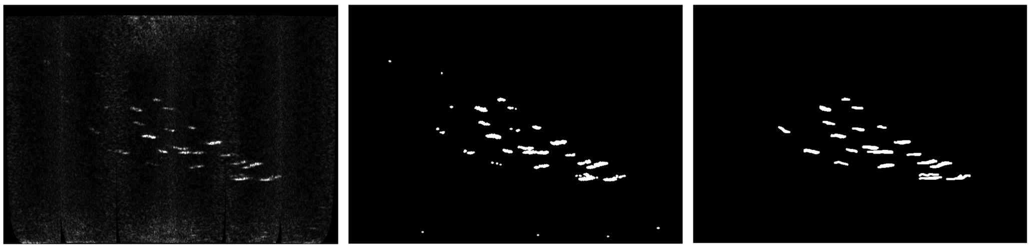


Figure 3.3: A sonar image of fish. From left to right: original image, segmentation of [WB14] and the ground truth data. ©2014 IEEE

Thresholding

Thresholding is a very simple and fast method to binarize an image. The classification of each pixel value $v = I(\bar{v})$ is usually done on grayscale images by a comparison against the predetermined thresholed T and gives the segmentation

$$Seg(\bar{v}) = \begin{cases} 1 & \text{if} \quad I(\bar{v}) < T \\ 0 & \text{else.} \end{cases} \tag{3.1}$$

The vector $\bar{v}$ contains the pixel location (i, j) in image I and in this work all parameters that denote vectors will have a line over them to distinguish them from scalar quantities. Thresholding can also be applied to color images if the norm of the vector of pixel values is taken instead of the single grayscale value. It is very difficult and important to get an optimal value for the threshold T and many methods have been proposed for this. One example is Otsu's method [Ots79] where the variance in each class (foreground/background) is minimized. The method has been extended since then in many different ways, e.g. in the two-dimensional Otsu's method [JWY91] where a two-dimensional histogram is used – the second dimension shows the average of a small area around each pixel. Thresholding usually only works for special tasks or simple images because the complexity of understanding natural images cannot be reduced to a single value. Nonetheless, it has been applied for some underwater tasks, e.g. in [Lab+12] where fish fry was counted in a small glass container. They used an adaptive threshold to account for the changing lighting situations in different parts of the scene and averaged the results over several frames of a video to reduce outliers. The counting was accurate for batches of up to 700 fish but fails at larger batches since the fish then constantly overlap, which cannot be handled by thresholding alone anymore. In this scenario thresholding was an adequate method since there was a very distinct difference between the dark fish and the white background. However, for more difficult scenes with complex backgrounds, this method cannot be used anymore because it becomes impossible to define the difference between foreground and background by a simple threshold.

In [RGS12] different thresholding techniques were tested and combined with image enhancement on underwater scenarios. In contrast to the standard global thresholding, a multilevel approach was used which marks all pixels as foreground that are between two values,

$$Seg(\bar{v}) = \begin{cases} 1 & \text{if} \quad T_1 < I(\bar{v}) < T_2 \\ 0 & \text{else.} \end{cases} \tag{3.2}$$

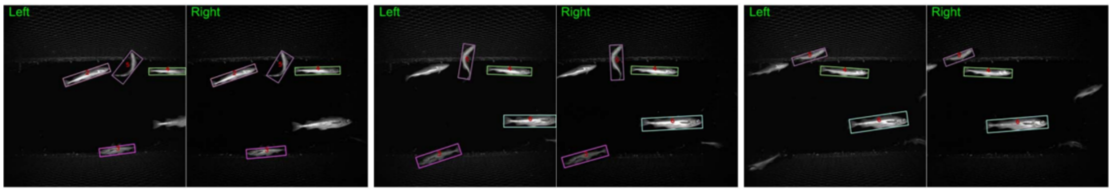


Figure 3.4: Results of the fish detection and tracking algorithm from [Chu+15]. ©2015 IEEE

For the image enhancement, a Contrast Limited Adaptive Histogram Equalization (CLAHE) was chosen and could significantly improve the results because the foreground areas were more distinct from the background regions and therefore it became easier to find a threshold which separates these two. This addition could mitigate the problems of thresholding in complex scenes just mentioned but are not enough to make it a viable option.

Thresholding with Histogram Backprojection

A stereo camera system with LED strobes was used in a trawl by Chuang et al. in [Chu+15]. Due to the lighting conditions and LED strobes the video shows a completely black background with partially bright fish in the foreground. First, an ellipse was constructed around bright areas (which are probably fish) so that a local thresholding with Otsu's method could be applied there. Two segmentations are computed with this method, one with a higher and one with a lower threshold. Both segmentations are then 'binned' locally, that means counting how many pixels in an area fall into each category (here foreground or background). With this, a ratio histogram is created which compares the number of pixels in the bins for the foreground areas in both segmentations. This histogram is then back-projected onto the frame which means that each pixel gets assigned the value of the ratio histogram that corresponds to its own pixel value. Finally, the frame is segmented by thresholding these ratios. This rather complicated thresholding process is necessary to deal with the varying lighting conditions, where the fish are partially very bright but also contain parts which are dark. The results are further enhanced with a spatial component to eliminate outliers and then a stereo matching and tracking are applied. The stereo matching is done with a block matching, where each fish is divided into four blocks along its horizontal axis and is matched with the best corresponding block in the other frame. Afterwards, a temporal matching is done with four cues (vicinity, area, motion and histogram distance) which are used in a Viterbi data association algorithm (first proposed in [Vit67]). Some results of the whole approach can be seen in Figure 3.4. Almost all fish could be detected by their algorithm (Precision: 0.98) but the accuracy of the segmentations is meager for the relatively easy conditions with a Precision of 0.746. Again, this method has the problem that it depends on a completely black background so that everything bright can be assumed to be a fish, e.g. in the first step where the ellipses are constructed around the fish. Therefore, it cannot be applied to general scenarios.

Mean Shift

The mean shift algorithm was introduced by Fukunaga and Hostetler in [FH75] and is a nonparametric clustering technique which does not require prior knowledge about the number of clusters

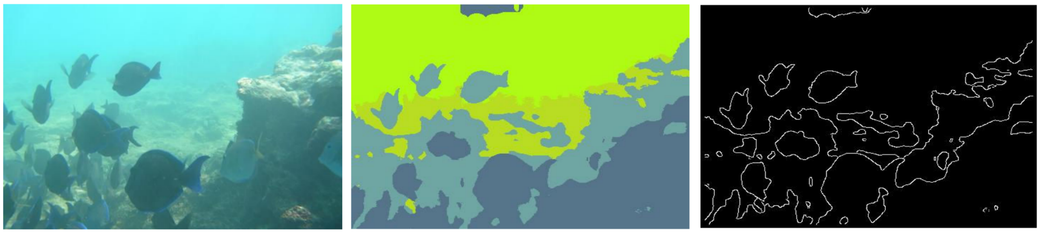


Figure 3.5: Segmentation with the mean shift algorithm from [MS08]. On the left is the original image, in the middle the result after the clustering and on the right the detected edges based on the clusters. ©2008 IEEE

in the data. The mean of a cluster with the initial estimation of the cluster location $\bar{x}$ is defined by

$$\zeta(\bar{x}) = \frac{\sum_{x_i \in N(\bar{x})} K(\bar{x}_i - \bar{x})\bar{x}_i}{\sum_{\bar{x}_i \in N(\bar{x})} K(\bar{x}_i - \bar{x})} \tag{3.3}$$

with a kernel function $K(\bar{x})$ which is often the Gaussian kernel $K(\bar{x}) = e^{-c\|\bar{x}\|^2}$. The mean shift vector is then $\zeta(\bar{x}) - \bar{x}$ and gives the direction to the area with the highest density. The algorithm converges to static points, where the mean shift vector is zero, which then defines the clusters.

The mean shift approach was used in many different scenarios for the segmentation of underwater images, mostly in combination with prior image enhancement and other segmentation algorithms. For example, in [MS08] for the segmentation of fish and corals in combination with histogram equalization for color correction and a Median-Cut. An example of this is depicted in Figure 3.5. The segmentation of fish was also the objective of [BBN14] where the mean shift was used after an image enhancement step to divide the image into subregions which were then classified by an object detection algorithm. In [Liu+10] the algorithm was used on grayscale images to detect feature points in weld images.

Overall, the mean shift algorithm can be a valuable first step to divide the image into several regions which are then further processed by other algorithms. It cannot, however, deliver useful segmentations on its own since there is no further information given about the clusters computed, e.g. which is a fish and which cluster is background. Additionally, there has to be a quite distinct difference between the foreground object and the background so that they do not end up in the same cluster, some examples of this can be seen in Figure 3.5.

Fuzzy C-Means

Fuzzy C-Means was first proposed by [Dun73] and is a clustering algorithm related to the K-Means approach. In both algorithms, the number of clusters must be defined by the user (K or C) and then each of these clusters is initialized. Subsequently, all data points have to be assigned to one of these preliminary clusters and then an energy function is minimized to optimize this assignment and the clusters. For the K-Means algorithm, this function is the sum of distances of all data points belonging to a cluster to the center $\bar{c}$ of that cluster

$$E_{\text{K-Means}} = \sum_{i=1}^{K} \sum_{j=1}^{N^{(i)}} \|\bar{x}_j^{(i)} - \bar{c}_i\|. \tag{3.4}$$

The variable $N^{(i)}$ describes the number of neighbors of cluster i with the center $\bar{c}_i$ and $\bar{x}_j^{(i)}$ is the i-th neighbor of $\bar{c}_i$. Each point $\bar{x}_j$ will only appear once in the sum above as it is assigned to one cluster exclusively. In the fuzzy C-Means approach the membership of one data point $\bar{x}_j$ to a cluster is not binary anymore but a membership variable. The value m_{ji} defines the degree of membership the data point $\bar{x}_j$ has to the cluster i. The values of m_{ji} are limited to $[0,1]$ and $\sum_{i=1}^{C} m_{ji} = 1$. The corresponding energy function then is

$$E_b = \sum_{i=1}^{K} \sum_{j=1}^{J} m_{ji}^b \|\bar{x}_j{}^{-}\bar{c}_i\|, \tag{3.5}$$

where J is the number of data points overall and the cluster center $\bar{c}_i$ is then defined by

$$\bar{c}_i = \frac{\sum_{j=1}^{J} m_{ji}^b \bar{x}_j}{\sum_{j=1}^{J} m_{ji}^b}. \tag{3.6}$$

The parameter b is the fuzzy exponent and regulates the impact of the membership variable, $b \in [1, \infty)$.

Fuzzy C-Means was used to segment underwater images in the work [Bai+16] in combination with a morphological component analysis (MCA). With MCA they could separate the image into two parts by using sparse representation – a noise part and a texture part – and thereby retrieve a cleaner image which is better suited for the following clustering. For the segmentation, they combine the fuzzy C-Means energy function with the idea of active contours. They limited themselves to two clusters (to retrieve a binary segmentation) but added a factor to the energy function which minimizes the length of the contour between the two clusters. As the membership is fuzzy, the contour is always defined by the two regions of pixels with less or more than 0.5 membership to a cluster. With this energy function, they could segment simple objects spatially coherent in underwater scenarios, an example is depicted in Figure 3.6.

To improve usage of the color information the $I_1I_2I_3$ color space was applied in [Wan+11] and combined with a special metric. The distance between the pixel $\bar{v} = (v_1, v_2, v_3)$ and cluster center $\bar{c} = (c_1, c_2, c_3)$ was defined as

$$d_{CS}(\bar{v}, \bar{c}) = \sqrt{\sum_{i=1}^{3} d(v_i, c_i)^2} = \sqrt{\sum_{i=1}^{3} \left[v_i \log_{10}\left(\frac{v_i + 1}{c_i + 1}\right) + c_i \log_{10}\left(\frac{c_i + 1}{v_i + 1}\right)\right]^2}. \tag{3.7}$$

Furthermore, the energy function was enhanced with a spatial component that takes into account the neighborhood of a pixel and enforces similarity in the membership variable of that neighborhood. The method was able to segment single, simple objects in turbid underwater scenes. In comparison to the mean shift clustering this approach has a higher flexibility because the number of clusters can be fixed – e.g. to two when a binary segmentation is necessary – and the fuzzy

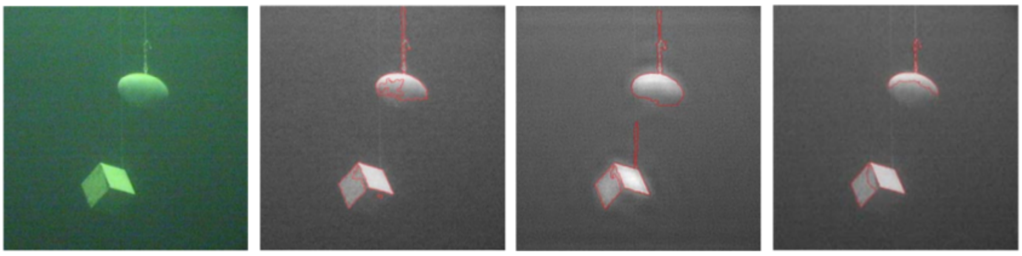


Figure 3.6: One result of the approach of [Bai+16]. One the left is the original underwater image, the next image shows their segmentation on the grayscale version of this image, the next image shows the MCA enhanced version of this image and their segmentation on it and the last image on the right shows a local binary fitting method for comparison. ©2016 IEEE

membership enables a smoother optimization process. However, the need for prior information limits the usability.

As a whole, it can be stated that methods like clustering or thresholding can be useful in special cases or for simple objects, but usually, fail in more demanding scenarios. An example would be a complex background, which makes the usage of these methods futile as they cannot differentiate between foreground objects and unimportant background objects. Therefore, combinations with other methods are necessary for the application on complicated images or videos, e.g. with background subtraction [SHP12; PM16].

Neural Networks

Machine Learning algorithms have gained strong popularity in recent years because of their ability to solve a great variety of very different problems with high accuracy. These algorithms always have a learning phase at first, in which the algorithm is given annotated data from which it can learn what it should detect/distinguish in the images. The statement of the problem, as well as the selection of the data, are crucial points for the success of these methods. A very popular approach at the moment is neural networks which try to loosely model the learning and problem-solving abilities of the biological brain, a small example can be seen in Figure 3.7. There the input layer (red) gets data from the outside which activates the neurons on that layer, e.g. each neuron could represent one pixel of an image and is activated based on the color values of that pixel. The neurons then emit signals which are the input for the next layer and so on. At the end is the output layer which tries to give the desired information, e.g. for each pixel if it is classified as background or foreground.

The important parameters in such a model are the *activation functions* which converts for each neuron the input signals into an output signal and the interconnection between the neurons which determine how much influence a neuron has on other neurons. These values are determined during a training phase and there exist many different algorithms for an optimal parameter learning, see [LBH15]. Images are usually too large to model each pixel as one neuron even with today's hardware and therefore convolutional neural networks (CNNs) are used. There, smaller parts of the image are first processed separately and in later layers tiled together for a better representation

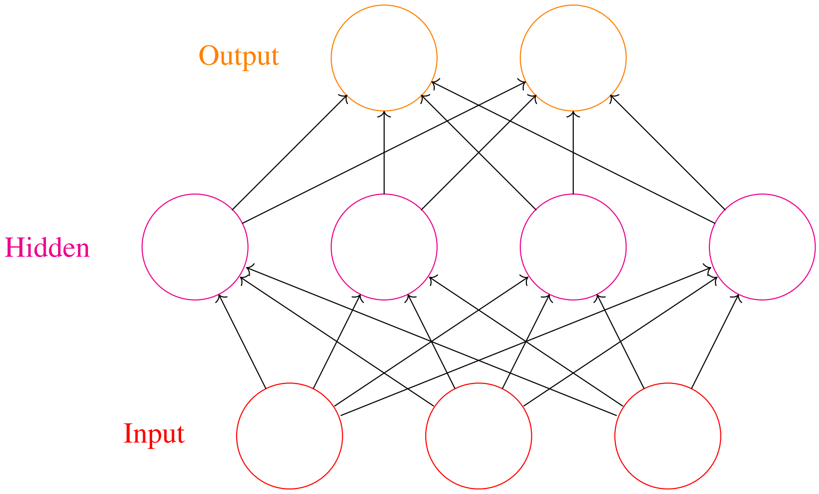


Figure 3.7: A depiction of a small neural network. Each circle represents a neuron and the arrows are the connections between them. The neural network consists of three parts, the input layer (red), one or several hidden layers (magenta) and the output layer.

of the whole image [Cir+11; KSH12].

In the underwater context, they have been used for automatic abundance estimation of cold-water corals and sponges in [Pur+09]. An ROV was used to obtain a video of the seafloor and single frames were then taken for the evaluation of the coral and sponge coverage. A training set of 250 examples for each species was used to train the algorithm but instead of using neural networks on the images themselves they extracted a 30-dimensional feature vector of the images and trained the algorithm on them. The approach had some problems with changing illuminations and was evaluated against three other methods which required user interaction and could provide similar results. Nonetheless, it is a completely automated process (after the training phase with expert labels) and therefore could be effortlessly applied on a very large number of images. A similar approach for the segmentation of corals was used in [ONN16]. There one specific patch of the seafloor was observed for more than one year and every hour a high-resolution image was taken. In a preprocessing step, the images were all aligned and color corrected. For each of these images, the corals were segmented and only pixels that were marked as coral in every single image were taken as coral-pixels. On these pixels, the color change was analyzed to gain data about the developments of corals over longer periods of times (blooming phases etc.). An example can be seen in Figure 3.8.

In [MB16] CNNs were used to detect and classify benthic fauna. High-resolution photo mosaics of the Pacific continental shelf captured by an ROV were used and ten different classes should be detected (e.g. coral, crab, flatfish). The overall accuracy was quite high with up to 84.7% but varied greatly between the different classes. Flatfish had a low detection rate because of their camouflage behavior and other classes were quite similar so that the objects got detected but then wrongly classified. They also compared two different toolkits for the usage of neural networks, the *DIGITS* toolkit from Nvidia and *TensorFlow* from Google. The performance of *DIGITS* was in all tests slightly better but the Python interface of the *TensorFlow* toolkit made it

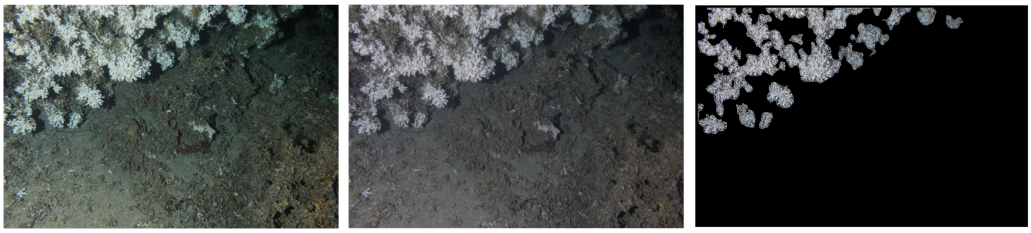


Figure 3.8: Cold-water coral segmentation of [ONN16]. On the left is the original image, in the middle the aligned and colored corrected version and on the right the result of the coral segmentation with neural networks. ©2016 IEEE

easier to use.

Neural networks or machine learning, in general, can achieve great results in diverse scenarios but has always the need of a training phase with annotated ground truth data which limits its usability. It can help in the detection of specific objects in large databases and thereby relieve the human of tedious work (see [Pur+09]). However, it would be difficult to implement this as a general foreground detection and furthermore, the detection is usually not pixel accurate but rather a bounding box around the object.

CNNs and Feature Pooling

Li et al. tried to counter some of the weaknesses of neural networks by combining them with ideas from the semantic segmentation. In [Li+16] a special CNNs was trained on the *LifeCLEF 2014 Fish Task* dataset [1] which finds regions that contain fish with a high accuracy. From these regions SIFT features and local binary patterns are extracted and afterwards pooled, that means that all features of a region are combined by an operator to form a region descriptor. Examples of such an operator could be the average or maximum function (for details see [Car+12]). These regional descriptors allowed them to automatically learn fish features without adding features that described the background as these got overruled in the pooling process. Also, this combination allowed them to retrieve pixel-wise foreground-background segmentation results. The accuracy of this pixel-wise segmentation was not measured in the paper but the detection and classification accuracy on the *LifeCLEF* dataset was 82.7% and thereby higher than for many previous methods. An example of a segmentation result is depicted in Figure 3.9. Overall, the proposed combination could fix one of the shortcomings of neural networks by allowing the creation of pixel-wise segmentations and could also increase the accuracy slightly. However, the accuracy of these segmentations was not measured and there is still the need of hand-annotated data for a learning phase.

Active Contours

Active contours, proposed by Kass et al. [KWT88] and also known as snakes, is a concept of the computer vision field which optimizes an estimation of a shape (e.g. a bounding box) to the real

---

[1] `http://www.imageclef.org/2014/lifeclef/fish`

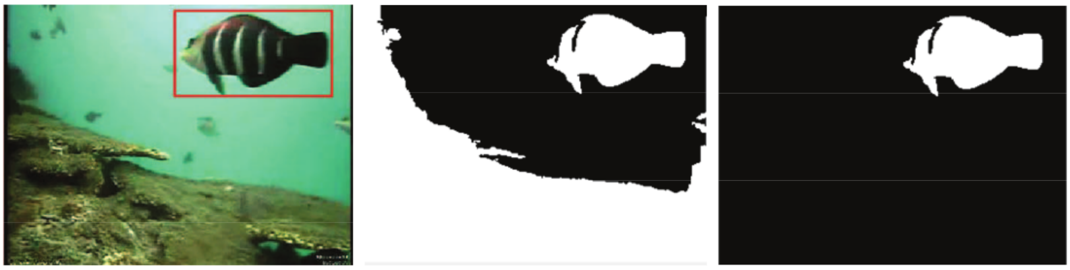


Figure 3.9: One example of the approach from [Li+16] on the *LifeCLEF* dataset. On the left is original image with the region found by the neural network, in the middle is the segmentation based on support vector regression and feature pooling and on the right side shows the combination of both. ©2016 IEEE

edges of the object. It consists of a spline $s(x)$ defined by a set of points $\bar{x}_i$ on which an energy function is defined. There exists a great variety of proposed energy functions, but usually, they consist of two terms:

$$E_{snake} = \int_0^1 E_{internal}(s(x)) + E_{image}(s(x))dx. \tag{3.8}$$

The integral goes over the whole length of the spline and evaluates different properties of it. The internal energy $E_{internal}$ evaluates the shape of the spline, e.g. it is usually desired that the spline is smooth as this corresponds with the smoothness of natural objects. This can be enforced by minimizing the second derivate of the spline and therefore $|\frac{d^2s}{dx^2}|^2$ is one common part of the internal energy function. The other part of the energy function, E_{image}, should adapt the spline optimally to the image. Usually, this means that the spline should lie on edges of the image and for these areas high gradient values are characteristic. Therefore, by adding the term $-|\nabla I(i,j)|^2$ to the energy function the spline will tend to stay on edges. There are many more possible terms for the energy function, but the exact configuration heavily depends on the application.

After defining the energy function the shape of the spline will be changed gradually so that the energy is minimal with the aim that the spline adapts to the contours of an object in the image. This concept was used for underwater images in [Zha+15a]. After applying an image restoration method based on a physical model on the frame the object was detected with Canny edge detection. The objects they want to detect are light sources which are visible during a docking process of an underwater vehicle. To get the exact shape and position of these light sources a snake is on the detected edges and then the shape is iteratively optimized. In general, active contours are a great tool for the refining of object shapes but rely heavily on good detection results and sharp pronounced edges between the object and the background. This proved to be successful in the scenario of [Zha+15a] in a pool but it seems doubtful if active contours can be a useful tool for underwater images in general because they are usually blurrier than in in-air images.

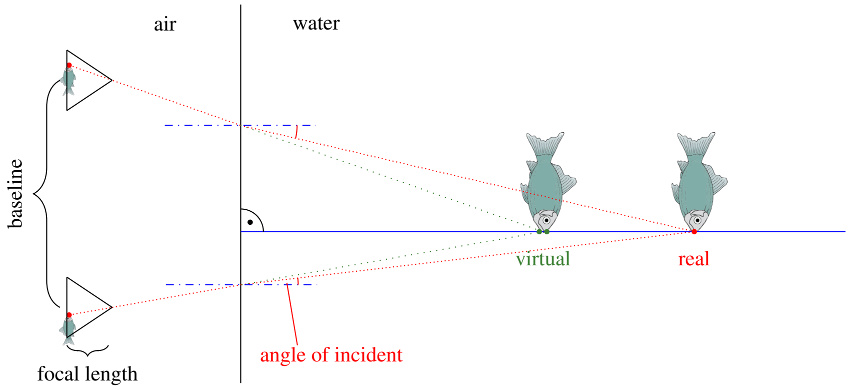


Figure 3.10: A simplified depiction of the effect of refraction on underwater stereo systems. Due to the different refraction indices of water and air the light ray (red) is bent at the transition of these mediums. If this effect is not considered the fish will seem to be closer to the camera system than it actually is (virtual position). The computation and cancellation of this effect is very difficult since it is nonlinear.

Depth Information via Stereo-Vision

Obtaining depth information in underwater scenarios is extremely difficult since water as a medium heavily disturbs all sensors. The Kinect has been used for this purpose in [Dan+14] but the infrared light it uses gets absorbed very quickly by the light. The same is true for a time of flight sensors which often use light of the infrared spectrum. In [Tsu+14] a maximal depth of one meter was achieved with such a sensor. Stereo systems do not suffer that strongly from absorption properties of water and can achieve higher ranges but have to deal with refraction which distorts the depth measurements in a nonlinear way and is therefore mathematically difficult to eliminate [DK14; DLK15]. The effect is illustrated in Figure 3.10.

Nonetheless, it was used in the work of Skinner and Roberson [SJ15] for the segmentation of archaeological sites underwater. They used two approaches in parallel, the first part uses clustering to create super pixels and then classifies them based on their structure and a learned dictionary. The second part segments the image based on the depth map they acquired with their stereo system. How exactly this was done is not explained in detail, e.g. if refraction was taken into account or just an in-air stereo algorithm applied. Nonetheless, instead of taking the absolute value the gradient of the depth map is computed and a threshold is set based on these values. This proved to be a good indicator since archaeological structures often have walls rectangular to the seafloor. With these two methods potential areas could be detected and from them, human-made structures were further filtered with a RANSAC algorithm which looked for the best straight lines.

A stereo system was installed in a big indoor tank ($10 \times 6 \times 5$ meters) in [Mar+13] to track an AUV in it. The detection of the AUV could be achieved with ordinary template matching or background subtraction since the water was clear and the background uniform. Afterwards, these detections were used for the estimation of the exact position of the AUV in the water tank via stereo matching. However, they used in-air methods for their camera calibration and therefore

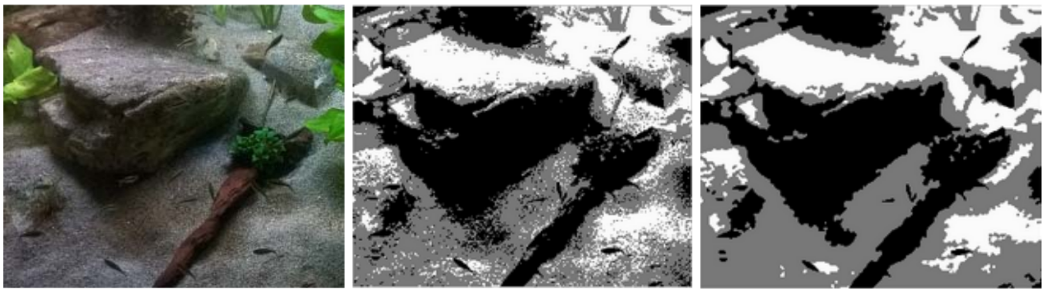


Figure 3.11: On the left is an underwater scene from [Wan+15] and in the middle the result of a K-Means clustering on that image. The segmentation is very fragmented and scattered which complicates a further usage and interpretation of that result. By using the MRF model on such clustering results the spatial coherency could be improved greatly, which can be seen on the right side. ©2015 IEEE

refraction was neglected which leads to measurement errors of up to 10%.

Underwater stereo systems are expensive (underwater housing, two cameras etc.) and difficult to set up (synchronization, calibration) but can provide useful depth information. With standard in-air algorithms, the absolute depth values are not very accurate but the relation between different values is still accurate enough for simple segmentation tasks like in [SJ15]. More complex and general segmentation scenarios would require the use of advanced algorithms on the depth maps (e.g. background subtraction) which usually promise better results when they are used directly on the color images. Overall, depth maps can be an additional source of information which is useful to validate and enhance the results on the color images. However, it is very expensive to create these maps from both the hardware and the computational perspective. It seems only advisable, especially in the underwater case, if the depth information is necessary later anyway, e.g. to determine the size of objects.

Markov Random Fields

Markov Random Field (MRF) models and how to optimize them with Belief Propagation are explained in detail in 4.2.1. They are a great universal tool but need prior information about the scene, either in form of a model that describes the foreground and background or a different algorithm that delivers a first result which is then refined. In [Wan+15] the MRF model has been used in conjunction with a clustering algorithm to segment objects lying on the seafloor in frames captured by an AUV. The clustering they used had no spatial component and hence the segmentations based only on these clusters were very scattered, especially in more complex scenes. The MRF can model spatial relationships very good and was therefore ideal to improve this. The model needs probabilities of each pixel to belong to a certain class/cluster as a starting configuration and these were computed by using the mean and variance of each cluster. The model was then optimized with a simulated annealing approach and afterwards, the mean and variance of the new clusters were computed again and the process was repeated until a steady state was achieved. They compared their method against other clustering algorithms and could achieve segmentations that were spatially more coherent due to the neighborhood relationships of

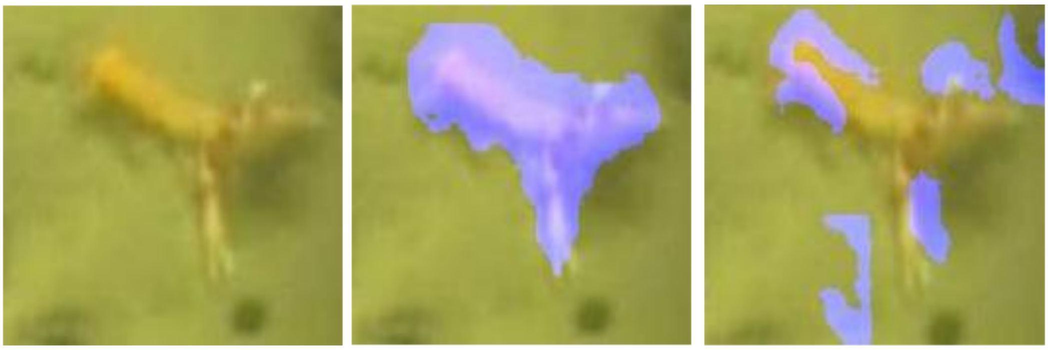


Figure 3.12: Lobster segmentation of [Soo+14]. On the left side is the original frame, in the middle the result of the proposed approach and the right side shows the result of the approach from [Lau+12] for comparison. ©2014 IEEE

the pixels in the MRF model, this result can be seen in Figure 3.11.

It is difficult to use an MRF model alone for the segmentation of images since it needs prior information. However, it is a great model of the spatial relations in natural images and can, therefore, improve the accuracy of many segmentation methods by using their results as input, similar to the approach in [Wan+15]. The optimization of these models is very demanding which makes it impossible to use them for real-time applications.

Bayesian Framework

A Bayesian framework is a statistical approach resembling the MRF method, however, a Bayesian network is acyclic and directed in contrast to an MRF and therefore the optimization (Maximum A Posteriori estimation) is easier. In [Soo+14] lobster should be detected in large mosaic images of the seafloor. Like the MRF model the Bayesian framework needs prior information about the scene and therefore they learn the characteristic brightness, hue and saturation of the lobster and of the background and from these values infer a probability for each pixel of being lobster or background. To ensure spatial smoothness a Gibbs energy function is used over a 3×3 neighborhood. An example of the result is depicted in Figure 3.12. Overall, the approach is comparable to the MRF model discussed above; the optimization can be done faster but therefore the spatial relationships are not modeled as accurately.

In general, statistical inference models like these can very successfully fuse the input data from different sources, either spatially like in the MRF example or from the different models like in the Bayesian framework. However, without a good model and input data it is not possible to create accurate segmentations and often the creation of them is the most difficult part. Furthermore, the inference is mathematically complex and computationally expensive.

Evaluation of Optical Imaging Approaches

Optical imaging can provide higher resolutions than sonar imaging and therefore the algorithms in general reach more accurate results than their sonar counterparts. Also, single images can be stored and processed in time much easier than whole videos (for which algorithms will be discussed hereafter). On the downside, the range of optical imaging is very limited underwater and single images miss the important time domain so that the detection of objects becomes much harder and behavior analysis almost impossible.

There are algorithms like Clustering (K-Means, C-Means), Thresholding or Edge Detection that segment underwater images without any special prior knowledge (e.g. training of specific object feature). However, on single images it is very hard to find a universal distinguishing feature between ’interesting’ objects and the background. Therefore, assumptions that only hold for a few special cases are common in this area, for example, that the foreground is always brighter than the background. Overall, these algorithms are very fast and often easy to implement and use but lack heavily in generality since the necessary assumptions usually do not apply for different images.

Algorithms that require a prior training phase (like Neural Networks) are more distinctive and can detect almost every object with acceptable precision when provided with enough training data. However, the need of a training phase makes them insufficient in generality and ease of use since for every new object training data has to be created (often by hand) and incorporated into the existing model.

Depth information is hard gather in the underwater context due to the harsh conditions and simple segmentation algorithms applied on the depth data are usually not sufficient to get precise detections. The discussed statistical approaches (MRF or the Bayesian Framework) are often computational demanding but are also accurate spatial models which often increase the segmentation accuracy greatly. However, they can only be supplementary to an already existing segmentation algorithm since they are dependent on its segmentation data.

Overall, no single image algorithm can solve the underwater segmentation problem satisfactorily. The most promising ones are learning based algorithms like Convolutional Neural Networks which can achieve high accuracies but their need for training makes them difficult and cumbersome to use even for experts. For example, for every different fish species the creation of a large amount of training data and a computational intensive training phase is necessary. Especially Neural Networks are improving rapidly at the moment and might be a viable option in a few years, however, at the moment the segmentation of videos is the better option for the segmentation of fishes underwater since it allows the precise extraction of more general features like movement and the evaluation of the detection over time which can be used to analyze the behavior of different fish.

3.3 Video Segmentation

When a whole video is available for the segmentation process a completely new source of information is available, the temporal change of a pixel. With this, new approaches are possible, e.g. the computation of an optical flow, and these sources of information should be used for the segmentation although their computation is often very demanding. Only methods that use this additional temporal information are considered as video segmentation methods in this work, even though single image approaches are sometimes used as well on whole videos, see [Lab+12] for example where thresholding was applied on every single frame of the video but the temporal information were not used at all. However, more common are combinations of single image and video segmentation methods, e.g. thresholding after background subtraction or Clustering and graph cut approaches to align the segmentation derived from the temporal cues to the edges of the current frame of the video [Zha+15b]. Another possibility is slightly adapted image segmentation approaches that also use the temporal information of the video, e.g. Graph Cuts in a 3D image volume [He+15].

Since underwater video segmentation is a niche topic and only very few algorithms have been created especially for this cause, this section will begin with a general overview of the field of video segmentation. The advantage of the in-air scenario is also that there exist several common datasets for these cases which enables a good comparison of different approaches. In the second part of this section some algorithms are presented that were designed for and used on underwater videos.

3.3.1 General Video Segmentation

For the segmentation of videos two cases should be distinguished generally, a freely moving camera and a static camera. The segmentation of a video made by a moving camera is much more complex than that of a static camera as many algorithms exploit the non-moving parts of a video (e.g. Background Subtraction) or work more reliable in a static setting (e.g. optical flow). Approaches to solve the inherent difficulties of a moving camera are shown in [ZYD14; ZZY15] where the motion of the camera is estimated from the video based on the assumption that most of the scene is static background. Then, either the camera motion is subtracted from the frames which then makes it possible to use slightly modified versions of the segmentation algorithms for static camera scenarios or the segmentation can be based directly on the camera motion and all objects which have a deviating direction of movement will be marked as foreground.

Nonetheless, a freely moving camera is a great challenge, since, for example, new parts of the background become visible all the time so that the background model can usually only be based on the last few frames. Also, when the background consists of many objects at different distances to the camera the perceived motion induced by the camera movement is not uniform but extremely diverse. If the motion vector of an object and the perceived motion of the whole scene have the same direction, it becomes almost impossible to decide whether the object is close to the camera and stationary or far away from the camera and moving. In Figure 3.13 some examples are shown where these problems were addressed in [ZYD14] and [ZZY15]. On the image on the left side, the perceived motion of the parking cars is a lot stronger than that of the trees and other background parts because they are farther away from the camera. This substantially complicates the detection of the cars that are actually moving. The only advantage is that the camera moves quite slowly and, therefore, the perceived motion of stationary objects is, in general, an order of magnitude lower than that of the moving cars. The images on the right side show a different approach where the background was reconstructed and then a background subtraction method applied. The scenarios shown there are easier overall because the distance between the background objects and the camera is greater, this makes the whole scene less volatile over time.

The other case, the segregation into foreground and background of videos taken by a static camera, will be discussed in the following and can be approached in a very different way. Since the camera is static it can be expected that the background of the scene will remain almost

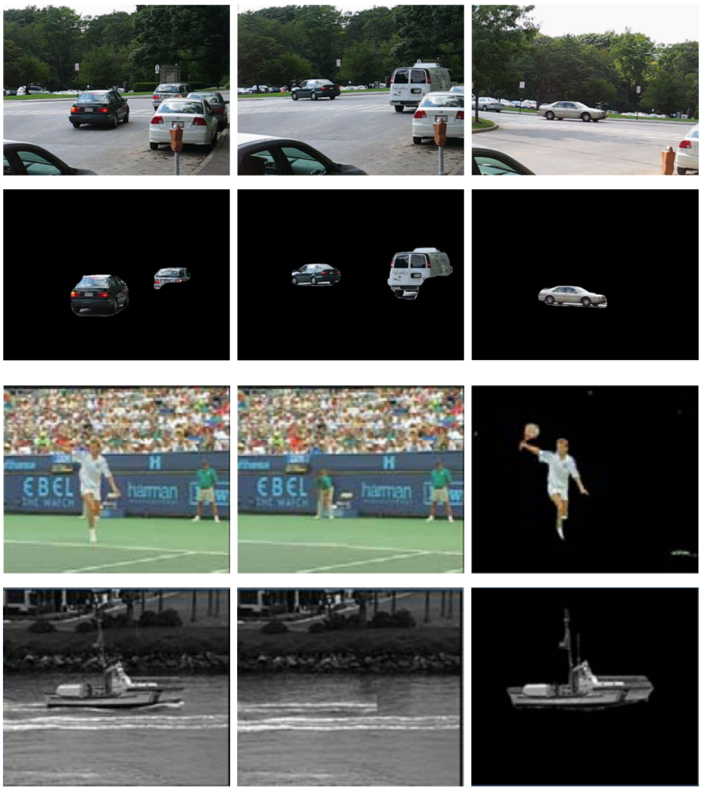


Figure 3.13: Segmentation of a video made with a moving camera. The top row shows the original frame and below are the detections of [ZYD14] depicted. The scene is difficult since various stationary objects at different distances to the camera exist. In the bottom rows is the approach of [ZZY15], it shows from left to right the original frame, the background model and the segmentation. ©2015 IEEE

constant for the duration of the video. Therefore, a model of the background can be created over time which only contains the static background parts but excludes all moving objects as they can only be seen for a relatively short duration. A comparison between this background model and the current frame of the video then allows a detection of all moving objects. In Figure 3.14 an example of this can be seen.

Background Modeling and Subtraction

Background modeling and subtraction is used for a long time already in computer vision. The first approaches date back to the beginning of the 90ths [RMK95; SS93] and commercial applications followed soon. An example is the Patent [GM99], where background subtraction is used for video compression. The background model is used to detect changing areas (areas of interest) in the current scene and then only these areas will be updated in the video stream. This could reduce the bandwidth requirements dramatically e.g. for real-time video calls.

Common problems with this approach are image noise and shaking cameras. Random variations of brightness or color information in images are called image noise. These random variations can cause false detections during the Background Subtraction approach which can be seen in Figure 3.14. A way to reduce image noise is an improvement of the lighting conditions in the scene or the usage of a camera with a better sensor. Shaking cameras are also a problem since then it seems for the background subtraction algorithm that the whole scene is under movement when in reality only the camera is slightly moving. Therefore, a good fixation of the camera is a necessity for Background Subtraction algorithms.

Single Gaussian

The most widespread way to model the background of a scene is by assuming that each pixel value changes according to the probability function of a normal distribution (also called Gaussian). The idea is that noise of any kind which disturbs the imaging system has a Gaussian distribution and therefore it can be assumed that a pixel value in a static scene will also show a Gaussian distribution over time. However, if a real change in the scene occurs, which is not induced only by the noise of the imaging system, then the pixel value will differ significantly from that previous distribution. A simple way to use this for the detection of moving objects is the Single Gaussian method which was used e.g. by Wren et al. [Wre+97] (short: *Single Gaussian*) in their real-time tracking system. They used a simple updating scheme

$$\mu_t = \alpha \cdot V(i, j, c, t) + (1 - \alpha)\mu_{t-1} \tag{3.9}$$

where μ_t is the mean value of the Gaussian distribution at position (i, j) and time t for channel the channel c. The update rate α defines how fast the mean adapts to changes in the scene. A similar scheme can be applied for the variance and with these parameters (mean and variance) the Gaussian distribution is fully defined. This simple model made real-time applications possible even 20 years ago but showed problems with more difficult scenarios, e.g. changes in the background of the scene (e.g. by a swaying tree) or the constant presence of foreground objects which could impair the accuracy of the background model.

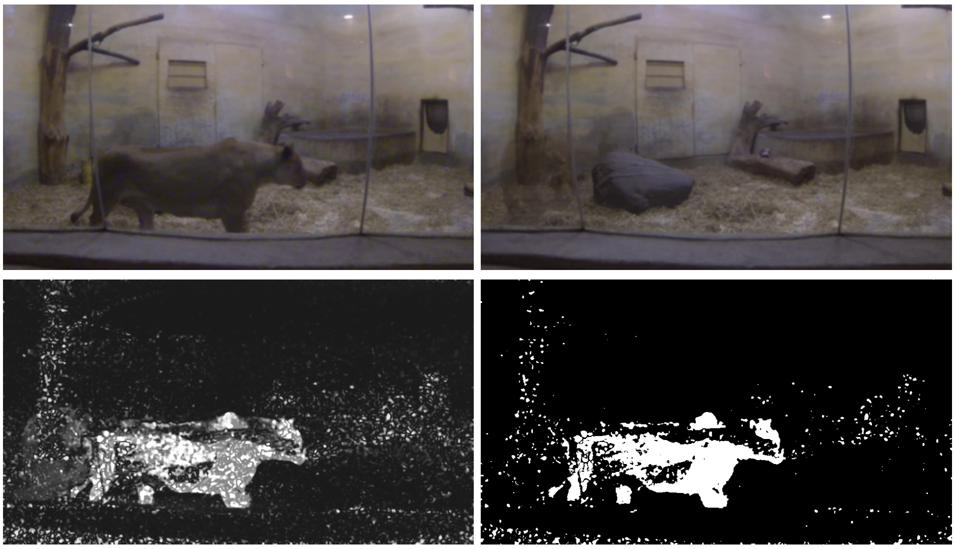


Figure 3.14: Background Subtraction on an example video of a lion from the zoo of Rostock. The left top image shows the current frame of the video and the right top image depicts the background model that was created from the past frames of the video. A map of the differences between the model and the current frame can be seen on the bottom left and applying a threshold on this creates a binary segmentation of foreground and background.

Mixture of Gaussians

To account for these problems an adaption of this algorithm was proposed by Stauffer et al. [SG99] (short: *Adaptive MoG*) which used several Gaussians for each pixel (and each channel) instead of only one Gaussian per value. This additional complexity made the modeling slower and increased the memory demand, but also enabled the process to accurately model many difficult situations which cannot be handled by the Single Gaussian approach. The key idea is that there is sometimes more than one exact background color which should be learned by the background model for each pixel. Examples could be the swaying of tree in the wind – where the color of a specific pixel could alternate between the green of the leaves, the brown of the branches and the blue of the sky – or an outdoor scene where the clouds sometimes block the direct sunlight and sometimes do not block it, so that each pixel alternates between bright and dark.

By using several Gaussians there are basically several models for each pixel and each new value (from a new frame of the video) is put into only one of the Gaussians/models, namely the one that it fits the best. In the swaying tree scenario, ideally, there would be one Gaussian for the green color of the leaves, one for the brown color of the branches and so on. In this way, for every new pixel value a fitting Gaussian will be found or otherwise, a new Gaussian will be created as it will be assumed a new object appeared. To now create a segmentation a weight must be assigned to each of these Gaussians. The weight indicates how often a value could be matched to a specific Gaussian in the past and as background values are assumed to appear often and regularly, only Gaussians with a high weight can represent the true background. A match

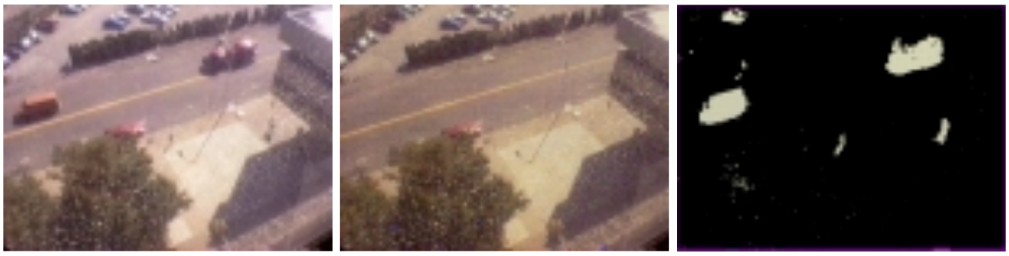


Figure 3.15: An example of the *Adaptive MoG* background subtraction of [SG99]. On the left is the actual frame, in the middle an illustration of the background model and on the right the segmentation. Although it is a difficult outside scene with shadows, changing lighting conditions and a tree close to the camera the algorithm could still model the background quite accurately. ©1999 IEEE

with a low weight Gaussian means that the value could not be matched to any background value and, therefore, belongs to a foreground object. In Figure 3.15 the accuracy of the MoG method on a difficult scene is shown.

Background Subtraction methods, in general, have difficulties with shadows and too many parameters that have to be adjusted. In [Ziv04] (Short: *MoG 1*) a standard MoG model is used but important parameters of this model - for example the update rate or the number of Gaussians - are determined automatically by using the Maximum Likelihood estimation. This makes the method more robust to different situations and easier to use for non-experts. In [WS07] (Short: *MoG + PSO*) the problem of tuning the (often many) parameters of a MoG method was approached differently. Particle swarm optimization - a method to find extrema in higher dimensional functions - was used to optimize parameters like the update rate α or the sensitivity β. The problem with this is, that ground-truth data is necessary to evaluate the segmentation results and adapt automatically the parameters. Therefore, White und Shah require a ground-truth segmentation every 100 to 200 frames of the video to recalibrate the parameters. In realistic application this is hardly achievable and, furthermore, many background subtraction approaches without this requirement could achieve better results.

The problem of the unwanted detection of shadows is addressed in [KB02] (Short: *MoG 2*) where a special condition for the RGB color space was developed to classify detected foreground pixels into real objects and shadows. The condition checks whether there was only a change in the intensity of the pixel or also a color change, the first would be an indication of a shadow that just darkened the pixel. Also, a new updating scheme is developed that improves the important initialization of the model so that precise segmentations can be achieved with a shorter training phase. The especially developed IHLS color space was used in [Set+06] (Short: *MoG + IHLS*) to detect shadows. It splits the information of each pixel into hue, luminance and saturation. From this, a chrominance vector is generated based on the saturation-weighted hue values. A shadow should now only affect the luminance value of a pixel but not the chrominance vector and can, thereby, be easily distinguished from real objects. An example of the accuracy of the shadow detection can be seen in Figure 3.16.

The Mixture of Gaussian (MoG) is still often used today because of its high accuracy even in difficult situations and its real-time capability. However, it has one disadvantage which is present

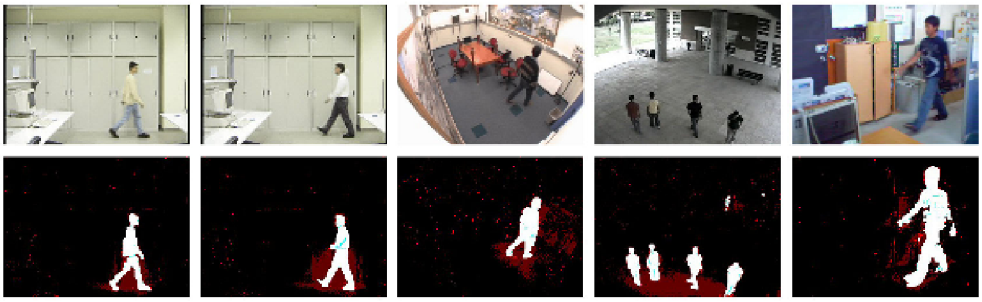


Figure 3.16: The top row shows the original frames and in at the bottom the segmentation results of *MoG + IHLS* can be seen. Shadows that have been detected thanks to the IHLS color space are shown in red.

in all background modeling and subtraction methods. They all model each pixel separately and thereby forgo all spatial aspects of the scene which are often relevant for segmentation tasks. Images of the real world are smooth and objects in them are usually quite big (several hundredths of pixels or more) and this knowledge should be used for the segmentation process. An example would be an area where all pixels were classified as background by the background subtraction except one pixel that was marked as foreground. If now the neighborhood as a whole is observed, it becomes obvious that this one foreground pixel is not an object that should be detected but almost certainly some noise and therefore should be marked as background. To incorporate these spatial aspects of the segmentation the MoG is nowadays often combined with a spatial method.

MoG + Markov Random Fields

One common method to model the smoothness of natural images is the MRF (see 3.2 and 4.2.1 for a detailed explanation) which was used e.g. by Schindler et al. [SW06] (short: *Enhanced MoG* and *Enhanced MoG + MRF*). As discussed earlier, the MRF needs input data which is then enhanced by using the neighborhood information of the pixel. In the case of image or video segmentation this is usually a probability map (not a binary segmentation), and therefore the background subtraction output has to be adapted to this. Schindler et al. computed the foreground probability $P(\bar{v})$ of the pixel at location $\bar{v} = (i, j)$ in the following way

$$P(\bar{v}) = \sum_{l=1}^{K} \frac{w_l(\bar{v})}{\sqrt{(2\pi)^n |S_l(\bar{v})|}} \mathrm{e}^{-\frac{1}{2}(I(\bar{v})-m_l(\bar{v}))^T S_l(\bar{v})^{-1}(I(\bar{v})-m_l(\bar{v}))}. \tag{3.10}$$

Here, K is the maximal number of Gaussians per pixel in the MoG model, $w_l(\bar{v})$ is the weight of the l-th Gaussian of pixel $\bar{v}$, $S_l(\bar{v})$ is the corresponding covariance matrix, $I(\bar{v})$ is the vector with the pixel values of $\bar{v}$ and $m_l(\bar{v})$ is the vector with the mean values of the l-th Gaussian.

Exemplarily the effect of the MRF from [SW06] is depicted in Figure 3.17. Small false detections due to noise could be removed and holes or openings in the objects were mostly closed without impairing the shape. Overall, this combination can provide very promising results and [SW06] were able to create excellent results on the Wallflower dataset (the best so far). However, the MRF model heavily depends on the results of the background subtraction and is not able

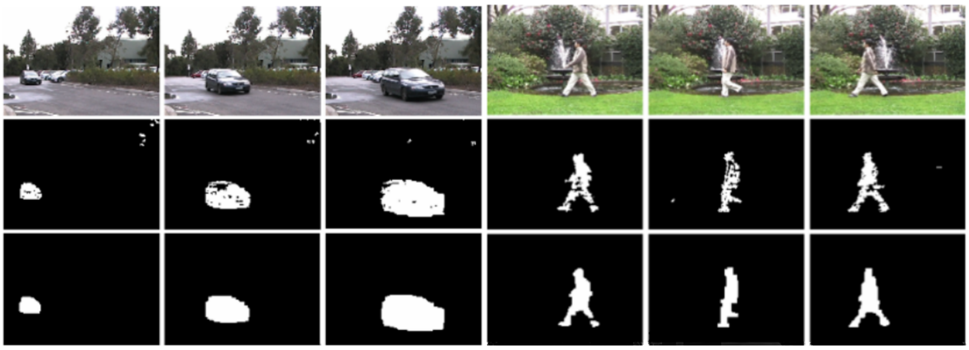


Figure 3.17: Effect of an MRF model on the background subtraction segmentation. Images taken from [SW06].

to correct larger errors which other approaches can achieve to some degree, e.g. graph cuts by adapting the segmentation to the edges in the frame. Moreover, the quality of the results has a drawback in the computational complexity of the MRF model which prohibits most real-time applications.

Non-Gaussian Models

Apart from the different Gaussian models for modeling the background in videos there also exist some non-Gaussian models which will be discussed in the following. The basic idea of these methods is the same but the pixel values are not modeled anymore through a probability distribution but by other means, e.g. by saving a large number of past pixel values and deducing a prediction for the current frame from them.

Wiener Filter

Toyama et al. [Toy+99] defined the most prominent background modeling problems and created a segmentation algorithm especially designed for these difficulties (short: *Scale-Based Wiener Filter*). It consists of three different levels, with the lowest level (pixel level) being the standard pixel-wise modeling of the background. Instead of the common Gaussian model they use however a Wiener filter [Wie64] with the 50 most recent values to derive a prediction for the background value of each pixel.

The second level is a regional approach which tries to segment whole objects based on the pixel-wise segmentation of the first step. The idea is to get at least one seed region of each foreground object and then grow them until they reach the edges of the whole object. The assumption here is that the foreground objects are homogeneous in their color, which is mostly true for this dataset but in many other cases not. The third and last level is frame wide and tries to deal with changes of the whole scene, the example here is an indoor scene in which the light is turned on or off. They argue that these events must be treated globally and use several background models for these situations. The first step in creating these model is a learning phase where the frames are clustered via k-means [HW79] to train k different background models. Afterwards, the segmentation algorithm detects these global changes – if more than 70 percent of all pixels are marked as foreground – and switches to the most appropriate background model.

The method showed good results on the dataset provided by them and contains some interesting ideas but is too much engineered for this specific dataset. For example, for the k-means clustering the number of different global states has to be known beforehand and it should be ensured that each of these states occurs during the training phase, otherwise the clustering would create meaningless results and may even interfere with the normal background subtraction.

Subspace Learning

Subspace learning methods do not look at the videos pixel-wise but take the whole scene at once into account. An often-used approach is the Principal Component Analysis (PCA) in which the last k frames of the video are taken as data input. This data is then reduced drastically in dimension and because of this reduction only large and prominent parts of the image will still be part of the lower-dimensional data. The idea to use this for video segmentation comes from the fact that moving objects are in a different location in every of these k frames and, therefore, they are no consistent and prominent part of the scene. Hence, only the background part of the video should be modeled by the lower-dimensional data. In [ORP00] (Short: *Eigenspace Model*) a background model was created with PCA and then a simple thresholding was used to create a binary segmentation. They used the results for tracking and behavioral analysis but overall the accuracy is lackluster in comparison to other approaches.

The PCA approach of reconstructing the background was combined with a Linear Discriminant Analysis (LDA) in [MTR12] (short: *PCA + LDA*). LDA works similar, however, it does not model the background of the scene but rather tries to detect changing parts of the image directly in the lower-dimensional data. Each of these methods has very different advantages and disadvantages and by combining them Marghes et al. could achieve competitive results to most Background Modeling approaches.

Instead of LDA the Maximum Margin Criterion (MMC) was used in [FMB12](short: *IMMC*). The computation of the MMC is easier and faster in comparison to the LDA since no inverse operations are necessary. Additionally, they developed it further into an incremental approach which makes a batch computation with the last k frames unnecessary for updating the background model with the current frame. They tested it thoroughly and showed e.g. the great improvements that can be obtained by enlarging the history that is used to create the model (increasing k). Nonetheless, overall the PCA + MMC approach performed slightly worse than PCA + LDA on the background subtraction task and the execution time is still poor with only 7 frames per second on a 240×360 video.

Another subspace learning approach is the Independent Component Analysis (ICA) in which an image (or in general any signal) is seen as a mix of its independent components (source signals). In [TL09] (short: *ICA*) this property is used in a method that resembles the background subtraction approach. The independent components of two images are computed at the same time with ICA and then the independence of these signals against the signals in the other image is measured. Each signal should correspond to one object in the image and, therefore, if a signal from the first frame is independent to all signals from the second frame - the corresponding object has only appeared in frame one. To segment a video, an image which only contains background is compared to the current frame of the video in this way and all objects that are not present in the background image are marked as foreground. The problem with this approach is the need of a

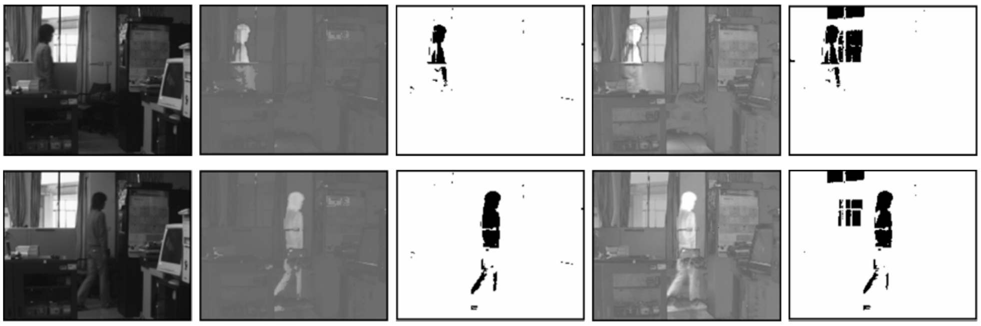


Figure 3.18: Results from the method proposed in [TL09]. The first image in each row shows the original frame. The second image shows the result of comparing the original frame with the background model via ICA; a brighter pixel means a greater difference to the background was detected. The third image is a binarization of the second one. The fourth and fifth image show the results of a different method based on ICA that was used for comparison. ©2009 IEEE

background image which is often hard to obtain and also does not adjust itself to slight changes in the scene like the background modeling methods discussed earlier. An example of the results on a grayscale video can be seen in Figure 3.18.

In [BG09] (short: *INMF*) a non-negative matrix factorization was used on videos to lower the dimension of the data. The non-negativity restrain makes the data more meaningful (e.g. an object cannot have a negative presence in a scene) but also even more complex to compute. To address this problem, they proposed an incremental updating approach that can update the matrix factorization for each new incoming frame of the video with a much lower complexity than the standard batch calculation approach. Nonetheless, the results on change detection datasets were poor and could not justify the additional complexity. Lastly, in [Hu+11] (short: *Tensor Subspace Learning*) a tensor is used with an incremental singular value decomposition to obtain the most significant background components of the scene. Although they also use an incremental updating approach to lower the computational load, the approach is still much slower than background modeling and subtraction methods.

The biggest drawbacks of all these methods, in general, is their computational complexity since the decomposition of large matrices is not an easy task, even for today's computers and with the adaptations made to some of the algorithms. In return, the algorithms promise a spatially precise model of the background since the scene is modeled as a whole and not each pixel separately. Nonetheless, the results in binary video segmentation are not more accurate than background subtraction or optical flow approaches.

Sample Consensus Methods

These methods model the background by keeping a set of samples of each pixel instead of modeling the color directly. One of the earlier examples for these algorithms is provided in [ZH06] (short: *KNN*) where pixel values of previous frames are stored in different memories - one for the short term and one for the long term. In a variable sized kernel region a search for matches to the current pixel value is performed and if enough matches can be found the current pixel will be classified as background. In [EHD00] (short: *Non-parametric Model*) a method resembling this approach is presented. They also use one short-term and one long-term model, each updated differently. The comparison of the current pixel is done with a kernel function that evaluates and thresholds the distance to all samples in the model at once. Additionally, they are suppressing false detection by considering the whole neighborhood of a detection to evaluate if the area just moved a few pixels in one direction - e.g. a tree branch moving in the wind.

The *ViBe* algorithm [BD11] is another example for this class, it updates the saved old samples for each pixel randomly so that even old values can have an influence on the current segmentation (although with a low probability). Furthermore, the updating process diverges spatially so that an update of one sample can influence the neighboring samples which makes the model spatially coherent to some degree. The segmentation itself is done by counting the number of values that agree with the current value (are closer to the value than a threshold) and if that number is large enough the pixel is assumed to be background. The approach by St-Carles et al. [SBB15] is similar to that, however it does not store the pixel values directly but rather an LBSP-feature (Local Binary Similarity Patterns) vector that describes the pixel and its neighborhood. Furthermore, they used a sophisticated scheme to adapt their parameters to the current situation based on a regional classification.

The sample consensus methods can achieve results similar to MoG background modeling approaches and are also real-time capable. They tend to need more memory to save the past pixel values or feature vectors, however, this depends heavily on the parameters of the algorithm and is usually not a problem with today's hardware. They not only have very similar advantages to the MoG methods but also the same problems because they model each pixel separately. By using local features instead of pixel values (see [SBB15]) this can be countered partly but it also increases the computational complexity.

Fusion of Segmentation Methods

This is a completely different approach which does not try to model the background by itself but relies on other algorithms and tries to combine their results so that an improved segmentation can be created. Examples of this are [Mig10], which used a Bayesian Model and a Rand Estimator to fuse different segmentations or [WZW04] which used MRFs to fuse segmentations of medical images. A quite current approach is [BCS15] which used the large database of different segmentations of the *changedetection.net* dataset and fused the best performing of them. The fusion process itself was not done by a Bayesian Model like in the other cases but with a genetic algorithm. The genetic algorithm had the segmentations and a set of functions it could apply to them and tried to find the best combination. These functions were e.g. morphological erosion or dilation, logical *AND* or *OR* operations, or a majority vote. In this way, they could improve the already very good results of the top algorithms. However, to run their genetic algorithm ground truth data is necessary and therefore they used one video of each category and the corresponding ground truth data to find the best combination of segmentations and functions. This is not the original idea of the dataset and therefore their results are not comparable with other algorithms who have not partly used the ground truth data to train their algorithms.

Overall, these algorithms can partly improve already accurate segmentation results but rely on several good segmentation algorithms which have to provide the input data. This makes them impractical for real usage as the calibration and computation of these algorithms is very time consuming and is not justified by the usually small gain in accuracy over the input methods.

Optical Flow

A video segmentation method which is not based on the creation of a background model and was already discussed in conjunction with other approaches is the optical flow. A video is here viewed as a 3-dimensional image volume V in which each frame is one layer. For the location $\bar{v} = (i, j, t)$ in such an image volume the optical flow $f(\bar{v}) = (f_x, f_y, f_t)$ is usually computed with the formula

$$\frac{\partial V(\bar{v})}{\partial x} f_x + \frac{\partial V(\bar{v})}{\partial y} f_y + \frac{\partial V(\bar{v})}{\partial t} f_t = \nabla V(\bar{v})^T f(\bar{p}) = 0. \tag{3.11}$$

This condition can usually be fulfilled only approximately and will be checked over a small 3D image patch $\Lambda(\bar{v})$ to make the method more robust, e.g. against aperture. Using a least-squares error measure this leads to the following minimization problem

$$Err(\bar{v}) = \int_{\bar{n}\in\Lambda(\bar{v})} (\nabla V(\bar{n})^T f(\bar{v}))^2 W(\bar{v},\bar{n}) d\bar{n} + \lambda(1 - f(\bar{v})^T f(\bar{v})), \tag{3.12}$$

where $W(\bar{v},\bar{n})$ is weighting function. The last term enforces a normalization of the optical flow as otherwise $f(\bar{v}) = 0$ would always be the best solution. This is then usually reformulated as an eigenvalue problem and the optimal solution is estimated. However, the computation of the optical flow by solving an eigenvalue problem is very costly and therefore many adaptions and different formulations have been made that try to ease the computational burden without losing too much accuracy [Wan+14a; Sey+16a]. To derive segmentations the resulting optical flows can be thresholded so that every pixel that moves/changes gets marked as foreground. If the camera is not static this can still be applied by computing and subtracting the camera motion although this gets very difficult for complex scenes.

To make the computation of the optical flow faster Wulff and Black proposed in [WB15] a method based on sparse feature matching which was enhanced to a dense optical flow (DOF) by trained data. First, features were found and matched in two consecutive frames to create a sparse optical flow. Afterwards, they used a trained basis B to estimate the DOF from that. The basis was trained with four movies for which the DOF was computed with a standard method. Ground truth data would have been better for the training but is not available to this extent. The basis B could then be used to reconstruct the DOF so that it best matches the movement of the found features. The biggest problems they had were outliers, which are unavoidable to a certain extent. To deal with this the robust Cauchy function

$$\rho(x) = \frac{\theta^2}{2} \log(1 + \frac{x^2}{\theta^2}) \tag{3.13}$$

was applied to reduce the sensitivity to errors. If a segmentation of the frames is required (in addition to the DOF) the found features are clustered into M classes and for each of them the DOF is calculated separately as each class is expected to move independently. Then each pixel is assigned to one of the layers/classes based on an energy function which considers smoothness, color and the position of features.

The algorithm was evaluated on the MPI Sintel and KITTI datasets and gave excellent results in comparison to the computation time is needed. Nevertheless, although the approach is already designed for speed it still takes about 3 seconds for the segmentation of an image and has the need for prior training. For underwater scenes, the detection and matching of features are also difficult in general because of all the degradations on the videos (blur, color attenuation etc.).

Flux Tensor

The Flux Tensor is a different way to compute the optical flow in a video, however, it uses a slightly different formulation for the task so that only the movement speed is calculated and not the direction of movement. Because of this, it is faster than normal optical flow methods and can be computed in real time. In [Bun+07] the approached was used on infrared data and combined with geodesic active contours which were used on the visible light spectrum of the image. A meaningful comparison is hardly possible since a dataset with videos in the visible and infrared spectrum is necessary. Nonetheless, the Flux Tensor alone seems to perform quite bad since it tends to only segment the edges of the objects but can be a useful intermediate step in a combination of algorithms since it can be computed quite fast for an optical flow. Of course, it can also be used on normal optical videos and details about the computation of the Flux Tensor can be found in Section 7.2.1.

Optical Flow + CNNs

A combination of CNNs and an Optical Flow was used in [HR16] to gather information in scenarios with a moving camera. To create the optical flow each frame was first over-segmented into superpixels and then for each superpixel a fundamental matrix was estimated which gives the relation from the current frame to the last frames. From this matrix, the optical flow can be easily derived. For the estimation of the fundamental matrix the semantic information was essential. If the superpixel was classified as a static object (e.g. street or building) there could still be perceived motion due to the camera movement but only motion that obeys to the epipolar constraints, which limits the possible movement vectors drastically. For an object that moves through the scene (humans, cars etc.) this constraint cannot be applied. The semantic information was extracted with the CNN from [LSD15].

These constraints were combined with other auxiliary conditions about the occlusion and connectivity between the superpixels and lead to a complicated optimization problem. The optimization was done with a PatchMatch Belief Propagation [Bes+12] which is similar to the normal Belief Propagation but can work in the continuous domain instead of a set of discrete labels, which is necessary for the optimization of the fundamental matrix. An example of the results is depicted in Figure 3.19.

The whole process is complicated and computational expensive but necessary as they had difficult scenarios with a non-static camera. The inclusion of the CNN allowed the approach to better handle the ego-motion of the camera as it made it possible to detect static areas. Although this greatly improves the quality of the optical flows as well as the detection accuracy of moving

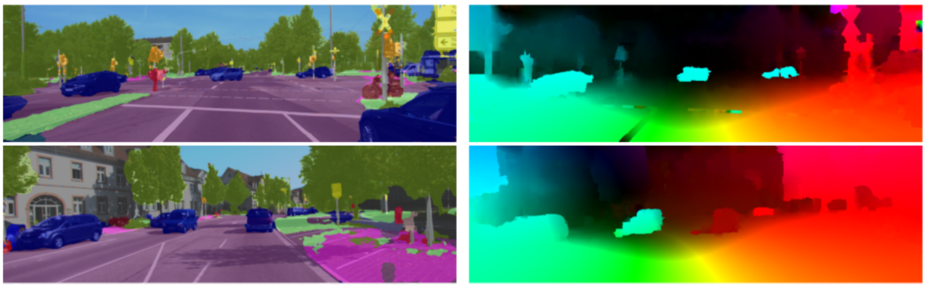


Figure 3.19: An example of the approach from [HR16]. The left side shows the original frame with the semantic information overlaid and the right side depicts the optical flow.

objects, it also limits the generality of the approach since now training data is necessary and the trained CNN will also only work for one scenario (e.g. car driving in a city) but not for others (e.g. human inside a building or bike driving through a wood).

Optical Flow + Stereo-Vision

Geiger et al. combine the idea behind the optical flow with a stereo setup in [GZS11] for fast 3D reconstruction. They detect several features in each image, match them in consecutive frames and then find loops in these matches (current left frame $\leftrightarrow$ previous left frame $\leftrightarrow$ previous right frame $\leftrightarrow$ current right frame $\leftrightarrow$ current left frame). Such a loop allows for higher confidence in the features matches and the stereo matches also allow for a depth estimation. The detected points are then structured and clustered by using a Delaunay triangulation. In [Len+11] this approach was applied to detect moving objects in urban environments and predict their movement. Together with the depth information this should be used for collision avoidance in driver assistance systems.

Overall their results were convincing and they could clearly enhance the optical flow data by combining it with stereo matches. Nonetheless, problems still exist with the tracking of objects over longer periods due to false detection or occlusions of objects in crowded scenes. Furthermore, the need for a stereo camera system makes the whole approach expensive and complicated, especially in the underwater case where refraction must be taken into account.

Optical Flow + Feature Tracking

A different way to obtain feature matches with a high confidence was proposed by Ochs et al. in [OMB14]. They only had a single camera system and, therefore, instead of using the stereo matches to verify their optical flow they utilized the time dimension extensively. They used the approach from [BM11] to compute a sparse optical flow between two frames but then combined these matches over many frames to obtain feature trajectories. If these trajectories go over hundreds of frames the feature becomes very reliable. A distance measure between these trajectories is then defined and a spectral clustering applied so that similar trajectories are grouped together as they most likely belong to the same object. In the last step, the sparse features are used to get a classical pixel-wise classification/segmentation of the frame. This is done by optimizing a Potts model which minimize the length of the borders as well as the number of

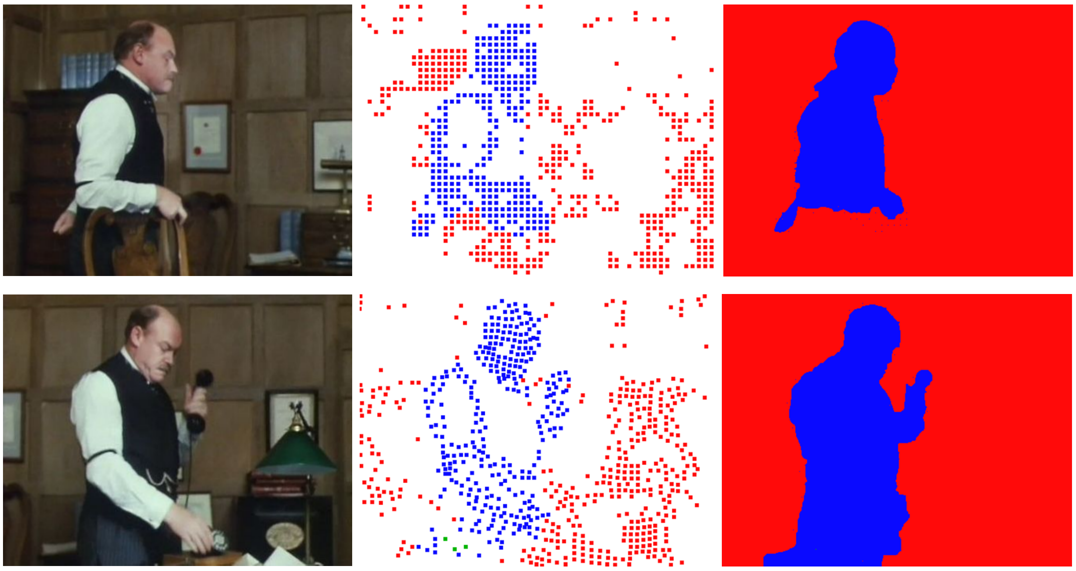


Figure 3.20: One example from [OMB14] of the segmentation of a video. On the left are the original frame, in the middle are the features which were found in these frames and on the right side is the dense segmentation. ©2014 IEEE

false classifications inside of regions. The code of the algorithm is freely available[2] and some examples are depicted in Figure 3.20.

The advantage of this method is that it can detect and separate any moving objects without a training phase even if the camera is moving. However, the computation time is nothing close to real-time and it is always necessary to collect a batch of frames first to compute the trajectories on them, so it can never be used as an online algorithm. When used on underwater images the feature detection was often a problem as not enough features could be found to create meaningful segmentations.

Overall, the computation of an optical flow is often very costly, especially for whole videos, and can only be done in real-time if a lot of accuracy is sacrificed. It has some advantages in contrast to background subtraction methods, e.g. it does not need a learning phase, handles illumination changes better and can also be used in videos captured by a moving camera because a static background (which can be modeled) is not necessary. Nonetheless, scenarios with moving cameras are quite difficult even for optical flow approaches since it is very delicate to differentiate between real movement and movement induced by the camera. Problematic for optical flows are in general the detection of foreground objects that are barely moving for a period of time or a lack of detectable features in the video.

MoG + Optical Flow

Another way to adapt the segmentations to the smoothness of natural images is the usage of the optical flow as it is inherently very smooth. Furthermore, it provides a second cue for the

---

[2] https://lmb.informatik.uni-freiburg.de/resources/software.php

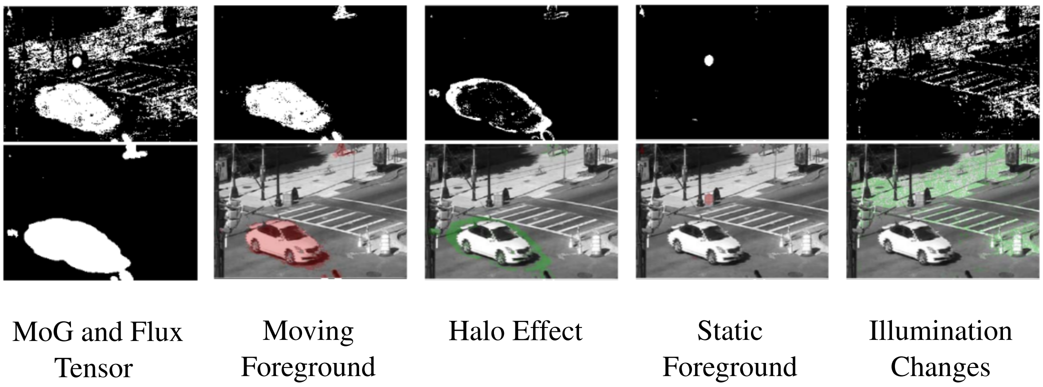


Figure 3.21: Example of a segmentation from [Wan+14a] where the two source segmentations (MoG background subtraction and Flux Tensor) are shown and four different derivatives from them. ©2014 IEEE

segmentation which is based on spatial change instead of temporal change (like background subtraction) which can also be used to compensate for the specific weaknesses of the background subtraction approach, e.g. is the optical flow not as sensitive to changes in the illumination. Wang et al. used the Flux Tensor to compute the optical flow in [Wan+14a] and used it together with a modified MoG background subtraction. They could achieve state of the art results on the *changedetection.net* dataset.

Figure 3.21 shows how much information can be gained by combining these two segmentation methods. The MoG segmentation (left top) is prone to noise and false detections due to illumination changes but gets the borders of the objects very accurate. In contrast to that, the optical flow (left bottom) has problems with foreground objects which do not move for a short time and does overestimate the size of moving objects in the direction of movement (halo effect). From these two segmentation results from different methods, a lot of information can now be extracted, pixels that are classified as foreground by both approaches belong quite certainly to moving foreground objects and can safely be marked as foreground (Moving Foreground in Figure 3.21). Pixels which were marked as foreground by the optical flow but not by the background subtraction belong to the halo effect of the optical flow. Pixels that were marked as foreground by the MoG but not by the optical flow can fall into two different categories. Most of them are illumination changes which were falsely detected by the MoG but some of them can belong to static foreground objects as they are undetectable for the optical flow. To differentiate between these two categories Wang et al. used a foreground model if the pixel coincides with the foreground model no illumination change has happened recently and therefore the pixel belongs to a static foreground object.

This extracted information about different classes is not only directly useful for the creation of the segmentation of the current image, but can also be very helpful in the updating process of the background model so that the adaption to illumination changes can happen faster and the addition of foreground information to the background model can be prevented better. Altogether this combination can produce a segmentation quality which is similar to the combination of MoG with MRF. The difference is that here the focus is not on the spatial relationship of the pixels

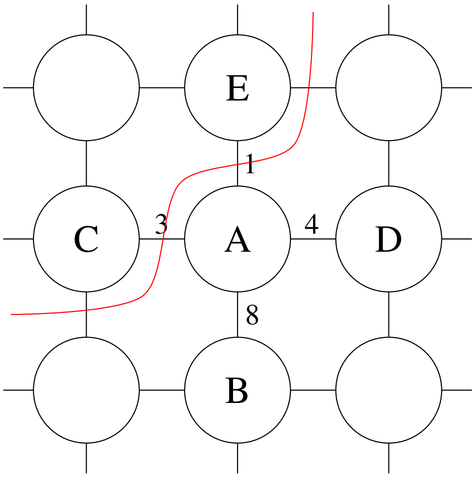


Figure 3.22: Graph representation of an image with a four-connected neighborhood. The red line is a cut through this graph that segments the image into two parts.

and therefore the spatial coherency is not as perfect as with an MRF model. Nonetheless, the additional value of having a second, completely different segmentation cue makes up for that. As the computation of an optical flow is very complex a usage for real-time applications is again very difficult, there exist some methods which can compute optical flows in real-time but they often lack in accuracy [AM16; Sey+16b]. Similar approaches with different optical flow algorithms can be found in [LH14] where it was combined with super pixels and an MRF or in [RC14] where alpha matting was added.

3D GraphCut

Graph cuts originate from the single image segmentation and have there been used for a long time already. The image is interpreted as a graph $G = (N, \Upsilon, w)$ with the nodes N that correspond to the pixels and the edges Υ which model the neighborhood relations, usually a four-connected neighborhood is chosen. The function w weights the edges based on the similarity of the two pixels that are connected by the edge. The segregation into two classes in this model is done by looking for an optimal cut through the graph. There are many criteria for the best cut (e.g. NCut [SM00]) but usually, this means that edges with low weights are cut while edges with high weights stay untouched as they signify a high connectivity. An example is depicted in Figure 3.22 where the pixels C and E are in a different class than A, B and D because their connectivity/similarity to A is very low.

This idea has been adopted for videos by analyzing a whole batch of frames at the same time instead of just a single image. The video is viewed as an image volume and a three-dimensional graph is created that represents this image volume. An optimal cut is then computed in a similar manner as in the two-dimensional case. This way a temporal smoothness of the segmentation can be enforced but the computation of such an optimal cut usually is very costly. In [Tia+11] the frames of the video were first over-segmented with a mean shift method and then a graph was created where each node would represent one of the segmented areas instead of single pixels. Thereby, they could reduce the size of the three-dimensional graph drastically without losing much accuracy since the borders of the areas which were separated by the mean shift approach would coincide with the edges of the objects in the frame. For each area, a foreground probability is then calculated based on the histogram and an optical flow. This calculation works on with the assumption that sudden changes (visible in the histogram) or movement (visible in the optical flow) are indicators for foreground objects. Together with a smoothness term between the segmented areas, this allows for a smooth segmentation of moving objects.

The computation time of this approach is enormous since it not only needs the optical flow maps of all the frames but then also has to find the optimal cut in a very large 3D graph. Also, only one object can be detected in the graph at once, which is why the method was only tested on simple and small videos with one foreground object. The presence of several foreground objects would have a severe impact on the detection accuracy and require the computation of several cuts (one per foreground object) through the graph.

Often this idea has also been used for medical image segmentation. Usually, not videos are segmented there but volumes of two-dimensional slices of body parts, but the general idea is similar. An example is [Nan+16] where nuclei from microscopic images were segmented. First, in each image/slice, the nuclei were segmented separately. This was done with by extracting seed points from the images and then applying a two-dimensional graph cut. With these segmentations, a first three-dimensional segmentation can be obtained and the nuclei can be separated from each other. However, the segmentation is not smooth between the different slices. Therefore, for each detected nucleus a directed three-dimensional graph is created and the minimum $s-t$ Cut in it computed. With the correct problem formulation, this method extracts the 3D Volume with the minimal surface that includes the nuclei.

With this method, volumes which contained many nuclei could be segmented accurately but the method is very specialized and cannot be easily converted to other segmentation problems. Overall, it can be stated that these algorithms are not suitable for general video segmentation as they are too computationally expensive and at the same time too limited in the number of foreground objects which can be handled at the same time. However, for special applications with fixed conditions, like in medical imaging, they can be very useful because graph cuts, in general, are a great tool for segmentation. This can be observed in the single image segmentation task where they have been used for many years and very regularly.

Evaluation of General Video Segmentation Approaches

There are two main ways to use the temporal information provided by a video efficiently for the segmentation task, computing the optical flow or modeling the background and subtracting it from the current frame. Other methods like 3D GraphCuts or subspace learning are hardly used, mostly because of their complexity (both: computational and algorithmic) and, additionally, their results cannot provide any improvements in accuracy over the other two methods in general.

The optical flow is solely based on the current movement in the video and will mark all moving pixel as foreground. One of the advantages is its great versatility: it can be used even if the camera is under movement or shaking since this movement of the complete scene can be detected and subtracted. Also, changes in the lighting conditions do only affect the detection accuracy for a very short time because only the movement at the moment is analyzed. One of the disadvantages is the computational load, which is usually much higher than for background subtraction approaches. Furthermore, the optical flow has problems with detecting foreground objects that stand still for a short time (e.g. a car that halts for a red light), objects that are "deforming" (e.g. a human which arms are alternating between being visible and being concealed by the body) and large uniform areas under movement.

Background modeling is faster and easier to achieve in real-time but requires a static camera since movements from the camera are very difficult to compensate. Unlike the optical flow, it has no problems with foreground objects that are not moving for a short while, deforming objects or large uniform areas. However, the background modeling has its own obstacles. When sudden changes in the lighting conditions occur a rather long period (hundreds of frames) is necessary to adapt the background model to the new conditions and get accurate results again. Furthermore, often bad detection results lead to a deterioration of the background model which then impairs the accuracy of future segmentations.

Overall, both methods are capable of creating accurate pixel-wise segmentations and should always be taken into consideration for binary video segmentation tasks. For the specific purpose that is discussed here, the background subtraction method was chosen since a static camera is always used anyway to observe the fish. Furthermore, real-time capability might become an issue when it is applied and, therefore, the fast background subtraction algorithm is more adequate.

3.3.2 Underwater Video Segmentation

The problem of detecting change in underwater videos has not been the focus of attention in recent years. One reason for this is that underwater scenes are often not available to scientists and the creation of them would be expensive due to the high costs of underwater imaging equipment. Furthermore, when underwater scenes are investigated it is not uncommon to ignore the special difficulties of these scenes and just use standard algorithms. Sometimes this can be justified, e.g. in an aquarium when the water is very clear so that there is no marine snow and hardly any color attenuation or blur, but often it is not and the existing problems are just ignored. In the following, some of the few existing underwater video segmentation approaches are discussed.

Spatio-Temporal Bayesian Framework

A fixed camera was installed in a pool to observe human swimmers in [KF10]. Although the camera is static, a background modeling is not possible in a situation like this because of the waves and reflections on the water surface as well as the turbulences in the water caused by a swimmer. Therefore, different regions were identified previously and a Bayesian framework was trained with the derived data. The extraction of these different regions was done based on their specific properties, e.g. the clear water was extracted by a segmentation of the blue channel of the image and turbulent water by a thresholding of the variance of small neighborhood regions. They used geodesic distances to achieve a spatiotemporal coherent segmentation of their videos. This was done by lowering the probabilities of being foreground in their Bayesian framework drastically if a pixel was far away (in the geodesic distance measure) from any foreground region in the previous segmentation. The reasoning behind this is that the movements of the human

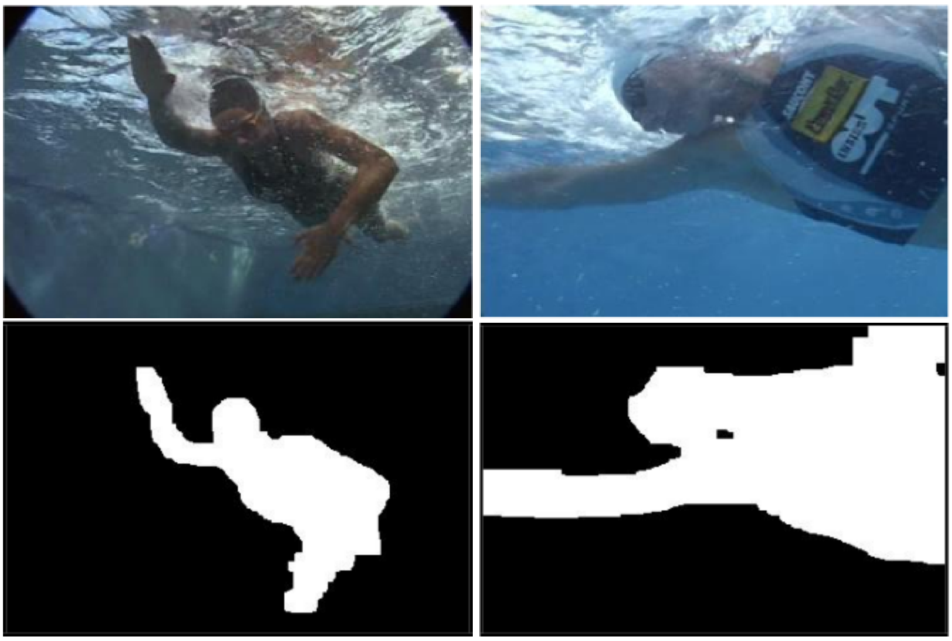


Figure 3.23: Two examples of the swimmer segmentation from [KF10]. At the top the reflections on the water surface as well as the turbulences due to the movements of the swimmer are clearly visible. ©2010 IEEE

swimmer have a finite velocity and therefore a region that was far away from any foreground region cannot be foreground in the next frame. This is a valid assumption but is only useful when the previous segmentations are close to the ground truth and the framerate is high enough. An example is depicted in Figure 3.23.

The presented approach is very specialized and adapting it to other scenarios would be difficult and time-consuming. Their segmentation results are visually convincing but no real comparison or evaluation could be done since ground truth data for this situation is not available. Nonetheless, the method illustrates the difficulties of underwater segmentation quite well, e.g. because background subtraction cannot be used although the camera is static due to all the turbulences and reflections.

Optical Flow with Focus of Expansion

Another very specific approach, this time for the detection of jellyfish, was suggested by Wang et al. in [WWW11]. They allowed camera motion in their approach but restricted the movement to be mainly in the z-direction (meaning: forward, into the field of view). Under this motion, there exists a point from which all other points seem to move away, the Focus of Expansion (FoE). With this in mind, the optical flow was computed with a standard method [HS80] and then the results were set in relation to the FoE point. If the movement of a pixel was not directly away from the FoE point it had to belong to an object which was under motion itself should, therefore, be marked as foreground. In their videos, the moving objects were mostly jellyfish but a classification step was still necessary. The classification was mainly based on the average intensity

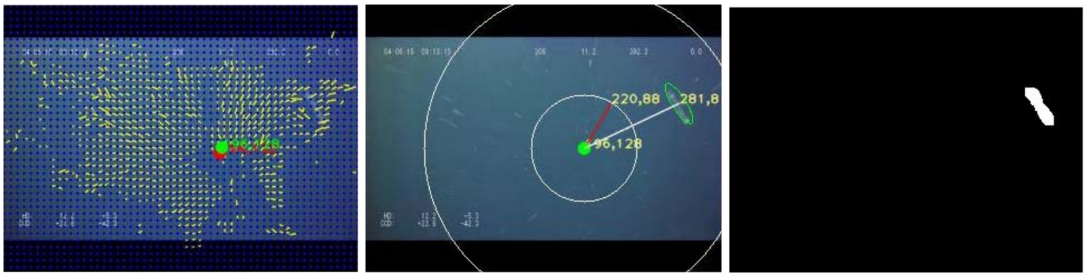


Figure 3.24: The left image shows the optical flow and two estimations of the FoE point (green and red dot) from the approach of [WWW11]. In the middle are the different areas around the FoE depicted, different thresholds are used in each of these areas. The final segmentation is depicted in the right image. ©2011 IEEE

of the detected object in relation to the distance to the FoE point. This distance is important because jellyfish usually appear brighter near the FoE and darker at the corners, hence different intensity thresholds were used for different regions. The optical flow, FoE point detection and an example of a segmentation are depicted in Figure 3.24.

The whole method is specialized for one scenario and therefore hard to evaluated against other approaches, similar to the previous swimmer segmentation. It would be easily possible to use the idea of the FoE point to segment all moving objects in such a scenario with a non-static camera. However, the ROV (or car, or plane) on which the camera is mounted, is then only be allowed to go straight forward as otherwise the FoE would change and this deteriorates the segmentation results. Moreover, the results are completely dependent on the quality of the optical flow which is often problematic in underwater scenarios because they need to find features in the frames. The degraded underwater images often have less features (that can be found) which makes an accurate computation of the optical flow difficult.

Background Modeling

Background modeling and subtraction is the most common approach for underwater video segmentation because it is not as strongly affected by the degradation effects in underwater scenarios as other approaches, e.g. optical flow. Nonetheless, there still are negative effects like the lack of color information (due to color cast) which makes it more difficult to distinguish between foreground and background or the presence of marine snow which complicates the modeling process. An example of the problems can be found in [Spa+08], where first a simple Single Gaussian approach was used for the background modeling. However, the quality of the results was not satisfactory due to the harsh conditions of the underwater videos and a more advanced approach had to be used. The MoG model from [Ziv04] was used by them afterwards and proved to cope better with the problems because it could model the foreground objects separately (in different Gaussians) which results in a better model and allows a stricter threshold after the background subtraction. Overall, the detection rate of the fish (not accuracy) was around 85%. Other examples are [Sha+15] where the background modeling and subtraction from [SG00] was used or [Liu+16] where the approach from [HHD00] was applied. However, in none of these

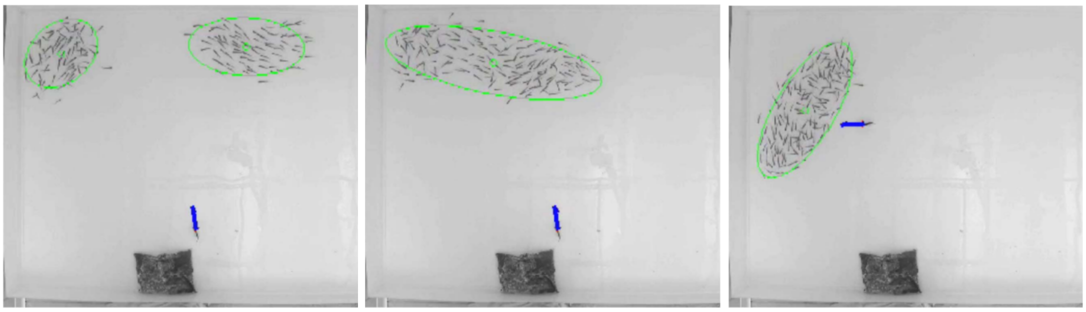


Figure 3.25: The fish basin of [SCL12] with the detected fish school marked by a green ellipse and the robot fish by a blue line.©2012 IEEE

a satisfactory evaluation of the results was done because of the lack of ground-truth data (e.g. comparing just the number of detection against the number of fish).

Background Modeling in real-time scenarios

A real-time feedback loop was created based on background subtraction in [SCL12] to control a small robotic fish. First, a small group of fish was detected and their movement was estimated. With this information, a robotic fish was controlled and used to influence the behavior of the whole fish school. Since the fish are dark and swim in very clear water in front of a white background the modeling and subtraction were straightforward in this scenario and a high accuracy could be achieved. A very important aspect was also the runtime because the robotic fish had to be controlled in real-time. Here the efficiency of the background subtraction could shine, the tracking and motion prediction of the fish was much more computationally expensive than the segmentation. Three examples of the fish basin are shown in Figure 3.25. A similar setup was used in [TNL09] to count fish in a small basin. There different methods like edge detection or thresholding were evaluated and background subtraction was the approach that could provide the best results.

In these two cases, background subtraction was used for underwater scenarios but in very artificial conditions which were very similar to in-air videos and did not reflect the special difficulties of most underwater videos. The method was mainly chosen because of its runtime to handle the real-time conditions.

Background-Foreground Modeling

An adapted algorithm was introduced in [Mor+05] where two MoG models were used, one for the background and a second one for the foreground. The background MoG consisted of four Gaussians, which is a common value for a MoG approach, and was trained beforehand on selected images which showed only the background. The approach was not pixel-wise as each of these Gaussian mixtures was modeling small areas of the scene. The foreground model was also trained beforehand on selected parts of images which only contained fish. In contrast to the background model, there was only one foreground model for the complete scene which was applied on all areas. The reason for this is that there was just one fish species which should be detected which made it possible to model the foreground by one MoG. However, to be able to model all different

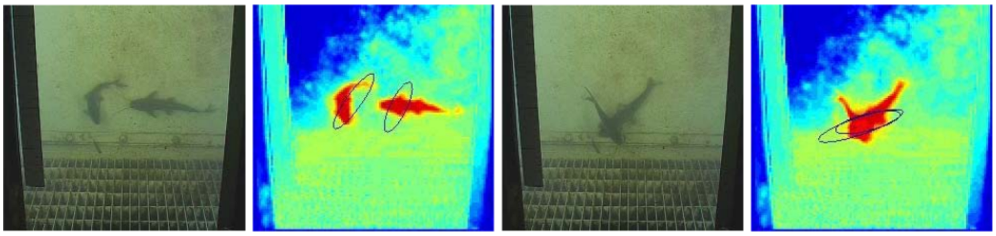


Figure 3.26: Depicted are two frames from [Mor+05] with their corresponding foreground probability maps. The black ellipses show the estimated fish position by the tracking approach. ©2005 IEEE

parts of the fish at the same time this foreground MoG had 25 Gaussians, an example of the results is displayed in Figure 3.26.

The segmentations were used to count the fish in the scene and analyze their movement. Although the water conditions were poorly they could achieve respectable results due to their offline training on hand-annotated data. Their detection accuracy reached 81% on their own dataset and they could track the fish positions with an error margin of only a few pixels. The usage of ground truth data for the training process, however, limits the usability severely and also scenes with several different kinds of foreground objects would be very problematic for this approach.

Background Modeling with Image Enhancement

An automatic fish counting on deep-sea videos from the ocean networks Canada [3] was done in [FAH14] with a combination of underwater image enhancement, background subtraction and tracking. As the conditions in the deep-sea videos were very harsh they used several preprocessing steps to enhance the quality of the videos. Among other things they reduced the effect of the marine snow with median filtering [HMY99], the noise was reduced with a bilateral filter and the contrast stretching was applied. The background subtraction was done with a standard MoG method but solely on the green channel of the image because, according to them, this channel was the least affected one by noise. The threshold for the background subtraction step was adaptively computed with Otsu's method. Lastly, to also count fish correctly that overlap each other, a blob tracking was introduced which uses the information from the DOF of Farnebäck [Far03] to predict the movements of the blobs.

They evaluated their data on 100 randomly selected videos from the ocean networks Canada and could achieve a fish detection precision of 65.8%. This is substantially lower than the previous approach ([Mor+05]) which could attain 81% in very harsh conditions. However, the advantage of this approach is its generality. It is not limited to one foreground class and does not need any prior training on ground truth data. Unfortunately, there was no evaluation whether the preprocessing had a positive effect on the results and it was not mentioned how computationally expensive it was.

---

[3] http://www.oceannetworks.ca/

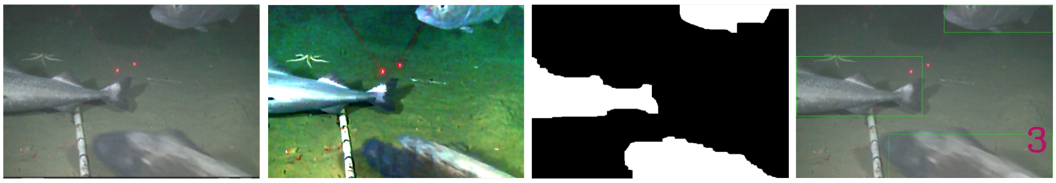


Figure 3.27: Results of the approach from [FAH14]. On the left is the original image, then the enhanced image, the segmentation result and the last image shows the tracking result overlayed over the original image. ©2014 IEEE

Background Modeling in Multi-Camera Systems

Multiple cameras in a small pool was used to detect and track robot fish in [Zha+16]. The fish were detected with a standard MoG background subtraction separately for each camera. They did not give any details about their accuracy but used these results (foreground blobs) to track the fish over time. The fish were tracked with a Kalman Filter approach over ten features for every camera and afterwards the detected fish were registered between the different cameras with a SURF-RANSAC algorithm (SURF: Speeded Up Robust Features) and thereby the different tracks of the cameras could be unified.

This paper could demonstrate that it is possible to use multi-camera systems for underwater scenarios at least under laboratory conditions (robot fish in a pool) and unify the results of the different cameras. Consistent trajectories could be computed for up to five fish at the same time but a real evaluation was not possible since no ground truth data is provided. Therefore, it is also not clear how the multi-camera approach could improve the accuracy over single camera methods.

In conclusion, it can be said that background modeling methods can achieve good segmentation results for underwater videos even under harsh conditions but although they were used in different scenarios no common dataset is available and therefore no real qualitative comparison possible. Some approaches created their own ground truth data for the evaluation but then did not publish it, which is unfortunate. Furthermore, the circumstances of the videos used are very different: from small basins with clear water in a lab, where effects like color cast or marine snow do basically not exist, to deep sea stations in the ocean several hundred meter below the water surface, where these effects are very prominent and interfere with the segmentation process. Nonetheless, for general change detection, the background modeling and subtraction approaches seem to be the best choice if the camera is static. The only methods that could compete with them on the in-air datasets were based on the optical flow and these are in general slower and suffer more from underwater degradations.

3.4 Conclusion

Approaches about the segmentation of images or videos of underwater scenes are strongly underrepresented in the scientific community due to the difficulties of obtaining them and because there are fewer use cases in the underwater world. Many of the papers that deal with these scenarios concentrate on sonar imaging since it is a special approach that is only viable in the underwater world. For in-air situations, sonar imaging grants no advantages over imaging with electromagnetic waves. In water, on the contrary, sound waves have a specific benefit over electromagnetic waves and especially the visible light, they are not absorbed by the water as quickly and therefore have a vastly larger range. Nonetheless, their spatial resolution and frame rate are not high enough to provide information that could differentiate exactly between various fish in a scene. Therefore, this technology is not viable for the detection of single fish but rather for the identification or larger objects like fish swarms or submarines.

Accordingly, this related works was focused mainly on approaches that used optical imaging and try to deal with the degradation effects and limited range of the underwater world. For the single image segmentation many algorithms were examined but often they could not comprehensively differentiate between background and foreground and just clustered pixels into different groups (e.g. thresholding or mean shift). These can only be used in very simple situations directly for a detection of change (e.g. when the background is completely white) but fail in more complex scenarios. For these scenarios, the mentioned algorithms can mostly only be used as pre- or post-processing steps to aid other more sophisticated approaches. There are also single image approaches that directly rely on provided data (e.g. MRF on foreground probability maps) and, therefore, fall in the same category of supplementary algorithms. Only a few approaches could really create foreground-background segmentations on single images, and they were dependent on a training phase (e.g. CNN) so that they could learn the difference between a special object category and the rest of the scene. To create a general approach that works in most scenarios without any prior learning a whole video with its additional time dimension is therefore necessary.

Video segmentation approaches, in contrast to single image algorithms, exploit the temporal information of a video and do not work on one frame alone. Of these approaches, only very few were adjusted for underwater scenarios and, therefore, the section about video segmentation started with a summary of in-air methods. Also, in contrast to the underwater case, for the in-air scenario several common datasets exist so that an evaluation of different approaches is possible. Basically, there exist two different techniques for the change detection in videos, the computation of the optical flow or the creation of a background model. For both several methods and combinations with other approaches exist. However, in general, they have quite specific advantages and disadvantages. The background modeling is fast and accurate even in complex scenarios but is limited to static cameras, has the necessity of a training phase and has no inherent spatial coherency. The optical flow, on the other hand, is spatially coherent, not limited to static cameras in general and can also handle complex scenes, however, it is slower than the background modeling, usually not as accurate and has specific problems e.g. with large homogeneous areas.

For underwater scenarios the situation is very similar. Both methods, the optical flow and background subtraction, suffer from the degradation effects of the underwater world but the negative impact is more pronounced for the optical flow. Especially the blur and haze make it difficult to find enough features in the scenes which can be matched among frames to create an accurate optical flow. The background modeling is, of course, also affected by the degradations, e.g. the lack of color information makes the differences between background and foreground less pronounced, but this effect is often marginal. In the current works about underwater video segmentation, these specific problems were often only insinuated because of the lack of a common dataset to qualitatively compare different methods. A detailed discussion of this is done in section 6 of this work. Overall, background modeling is faster, at least as accurate as optical flow approaches and suffers less from the underwater degradations and is therefore preferable for the change detection in underwater scenarios with a static camera, which is the scenario that is discussed in this thesis. Hence, a background subtraction approach will be used in this work to create segmentations.

Open Problems

The main challenge for background subtraction methods is the updating of the background model and existing updating methods are often complicated (e.g. many input parameters are required) which makes them difficult to apply and also impacts their runtime. Therefore, approaches with low and very stable runtimes which also have fewer parameters are sought-after. Even if a good background model can be obtained, there is still the lack of spatial coherency that impacts the segmentation quality because of the pixel-wise nature of the background subtraction method. This is especially important for further processing steps (e.g. tracking) since otherwise an object might be split into many small parts. At the moment, algorithms which deal with that problem are all based on single images and either unsophisticated (simple smoothing) or very slow (Markow Random Fields). Hence, a method that delivers accurate and spatially coherent segmentations in real-time is not available yet.

Another set of problems that will be examined are the underwater degradations in videos. Their effect is almost completely unaddressed in previous works and, therefore, it is an open question how grave their effect on the segmentation quality is, how to deal with them during the segmentation and if, for example, image enhancement methods can help in this regard. A special problem in this scenario is crowded scenes. They appear especially often underwater (fish swarms) and need to be addressed specifically because the main assumption of the background subtraction approach - the background is visible more than 50

After the segmentation, to gain more high-level information, a tracking over time of the segments is advisable. Existing tracking approaches often use pre-trained detectors for specific objects. To not be limited to specific objects a more general approach like blob-tracking is preferable. The limiting factor while tracking general blobs/segments is that small errors (e.g. one object is sometimes detected as two blobs) can lead to great confusion during the tracking. For this, novel algorithms are necessary that deal with these unavoidable errors during the segmentation phase and correct the tracking accordingly.

Outlook

To address these just mentioned problems, a novel background updating algorithm will be developed in the next chapter which can detect and precisely segment fish underwater. It will be combined with a heuristic to detect changes in the lighting conditions of the scene which allows a fast adaption of the model to these new conditions. Furthermore, the model will be applied to a special color space which facilitates shadow detection. Overall, this combination provides a versatile and accurate background subtraction method that can later be expanded and adapted to the specifics of underwater scenarios.

The lack of spatial coherency is discussed next. To mitigate this problem and improve the segmentations three different and novel methods will be proposed. Each of them has their own advantages and deficiencies and should be chosen according to the specific task. First, an advanced MRF approach that uses a larger neighborhood than previous methods is introduced. Next, an adaption of the normalized Cut to the video segmentation problem will be proposed and, lastly, a method that uses the optical flow to combine the information about the segmentations from several frames.

The first underwater specific problem is the lack of a common dataset to evaluate algorithms fairly. Hence, a new underwater dataset will be presented here with fish as foreground objects. The proposed and other state-of-the-art algorithms will be tested on it, for example to evaluate if background subtraction or optical flow approaches suffer less from the underwater degradation. Furthermore, the addition of four different image enhancement techniques and their influence on the segmentation accuracy is thoroughly reviewed.

The next part deals with crowded scenes and their implications for background subtraction. Standard background modeling algorithms usually fail in these situations because they start to model the foreground objects in these scenarios instead of the background. To solve this, a special preliminary step is proposed which segments the frames roughly based on their optical flow. The background subtraction method then only gets the remaining (background) parts of the frame which circumvents this problem. Additionally, the effect the image degradation has on the performance of the spatial methods has never been tested. Since methods like the Ncut often rely on prominent edges in the images, a strong decline in effectiveness is to be expected. Therefore, all three proposed spatial methods are evaluated on underwater videos to see which of them can handle the difficult conditions better and can be used together with the specially adapted background subtraction.

In the end, the presented algorithms are combined to make accurate segmentations of fish and then a novel general-purpose tracking method is developed. This is necessary to gain important data about the behavior of fish (e.g. velocity, direction movement, average velocity) which could then be used for the detection of illnesses or other anomalies. To keep the generality of the presented approach, the tracking - unlike most other approaches - does not use any specific data about the fish that should be tracked but just uses the detections made from the background subtraction method. This allows an easy adaption to other fish-species or even completely different objects or contexts.

4 General Change Detection

In this first part, a new method for background modeling for general change detection will be introduced which is the foundation for all subsequent algorithms. It is based on two differently updated models and by comparing them errors in the modeling process can be found and corrected. Since one of the biggest drawbacks of the background subtraction method is its lack of spatial coherency, this will be addressed in this chapter as well by introducing three distinct spatial methods and analyzing their particular advantages and weaknesses. Although this thesis tries to provide solutions for the underwater change detection problem, this first part will deal with the general in-air case. This is justified because of the general background modeling approach, as well as the spatial model, which are in both scenarios almost identical. For example, in both situations it can be assumed that natural objects are almost always smooth, without sharp edges or holes, and should have a minimum size to get detected. Therefore, addressing the general case at first is very beneficial since it allows an extensive evaluation on common datasets as well as a qualitative comparison with many other methods. A dataset for the underwater case will be introduced in the next section and afterwards, the here proposed algorithms will be extended and adapted to handle the specific difficulties of underwater scenes.

The idea behind the detection of change in a video is that moving objects are usually the most interesting parts of a scene and it can be very useful to separate these objects from the rest of the scene for further analysis, e.g. classification. The advantage of this approach is that change is a very general concept so that it can be used in many different situations without a special adaption of the algorithm. Common examples are the detection of people walking by in a pedestrian area, the counting of cars passing an intersection or the surveillance and burglar detection in deserted areas and/or at night. The one limitation of this technique is that it requires a static camera so that there is no perceived motion from stationary objects. Otherwise, it becomes very difficult to differentiate between the motion of objects and the motion of the camera and in these situations often other approaches perform better. Therefore, there is always a static camera assumed for the presented algorithms.

Background subtraction is by far the most used method for the task of change detection at the moment and its popularity comes from two facts. First, it is very efficient and can run in real-time even on cheap hardware or one that was designed for low energy consumption. The second point is that the principle behind it is simple and easy to understand but is able to create very accurate results in most situations. Other approaches which could be used for the detection of change, e.g. optical flow, are more complicated in their development and usage but cannot achieve better results for static camera scenarios. The basic principle behind background subtraction is a comparison between a model of the scene that includes only stationary objects and the current frame of the video. All objects that are moving, and thereby creating change in the video, can then be detected as a difference to the model.

A drawback of this approach is that it only analyzes each pixel separately and does not use the spatial relations in a frame. The objects that should be detected are usually rather big (several hundred pixels), connected and have smooth borders. This knowledge is ignored if the background subtraction approach is used individually but can and should be used to improve the segmentations afterwards, e.g. by using MRF or the optical flow to delete small false detections and close holes in segmented objects. This simplifies the further processing of the segmentations drastically because it reduces the number of (falsely) found objects and overall increases the accuracy. Therefore, this step, although sometimes computational intensive, cannot be neglected and is an important part of the segmentation process with background subtraction.

At first, a new background modeling and subtraction algorithm will be introduced which combines the efficiency and simplicity of Single Gaussian approaches with the accuracy of Mixture of Gaussian models. Afterwards, three methods are introduced which model the spatial coherency of the segmentations and make them coincide with the aforementioned assumptions. The first is a Markov Random Field which uses larger neighborhoods than the 4-connected (von Neumann) neighborhood previous approaches used to better model the relations between the pixels and regions. The second approach uses the idea behind *NCut* and adapts it for the usage in videos instead of single images. Lastly, a method is proposed which adds a temporal component by tracking a pixel in the following (and previous) frames. For this tracking, dense Optical Flows are used which allow the usage, not only of the spatial neighborhood but also of the temporal neighboring pixels (e.g. pixel in the previous frame).

4.1 Gaussian Switch Model

The most important but also most difficult part of a background subtraction approach is the modeling of the background. If an adequate model is available, the creation of a segmentation via subtraction and thresholding is usually unproblematic. As described in Section 3.3 most of the current algorithms use statistical methods for the modeling process and usually for each pixel one or several Gaussian distributions are used to model the color of that pixel. The usage of Gaussian distributions is justified by the fact that the intensity of a pixel in a completely static scene will theoretically vary according to a Normal distribution $\mathcal{N}(\mu,\sigma^2)$ with mean μ and variance σ^2 due to the measurement errors inherent in the camera system.

Therefore, Gaussian distributions will be used for the proposed approach as well but instead of using many Gaussians for each pixel and channel the aim is to create a genuine model of the background by using only a few Gaussians by using an intelligent updating scheme. This helps to minimize and bound the memory consumption and runtime as well as simplifies the algorithm so that fewer parameters are necessary. In the end, a model M stores all the Gaussians that describe the background of the scene and it has the same dimensions as the current video V. A specific Gaussian can be addressed with a location $\bar{v}$ and a channel c. Additionally, the required parameter of the Gaussian must be stated, mean μ or variance σ. Thus, $M(\mu,100,100,r)$ is the mean of the red channel at location $(100,100)$ in the video.

The first step is always the initialization of the model. The most common and general approach was chosen which is to take the first image as the initial model. There are other and more sophisticated ways to initialize the model, but they require extra knowledge or expensive computations which cannot be done in real-time and if there is an advantage of such an initialization it vanishes quickly as the importance of the first state decreases with each new frame. For the pixel at location $\bar{v}$ the mean and variance of corresponding Gaussian in the Model M is set to

$$M(\mu,\bar{v},c) = V(\bar{v},0,c) \qquad \text{and} \qquad M(\sigma,\bar{v},c) = \zeta \tag{4.1}$$

as initial values. It is also possible to treat each frame as a grayscale frame and model only the intensities, this saves almost $\frac{2}{3}$ of the computation time and still gives good results. Nonetheless, in difficult situations, the color information can be very valuable e.g. to differentiate moving objects from shadows. The variable ζ is a constant value and was set to 0.001 but the importance of this variable is minor as the variance gets adjusted quickly.

After setting up the model, it needs to be updated with every new frame from the video. One reason is that there can be foreground objects in the first frame which are part of the model after the initialization and now have to be unlearned and another aspect is that the model has to adapt to changes in the background, e.g. changing light conditions or new background objects (objects that are static for a long time). Therefore, with each new frame at time point t the model is updated in the following way

$$\begin{aligned} M(\sigma,\bar{v},c) &= \alpha \cdot M(\sigma,\bar{v},c) + (1-\alpha)\cdot (V(\bar{v},t,c) - M(\mu,\bar{v},c))^2, \\ M(\mu,\bar{v},c) &= \alpha \cdot M(\mu,\bar{v},c) + (1-\alpha)\cdot V(\bar{v},t,c). \end{aligned} \tag{4.2}$$

The update rate $\alpha \in (0,1)$ controls the impact of each new frame on the model and is therefore very important for the model building process. Usually, it is set close to one so that every new frame has only a small impact on the model. This ensures that the model is not dominated by the newest frames but an average of the last several hundred frames. Also, the data from the different frames is inherently weighted by that updating scheme so that newer frames have a greater influence than older frames.

The problem with this model building is that it does also include the information from foreground objects into the model and therefore corrupts it. Especially when there is a constant presence of many foreground objects, the background model gets influenced and corrupted heavily by them. Usually, these problems are tackled with a Mixture of Gaussians (MoG) approach in which several Gaussians are used for each pixel and each channel. The idea is that the background is modeled in one Gaussian and the foreground objects separately in different Gaussians.

Another approach to solve this is to make a *partial update*, this means that instead of updating the complete model (all pixels) only the Gaussians are updated that correspond to pixels which were classified as background. Ideally, now only background information should get included into the model which should make the modeling process more robust and precise. To achieve this the segmentation of the current frame must be computed by the background subtraction before the model is updated with the information from the new frame. Then this segmentation can be used to only update the background pixels and exclude foreground objects from the updating process. In general, this improves the segmentation and makes the model more accurate, but as the model is used to improve the updating process for the model itself, a kind of self-fulfilling prophecy can occur.

An example of this is the presence of a foreground object during the initialization. This foreground object is a part of the model in the beginning and should slowly be overwritten with background information during the updating process. However, when partial updating is applied this regularly does not happen because the actual background in that area will be marked as foreground and therefore not get included in the model. The true background will be classified as foreground because it is very different to the current model, which has the information from the foreground object of the first frame. An example of this and a depiction of the problems of the partial and complete update approach can be seen in Figure 4.1.

The same problem occurs when a new object becomes part of the background and should, therefore, get included into the background model over time. An example would be a car that comes into the scene and parks, it is first a foreground object since it is moving and then becomes part of the background because it becomes permanently immobile. Objects like these never get included into the model when partial updating is applied because they are recognized as foreground objects and therefore the model is not updated. To still get the benefits from the partial updating without these problems, the Gaussian Switch Model (GSM) is proposed which uses exactly two Gaussians to model the background. The first Gaussian is partially updated and is standardly taken as the background model while the second Gaussian is fully updated with every frame. This second Gaussian is used to detect the problems that occur with the partial updating method. By comparing the two Gaussians the errors in the partially updated model can be detected since they always show the same characteristics:

- The means of the two Gaussians slowly diverge from each other as the model that is fully updated adapts to the new background and the other stays constant.
- For many successive frames a foreground object is detected at the same position.

If these characteristics are true for a specific pixel, the partially updated Gaussian for that pixel gets overwritten with the values of the full updated Gaussian as it does not reflect the true background anymore. Furthermore, this switch to the fully updated Gaussian only happens when the variance of this Gaussian is small because this indicates that there has not been much noise or many foreground objects recently and therefore its values can be taken as a reliable background model.

An example of this background modeling with the GSM can be seen in Figure 4.1 (the mean values of the Gaussians are displayed to represent the background model) where it is compared to the full and partial updating schemes on a video with many foreground objects. The parameters of the modeling (e.g. α) are the same for all three methods and it can be seen that the complete update created a model which is corrupted with many of the current foreground objects of the scene. The partial update eliminates this problem but many objects from the first frame can still be seen in the model because they have never been overwritten with the real background information. The GSM can combine the advantages of both methods and can create an almost uncorrupted background model.

The next step, after the creation of the model, is the subtraction of the model from the current frame of the video. This gives a difference frame which can be thresholded to generate a binary segmentation. The information from the Gaussians of the model can be used to create an adaptive individual threshold for each pixel instead of one global threshold. The important information

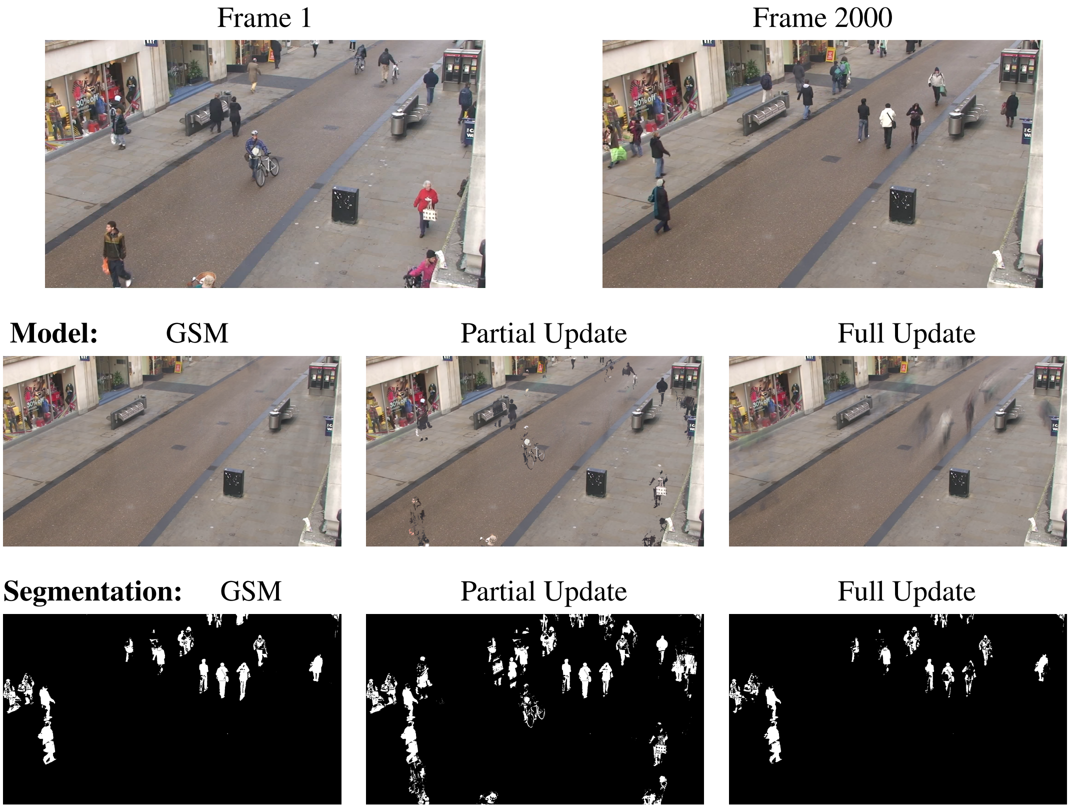


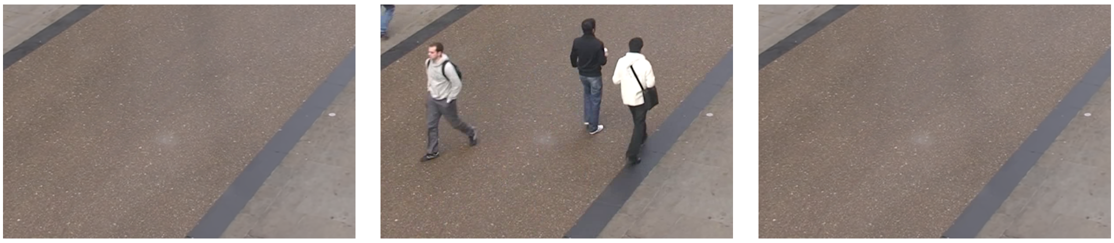
Figure 4.1: Comparison of different update schemes for the background modeling. In the top row are the first and 2000th frame of the Town Center video from [BR11]. In the next row are three background models for the 2000th frame of the video created with the same parameters but different updating mechanisms: the first was created with GSM model, the second with partial updating and the last one with a complete update for every frame. The last row shows the corresponding segmentations for every model.

here is in the *variance* as this encodes the expected deviation from the mean and hence is a good reference for the threshold. Therefore, for each pixel and channel the inequality

$$(M^p(\mu,\bar{v},c) - V^p(t,\bar{v},c))^2 < \max(\beta \cdot M^p(\sigma,\bar{v},c),\gamma) \tag{4.3}$$

is checked. Here M^p stands for the Model of the partially updated Gaussian, β is a parameter which defines the sensitivity of the algorithm and γ is a lower threshold since the variance can attain very small values when there is no foreground object for a long time and this would lead to a very high sensitivity and a noisy segmentation. Each channel is checked separately and afterwards a voting algorithm is applied to unify the results. If at least for two of the three channels of a pixel the inequality is true, the pixel will be classified as background, otherwise as foreground.

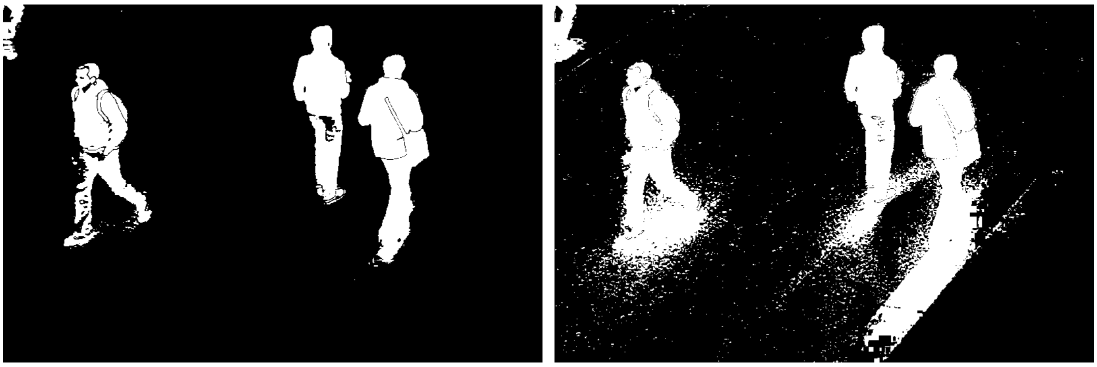
An advantage of these separate handling of the channels is that shadows can be detected easier. Shadows are a serious problem for change detection in all scenarios as they are real changes in



Model: converted color space

Original

Model: RGB color space



Segmentation with converted color space

Segmentation with RGB color space

Figure 4.2: Comparison between different color spaces and their effect on the segmentation on the Town Center video. On the left top is the model created with the converted color space (see 4.4) and on the top right is the same model but created with the standard RGB color space. In the middle is the original frame where three persons are visible. The bottom row shows the corresponding segmentations. Although both models seem very similar the misdetections of shadows are more pronounced when the RGB color space is used.

the scene but should usually not be detected. The only difference to a real object is that a shadow only darkens the background and does not change the color of that area (if the shadow is not too strong). This characteristic is used to differentiate real moving objects from their shadows.

For this the color space of the images is changed from the common *RGB* to a color space which has the brightness information encoded in one channel and in the other two channels are then pure color information. The conversion is done with the following equation similar to [Li+08]

$$\begin{aligned} L &= R+B+G, \\ C_1 &= R/L, \\ C_2 &= B/L. \end{aligned} \tag{4.4}$$

Afterwards the Intensity L is scaled with the factor $\frac{1}{3\cdot 255}$ so that all values are in the range $[0,1]$. This color space is similar to the YUV space because both have one intensity channel and two channels with color information, however, the described color space does not have any weighting parameters to adapt the colors to the human perception as this is not necessary for the automatic image segmentation task.

After this division into brightness and color a shadow should only be visible in the intensity channel L and not in the two other channels and therefore will not be classified as foreground since the two color channels overrule the intensity channel in the voting algorithm. However, this will not work when the shadow is very strong since then the color information is lost completely and the channels C_1 and C_2 might also indicate a change in that area. Figure 4.2 shows a comparison of the color space used here with the standard RGB color space. The models for both color spaces are similar and accurate but the derived segmentations are different and show that new color space suppresses the segmentation of shadows quite well.

In natural scenes, the lighting conditions are often very dynamic (clouds blocking the sun, reflections, shadows etc.) and consequently the changes in the lighting are in general at least an order of magnitude higher than the variations in the color of the background. Hence, the intensity channel should be treated slightly different than the other two channels, for example in equation 4.3 the parameter β should have a higher value when the channel with the brightness information is checked than for the color channels to incorporate the higher variations of the intensity values in the model.

Another common problem are rapidly changing lighting conditions of the whole scene, e.g. induced by a light that is turned on or off, the sunset or a cloud blocking the sun. These events often result in the classification of almost the whole scene as foreground and afterwards, it takes the model a long time to adapt to the new conditions. To improve this behavior [Toy+99] proposed a method which checks for these events and then retrains the model rapidly for the new lighting conditions. This idea was adapted for the GSM algorithm by checking in every new frame if more than 75% of the pixels are classified as foreground but only the intensity channel is taken into consideration for this because these rapid changes usually only affect the brightness. If this is the case the update rate α will be lowered to 0.5 for this specific frame and every pixel of the model will be updated (no partial update). This increases the adaption speed of the model drastically and improves the results of the algorithm in the difficult lighting situations.

4.2 Spatial Coherence

A drawback of the background subtraction method for change detection is that it does not take into account the smoothness of natural images. Each pixel is treated separately and the neighborhood of that pixel is not taken into account at all. However, for real world scenes some assumptions can be made, e.g. objects are larger than a few pixels, borders are smooth (not zick-zacking) or objects do not have (small) holes. These assumptions can then be used to improve the segmentations derived by the background subtractions. In the following sections, three different methods are proposed that add spatial coherency to the segmentations in very distinct ways and each has its own advantages and disadvantages which will be analyzed.

4.2.1 Higher Order Markov Random Fields and Belief Propagation

The MRF is a well-established and widely used statistical model which can describe the dependencies between various random variables. It originates from the work of Ising [Isi25] on ferromagnetism but was since extended and adopted to many different problems, especially in image processing and computer vision.

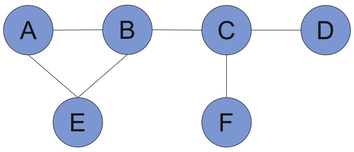


Figure 4.3: A graphical depiction of a MRF. Here E depends on A and B; B depends on A, E and C; D depends only on C and so forth.

A small example of an MRF represented as a graph can be seen in Figure 4.3. The random variables are depicted as circles and the edges show the dependencies between them. This easy and graphical way of modeling the relations between random variables can be useful in a great variety of applications, one example is the spatial relation between pixels in an image. In this case, every pixel is represented by one random variable for which the state is unknown and which has dependencies on all neighboring pixels. Thereby, the state of a random variable could indicate if the corresponding pixel is in the foreground or background of the image, denote the optical flow at that point or any other information which shall be obtained for a single pixel.

A crucial point for this is the neighborhood system which is chosen, a small system like the von Neumann neighborhood (four-connected neighborhood) might be unable to model all complex relations between the pixels and a larger system will soon create models which are unmanageable. For image processing or computer vision algorithms, the von Neumann neighborhood is almost always chosen because it will create a pairwise MRF. In contrast to higher order MRF, these have the advantage that they only have cliques of one or two nodes which makes the following optimization computationally much easier. The small example MRF in Figure 4.3 is not pairwise because A, B and E are all connected among themselves and, therefore, already form a clique of three nodes.

In the optimization step the maximum a posteriori probability (MAP) should be estimated, which is the most likely state of the overall system based on prior knowledge (natural images are smooth) and observations (the current frame). The computational difficulties derive from the fact that in every clique all members will be influenced by all the others. To deduce an approximation of the MAP the Belief Propagation algorithm will compute messages from every possible clique to all of its members. This means that in Figure 4.3 node E will receive one message depending only on A (because E and A form a clique), one depending only on B but also a third message depending on both of them (because A, E and B form a clique). For larger neighborhoods, each node can be in tens or even hundreds of different cliques which will make the computation and especially the storage of all the messages nearly impossible.

In the Moore neighborhood (eight connected neighborhood), which is the next larger system

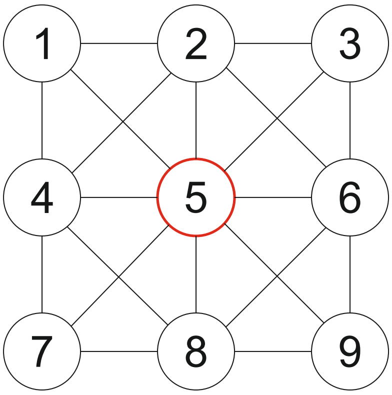


Figure 4.4: Here the Moore neighborhood of node five is shown. There are eight cliques of size two, twelve cliques of size three and four cliques of size four in which this node is a member.

which is commonly used for images, every node is already a member in 24 cliques (see Figure 4.4). Also, it has to be noted that there is not just one message from every clique to each member but one message for every possible state the clique(!) can be in. For example, if there is a clique with six nodes and every node has two possible states then each node can receive up to 2^5 messages because the other five nodes can be in this many different states. Every one of these messages will deliver information on how likely it is that the clique will attain one certain overall state. To reduce this heavy computational load some simplifications to the MRF model will be introduced later which will make it possible to compute good approximations of the MAP even for advanced neighborhood systems.

Before discussing these simplifications, the model has to be completely defined and created. As a neighborhood system, the generalized Moore neighborhood (GMN) is chosen, which is a variable extension of the Moore neighborhood. The standard Moore neighborhood is shown in Figure 4.6 for two different nodes. For an arbitrary node N this neighborhood system is defined by a three times three square of nodes with N in the center of it. All nodes of this square are then the neighbors of N. The first order GMN uses a 5×5 square instead of a 3×3 one, the second order GMN then enlarges this to a 7×7 square and so forth (see Figure 4.5). This neighborhood system has the advantage that it ensures the homogeneity of the MRF since it is symmetrical in all axes and can at the same time easily be changed in size. Homogeneity is important because it ensures that all pixels have a neighborhood which is structured in exactly the same way. This regularity makes the computation and implementation much easier and can be guaranteed in this case because of the uniform structure of the image and neighborhood system, only the pixels close to a border are an exception and must be treated carefully. Furthermore, the variability of the size makes the GMN very flexible, but the number of cliques will increase drastically with the order of the GMN. For example, the second order GMN has already 477.439 cliques for each pixel which makes a computation and storage of all messages infeasible.

So far only the modeling of the spatial relationships of the pixels was discussed. To generate a

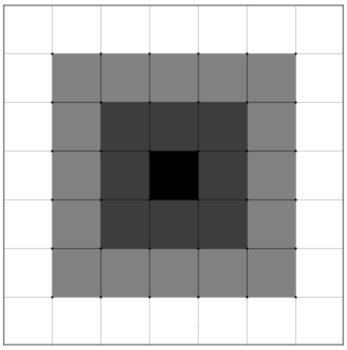


Figure 4.5: This figure shows three different neighborhoods for the black pixel in the center. In the standard Moore neighborhood only the dark gray pixel are neighbors. The first order GMN extends this by adding all the bright gray pixels and the second order GMN also includes the outer white pixels as neighbors.

good segmentation, the information given by the actual image also has to be included into the model. In the proposed method this will be a value generated by the background subtraction method denoting the probability of the pixel being in the background. Nonetheless, this could also be the color value of the pixel, a value given by an edge detector or any other data derived from the image depending on the task. To include this knowledge into the model a second node with a fixed/known state is created for every pixel. This new node is called an evidence node because it represents given information in the model. The other nodes are called hidden since they indicate an unknown state of the system which shall be deduced. In this case, the unknown state that shall be deduced is whether the pixels are in the fore- or background of the scene. Every evidence node will influence only his corresponding hidden node in a way that it will more likely attain the state favored by the given data. That means, if the background subtraction classified pixel v as foreground then the evidence node of v will have a constant influence on the corresponding hidden node to classify the pixel as foreground. However, if, for example, all the neighboring pixels/nodes are classified as background then they will also influence the hidden node of v with a favor for the classification as background and will probably overrule the evidence node of v and thereby correct an assumedly false classification of the background subtraction. A small example of a complete MRF model with a Moore neighborhood can be seen in Figure 4.6.

Now, that the model is fully defined, the input has to be specified. As input the MRF needs in principle two values for each pixel, one should indicate the probability of the pixel for being foreground and the other the probability of it being background. This data is necessary for the evidence nodes so that they can represent the result of the background subtraction algorithm. Here, just one value $p_{BS}(\bar{v})$ is used, which is the probability that the pixel at location $\bar{v}$ belongs to the background. As this value is already normalized the other probability can be simply set to $1 - p_{BS}(\bar{v})$. To get a floating-point probability instead of a binary value (which was computed in the previous Section 4.1) all three channels are used separately in the following way

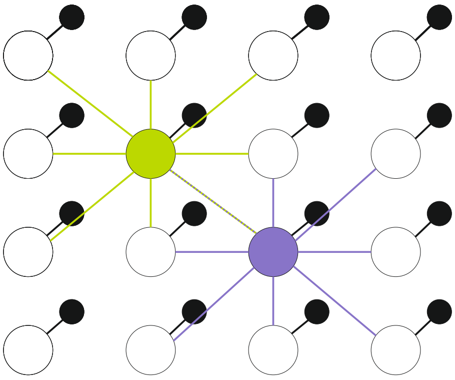


Figure 4.6: The evidence nodes are drawn as small black filled circles and each is connected with one edge to the corresponding hidden node. For the green and purple hidden nodes the edges to the neighboring nodes are also drawn.

$$p_{BS}(\bar{v},c) = \frac{e^{-(I(\bar{v},c)-M^p(\mu,\bar{v},c))^2}}{\max(\beta \cdot M^p(\sigma,\bar{v},c),\gamma)}, \tag{4.5}$$

$$\tilde{p}_{BS}(\bar{v},c) = \min(p_{BS}(\bar{v},c),\tfrac{1}{3}), \tag{4.6}$$

$$p_{BS}(\bar{v}) = \tilde{p}_{BS}(\bar{v},R) + \tilde{p}_{BS}(\bar{v},B) + \tilde{p}_{BS}(\bar{x},G). \tag{4.7}$$

By modifying and rearranging the equation 4.3 the probability $p_{BS}(\bar{v},c)$ can be obtained in equation 4.5. Because the channels are handled separately the influence of each channel has to be limited so that, for example, a big change in the brightness alone cannot produce a 100% probability of being foreground without any changes in the color. Therefore, the impact of each channel is limited to one-third in equation 4.6 so that a change in at least two channels is necessary to get a probability of over 50%. In the last equation 4.7 the overall probability $p_{BS}(\bar{v})$ of being foreground for the pixel $I(\bar{v})$ is computed by adding together the results for the different channels, since each of them is limited to one-third the overall probability cannot exceed 100%.

Furthermore, a cost function has to be defined on the model to measure how good a segmentation matches the MRF model. Therefore, a function $D_l(s)$ is needed with the variable s that describes a specific state of the evidence node l. For each possible state of s (here foreground or background) the function represents how good this matches the data delivered from the background subtraction. In this specific case the function is defined as

$$D_l(s) = \begin{cases} p_{BS}(\bar{v}), & s = \text{foreground} \\ 1 - p_{BS}(\bar{v}), & s = \text{backgroundround} \end{cases} \tag{4.8}$$

where $I(\bar{v})$ is the pixel that corresponds to the node l. There is a one-to-one correlation between the pixels and nodes, for every pixel $I(\bar{v})$ there exists one node l that models this pixel and which consists of the hidden node l and the evidence node l. Additionally, a second set of functions is necessary for the cliques between the hidden nodes. They are named C_k, the k indicating the size of the clique. These functions should represent the spatial relationship between the pixels and hence there can be different functions for all possible clique sizes and spatial arrangements. However, in this case the beneficial homogenous structure of the MRF can be exploited. Since all hidden nodes have the same neighborhood and a coherent segmentation shall be achieved over the whole image, the functions C_k can be identical for all nodes. A small exception are the boundaries, there the neighborhood is smaller and consequently the number of cliques decreases.

Even though almost all nodes have the same neighborhood the model still has an extreme flexibility as each of the cliques could potentially have a unique cost function. For example the four cliques of size three in Figure 4.4 ($\{2,4,5\}$,$\{2,6,5\}$,$\{8,6,5\}$ and $\{8,4,5\}$) could have each a special cost function, e.g. one that favors a configurations in which two nodes are foreground and one node is background for the first clique, a favor for all background pixels in the second clique and so on. To use this flexibility would, of course, complicate the approximation of the MAP extremely and is in most realistic scenarios not necessary. In the small example that was just given it does not make sense to define a function that favors states in which two pixels are foreground and one background since the aim is to have spatial coherent segmentations. However, if it was possible to go to very large neighborhoods the shape of a clique could be important for the expected smoothness (maybe by learning prior shape features) and their different cost functions could provide a benefit, but as the computation of the MAP is already very costly for the smallest possible neighborhoods this is only a theoretical possibility at the moment.

The aim is rather to reduce the computational burden and therefore very simple energy functions are used for the cliques. In a spatially coherent segmentation almost all cliques will have only background or only foreground nodes and hence the function should only favor these states for all cliques, independent of their size or location in the image. This can be achieved when the function C_k is returning 0 energy for the case that all nodes have the same classification and a static value (larger than zero) for all other cases in which foreground and background pixels are mixed. If $k = 4$ the energy function for the clique could look like this

$$C_4(s_1,s_2,s_3,s_4) = \begin{cases} p(s_1)\cdot p(s_2)\cdot p(s_3)\cdot p(s_4), & \text{if} \quad s_1 = s_2 = s_3 = s_4 \\ 0, & \text{elsewise} \end{cases} \tag{4.9}$$

where the variables s_1 to s_4 indicate the states of the four nodes contained in the clique and $p(s_1)$ is the probability that the corresponding pixel has the state s_1. For the probabilities the values derived from the background subtraction are taken. This energy function also has the great advantage that now only a differentiation between two states of the whole clique is necessary instead of 2^4 states.

If these cost functions are given, an evaluation of different segmentations is possible and thereby the best segmentation could be computed. For example, for a fully defined pairwise MRF model and the necessary input data (t_l is the data for the evidence node l, in this case, these are the probabilities from the background subtraction) the probability that a specific configuration of states occurs can be now computed with

$$p(\{s_l : l \in I\} \mid \{t_l : l \in I\}) = \frac{1}{Z} \prod_{l \in I} D(s_l) \prod_{r \in I} C_2(s_l, s_r). \tag{4.10}$$

Here I is the set of all nodes/pixels and the function $C_2(s_l, s_r)$ is set to one when the hidden nodes l and r are not connected. The variable Z is a normalizing constant that ensures that all probabilities add up to one. The k in C_k can only be two in this case since it is a pairwise MRF. Since the product is over all pixels of the image it is clear that the calculation of the probability is very cumbersome and time-consuming. If the MRF is not pairwise the calculation becomes even more difficult with

$$p(\{s_l : l \in I\} \mid \{t_l : l \in I\}) = \frac{1}{Z} \prod_{l \in I} D(s_l) \sum_{k=2}^{n} \prod_{l_1,\ldots,l_k \in I} C_k(s_{l_1}, \ldots, s_{l_k}). \tag{4.11}$$

The number of possible states (different foreground-background segmentations) of such a system is $2^{\#pixels}$ which is an extremely large number and to calculate the probabilities of all of them is not possible.

For this reason, an optimization algorithm is necessary that can at least estimate an optimal solution. In this case, Belief Propagation was chosen because its message passing approach suits the concept of the MRF model quite well. For the optimization, the model is converted to a factor graph first so that a loopy max-product Belief Propagation can be applied more easily in which the cost functions were incorporated into the messages. The details of this optimization technique are described in the following subsection.

Belief Propagation

There is a great quantity of literature about Belief Propagation (BP) in general and its specific implementation for different usages specifically. Two works can be recommended, which were also used for creating this paragraph about BP, a good general overview of the topic and different applications that are possible with BP is given by [YFW03] and a detailed description of an implementation for computer vision purposes can be found in [FH04].

BP is a local message passing algorithm which is used to solve inference problems. These are problems in which there exist several entities and they influence each other. Therefore, a change in one of the entities can have global effects which make it impossible to divide the problem into smaller sub problems which are easier to solve. This makes finding the optimal solution very complex and time-consuming. As inference problems come up in various scientific disciplines, e.g. computer vision, artificial intelligence (AI) or statistical physics, Belief Propagation, as an efficient way to solve them, was rediscovered several times in slightly different ways. Examples are the Kalman filter, the transfer-matrix approach in physics, the Viterbi algorithm or the sum-product algorithm [KFL01], which are all special cases of BP.

For acyclic networks like the Bayesian network in Figure 4.7 it is mathematical proven that BP will give the exact solution after a finite number of iterations. Solution here means that it returns the state of the nodes A to G which has the highest probability of all possible combinations. For cyclic graphs, there is no certainty to find the best overall solution anymore because the influence exerted by the nodes goes along these cyclic paths and cannot be modeled completely

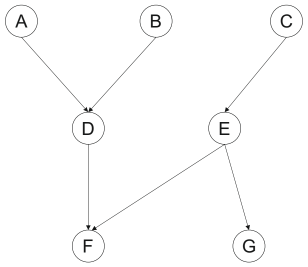


Figure 4.7: Small acyclic Bayesian network. Each node can attain a finite number of states and with BP the most probable combination of states can be computed exactly because of the acyclic nature of the network.

by a finite number of iterations. However, these cyclic networks are the most interesting ones for the computer vision community as images are usually modeled as Markov Random Fields, which are cyclic. Although BP cannot compute the exact solution it can still give satisfactory approximations after a small number of iterations and is, therefore, a useful tool.

An example for a pairwise MRF of an image can be seen on the left side of Figure 4.8. Pairwise means that the largest cliques in the MRF have the size two. The gray nodes represent evidence nodes which contain given information from the image and the hidden nodes (green) can attain several states (e.g. foreground or background). As opposed to the Bayesian network the MRF is an undirected graphical model and therefore always contains circles.

To approximate the most probable combination of states (MAP) of the hidden nodes the graph is first transformed into a factor graph (right side of Figure 4.8). This is not a necessary step but always possible for an MRF and it simplifies the description and implementation of the BP algorithm. A factor graph is a bipartite graph which is used to represent a factorization of a function, in this context of the probability distribution of the MRF. It consists of *vertices* (green circles), *factor vertices* (blue squares) and edges. The factor vertices represent functions and the normal vertices the variables. An edge can only be between a vertex and a factor vertex, because only a variable can be the input for a function and only a function can create a new variable, therefore the graph is always bipartite.

In this case, the factor vertices are neighborhood functions that collect the data from all neighboring vertices and unify them into one message. The normal vertices include the input data (from the background subtraction) to these messages and send them to the next factor vertex. The next step, after the conversion to a factor graph, is to actually compute the messages. For this let f_i be the factor vertex for the node i and n_k the vertex for node k. The corresponding node/pixel k can attain different states s_k and for each of these states the probabilities should be computed. Therefore, the messages from the factor vertices f to the vertices n (for a specific state s) are defined by

$$Q_{f_i,n_k}(s_k) = \sum_{s_z} C_2(s_k, s_z) \cdot Q_{n_z,f_i} \tag{4.12}$$

for a node k in a pairwise MRF, or

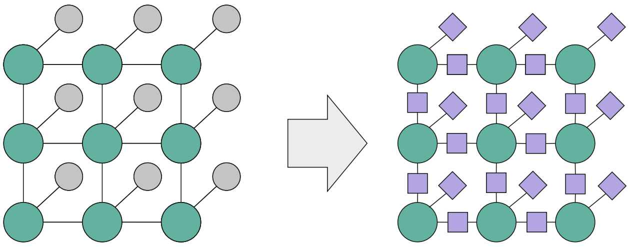


Figure 4.8: On the left a MRF of an image with a four-connected neighborhood and on the right side the same model as a factor graph.

$$Q_{f_i,n_k}(s_k) = \sum_{v=1}^{r} \sum_{s_{z_1},\dots,s_{z_v}} C_{v+1}(s_k, s_{z_1}, \dots, s_{z_v}) \cdot Q_{n_{z_1},f_i} \cdot \dots \cdot Q_{n_{z_v},f_i} \tag{4.13}$$

for a general MRF. The sum is over all possible states the other vertices in the clique can attain and the parameter r describes the maximum size of the cliques. The messages from vertices to factor vertices are defined as

$$Q_{n_k,f_i}(s_k) = D_k(s_k) \prod_{f_z \in N(n_k) \setminus f_i} Q_{f_z,n_k}(s_k). \tag{4.14}$$

The set $N(n_k)$ consists of all factor vertices which are connected to n_k. For the initialization all messages from factor vertices to normal vertices are set to one ($Q_{f_i,n_k} = 1$) so that $Q_{n_k,f_i} = D_k(s_k)$ at first. With this, the message passing can start but before sending the messages they always have to be normalized so that all messages from one vertex to another vertex sum up to one. This is important because there is one message for each possible state (the equations 4.12 and 4.14 must be computed for all possible states of s_k) and after the normalization, these messages can be interpreted as a probability for that state.

The BP will always converge to a minimum (although it might be a local one) but it is not advisable to let it run until it has found the exact minimum. This would be the case when a steady state is found, which would take many iterations. However, after very few iterations the changes of the probability for each state are usually already very minimal and do not influence the final decision (since it is irrelevant if the final foreground probability is 80.13% or 80.15%). Therefore, a threshold should be defined, either a maximum number of iterations or a minimum change in the probabilities or messages. After the threshold is met, the marginal probability that node/pixel k has state s_k can be expressed as

$$p(s_k) \propto \prod_{f_i \in N(n_k)} Q_{f_i,n_k}, \tag{4.15}$$

which gives an approximation of the MAP.

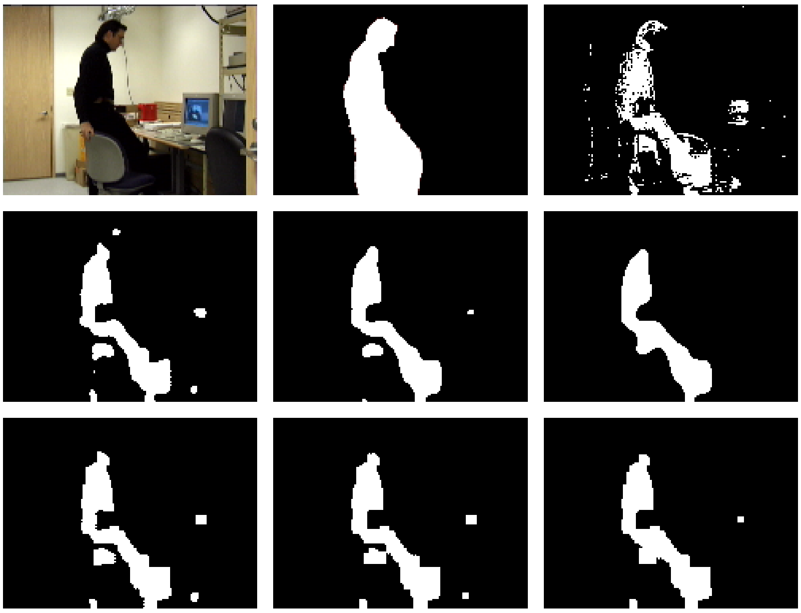


Figure 4.9: The first row shows the original picture, the ground truth data and the result after the background subtraction. In the second row, the results after the smallest clique Belief Propagation algorithm with an increasing neighborhood (Moore neighborhood, first and second order GMN) are depicted. The last row shows the same for the maximal clique Belief Propagation.

The reason why almost only pairwise MRFs have been used so far is that the formula 4.13 is very complex and time-consuming to compute, especially since it has to be computed for all possible states in which the neighbors $s_{z_1}, \dots, s_{z_v}$ can be. However, in this special case, the complexity can be reduced drastically because only the probability that all neighbors have the same state as node k is important, all other possible states can be condensed into an unwanted state with a mixture of foreground and background. Therefore, only two messages have to be sent from the factor vertices to the normal vertices, independently from the size of the neighborhood, which are

$$Q_{f_i,n_k}(s_k) = C_{v+1}(s_k, s_{z_1} = s_j, \dots, s_{z_v} = s_k) \prod_{t=1}^{v} Q_{n_{z_t},f_i} = p_k(s_k) p_{z_1}(s_k) \cdot \dots \cdot p_{z_n}(s_k) \prod_{t=1}^{v} Q_{n_{z_t},f_i} \tag{4.16}$$

for s_k either foreground or background where $p_{z_1}(s_k)$ is the probability that node z_1 has state s_k (compare with Equation 4.9). The messages from the normal vertices to the factor vertices are similarly reduced as they depend on the just discussed messages. This was a first great reduction in complexity of the computation, however, this is still not enough because for larger neighborhoods there are just too many cliques, and even if only two messages for every clique are sent it would still need an excessive amount of memory to just store them.

For example, if a 5×5 GMN should be applied on an HD image (1920×1080 pixels) each pixel would be in 1951 cliques (if the few special cases at the border are ignored) and therefore $2 \cdot 1920 \cdot 1080 \cdot 1951 = 8.091.187.200$ messages need to be computed and stored, which would require over 60GB of memory if every message is one double value. And these are just the messages from the factor to the normal vertices, the same amount is necessary for the other messages. It is therefore obvious that another drastic reduction in the number of messages is necessary. For this, the algorithm is reduced to cliques of the largest (or smallest) size so that the number of messages becomes manageable. The largest and smallest cliques are ideal because they either define the smoothness of details accurately or the smoothness in a larger area. Furthermore, the number of cliques of these sizes is very limited in contrast to other clique sizes, e.g. the 5×5 GMN has only 24 cliques of the smallest size. An example that shows the different effect the choice between the largest or smallest cliques has is depicted in Figure 4.9 for three different neighborhood sizes. Overall the results for the smallest cliques look more natural and this algorithm was therefore chosen in all further applications of the BP approach.

Combining the between Class Variance with Belief Propagation

After using a pixel-wise segmentation method (background subtraction) and combining it with an optimization over the whole image (MRF+BP) the last step is a local method which adapts the segmentation to local edges and corners. In [Ots79] a method is described to segment a picture into two classes by trying to maximize the variance between the two classes. The segmentation is based on thresholding and the variance between the classes is used as an indicator to find the best threshold value. If applied to a grayscale image this will lead to a segmentation in which all dark pixels are in one class and all bright pixels in the other class. This results in a useful segmentation only under very specific circumstances, namely when the objects of interest are always brighter or darker than the background. It is obvious that this assumption cannot be made in most real-life images.

However, in most cases, this assumption will hold if only a small area around a pixel is taken into account. The reason for this is that objects are usually uniform in their appearance at least locally and hence the between Class Variance would be maximized when all pixels of an object are in the same class. To use this to improve the segmentation, a local between Class Variance was coupled with the BP explained earlier to enhance the data taken from the background subtraction in each iteration and thus improve the overall result.

The between Class Variance can only be computed for a given segmentation because then there exist two classes (foreground and background) between which the variance can be computed. For this reason, a segmentation will be computed after every iteration of the Belief Propagation algorithm with equation 4.15. Afterwards, the algorithm will iterate through all pixels of the image and compute the average color of all background pixels (respectively foreground pixels) in an area around the current pixel. The area was chosen to be square sized and the current pixel was excluded from the averaging. Now the background probability p_{BS} will be increased if the color of the current pixel is closer to the background average than to the foreground average and lowered elsewise.

To be more precise, for a given patch of the image let $c_{bg} = (L, C_1, C_2)$ (see Equation 4.4) and c_{fg} be the average color of the background respectively foreground and $I(\bar{v})$ the color of the center pixel of that patch. Then, if

$$\|c_{bg} - I(\bar{v})\|_2 < \|c_{fg} - I(\bar{v})\|_2 \tag{4.17}$$

the between Class Variance will be increased when the current pixel is classified as background and hence the background probability will be increased. If the inequality does not hold a classification as foreground would increase the local between Class Variance and will therefore be favored.

However, the classification of the current pixel itself will not be directly changed according to the between Class Variance but the data obtained from the background subtraction will be altered to reinforce a classification that increases the local between Class Variance. This is done by computing the value $a_{\bar{v}}$

$$a_{\bar{v}} = \eta \cdot (\|c_{fg} - I(\bar{v})\|_2 - \|c_{bg} - I(\bar{v})\|_2) \tag{4.18}$$

which determines how much the probability $p_{BS}(\bar{v})$ of the current pixel will change, η is a parameter to control this influence. The value $p_{BS}(\bar{v})$ was computed in equation 4.7 and represents the probability that a pixel lies in the foreground. It influences the BP according to the energy function $D(\bar{v})$ from equation 4.8 and was a fix value until now. With the addition of the local between Class Variance to the BP this value is made adjustable to incorporate the additional information. The probability $p_{BS}(\bar{v})$ is adjusted in the following way

$$\tilde{p}_{bg}(\bar{v}) = \max(p_{BS}(\bar{v}) + a_{\bar{v}}, 0), \tag{4.19}$$

$$\tilde{p}_{fg}(\bar{v}) = \max((1 - p_{BS}(\bar{v})) - a_{\bar{v}}, 0), \tag{4.20}$$

$$p_{BS}(\bar{x}) = \frac{\tilde{p}_{bg}(\bar{v})}{\tilde{p}_{bg}(\bar{v}) + \tilde{p}_{fg}(\bar{v})}. \tag{4.21}$$

The maximum is required to avoid negative probabilities. This process adapts the segmentations locally to edges but is also computational expensive, especially the calculation of the average values c_{bg} and c_{fg} for every pixel in every iteration of the BP algorithm. To reduce the runtime, integral images (introduced by [VJ04]) are used to efficiently compute these values in a static time, independently from the size of the area over which the average is calculated. For every channel two integral images have to be created, one for all background pixels and one which only adds up the foreground pixels. After calculating these integral images, the actual averages can be obtained by a simple operation consisting only of three additions.

4.2.2 N^{2}Cut

Even though the model was drastically simplified, the approximation of the MAP of the MRF still is costly and not possible in real-time for typical video formats today (e.g. 1920 × 1080). Therefore, in this section, another method is proposed which increases the spatial coherence, so that the segmentations reflect better the smoothness of natural images but uses a different model and minimization technique which is fast enough even for real-time video analysis.

In many single image segmentation approaches the image is seen as a graph where each pixel is one node which is connected to its four neighbors (Moore neighborhood) and then an optimal cut through the graph defines a segmentation into different objects or parts of the images. For these graph cuts, there are many ways to define an optimal cut, the most prominent are the minimization of the cut-value by using the max-flow min-cut theorem [PL10] and the Normalized Cut (NCut) [Car+10; SM00]. The NCut overall gives better results for single images as it not only takes the cut through the graph/image into account but also the interior of the two areas that are divided by the cut. Therefore, NCut will favor cuts that create two areas of similar size and, as far as possible, have a uniform color which usually corresponds to how humans would segment natural images.

However, these algorithms have been used almost exclusively for single images as they do not incorporate any temporal information of a video. Therefore, they cannot detect the moving objects or any changes in a video but will instead segment the objects with the most distinctive edges, independently from their movement in the scene. Nonetheless, they can be used for the segmentation of videos to refine an already existing segmentation by adjusting it to the edges in the individual frames. To do this with the segmentations derived by the GSM background subtraction a graph is built for each frame where the edges between two pixels have a weight according to their difference in the color space. To avoid computational expensive exponentiation of floating point values the *1-Norm* is used for the computation of the weights. In comparison to the Euclidean norm there was no detectable accuracy loss but a significant gain in computation speed with this metric. Therefore, the weight is defined as

$$\omega_{vy} = |r_v - r_y| + |g_v - g_y| + |b_v - b_y|. \tag{4.22}$$

For graph cuts the RBG color space is more appropriate than the color space introduced in equation 4.4 since changes in intensity and color should be treated similarly. Thus, $r_{\bar{x}}$ is the value of the red channel at position $\bar{x}$, $g_{\bar{x}}$ the value at same position in the green channel and so forth. After this the graph is defined with a von Neumann neighborhood and its weights computed. Now, an energy function has to be formulated for this graph so that a cut which is optimal in regard to this energy function can be found. For this, the NCut is a good starting point although it has the drawback that finding the optimal solution is a NP-hard problem [SM00] which makes it necessary to use approximative methods for the optimization (e.g. spectral graph theory). If N is the set of nodes in the graph and FG and BG are two sets which constitute a partition of that graph (that means: $N = FG \cup BG \ \wedge \ FG \cap BG = \emptyset$), then the NCut is defined as follows

$$NCut(FG,BG) = \frac{Cut(FG,BG)}{Assoc(FG)} + \frac{Cut(FG,BG)}{Assoc(BG)}, \tag{4.23}$$

$$Assoc(FG) = \sum_{m \in FG, v \in N} \omega_{mv}, \tag{4.24}$$

$$Cut(FG,BG) = \sum_{m \in FG, v \in BG} \omega_{mv}. \tag{4.25}$$

This partition of the graph into the two sets FG and BG is similar to a segmentation, FG can be interpreted as the set of foreground nodes/pixels and BG as the set of background nodes/pixels and the NCut value is an evaluation of this segmentation. Although the NCut function uses the information better than a minimal cut approach by also taking into account the weights of the edges inside the sets FG and BG, it is not well suited for the evaluation of foreground-background segmentations in videos. One reason for this lies in the simple fact that a 100% background segmentation is not possible as it would result in a division by zero. This division would occur since the association ($Assoc(\cdot)$) of an empty set is an empty sum and hence zero. Therefore, the energy function inherently works with the assumption that there are always foreground objects visible. This assumption is often wrong for videos and will lead to erroneous results.

Another problem is that even segmentations with only a small amount of foreground are heavily penalized by this energy function. If there is only a small amount of foreground (or background) the association in that region ($Assoc(FG)$ in equation 4.24) will attain small values and the corresponding summand in equation 4.23 will become very large. Consequently, the NCut energy function tends to segment the image in roughly the same amount of fore- and background because the overall association ($Assoc(FG) + Assoc(BG)$) is almost constant. The more prominent the edges are in images (that corresponds to the possibility of high $Cut(FG,BG)$ values) the more unbalanced the amount of foreground and background can become, however, less than 20% of either foreground or background is extremely unlikely. In the segmentation of videos, there are often frames with zero or almost zero foreground and without prominent transitions. Therefore, the NCut is not an appropriate energy function for these tasks.

To overcome these problems, a modified NCut is proposed in this work which has no bias for any specific amount of foreground:

$$N^2cut(FG,BG) = \frac{Cut(FG,BG)}{nAssoc(FG)} + \frac{Cut(FG,BG)}{nAssoc(BG)}, \tag{4.26}$$

$$nAssoc(FG) = \frac{Assoc(FG) + 1}{\sum_{m \in FG, v \in N, \exists e_{mv}} 1 + 1}. \tag{4.27}$$

In equation 4.27 the association is normalized by dividing by the number of edges in the set (e_{mv} is the edge between m and v). This eliminates the preference for large sets because now only the average association between nodes/pixels is important regardless of the size of that region. The addition of one to the denominator and numerator of the fraction in equation 4.27 prevents the divisions by zero for empty sets. Therefore, even 100% background or foreground segmentations can be mapped by this function (see Figure 4.14). The normalization of $Cut(FG,BG)$ in the same way is not reasonable. At the moment the length of the cut is important and shorter cuts are preferred since the coincide with smaller cut values. The normalization of $Cut(FG,BG)$ would remove the dependency from the length of the cut which would lead to longer cuts with a high curvature (zig-zagging through the frame). Segmentations obtained from such cuts would not reflect the smoothness of natural images anymore.

Nonetheless, a background-foreground segregation purely based on this energy function would not be meaningful because the minimum for every image would be $A = \emptyset$ and $BG = N$ or $FG = N$ and $BG = \emptyset$, since the cut-value would then be zero and the association one. However, the borders of objects in the image constitute local minima of this energy function. This fact can be exploited by a local optimization method which is initiated with the segmentation provided by the background subtraction. This will adapt the segmentation to the edges of the nearest objects in the image and thereby it makes the results more spatially coherent.

Minimization

The first step in the minimization process is the computation of the N^{2}Cut value for the given segmentation that was computed with the GSM background subtraction. For this, the values $Assoc(FG)$, $Assoc(BG)$ and $Cut(FG,BG)$ as well as the number of edges inside each set have to be calculated. From these values the normalized associations and the N^{2}Cut can be easily derived and subsequently all these values are stored to enable the fast computation of the N^{2}Cut value for a slightly changed segmentation. This will then be used for the local minimization of the energy function by changing the state (foreground/background) of a single pixel. Assuming that the N^{2}Cut is already computed, the swapping of the node t from set FG to set BG can be expressed in the following way

$$Cut(FG\setminus\{t\},BG\cup\{t\}) = Cut(FG,BG) + \sum_{m\in FG\ \wedge\ m\in Neigh(t)} \omega_{mt} - \sum_{v\in BG\ \wedge\ v\in Neigh(t)} \omega_{vt}, \quad (4.28)$$

$$Assoc(FG\setminus\{t\}) = Assoc(FG) - \sum_{m\in BG\ \wedge\ m\in Neigh(t)} \omega_{mt}, \quad (4.29)$$

$$Assoc(BG\cup\{t\}) = Assoc(BG) + \sum_{m\in FG\ \wedge\ m\in Neigh(t)} \omega_{mt}. \quad (4.30)$$

The values $Cut(FG,BG)$, $Assoc(FG)$ and $Assoc(BG)$ are already known and all summations contain at most four summands because the von Neumann neighborhood ($Neigh(t)$) limits the number of neighbors to four. Larger neighborhoods are not common for graph cuts since they are unintuitive and would disproportionate increase the computational costs. Therefore, the updating of the associations and the cut-value has a very low computational overhead. The number of edges in each set can be updated in the same way and with these numbers the value of the new N^{2}Cut($FG\setminus\{t\},BG\cup\{t\}$) can be computed with a few simple operations.

With this approach, an efficient local optimization method is built. The image is scanned for a pixel lying on the boundary between foreground and background and then, by changing the label of this pixel from foreground to background or vice versa, it will be attempted to decrease the N^{2}Cut value. If the swap did indeed decrease the N^{2}Cut a better segmentation has been found and will be used for further optimizations. Otherwise, the new segmentation and all values (Cut, Assoc etc.) are discarded and the old ones restored. After scanning the whole image in this way the entire process is repeated until a steady state is reached. The optimization process is only locally and hence the information from the initialization (the segmentation from the background subtraction) is not lost but just adjusted to the strongest edges in the neighborhood.

This method can be extended by applying it to different scales of the image. By using a downscaled version, the effect of changing the classification of one pixel is increased and the range of the N^{2}Cut is extended, therefore, the impact of the spatial relations on the overall result will be increased. In order to use this effect, a low-resolution version of the image is created by the consolidation of four pixels into one, the new pixel will then have the average color of the four former pixels. Applying this process reduces the number of pixels by about $^3/_4$, only at the borders of the image it is sometimes necessary to consolidate only two or one pixels to a new pixel of the low-resolution image.

The scaling algorithm can be applied several times to create an image pyramid with decreasing resolutions. A corresponding down scaled version of the initial segmentation is needed as well as a starting point for the N^{2}Cut minimization at the lowest level. However, the binary character of the segmentation gives rise to a problem in the scaling process. When exactly two of the four pixels that should be consolidated are labeled as foreground, there is a tie and no sensible decision for the new pixel can be made based on the given data. In this case, it was reasoned that the actual change in the foreground pixels is a better indicator than the static behavior of the two background pixels and therefore the pixel was marked as foreground.

Eventually, two image pyramids exist with the same number of layers (usually two to four were used for the examples in this work). The optimization of the N^{2}Cut will start with the lowest resolution image and the corresponding scaled version of the initial segmentation. The resulting segmentation is scaled up and used as an initial segmentation for the next higher resolution version of the image. Therefore, only the lowest resolution version from the initial segmentation (derived from the background subtraction) is used. This process is repeated until the final segmentation (lowest local N^{2}Cut value) with the original resolution has been computed.

The run time increase caused by this multi resolution approach is at most minor as the amount of pixels decreases rapidly for different scales. Also, since an improved initial segmentation is available for the last stage of the optimization with the highest resolution image (which is by far the most expensive stage) it is very likely that fewer iterations are necessary until a steady state is found and the overall costs could actually decrease. The amount of memory required by the algorithm increases only moderately as well. If the special cases of pixels at the borders are ignored the memory usage is capped at $^4/_3$ of the memory usage of the single resolution version.

This multi resolution approach enables the user to control the influence the spatial relations have on the segmentation by changing the number of layers in the image pyramid. This makes the proposed method more flexible without increasing the run time or memory usage considerably. The main steps of the algorithm are also depicted as pseudo code. Algorithm 1 shows the structure of the minimization. It requires an initial segmentation as well as the original frame and with that two image pyramids are created so that the N^{2}Cut minimization can be performed on each layer, beginning with the lowest resolution layer. Algorithm 2 depicts the actual minimization of a specific layer. It receives an image and segmentation of the same size and first computes the N^{2}Cut value for this segmentation. Then, for each pixel, it is checked if the pixel is located on the border between foreground and background. If this is true, the segmentation is changed, evaluated and kept only if the N^{2}Cut value has been decreased.

Algorithm 1 HierarchicalN2cut

```
Require: img                                          ▷ original image
Require: segmentation                                 ▷ from background subtraction
Require: layers                                       ▷ parameter, int
  imgPy ← createPyramid(img,layers)
  segPy ← createPyramid(segmentation,layers)
  for i:=layers-1 to 0 step -1 do
      segPy[i] ← minimizeN2Cut(imgPy[i],segPy[i])     ▷ N^2Cut Optimization
      if i >0 then
          segPy[i-1] ← upscale(segPy[i])
      end if
  end for
  return segPy[0]
```

Parallelization

To make the approach better scalable a parallelization was implemented. This is often a complicated process and limited by the need for extensive information exchange between the different threads. In this case, however, the nature of the proposed algorithm makes it easy to implement and promises a considerable runtime reduction because the minimization of the N^2Cut is completely independent for each frame and can be executed concurrently. Only the initial segmentation and the image itself have to be sent to the specified thread.

The first step is the parallelization of the GSM background subtraction. Otherwise, the N^2Cut minimizer would be idle most of the time and wait for the initial segmentations because the computation of them takes almost as long as the following N^2Cut minimization. Since the background model is updated with every new frame the background subtraction cannot happen concurrently for different frames. To still get a benefit from the parallelization the image must be split into different parts and then for each of these parts the modeling and subtraction of the background can be done by a different thread. Based on the runtime of the other components and the number of available CPU's (8) the ideal number of threads for the background subtraction was three in this case and therefore the image was split into the three different color channels as they were handled mostly separate anyway. If more CPU's are available for the background subtraction the image can also be split into an arbitrary amount of smaller parts since each pixel is handled separately.

Afterwards, the complete initial segmentation with the data from the three threads and send to another thread designated for the N^2Cut minimization. For the minimization, several threads should be provided (roughly the same amount as for the background subtraction) which successively take and process the incoming tasks as there is no concurrency limitation at this stage. A depiction of this whole process can be seen in Figure 4.10. Altogether, this made combination of GSM and N^2Cut freely scalable with only a minor parallelization overhead. A detailed analysis of the runtime is done in Section 4.3.

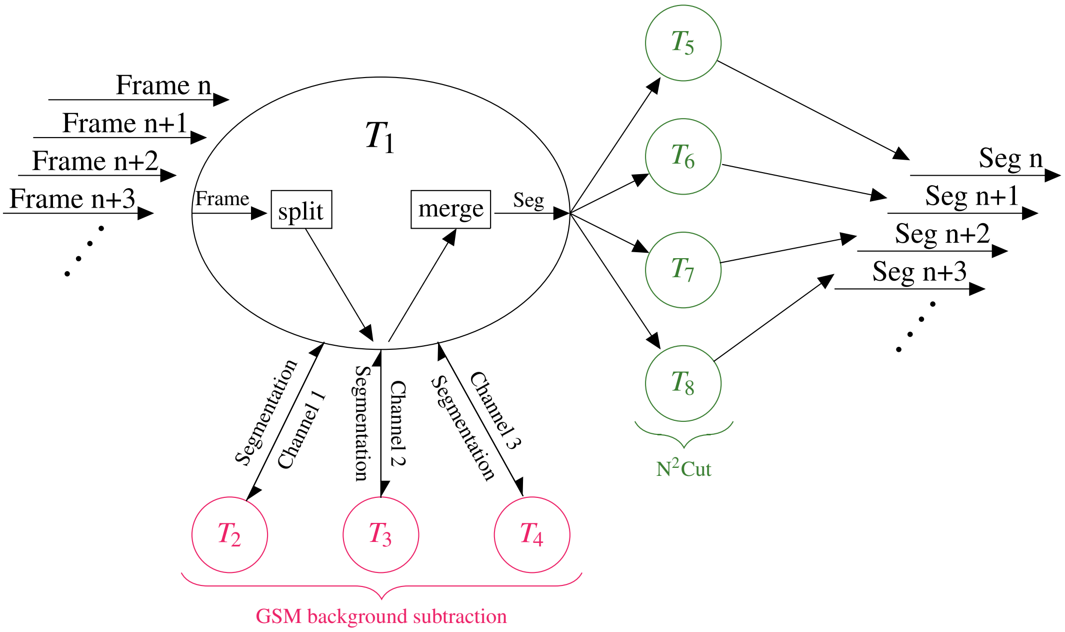


Figure 4.10: Visualization of the parallelization of the N^{2}Cut. On the left, several images are incoming from the video stream which should be segmented. They are first split into their different color channels and then each channel is segmented separately by the GSM background subtraction in the threads two to four. Afterwards, the different segmentations are unified again and send to the threads five to eight (depending on which is available at the moment) which are applying the N^{2}Cut minimization on them and afterwards send out the final segmentation for displaying or further processing. Thread one splits and merges the frames and coordinates the other threads. A time-flow of the parallelization can be seen in Figure 4.11.

Algorithm 2 minimizeN2cut

```
Require: img                                                  ▷ original image
Require: seg                                    ▷ from background subtraction
  repeat ← true                                                     ▷ bolean
  ncutValue ← ncutEvaluate(seg,img)
  while repeat do
      repeat ← false
      for i:=0 to imgWidth step 1 do      ▷ imgWidth = number of columns of the image
          for j:=0 to imgHeight step 1 do
              if IsOnBorder(seg,i,j) then
                  seg2 ← seg                                ▷ save old segmenation
                  ChangeSeg(seg2,i,j)                          ▷ change one pixel
                  if ncutValue>ncutEvaluate(seg2,img) then   ▷ new $N^2$Cut better?
                      ncutValue ← ncutEvaluate(seg2,img)
                      seg ← seg2
                      repeat ← true
                  end if
              end if
          end for
      end for
  end while
  return seg
```

4.2.3 Temporal Trajectories

In the last two sections, the segmentation accuracy was improved by adapting the spatial coherency to that of a natural image (general smoothness, short round edges etc.). Now a different view is taken which considers the lack of reliable data during the subtraction phase from the background model. The problem at this stage is that the model is usually based on hundreds or even thousands of past frames and therefore quite reliable because noise or errors are averaged out, e.g. see the picture in Figure 4.1 where the GSM could build a very accurate model of the background. However, the current frame, which is subtracted from this model, is only one single data point and therefore vulnerable to effects like camera noise. Hence it would be desirable to combine the information from more than one frame to get more robust information and consequently segmentations.

The basic idea is that a pixel that was foreground in the last two or three frames is more likely to be foreground again in the current frame. Similar to a pixel that is surrounded only by background pixels that then becomes more likely to be background itself. This combining of several images can be even extended to future frames so that, if the pixel is labeled as foreground in the next two or three frames, it becomes very likely that it should be labeled as foreground in the current frame as well. The problem with combining information from several frames like this is that the foreground objects move through the scene and therefore the information at a specific pixel location can change rapidly. However, the foreground pixels that were found in one frame do not

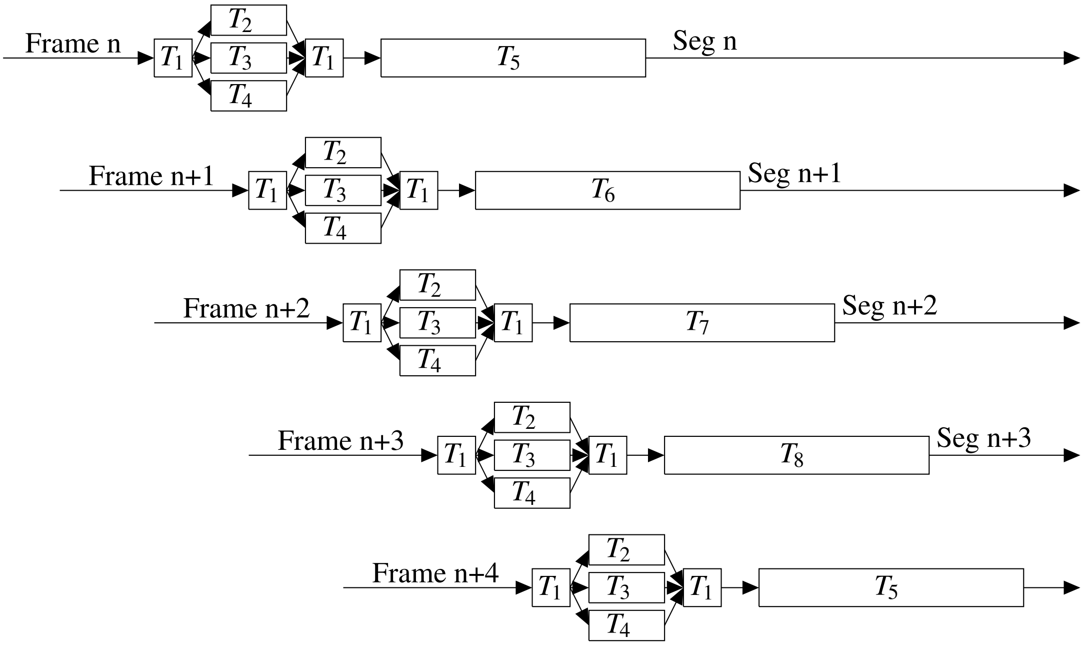


Figure 4.11: Development over time in the parallelized N^2Cut (see Figure 4.11). The time goes from left to right and therefore the length of the rectangles symbolizes the time the thread needs for the computation. T_1 stands for the first available thread, T_2 for the second and so on.

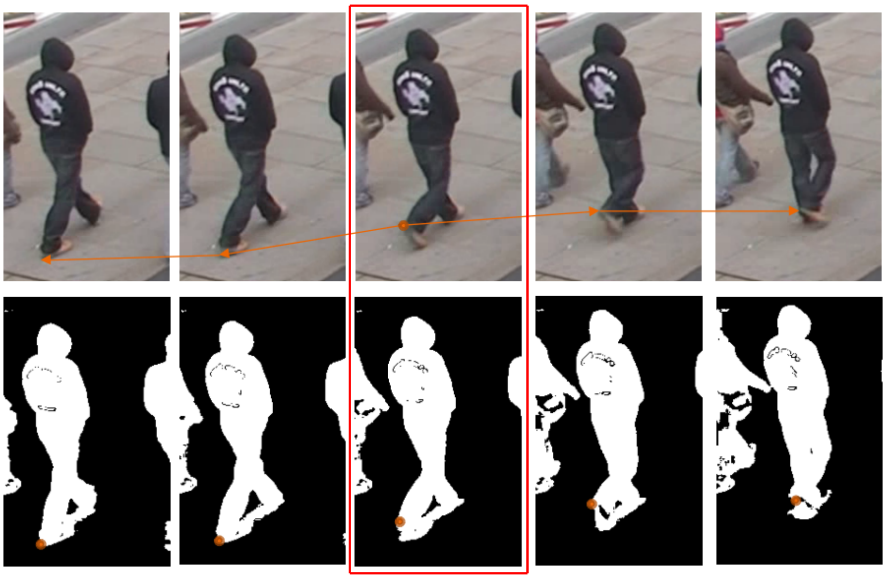


Figure 4.12: For the segmentation of the frame in the middle all pixel locations, e.g. the end of the shoe, are predicted in nearby frames and the segmentation from these frames are then used to enhance the segmentation accuracy in the middle frame.

vanish in the next frame, mostly they are just at a different location. Therefore, the movement of the pixels has to be tracked and their position in the next or previous frame estimated so that several segmentations can be combined. An illustration of this can be seen in Figure 4.12.

Dense Optical Flows

To genuinely track single pixels through the video dense optical flows (DOF) are necessary. In contrast to most optical flows, which only track certain feature from frame to frame, they estimate the movement of every pixel in the image. The output is usually a vector for each pixel which gives the estimated movement in up-down and left-right direction in pixel steps. With this information, a prediction of the position of a specific pixel in nearby frames can be easily obtained. A visualization of the result of a DOF can be seen in Figure 4.13.

In this case, the DOF is computed with the algorithm described in [Far03]. The idea behind this algorithm is to model both images as Polynomial Expansions and then, by solving a system of linear equations, create a displacement vector between these two models. Only quadratic polynomials in the form of

$$f_1(\bar{x}) \sim \bar{x}^T A_1 \bar{x} + \bar{b_1}^T \bar{x} + c_1, \tag{4.31}$$

were used as higher orders would be too computational expensive. The matrix A_1 is symmetric, $\bar{b_1}$ is a vector and c_1 a scalar. These coefficients were estimated with a weighted least square fit so the function will approximate the pixel values in an area around $\bar{x}$. If $f_1(\bar{x})$ now models a

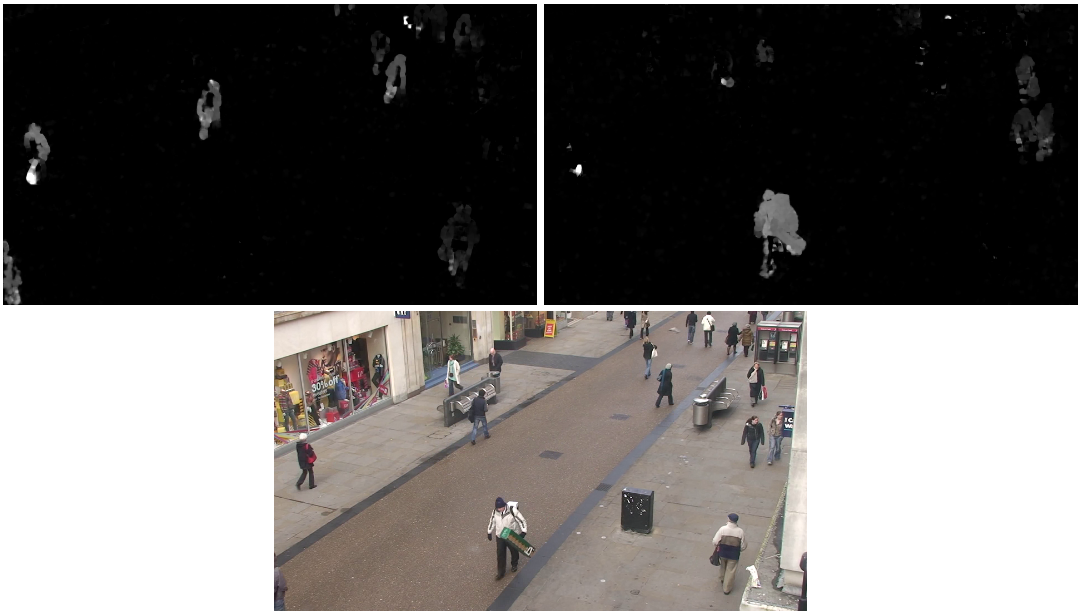


Figure 4.13: In the top row the detected movement with the DOF is translated into intensity values, a brighter pixel signifies a stronger movement at that location. On the left side, this is done for the right-left movement and on the right for the up-down movement. In the bottom row is the original frame from the video.

small area in frame I_1 which is also visible in frame I_2 but with a translation $\bar{d}$, then the function $f_2(\cdot)$ which models the same region in I_2 should be defined as

$$\begin{aligned} f_2(\bar{x}) &= \bar{x}^T A_2 \bar{x} + \bar{b_2}^T \bar{x} + c_2 = f_1(\bar{x} - \bar{d}) \\ &= (\bar{x} - \bar{d})^T A_1 (\bar{x} - \bar{d}) + \bar{b_1}^T (\bar{x} - \bar{\bar{d}}) + c_1 \\ &= \bar{x} A_1 \bar{x}^T + (\bar{b_1} - 2A_1 \bar{d})^T \bar{x} + \bar{d}^T A_1 \bar{d} - \bar{b_1}^T \bar{d} + c_1. \end{aligned} \tag{4.32}$$

Thus, the matrices A_1 and A_2 are identical and $\bar{b_2} = \bar{b_1} - 2A_1\bar{d}$ which leads to $\bar{d} = \frac{1}{2}A_1^{-1}(\bar{b_1} - \bar{b_2})$. Theoretically, the displacement vector can now be easily computed by solving a system of linear equations (which is necessary to get the invers of A), however, the practical computation still offers some problems. For example, the two matrices A_1 and A_2 are not identical in reality due to rounding errors but also because of small changes in the neighborhood of $\bar{x}$ over time. Therefore, a new matrix $A = \frac{1}{2}A_1 + \frac{1}{2}A_2$ has to be introduced which minimizes this problem. For other practical issues the reader is referred to [Far03].

This method has two important parameters, the first one is the size of the neighborhood that should be modeled by one quadratic equation. On one hand large neighborhoods are preferable as they are more robust (e.g. against aperture) but on the other hand, large neighborhoods cannot be modeled accurately anymore by a simple model like a quadratic equation. Experiments showed that for scenarios examined in this work a size of 15 × 15 pixels provided a good compromise between accuracy and robustness, this size was used in all further evaluations. The second parameter is a weight function for the evaluation of how similar two neighborhoods are. Since usually no exact match can be found between two images, the translation vector d which gives the best match for $f_1(x) \approx f_2(\bar{x}+\bar{d})$ needs to be found. For this, the Euclidean norm is used to compare the two neighborhoods but it is sensible to put an emphasize on the pixels close to the center as a match there is more important and reliable. Here the weight function comes into play which gives a different weight to different pixels in the neighborhood based on their distance to the center. For the experiments in this work, a two-dimensional Gaussian distribution with radially decreasing weights is chosen.

An advantage of this method is that it also allows the incorporation of a priori knowledge into the computation. If a prior estimation of the DOF is given, the algorithm only has to compute the displacement vectors from this estimation to the correct solution, which are usually much smaller. This is very beneficial because the calculation of large displacement vectors is computationally very demanding and especially prone to errors. This inclusion of prior knowledge can be used in two ways to improve the DOFs. Since the displacement vectors should be computed for all frames of the video, the optical flow from the last frame can always be used as a good first approximation for the current frame. A second way to use this is a hierarchical approach where the optical flow is calculated first on a low-resolution version of the image and the result of this calculation is then used as a start value for versions with higher resolution. By applying these two methods the optical flows can be computed faster and in a more stable way over the course of a video.

An example of the results of the optical flow can be seen in Figure 4.13. It can clearly be seen that optical flow better represents the smoothness in natural images as a background subtraction method. This comes from the fact that always whole neighborhoods are analyzed and not only single pixels. This alone can be used to improve the segmentations considerably and makes the optical flow an important second cue for the segmentation process. It has to be noted, however, that the computation of DOFs is always a complicated and computationally expensive task, especially if trajectories of pixels over several frames shall be created the errors will cumulate quickly. Therefore, the choice of the DOF algorithm is very important and should be well-conceived. The method based on the Polynomial Expansions was chosen because it provides a high flexibility with the option of using different patch sizes and different weight functions. This is combined with the possibility to use prior knowledge which is especially beneficial if whole videos have to be analyzed and it also aligns the results over several frames which is very important for the extraction of accurate trajectories. In the next step, these DOFs are used to enhance the segmentations derived from the background subtraction.

Using Dense Optical Flows as a Spatial Component

After calculating the DOFs the information from them should be used to support the background modeling and subtraction process. For this, the location of each pixel in nearby frames must be approximated to extract trajectories. For this let

$$DOF^{x}_{m-1,m}(\bar{v}) \quad \text{and} \quad DOF^{y}_{m-1,m}(\bar{v}) \tag{4.33}$$

be the optical flow of pixel $\bar{v}$ from frame $m-1$ to frame m. There are two float values for each pixel as one signifies the lateral and the other the vertical movement of the pixel. Therefore, the pixel that is in frame $m-1$ at location $\bar{v}$ will be at location

$$\begin{aligned}\bar{v}_{new} &= \bar{v} - DOF_{m-1,m}(\bar{v}) = (i,j) - (DOF^{x}_{m-1,m}(\bar{v}), DOF^{y}_{m-1,m}(\bar{v})) \\ &= (i - DOF^{x}_{m-1,m}(\bar{v}), j - DOF^{y}_{m-1,m}(\bar{v}))\end{aligned} \tag{4.34}$$

in frame m, according to the optical flow. The new location for the same pixel in the frames $m-2$, $m+1$ or $m+2$ can be calculated in a similar way. With this method, pathways for single pixels can be computed over several frames, however, the DOFs are never perfectly accurate and since errors accumulate radically over longer pathways, reliable information can only be derived for a small number of past or future frames. In this approach, the pathways for a batch of seven frames ($m-3$ to $m+3$) are computed and used to enhance the foreground probabilities of the m-th frame. This is done by using a Gaussian filter over the probabilities along the pathway centered at the m-th frame with a standard deviation of 0.75. The low standard deviation ensures that the importance of the values along the pathway decreases rapidly in both directions, this corresponds to the rapidly decreasing certainty of the correctness of the pathway. Longer trajectories are possible but the estimation accuracy, as well as the influence on the center pixel, decrease rapidly with the distance and therefore there was no detectable gain by using batches larger than seven frames.

Furthermore, the DOFs can be used as an indicator of movement in the scene themselves. For this purpose, the assumed location of a pixel $\bar{v}$ in the frames $m-1$ and m is taken and the Euclidean distance $d_{m-1,m}(\bar{v})$ computed. The same is done for the assumed locations in the frames m and $m+1$. The larger these distances are, the more certain it is that there is a moving object (foreground object) at this location and consequently the foreground probability derived from the background subtraction for this pixel will be adapted accordingly. If $p(\bar{v})$ is the current foreground probability of pixel $\bar{v}$, then it will be modified to

$$\tilde{p}(\bar{v}) = \frac{2}{3}p(\bar{v}) + \frac{1}{6}\min(\frac{d_{m-1,m}(\bar{v})}{\tau}, 1) + \frac{1}{6}\min(\frac{d_{m,m+1}(\bar{v})}{\tau}, 1). \tag{4.35}$$

The weights of the different components have been determined experimentally, the only hard condition is that they add up to one to ensure that the probability stays in the range of $[0,1]$. The distances are divided by τ which signifies that a movement of τ pixels from one frame to another is taken as a clear indication of a moving object. The value of τ is very dependent on the resolution of the video and therefore was set to $\frac{Width\ of\ Image}{100}$ so that it would be adaptive for different sizes of videos, this means that for an HD video a displacement vector of length ten would be a clear indication of a foreground object that moves.

The last step is a slight spatial smoothing of each probability map separately in which the differences of the value of the central pixel with that of all other pixel values in a 3×3 neighborhood are summed up, weighted and the result is added to the probability value of the central pixel. This eliminates outliers and in general increases the spatial coherency. These three steps – the smoothing over the trajectories, the adding of the optical flow prior and the spatial smoothing – are applied successively on a batch of segmentations which was first derived from the background subtraction. If several batches are necessary for the segmentation of a whole video it is advisable that the individual batches overlap slightly at the beginning and end, since the trajectories cannot be computed for the first and last frames of each batch and therefore their segmentation will be untouched by this algorithm.

Algorithm 3 Temporal coherence with DOFs

```
given: Batchsize k

** computing initial segmentations **
for i=1:k do
    $frames[i] \leftarrow getimage(video)$
    $UpdateBackgroundModel(frames[i])$
    $segs[i] \leftarrow BackgroundSubtraction(frames[i])$
end for

**get DOFs**
for i=1:k-1 do
    $DOF[i] = getDOF(frames[i], frames[i+1])$
end for

** get Trajectories from DOFs **
$Paths \leftarrow getTrajectories(DOF)$

** use $DOF$ and $Paths$ to improve results **
while $TerminationCriteria$ do
    $oldseg \leftarrow segs[k/2]$
    for i=4:k-3 do
        $segs[i] \leftarrow spatialsmoothing(segs[i])$
        $segs[i] \leftarrow AddDOFPrior(segs[i], DOF[i])$
    end for
    for i=4:k-3 do
        $segs[i] \leftarrow PathSmoothing(segs, Paths[i])$
    end for
    $TerminationCriteria \leftarrow CalculateChange(oldseg, segs[k/2])$
end while
```

A common batch size for this algorithm was 100 frames. Larger batch sizes would be slightly better for the quality of the results but also increase the memory demand of the algorithm. The whole algorithm is run several times over each batch, each time all three steps are executed. Several iterations can be useful because the changes in the segmentations in the first step influence the smoothing over the trajectories in the next steps and so on. The quality and smoothness of the probability map increase gradually over the iterations and with this also the trajectories provide better information in each step and thereby improve the segmentations further. This process, of course, needs a termination criterion which stops the iteration. Instead of a fixed number of iterations, the changes in the probability fields from one iteration to the next were measured and if these changes were smaller than a specified threshold the loop would terminate. The change was observed by measuring the mean of all foreground probabilities and calculating its variations over time. To reduce the computational cost, the measurement was constrained to the probability map of only one frame in the batch, the center frame. A summary of the whole segmentation process can be seen in algorithm 3.

4.3 Results

To assess the quality of a segmentation result it is necessary to have ground truth data. This data is usually created by hand by a human expert who marks all areas in a frame that are foreground and therefore should be segmented by the algorithm. The evaluation is then done by a pixel-wise comparison of the computed segmentations with the ground truth data. For the specific assessment of change detection algorithms, there exist several common public datasets on which scientists can compare their methods, here the Wallflower dataset was chosen for the evaluation of the just proposed algorithms since already many other algorithms have been tested there and an extensive evaluation is possible.

Wallflower Dataset

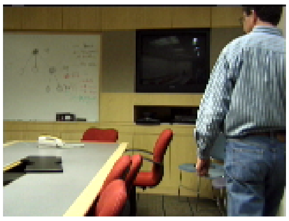
The Wallflower dataset was introduced by Toyama in [Toy+99] and is publicly available under a Microsoft research license[1]. It consists of seven videos which all feature very specific and distinct video segmentation problems. The videos have different length but all share the same resolution of 160×120 pixels. The advantages of this dataset are that each video features one very specific problem of change detection (e.g. a swaying tree in the background or aperture) and that a large dataset of algorithms exists for comparison. The drawback of it is that only one ground truth image per video exists, here a larger ground truth dataset would be desirable to get more universal results.

As an evaluation measure, the total number of false classifications (FP+FN) is chosen because advanced measures like the F1-Score or MCC cannot deal with one of the scenes of this dataset that does not feature any foreground. In this scene, the two values TP and FN are always zero (since there is no foreground) and therefore the F1-Score would either be 0 (the worst possible result) or a division zero by zero. The MCC measure has the same problem and would always result in a division by zero, see equations 2.10 and 2.11. Before the results of the proposed algorithms on this dataset are discussed a short description of each video and the distinct difficulties for the segmentation algorithms in them will be given.

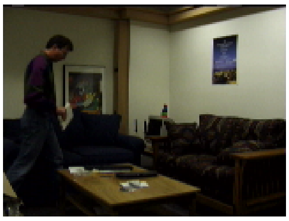
---

[1] http://research.microsoft.com/en-us/um/people/jckrumm/wallflower/testimages.htm

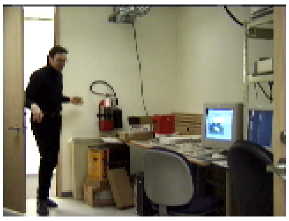
Moving Object (MO):
The stool is moved once by a person and afterwards has to be included into the background again in the new position.
Duration: 1744 frames
Learning Phase: 984 frames



Time of Day (ToD):
Indoor scene in which the light changes slowly from completely dark to normal illumination and back again. The foreground object is a person that enters the room.
Duration: 5889 frames
Learning Phase: 1849 frames



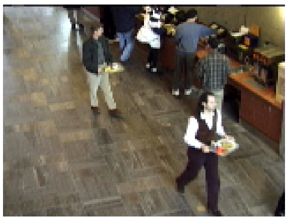
Light Switch (LS):
Another indoor scene with a flickering monitor and rapid light changes by someone turning on and off the light in the room. A person which enters the room and sits in front of the monitor should be detected.
Duration: 2714 frames
Learning Phase: 1865 frames



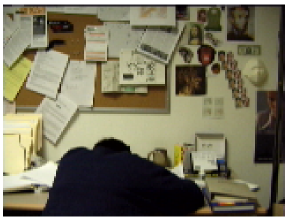
Waving Tree (WT):
Outdoor scene with a tree that sways slowly in the wind but should nonetheless be detected as a background object. The foreground object is again a person that enters the scene and stands in front of the tree.
Duration: 286 frames
Learning Phase: 246 frames

Camouflage (C):
Closeup of a flickering monitor which should be detected as background despite all the rapid changes which appear. Foreground object is a person which walks in front of the monitor and blocks the view.
Duration: 352 frames
Learning Phase: 250 frames

Bootstrap (B):
Indoor scene of a busy canteen with the constant presence of many different foreground objects (people who get their food) which aggravates the learning of an appropriated background model.
Duration: 3054 frames
Learning Phase: 298 frames



Foreground Aperture (FA):
A person sleeping on his desk who should be detected as foreground after he wakes up and starts moving. A special problem is the very homogenous color of the foreground object (completely black) which makes it difficult to detect changes.
Duration: 2112 frames
Learning Phase: 489 frames



Assessment of the proposed Algorithms

The GSM background subtraction was run on each of these videos and the results can be seen in Figure 4.14. There are also shown representations of the trained background model and the results of the GSM with the spatial methods discussed earlier in this chapter. The segmentations based purely on the GSM shows many single false detections, e.g. in the *Waving Tree* video many small areas get falsely classified as foreground because of the slight movements of the tree which complicate the background modeling. All of the spatial methods can eliminate these errors consistently and thereby improve the segmentation quality considerably which shows the necessity of these algorithms.

The pure Belief Propagation makes the segmentations and the objects look very natural/smooth due to the advanced neighborhood model used, but it works only based on the segmentation derived by the GSM background subtraction. Therefore, it does not take into account the actual borders of the images and hence cannot adapt the segmentations to these borders. This can be very well seen in the *Foreground Aperture* video (see Figure 4.14) where a large part of the human silhouette is not segmented although it is one consistently black object. This could be partially corrected with the improved BP by adding Otsu's method to the optimization process. Otsu's method favors segmentations in which the intra-class variance (e.g. the variance between pixels in the foreground) is low and thereby the segmentation adapts partly to the human silhouette because this reduces the intra-class variance. However, if the background, as well as the foreground objects, feature many different colors and brightness values, the intra-class variance is not a good indicator anymore. This can be seen on the *Waving Tree* video, where the segmentation gets worse.

An adaption mainly to the edges of the frame is done by the N^2Cut, which results in an even better segmentation on the *Foreground Aperture* video and additionally shows an improvement on the difficult *Light Switch* video. In both cases the edges between foreground objects and the background scene are very prominent and there the algorithm succeeds in adapting the segmentation to the real object edges. If the transition between foreground and background is not that prominent the algorithm has often problems stopping the expansion of the segmented objects. A good example of this is the *Time of Day* video where there is almost no difference between the background and the foreground object. Hence, there is no edge which would stop the N^{2}Cut from expanding (or shrinking) and as a result, the segmentation of the human degenerates to a simple square. Only in the upper part, the segmentation can adapt to the real object because there is a clear border between the dark head and the white wall. The part at the very bottom, where there are feet that should be detected, vanishes completely because there are again no borders which would stop the N^{2}Cut from growing or shrinking.

The last method, which uses trajectories derived from an optical flow as a spatial-temporal component, does also increase the spatial coherence but could not delete all small false detections in the *Waving Tree* example because they were so frequent that they were tracked over several frames. Likewise, in the *Light Switch* and *Camouflage* videos the constant errors due to the flickering monitor where a problem for this method. However, it represents the exact shape of the objects very well in most cases, e.g. in the *Ligh Switch* or *Time of Day* video, but still could delete most of the small errors.

The N^{2}Cut approach provided the best results with 5064 errors (false negatives + false positives) because it excelled in the *Light Switch* and *Time of Day* videos. This is also an immense improvement over the 9718 errors which were still present when only the GSM algorithm was used. The other methods to increase the spatial coherence could also improve the pure GSM results by a wide margin but not as much as the N^{2}Cut. For example, the smoothing with only the BP could already diminish the errors to 7169 because of all the small false detections due to noise which were removed. With the addition of Otsu's method to the Belief Propagation optimization, it could be further reduced to 6092 by enhancing the segmentation of homogenous objects. With 6078 errors the trajectories derived from optical flow produced similar results. The detailed results of the proposed algorithms are depicted in Table 4.2.

Overall, the proposed Belief Propagation algorithm can increase the segmentation quality immensely and creates natural-looking results. However, it is not good in correcting errors made in the background subtraction phase and computationally very demanding. By combining it with Otsu's method it became better at correcting errors in the provided segmentations, but this further increased the computational load and added additional parameters. The N^{2}Cut, on the other hand, has less computational complexity and is, therefore, usable for real-time applications (see the next paragraph for a detailed analysis of the runtimes). It is also very good at correcting large errors in the provided segmentations since it works at the original images and uses information about the edges in them. By the numbers it provided the best results, but the segmentation often is too edged and do not look natural. The last method, Temporal Trajectories, is in the segmentation quality slightly better than the Belief Propagation but also very computational complex since an optical flow has to be computed. It is very dependent on the computed optical flow and its potential can increase in the future when more accurate or faster optical flow methods become available.

An overview of the results and a comparison with some selected methods is given in the Figure 4.17 and a detailed depiction and evaluation of the results of other methods on the Wallflower dataset can be seen in the Figures 4.16 and 4.15 as well as in the Table 4.2. The results and images used there for the methods *Single Gaussian* [Wre+97], *Enhanced MoG*[SG99], *ICA*

[TL09], *IMMC* [FMB12], *INMF* [BG09], *PCA + LDA* [MTR12], *Tensor Subspace Learning* [Hu+11], *Non-parametric Model* [EHD00] and *Eigenspace Model* [ORP00] were taken from [MTR12]. For the methods *MoG 1* [Ziv04], *MoG 2*[KB02] and *KNN* [ZH06] the corresponding OpenCV implementation was used to create the results and the results for *Normalized Block Correlation*[MOH00] and *Scale-Based Wiener Filter* [Toy+99] were taken from [Toy+99]. For the other methods the results were taken directly from the respective papers (*Enhanced MoG + MRF* [SW06], *PSO* [WS07] and *MoG + IHLS* [Set+06]).

It can be seen that the standard methods cannot handle the difficult situations in the Wallflower dataset very well and had over 25.000 errors (e.g. Single Gaussian or the standard MoG methods from OpenCV). Approaches especially adapted for this dataset naturally could create better results and are typically around 10.000 errors, an example of this is [Set+06] with 9739 errors or the IMMC method from [FMB12] with 11974 errors. The only method that could create even better segmentations than the GSM+N^{2}Cut combination is from Schindler et al. [SW06] who used a special MoG approach which resulted in only 7340 errors and these results could be further enhanced by using Markov Random Field as a spatial method which in the end provided only 3808 errors in total.

This is a surprising result since every other approach that was available for comparison on this dataset generated more than double the amount of errors, even if they also used a MoG algorithm. This advantage probably comes from a careful implementation and the special adaption they made to account for some of the special difficulties, e.g. the quick global change in illumination in the *Light Switch* video. However, there are only hints regarding these modifications in the paper, so a definite answer is not possible.

Runtime

Another important aspect, apart from the pure accuracy of the segmentations, is the runtime of the algorithm. If the algorithm creates great results but takes hours or days for the segmentation process its usefulness is doubtful. The aim for video segmentation methods is often to be able to function in real-time so that they can be used online for the observation of production processes in factories, surveillance of certain areas or similar tasks that require real-time responses. Of course, there are also many applications without a hard real-time constraint, e.g. the analysis of scientific data collected in a mission, but even then a fast algorithm is beneficial.

The modeling of the background in a video and the subsequent subtraction of that model from the current frame is usually unproblematic and real-time capability can be achieved without much effort. Especially, since the whole process is done separately for each pixel and therefore a parallelization is easily possible if necessary, e.g. four processes which each take care of one-quarter of the image. The GSM algorithm has the advantage that it uses only two Gaussian models for each pixel value instead of five or more like in most MoG approaches. This reduces the computation time and memory usage and, furthermore, has the advantage that these values are fixed and not dependent on the complexity of the scene. For complex scenes, more of the available Gaussians in a MoG modeling method are actually used and this increases the memory usage and computation time compared to simpler scenes. The GSM background modeling and subtraction can process around 12 frames per second of a full HD video on a single 3.4 GHz Core. If the resolution is reduced to 960×540 it can handle up to 34 frames per second and a reduction to this (still high) resolution does not lose any information since the algorithm should not detect extremely small objects anyway.

The spatial models used in conjunction with background subtraction are often computationally more intensive and therefore make it difficult to maintain the real-time capability of the approach. One example of this is the proposed MRF model with the BP optimization which takes around 10 seconds per HD frame although the complexity of the model is already drastically reduced. The addition of Otsu's method into the optimization increases the already long runtime per frame only slightly. Other approaches that use the MRF model have similar run times, e.g. [SW06] use a smaller pairwise MRF on the Wallflower dataset and state that they need 14 milliseconds for the creation and optimization of the model. However, this measurement was done on the 160×120 pixel frames from the Wallflower dataset and not on an HD video, therefore, it is not really representative for today's cameras and the usual requirements. If it is assumed (very optimistically) that the runtime increases linearly with the number of pixels, the algorithm would need around 1.5 seconds per frame of an HD video for their MRF model and hence also lose real-time capability.

The N^2Cut was developed with this in mind and had the objective to be fast enough for real-time applications. Due to the sophisticated and efficient optimization process of the N^2Cut approach – which goes over different scales of the frame and only handles the edges of segmented objects – it achieves similar run times per frame than the GSM background subtraction. Due to the parallelization introduced in Section 4.2.2 all four cores (eight threads) of an *i7-4770* processor could be used fully. By doing so, it was possible to segment 20.5 frames of an HD video per second with a pipeline consisting of both, the GSM background subtraction and the N^2Cut. Therefore, the N^2Cut can be used for real-time applications even if the videos have a high resolution and only standard consumer hardware is available.

The last method that was introduced, which uses trajectories obtained by an optical flow to track pixel through the video, has a high memory demand since it first needs to collect a batch of frames in which then the trajectories can be computed. For example, for a batch of 100 HD frames, it is already necessary to have around 7.5 GB of memory since not only the frames have to be stored but also the segmentations and optical flows of all the frames. Also, this batch computation makes it inherently impossible for this approach to deliver quick responses since it needs to collect 100 frames first before it even starts with the calculation of the segmentations. The computational demand is not as high as with the BP optimization but it still takes around 400 seconds for every 100-frame batch, so four seconds per frame overall. This makes this method quite impractical for most applications.

Overall, the combination of the GSM background subtraction with the N^2Cut delivered the best results of the proposed algorithms and was also by far the fastest method (see Table 4.1). None of the many other approaches presented here could achieve a similar accuracy except than [SW06] which could produce an even higher accuracy but is also at least an order of magnitude slower. The advantages of the N^2Cut are especially in the frames with strong edges between foreground and background, since it then tries to adapt the detected foreground objects to the edges of the frame. When the background and foreground objects were diverse and complex the N^2Cut had problems with adapting the segmentations to the corrects edges, see e.g. the *Bootstrap* video. In these cases, the MRF model or the trajectories based on the optical flow could provide better results since they do not base their results on the current frame and the edges

Algorithm	**N^{2}Cut**	**MRF + Otsu**	**Temporal Trajectories**
speed (per frame)	0.15s	10s	4s
accuracy improvement	47.9%	37.3 %	37.5 %
memory usage	0.4GB	1GB	~7.5GB
result looks natural	$--$	$++$	$+$

Table 4.1: The numbers are all for one HD frame and without parallelization. For the temporal trajectories a 100-frame batch is assumed for the memory usage. The accuracy improvement describes how much the GSM result on the wallflower dataset can be improved by adding this method.

inside of it but solely on a foreground probability map or a dense optical flow. Although each of the spatial methods has its own strengths and weaknesses, all of them provided very accurate results and could significantly improve the already good results of the GSM approach. On the difficult Wallflower dataset, all of these combinations could provide improvements over almost all existing algorithms.

4.4 Conclusion

This chapter dealt with the standard change detection for in-air scenarios and focused on the prevalent background modeling and subtraction approach to solve this problem. First, a novel background modeling method was proposed that uses exactly two Gaussians which are differently updated – one partially and one fully – and creates an accurate model of the background by comparing and adapting these two. In comparison to the often-used Mixture of Gaussian approaches, it has the advantage of simplicity – which leads to fewer parameters that are needed for the algorithm – as well as a runtime and memory usage which are low and stable. Nonetheless, it proved to be very accurate on the difficult Wallflower dataset and could compete with other current approaches.

The main part of this chapter, however, discussed the problem of spatial coherency. Since the whole background subtraction approach is completely pixel-wise, it inherently lacks spatial coherency which diminishes the segmentation accuracy. Therefore, three different spatial methods are introduced which can be combined with background modeling and subtraction methods to improve the segmentation results. The first one uses an MRF model to describe the spatial relationships between pixels. However, instead of using a small pairwise model, like it has been done in previous approaches, a higher order MRF was used with a more complex neighborhood. This made it possible to adjust the degree of smoothness by choosing a smaller or larger neighborhood. However, to still be able to solve these complicated models the optimization step had to be simplified drastically. Overall, by applying this method the segmentation accuracy could be increased greatly but run-time increased as well to around ten seconds per HD frame so that real-time applications are not possible anymore. It was not the best spatial method by numbers, but it made the segmentations look most natural since it is a direct model of natural smoothness.

The second proposed approach, N^2cut, is inspired by the NCut image segmentation which was adapted and extended in this chapter for videos. In short, the method takes the segmentation from the background subtraction and adapts them to edges in the current frame. This has the advantage that it not only smooths the segmentations to a certain degree but can genuinely change their shape, e.g. in the *Light Switch* or *Time of Day* videos this led to great improvements. The effectiveness of the N^2cut decreases when there are no strong and clear edges present in the scene, but overall it gave the best results on the Wallflower dataset and was also the fastest method which can be even be run in real-time. Lastly, the temporal trajectories are a method that uses optical flows to join several successive frames of a video. This increases the amount of available data for the background subtraction but also makes the approach very dependent on accurate DOFs since otherwise the trajectories become flawed very quickly, especially at the edges of objects. With respect to accuracy and run-time, it is roughly equivalent to the MRF model but has a greater potential since it can profit from possible improvements in the optical flow computation. It remains to be seen, however, how necessary the whole background subtraction step is at all if at some point very accurate DOFs become available.

Overall, it could be shown that the proposed combinations of the GSM with spatial models are very effective, they could clearly outperform all but one of the previous methods on the Wallflower dataset. Furthermore, the evaluation showed that each of the three approaches to improve spatial coherence has its own strengths and weaknesses in certain scenarios so that they can complement each other very well. These accurate results will now be further used in and adapted for the difficult underwater scenarios so that precise segmentations can be obtained there also.

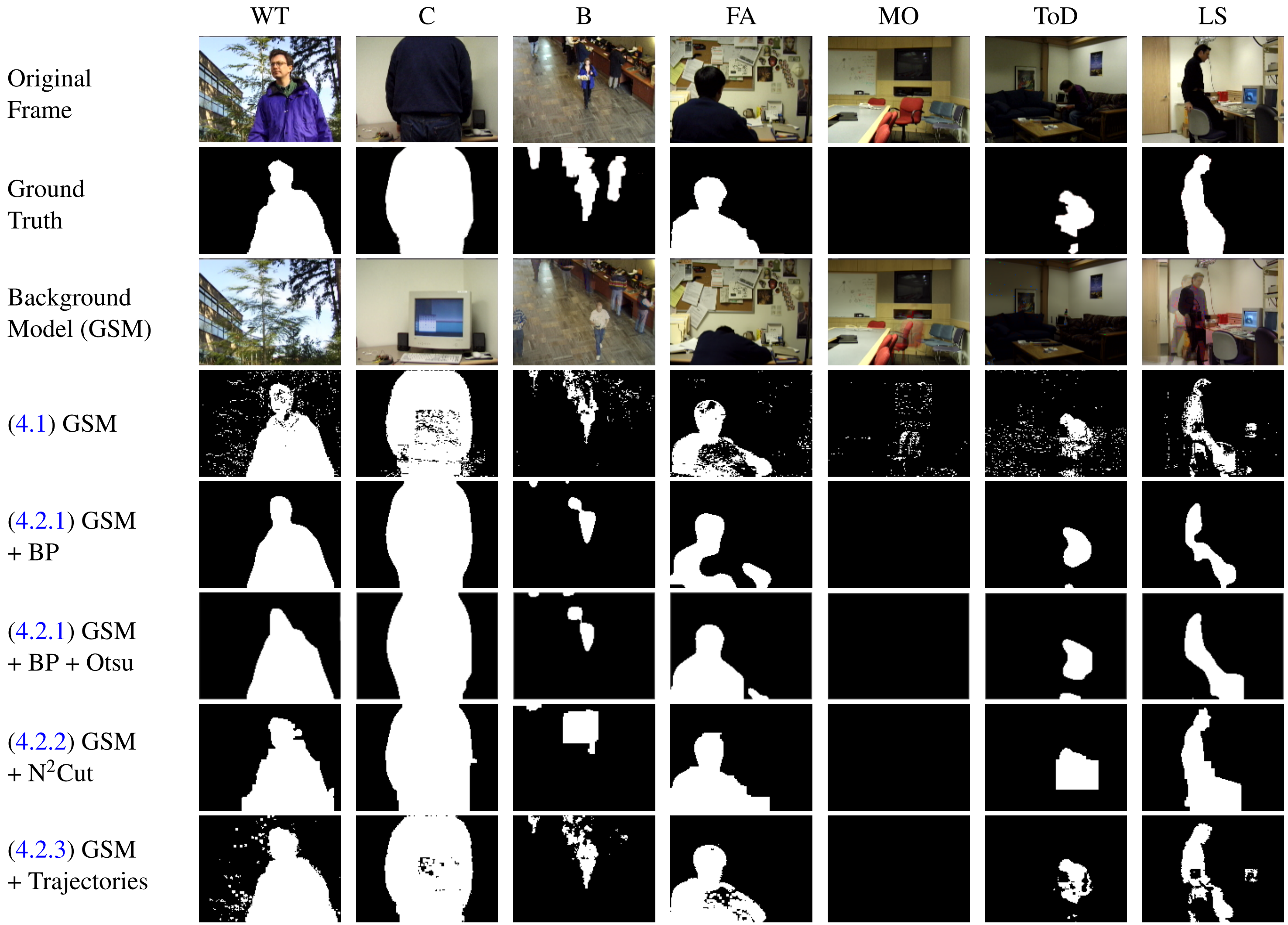


Figure 4.14: Shown are results from the proposed algorithms on the Wallflower dataset. Videos from left to right: Waving Tree (WT), Camouflage (C), Bootstrap (B), Foreground Aperture (FA), Moving Object (MO), Time of Day (ToD), Light Switch (LS).

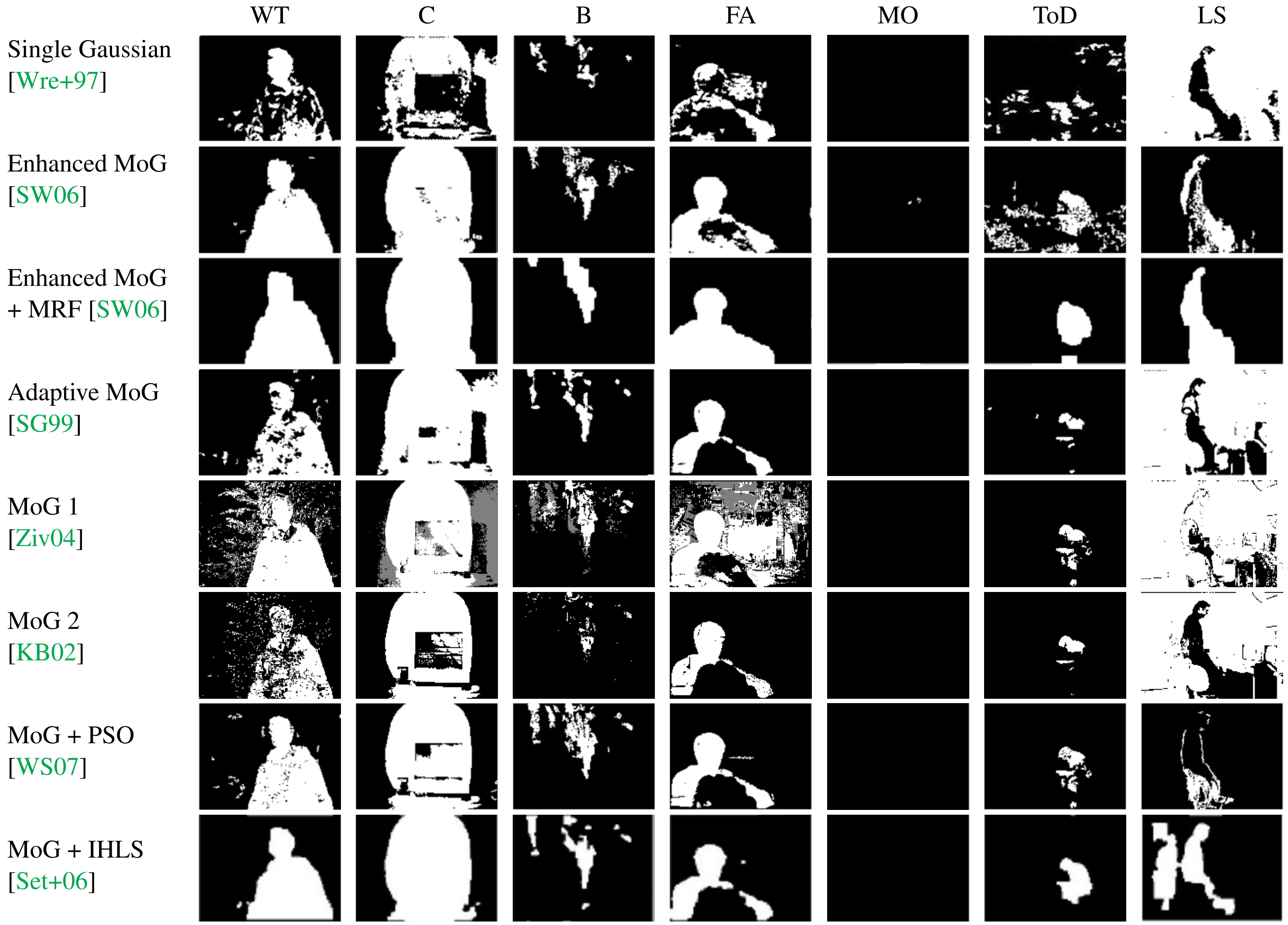


Figure 4.15: Results of other Gaussian models on the Wallflower dataset.

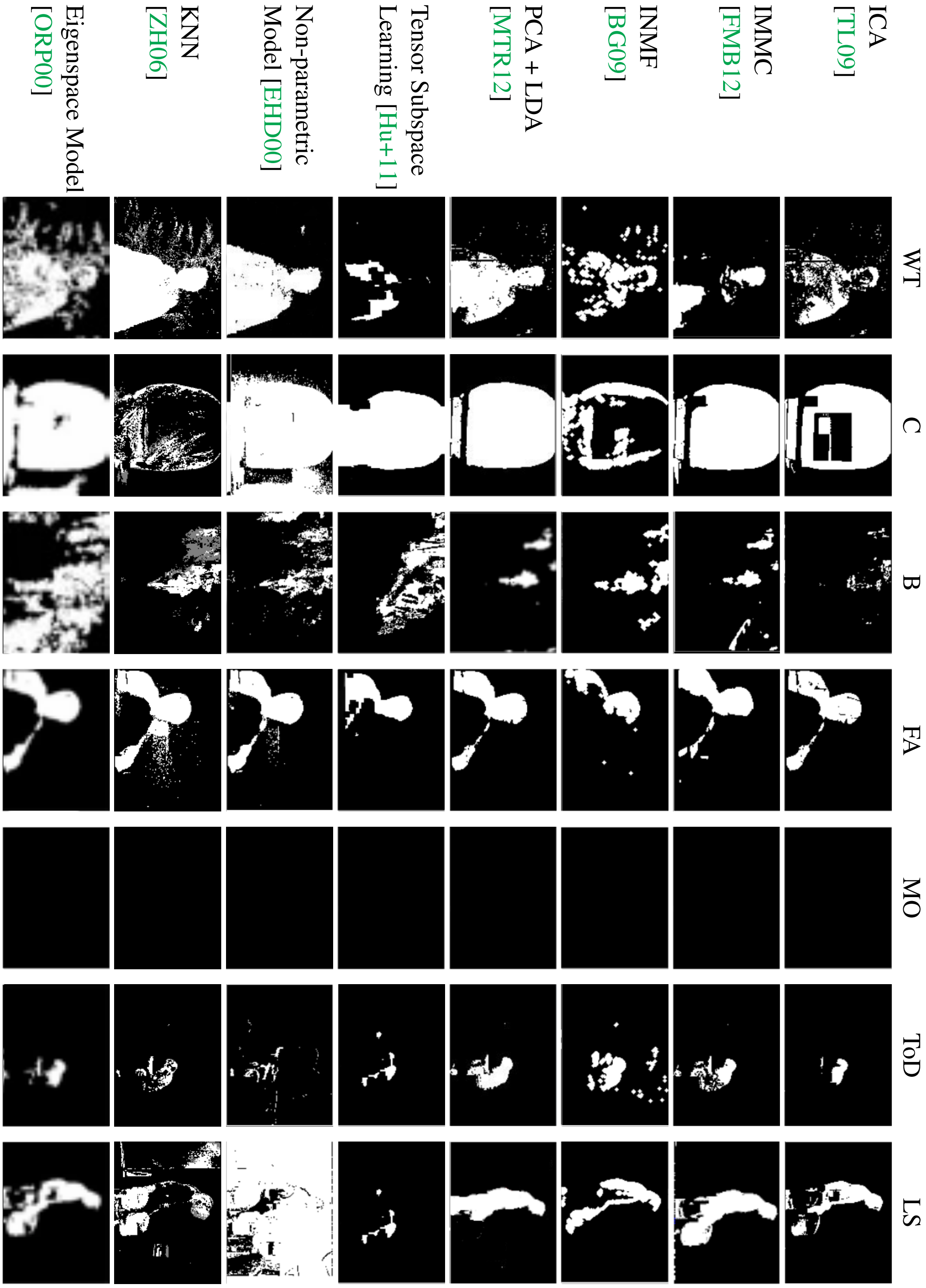


Figure 4.16: Results of non-Gaussian models on the Wallflower dataset.

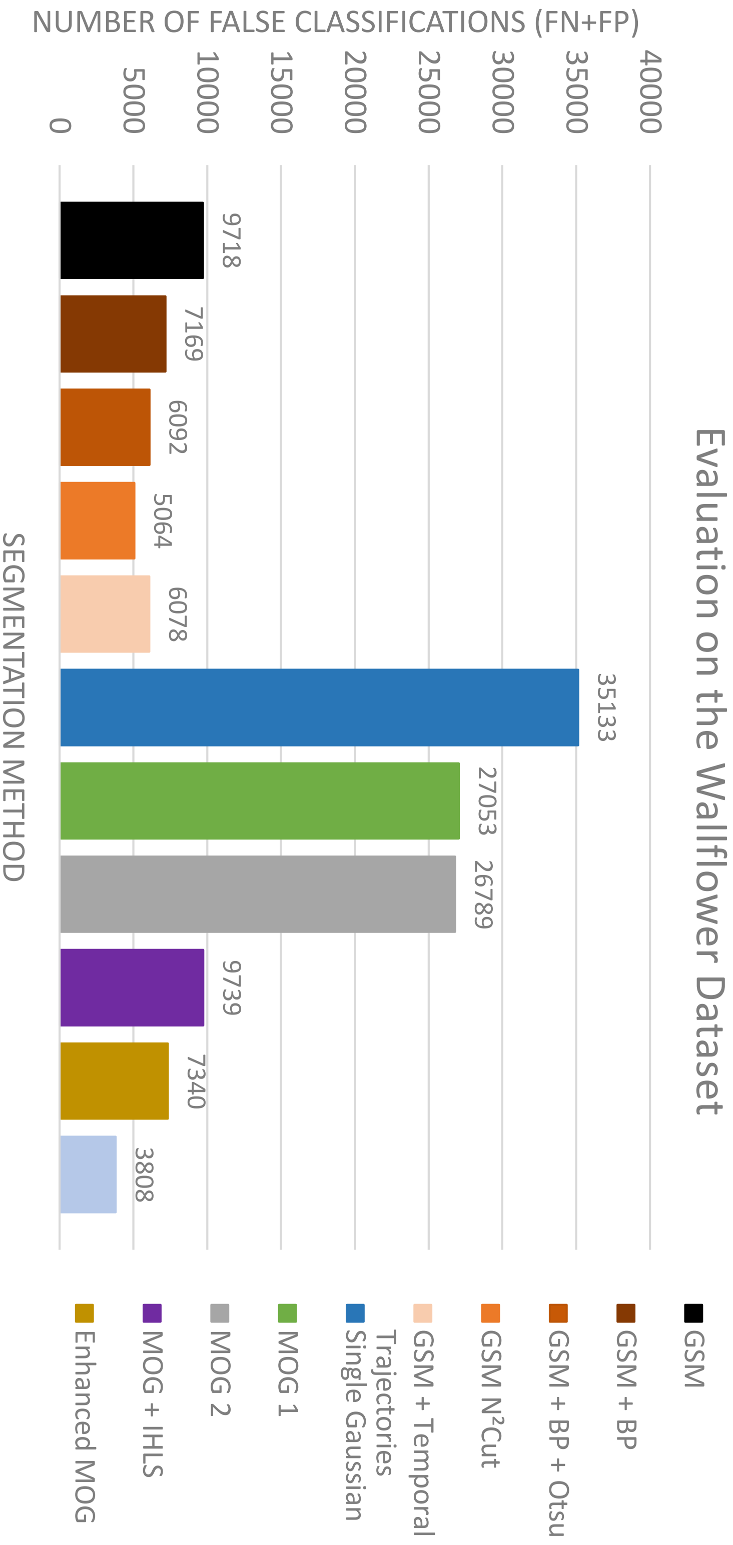


Figure 4.17: Evaluation of different background subtraction methods on the Wallflower dataset. From left to right: GSM (4.1), GSM+BP (4.2.1), GSM+improved BP (4.2.1), GSM+N^{2}cut (4.2.2), GSM+Optical Flow (4.2.3), Single Gaussian [Wre+97], MOG [SG99], MOG (OpenCV) [Ziv04], MOG+HLS Color Space [Set+06], Enhanced MOG [SW06] and Enhanced MOG+MRF [SW06].

Algorithm	**error type**	**Videos:**							**whole set**
		WT	**C**	**B**	**FA**	**MO**	**ToD**	**LS**	
GSM	FN	244	829	1708	1567	0	457	1636	6441
	FP	736	164	166	561	466	641	543	3277
4.1									**9718**
GSM+BP	FN	113	208	2064	1686	0	394	1789	6254
	FP	156	13	3	414	0	40	289	915
4.2.1									**7169**
GSM+ BP+Otsu	FN	174	246	2081	469	0	321	1383	4684
	FP	356	66	0	92	0	199	695	1408
4.2.1									**6092**
GSM+ N^2cut	FN	22	97	1634	70	0	145	36	2004
	FP	671	203	486	317	0	584	799	3060
4.2.2									**5064**
GSM+Optical Flow	FN	2	288	1557	476	0	443	1172	3938
	FP	1131	132	40	405	0	50	382	2140
4.2.3									**6078**
Single Gaussian	FN	3110	4101	2215	3464	0	949	1857	15696
	FP	357	2040	92	1290	0	535	15123	19437
[Wre+97]									**35133**
Adaptive MoG	FN	1323	398	1874	2442	0	1008	1633	8678
	FP	341	3098	217	530	0	20	14169	18375
[SG99]									**27053**
MoG 1	FN	266	229	1429	1463	0	775	784	4946
	FP	1484	192	309	5504	0	4	14350	21843
[Ziv04]									**26789**
MoG 2	FN	1887	2005	2397	2556	0	1043	2163	12051
	FP	411	346	38	462	0	2	13637	14896
[KB02]									**26947**
MoG + IHLS	FN	31	188	1647	2327	0	379	1146	5718
	FP	270	467	333	554	0	99	2298	4021
[Set+06]									**9739**
Enhanced MoG	FN	43	110	1159	1023	0	203	1148	3686
	FP	278	468	143	534	19	1648	564	3654
[SW06]									**7340**
Enhanced MoG + MRF	FN	15	16	1060	34	0	47	204	1376
	FP	311	467	102	604	0	402	546	2432
[SW06]									**3808**
Non-Parametric Model	FN	170	238	1755	2413	0	1298	760	6634
	FP	589	3392	993	624	0	125	14153	19816
[EHD00]									**26450**
KNN	FN	21	7524	1143	2331	0	852	1609	13480
	FP	753	82	117	1133	0	3	1198	3286
[ZH06]									**17242**

Algorithm	**error type**	**Videos:**							**whole set**
		WT	**C**	**B**	**FA**	**MO**	**ToD**	**LS**	
PSO	FN	219	929	1339	2300	0	737	2059	7583
	FP	267	433	546	631	0	9	96	1982
[WS07]									**9565**
Normalized Block Correlation	FN	3323	6103	2638	1172	0	1030	883	15149
	FP	448	567	35	1230	1200	135	2919	6534
[MOH00]									**21683**
Eigenspace Model	FN	1027	350	304	2441	0	879	962	5963
	FP	2057	1548	6129	537	1065	16	362	11714
[ORP00]									**17677**
ICA	FN	3372	3054	2560	2721	0	1199	1557	14463
	FP	148	43	16	428	0	0	210	845
[TL09]									**15308**
INMF	FN	4525	1491	1734	2438	0	1282	2822	14292
	FP	7	114	2080	12	0	159	389	2761
[BG09]									**17053**
PCA + LDA	FN	1426	922	2380	2025	0	443	1107	8303
	FP	958	204	142	576	0	103	49	2032
[MTR12]									**10335**
Tensor Subspace Learning	FN	4525	1491	1734	2438	0	1282	2822	14292
	FP	7	114	2080	12	0	159	389	2761
[Hu+11]									**17053**
IMMC	FN	4106	1167	2175	2320	0	626	711	11105
	FP	5	135	503	201	0	10	15	869
[FMB12]									**11974**
Scale-Based Wiener Filter	FN	877	229	2025	320	0	961	947	5359
	FP	1999	2706	365	649	0	25	375	6119
[Toy+99]									**11478**

Table 4.2: The results of different algorithms for the Wallflower dataset. Each row shows the number of wrongly classified pixels for one approach separated in false positives and false negatives.

5 A Dataset for Underwater Change Detection

The first step to adapt general change detection algorithms like the GSM for the specialties of the underwater case is the creation of a common dataset based underwater videos on which different algorithms can be evaluated and compared fairly against each other. The problem is that change detection in underwater environments has been a widely neglected subject so far, whereas the general change detection case has been researched intensively for several decades and is still used regularly as the foundation of many computer vision pipelines. Hence, no dataset with underwater videos exists at this point although there are many applications in which change detection is useful to automatically observe objects underwater. Some examples are the automatic counting of wild fish [TNL09] or the observation of aquacultures. For this reason, the specialties of change detection in this difficult environment should be investigated thoroughly to adapt and combine existing in-air algorithms for these tasks. To do this, the Underwater Change Detection dataset (**UCD** dataset in the following) is presented in this chapter and afterwards an evaluation of different image enhancement methods and their effect on the segmentation is done.

5.1 Existing Change Detection Datasets

There already exist several datasets for the change detection with a static camera in all kind of in-air scenarios which have been used for evaluation. The most prominent one at the moment is the changedetection.net dataset [Wan+14b; Goy+12] which comprises of dozens of videos and tens of thousands of ground truth images. Many algorithms have been tested on this dataset and a comprehensive evaluation could be made as all kinds of different scenes are reproduced in the data, e.g. difficult weather situations, shaky cameras or thermal videos. However, none of the 53 videos of the dataset depict an underwater scene and the same is true for all other known datasets and hence the unique difficulties of this environment have not been subjected to rigorous testing yet. In the following, some examples of other smaller datasets are given, which often have a special focus and therefore could be useful for specific applications, and afterwards, the UCD dataset is presented.

In [Pra+03] Prati et al. use five videos with several hundreds of ground truth images overall but focused on the difficulties of shadows in their algorithm and their dataset. An artificial dataset was created by Brutzer et al. [BHH11]. It consists of nine videos, each depicting one specific challenge of change detection. As artificial data was used they could easily create many perfect ground truth frames without any errors through human perception. However, the artificial data is far away from a perfect representation of the reality and therefore not a realistic test scenario. To create such an artificial dataset for underwater scenes would be especially difficult because of all the special physical effects the medium water has (refraction, haze, blur...). Another dataset

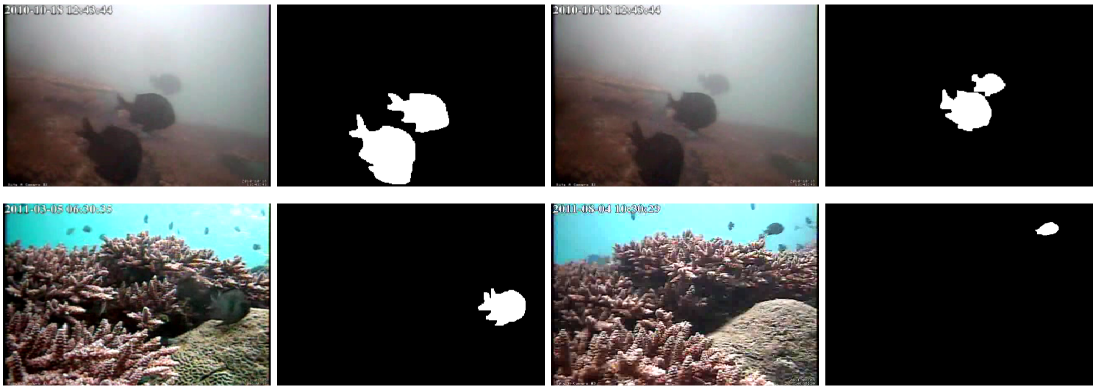


Figure 5.1: Depicted are some examples from the dataset presented in [Kav+14] and the corresponding ground truth data. The upper row shows two frames of the same video, in the first case the uppermost fish was not correctly marked in the ground truth data and in the next frame the lowermost fish. In the bottom row two crowded scenes are depicted and in each the non-expert humans only segmented the most prominent and biggest fish in the scene and left out many smaller fish.

is available from Microsoft [1] and was already used in several papers, e.g. [Cri+06; Yin+07]. It consists of many short videos which are mostly close-up shots of humans with some moving objects or persons in the background. And of course, the Wallflower dataset from [Toy+99] which was already discussed in Section 4.3.

There exists so far one Dataset on change detection which uses underwater scenes, however, it was created more incidentally by evaluating a platform for collaborative web-based video annotation. In [Kav+14] a tool is presented that has the aim to make it easier for computer vision experts to create ground truth data by using larger groups of non-experts and a collaborative platform. From several non-expert annotations for the same frame, the tool would calculate the most likely correct annotation. This method was evaluated on videos from the Fish4Knowledge project[2] in which the exact shape of the fish should be marked. The result was a change detection dataset for fish which consists of six different categories – e.g. crowded, blurred or complex background – which each comprises of several short videos, it can be downloaded at `http://f4k.dieei.unict.it/datasets/bkg_modeling`.

However, the problem with this dataset is that it was not made for the actual comparison of segmentation methods. For example, in some videos, there is no training phase included although the videos are quite long (5 to 10 minutes). But more importantly is the bad quality of the ground truth data. The shapes of the segmented fish are mostly quite accurate but often some fish are not segmented at all. Some examples of this are shown in Figure 5.1. This is probably because the non-expert humans misunderstood the task or were just not motivated enough to mark all small fish. Therefore, this dataset is not suited for a qualitative evaluation of segmentation approaches and was only used so far in one paper for this purpose ([QPL14]).

None of these datasets reflect the special difficulties of underwater scenarios and allow an

---

[1] `http://research.microsoft.com/vision/cambridge/i2i/DSWeb.htm`

[2] `www.fish4knowledge.eu`

extensive and fair comparison on underwater videos, therefore, the here presented dataset will focus exclusively on underwater videos to address this gap. The UCD dataset includes five videos of different scenes with fish as moving foreground objects. For each video, the first 1000 frames are used as a learning phase and are followed by 100 frames for which hand-segmented ground truth images were created. These images are labeled by three colors: white for *background*, black for *foreground* and gray for *border/unsure* where both labels are correct. The third category is especially needed in the underwater context as the blurriness often does not allow for a clear distinction between object and background at the edges. Also, when a fish swims into the scene from the background it is not clear (and quite subjective) when an algorithm should start detecting it. The videos of this dataset were all taken with a GoPro Hero 3 consumer camera and have a resolution of 1920×1080. This camera was chosen as it is cheap, easy to use and very robust and therefore can be utilized in various scenarios without problems. Furthermore, the success of a change detection algorithm should not depend on some special or very expensive hardware, the aim should be a universal approach.

5.2 Difficulties in the Underwater Context

In the following, the different special challenges in the underwater world are explained more detailed and afterwards the five videos of the UCD dataset are presented and it is described how they correlate to these challenges.

- *Blur*

  It reduces the sharpness and contrast of the image and appears in pictures taken in air usually when the camera is out of focus. Another reason for blur in images can be an object which moves fast through the scene (or a too long exposure time) which leads to motion blur. Both of these cases can happen in the underwater context as well and have to be avoided. However, the forward scattering in water is an additional source of blur which cannot be avoided by conventional means. To still get a sharp and clear underwater image, deblurring algorithms are an absolute necessity. (See also Figure 5.2)

- *Haze*

  Small particles in the medium, in which the image is taken, cause backscatter and have an effect which is similar to a sheer veil in front of the scene. It is a general effect and can often be observed in normal images, especially when there is a strong air pollution. In underwater images, the effect is often quite strong even for short distances and can be enhanced by stirred up sand or similar events. (See also Figure 5.2)

- *Color Attenuation*

  Attenuation of light happens in any medium but is neglectable in air for visible light. The absorption in water is several magnitudes stronger and therefore it is mandatory to be close to the object of interest to take an image underwater. Also, the absorption effect for visible light depends on the wavelength and this leads to underwater images with strongly distorted and mitigated colors. In clear and open water blue light is absorbed the fewest because it has the shortest wavelengths and red is absorbed the fastest. That is also the reason why water and objects inside it usually appear blue. However, coastal waters contain phytoplankton which absorbs light with short wavelengths (blue) especially well. Hence the near-shore waters can also appear greenish instead of blue. (See also Figure 5.3)

- *Caustics*

  They are a very special phenomenon of underwater images and videos taken in natural light. Ripples or waves on the water surface refract the light strongly when entering the water and therefore create a very unevenly illuminated scene. Also, the illumination conditions change constantly and rapidly, according to the changes of the ripples or waves on the water surface. (See also Figure 5.2)

- *Marine Snow*

  Small floating particles which are mostly organic and strongly reflect light are called Marine Snow. They are, in fact, small moving objects in the scene but are usually not the interesting part of the image and should therefore not be segmented. Mostly, they are small enough that they are filtered out/ignored during the segmentation process, however, they still corrupt the image and complicate for example the modeling of the static background for background subtraction algorithms. (See also Figure 5.2)

Furthermore, fish, especially the species produced in aquacultures, often have a color similar to the background to hide from predators and/or swim in big swarms. Both circumstances complicate the precise detection of single fish enormously. Also, shadows are disconnected from the objects which cause them. The reason for this is that fish float in the water and do not stand on the ground. This makes it difficult to differentiate between a dark fish and a shadow because the assumption that the shadow is always directly beneath the object itself is not valid anymore. It is possible, for example, that the actual fish swims above the field of view of the camera and only the shadows on the ground are visible in the scene. This can be confusing for a human observer and is a challenging task for computer vision algorithms. In the videos of the UCD dataset, this and most of the other difficulties have been reproduced, some examples can be seen in Figure 5.5.

5.3 Underwater Change Detection dataset

Below is a list of all five UCD videos and a short description of the special difficulties they feature.

- *Caustics*:
  Scene with a complex, partially moving background and light effects produced by caustics. The fish are very similar to the background and even for a human observer hard to detect.

- *Fish Swarm*:
  Fuzzy greenish river water with a swarm of fish constantly present in the scene and thereby complicating the modeling of the background, especially the initialization.
- *two Fish*:
  Basic scene with two fish in a fish tank slowly moving. The only real difficulty is that the fish have a color that is similar to the background.
- *Marine Snow*:
  Features a scene in an aquarium with good visual conditions and many different fish, however, the permanent presence of Marine Snow aggravates the segmentation task.
- *small Aquaculture*:
  An aquaculture used for scientific purposes, easier background and clearer water than in *Fish Swarm* but even more fish.

An example frame from each video can be seen in Figure 5.4. The videos were taken at the Fraunhofer EMB (facility for marine- and biotechnology) in Lübeck, the Ozeaneum in Stralsund and the Natur- und Umweltpark in Güstrow (natural- and environmental park). The whole dataset is available under underwaterchangedtection.eu and everyone is encouraged to evaluate their own algorithms and methods on it to get a better overall picture of change detection underwater.

5.4 Conclusion

In this chapter, the different degradation effects of videos and images in the underwater environment were described. Afterwards, the first change detection dataset was introduced that focuses on the underwater context. It contains all the described image degradation effects, which will allow, for the first time, a conclusive evaluation of change detection in the underwater context. The dataset consists of five videos with each featuring a distinct scenario and having 100 hand-segmented ground truth images for the evaluation. The moving objects are always fish and are usually very similar to each other as well as to the background. This data will be used in the next chapter to evaluate five different change detection methods in combination with different underwater image enhancement techniques.

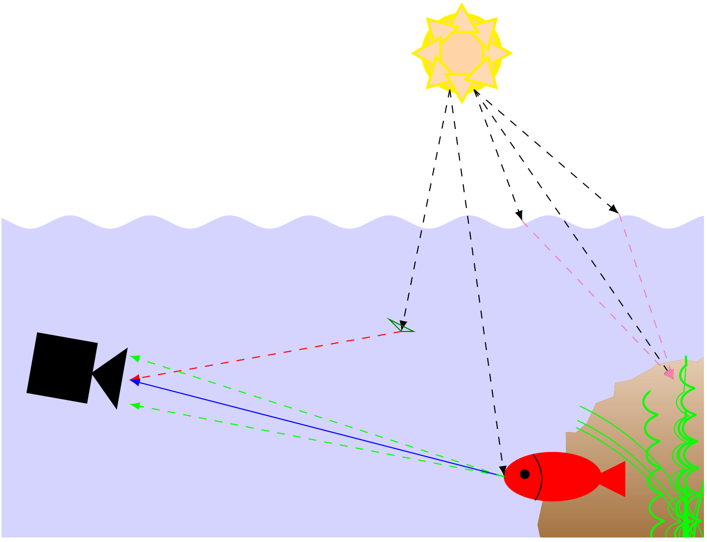


Figure 5.2: In this image several degradations effects of the underwater world can be seen. The leftmost sun ray hits a very small particle in the water and gets thereby reflected towards the camera (haze). If the particle is larger (not microscopic) the effect is similar but then the object itself can be seen instead of just a degradation effect, the object is then called Marine Snow. The sunray in the middle is reflected from the fish towards the camera (blue arrow) but the water diverts part of the light slightly (green arrows) which causes a blur effect. On the right side, the Caustic effect is visualized, the magenta rays got refracted differently at the water surface due to the waves. Now several rays hit one spot on the rock and make this area consequently very bright whereas in other areas almost no rays are incoming.

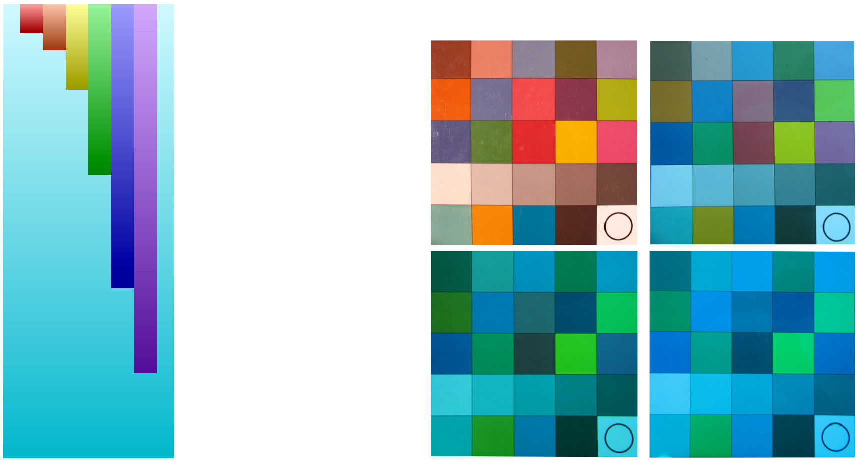


Figure 5.3: On the left side is a Figure that illustrates the different ranges of light based on the color/wavelength. On the right side is a real checkerboard with different colors on it photographed in air (top left), in five-meter-deep water (top right), in 10m depth (bottom left) and 20m depth. Images courtesy of Eik Deistung.

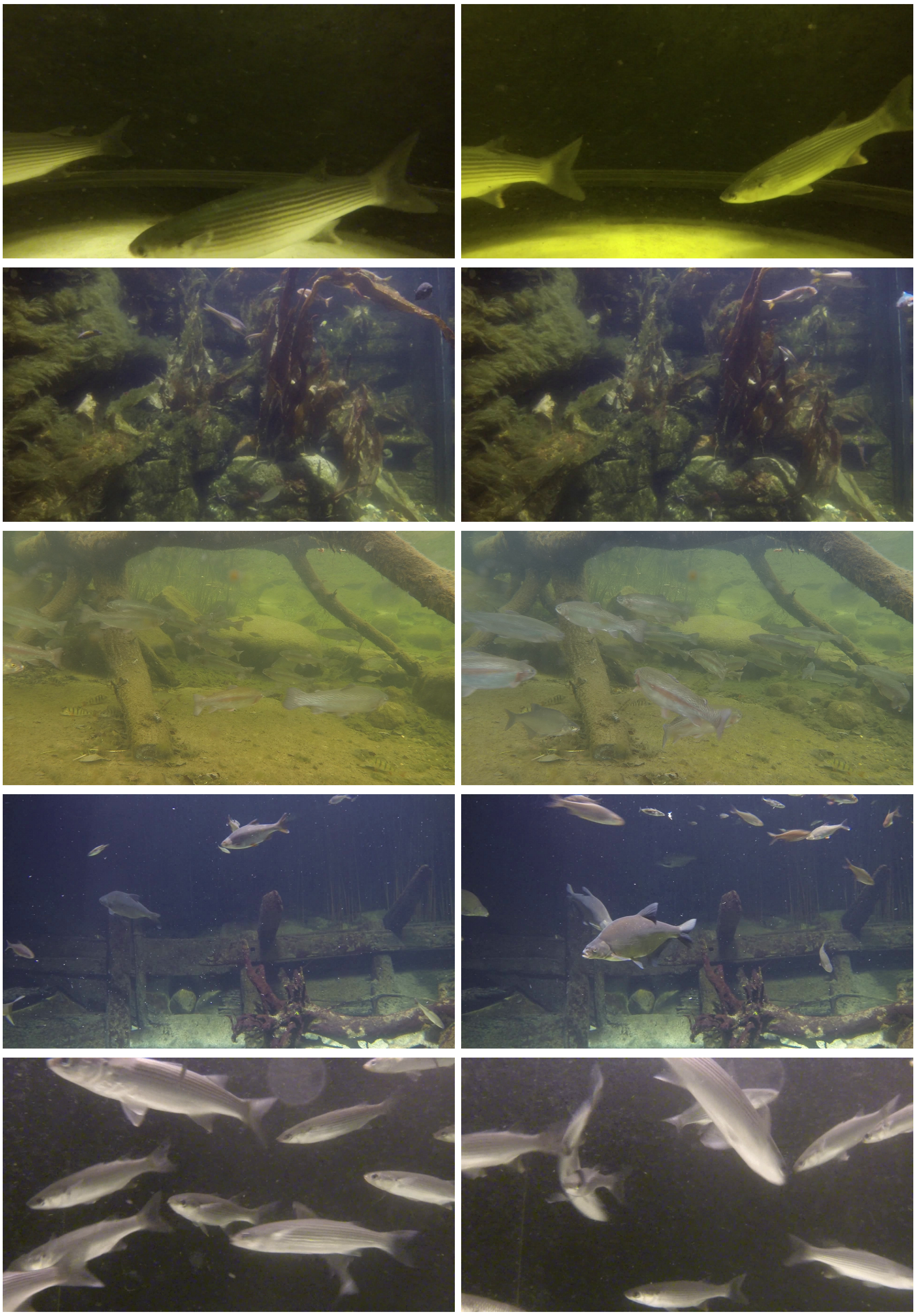


Figure 5.4: One example from each video of the UCD dataset. From top to bottom: *two Fish*, *Caustics*, *Fish Swarm*, *Marine Snow* and *small Aquaculture*.

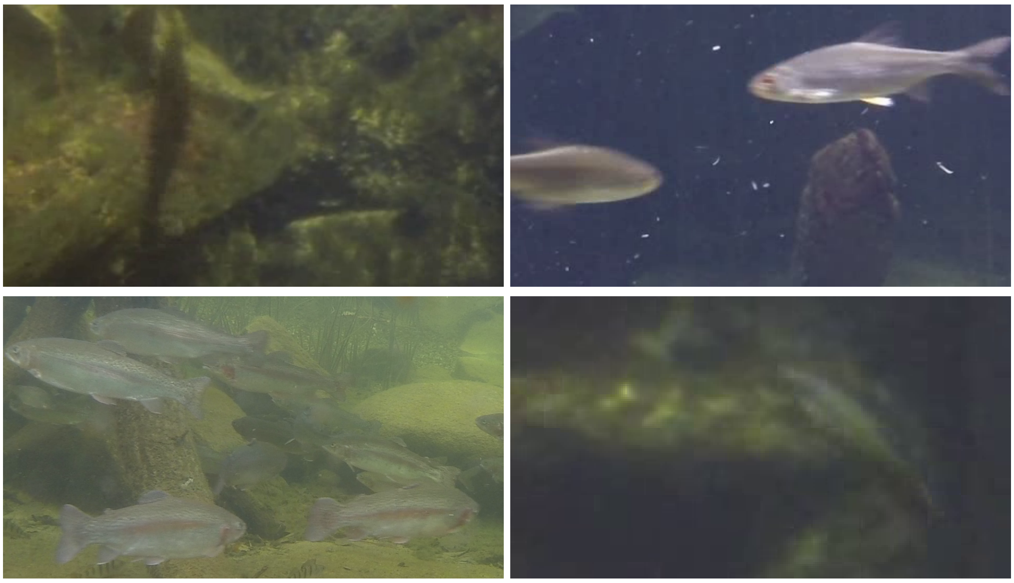


Figure 5.5: Selected difficult situations from the UCD dataset. In the top row are a shadow of a fish (left) and Marine Snow (right) depicted. At the bottom is a fish swarm (left) as well as a single fish which is barely distinguishable from the background (right).

6 Image Enhancement on Underwater Change Detection

With an established dataset, it is now possible to evaluate the effect of underwater image enhancement on different segmentation algorithms. These image enhancement methods can deliver great results for a human observer in many situations, however, it is mostly unknown if algorithms for change detection can profit from them in a similar way. First, there will be a short explanation of the underwater image enhancement methods that were used and afterwards their effects on the videos will be evaluated. For this, five different segmentation algorithms are combined with the enhancement methods and their segmentation results on the UCD dataset are compared.

6.1 Enhancement Methods

6.1.1 Marine Snow Removal - MSR

Marine Snow is a problem which is present in three of the presented videos, it has a strong impact in the *Marine Snow* video as well as the *small Aquaculture* video and is also visible in the *Fish Swarm* video but there mainly in regions without fish. To remove these unwanted objects from the image a method which is loosely based on the approach of [Ban+14] was implemented. The method uses the high reflectivity of the Marine Snow particles which makes them stand out prominently from their neighborhood. The algorithm looks for outliers in the brightness, this is done by using a search window that compares the brightness of the center pixel with the mean brightness of all other pixels in that window. To account for the different sizes Marine Snow can have the search process has to be repeated several times with varying sizes of the search window. Typical sizes used for the Marine Snow removal (MSR) were from 11×11 to 21×21.

If a pixel with an unusual high brightness was found the color was also checked to avoid wrong detections. Only if the three color channels (red, green and blue) had a similar intensity – which means a color close to white or gray – was the pixel considered as belonging to Marine Snow. Afterwards, the pixel value was adjusted to its neighborhood by median filtering but the immediate neighborhood was omitted in the filtering process. The reason for this is that these pixels are very likely to also belong to the Marine Snow particle and therefore would distort the average color of the background. The effect of this enhancement on the human observer is a visually more pleasant image and the pixels where Marine Snow was replaced by median filtering are hardly recognizable anymore (see Figure 6.1). However, there are also some false detections which change parts of the image that do not belong to Marine Snow particles and the median filtering, in general, creates some noise in the detected regions.

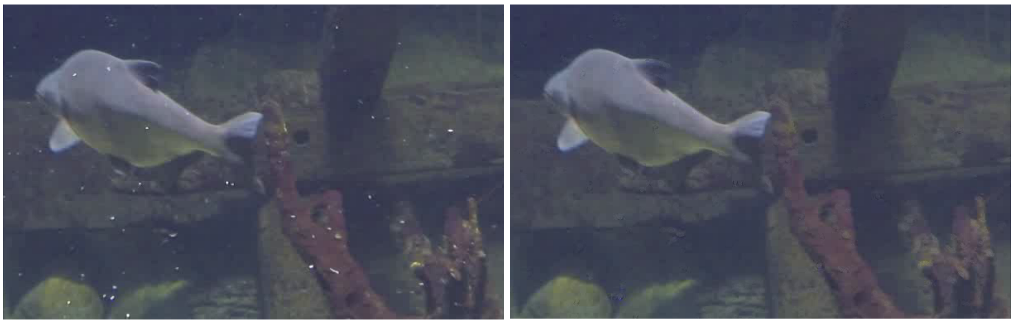


Figure 6.1: The effect of the MSR algorithm on a frame of the *Marine Snow* video.

6.1.2 Learning-based Deblurring - LbD

Underwater images often suffer from a strong blur of the present features due to forward scatter of light from the object to the camera. To address this issue, in [FR16] a dictionary learning algorithm based on sparse representation was proposed by us. The advantage of a learning-based algorithm is that it can extract useful knowledge of the physical properties from a given dataset although an explicit model is not available. The formulation of an explicit model would be difficult since many different physical effects interact with each other (see Figure 5.2) and at best the result is a mathematically complex formulation that describes some of the occurring effects. By having an over-complete dictionary (a matrix of vectorized image patches) the sparse representation theory allows the representation of a patch over this dictionary using only a few elements of this dictionary. Using this, together with the dictionary learning algorithm (K-SVD [AEB06]), the relation between a set of degraded training image patches and corresponding high-quality ones is learned and can then be used for the restoration of any underwater image. In a next step, the blurriness of each patch is measured first and then the deblurring is adapted to the specific amount of blur that is present in each patch. This adaption gives more accurate results and allows the creation of a coarse depth map since blur in underwater images is a function of distance. The result of the Learning-based Deblurring (LbD) approached is an image which is sharper and has more visible details. However, not only the edges are enhanced but unfortunately also the noise, which is more prominent in the results (see Figure 6.2).

6.1.3 Adaptive Gray World - AGW

The next two methods try to counter the loss of color information in underwater videos. In [FZL15] a simple but still effective color mapping scheme was proposed which is based on the gray world hypothesis. It is claimed that discarding the extreme cases in the deep sea, where red or even green colors are completely lost, the gray world hypothesis is still valid. However, remapping the colors in underwater images the same way as in-air images are remapped with the gray world hypothesis does not work. Usually, too much color information is lost, especially in the red channel, which would lead to a concentration of specific values. To avoid this effect an adaptive gain factor is introduced. Using this gain factor each pixel is sketched with a different weight and as a result, not only the colors are sketched based on their wavelength but also the

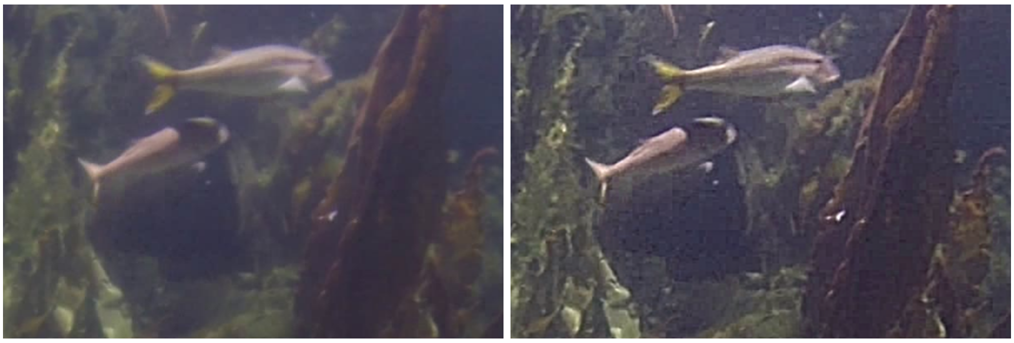


Figure 6.2: These images show the effect of the LbD algorithm. The image is not as blurry anymore and more details can be seen, for example at the wood on the right side. However, it can also be seen that the image becomes noisier.

whole dynamic range is used. An example is given in Figure 6.3.

6.1.4 Automatic Color Equalization - ACE

Automatic Color Equalization is a method introduced by Gatta et al. [GRM02] which combines the gray world hypothesis and the white patch approach for a fully automatic image color restoration. The original algorithm is quite slow ($O(N^4)$ with N as the number of pixels) since a slope function - which amplifies small differences and saturates large differences - has to be computed between every two pixels and is weighted with the distance between these two pixels. Therefore, the original approach is too slow for an HD-Video and here the implementation of [Get12] is used which reduces the complexity to $O(N^2 logN)$ by using polynomial approximations for the slope function. An example can be seen in Figure 6.3, there the difference between the fish and the background is more distinct than in the original frame. The overall effect on the image of ACE is much stronger than AGW.

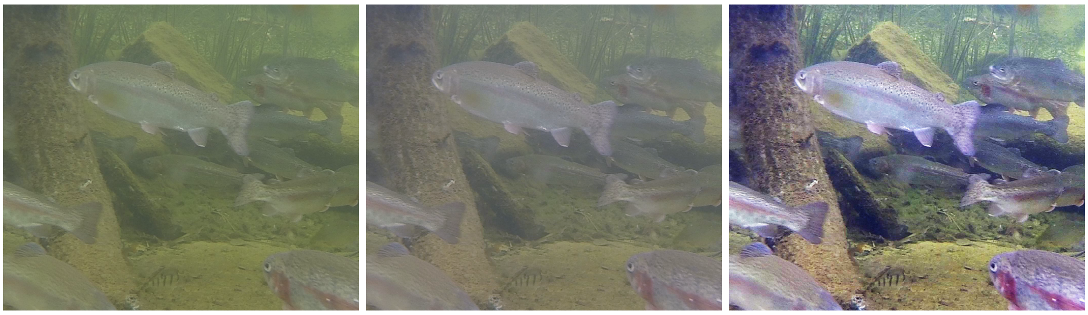


Figure 6.3: Depicted are, from left to right, the original snippet from the *Fish Swarm* video, the color corrected version with the adaptive gray world approach and the result of the automatic color equalization.

Segmentation Methods

The focus here was put on the background subtraction methods since this work mainly deals with this approach, however, another method which is solely based on the optical flow was also included for comparison. One of the background subtraction approaches that is evaluated with these enhancement methods is the proposed GSM algorithm described in Section 4.1. Alongside this, three other background modeling and subtraction methods are evaluated to get a more complete and thorough overview of the benefits of image enhancement. The spatial methods are evaluated later on underwater footage (see Section 7.3). In this section the focus is on three points:

- The general performance of background subtraction methods in the underwater context.
- Can the segmentation accuracy be increased by combining background subtraction with image enhancement methods?
- An comparison against optical flow-based methods. Which algorithms can deal better with the difficult underwater environment and which work together well with image enhancement?

The three background subtraction methods do not use spatial methods as well and their implementation is available in OpenCV[1] (MoG 1 [Ziv04], MoG 2 [KB02] and KNN [ZH06]). The last method, the segmentation by motion, is from the paper [OMB14] and the code they used for the segmentation is available as well[2]. It was chosen because it is based on the optical flow - the other widely used foreground detection method - and therefore allows an evaluation of how good these different approaches can deal with the difficult underwater environment and how they can profit from the image enhancement. These approaches were chosen for comparison because unlike most methods existing implementations of them are freely available, this ensures a fairer evaluation than a reimplementation.

6.2 Evaluation

A comprehensive evaluation of these segmentation algorithms is made in the following to compare the accuracy of them. First, they were all run on the normal UCD dataset and it will be analyzed how the degradation of the video quality in underwater scenarios influences the segmentation quality. Afterwards, the image enhancement will be added to see if it can have a positive effect on the segmentation accuracy.

Background Subtraction on Underwater Videos

The results of the four different background subtraction approaches on the UCD dataset can be seen in Table 6.1. There the four basic numbers for binary classification (*true negative*, *true positive*, *false negative* and *false positive*) are given and the deduced values *F1-Score* and

---

[1] www.opcencv.org

[2] http://lmb.informatik.uni-freiburg.de/resources/binaries/ under Motion Segmentation

MCC (see Chapter 2 for details). In the following tables only the MCC value will be used for the comparison because both values - although slightly different - lead in the end to the same conclusions. For the gray/unsure areas in the ground truth images both classifications were correct, so if the pixel is classified as background by the algorithm it would count as a *true negative* and otherwise as a *true positive* pixel.

Algorithm	True Negatives	True Positives	False Negatives	False Positives	F1-Score	MCC
			***two Fish* Video:**			
GSM	180,375,360	17,611,623	3,144,660	6,228,357	0.7898	0.7669
MoG 1 [Ziv04]	180,197,503	17,876,878	3,203,631	6,081,988	**0.7938**	**0.7708**
MoG 2 [KB02]	186,601,671	12,668,199	7,377,859	712,271	0.7579	0.7555
KNN [ZH06]	171,035,889	19,855,054	1,233,689	15,235,368	0.7068	0.6930
			***Caustics* Video:**			
GSM	199,856,886	2,845,746	399,016	4,258,352	0.5499	0.5841
MoG 1 [Ziv04]	202,367,357	2,922,016	396,985	1,673,642	0.7383	0.7435
MoG 2 [KB02]	204,526,698	1,441,280	1,260,559	131,463	0.6743	0.6964
KNN [ZH06]	202,611,023	2,869,860	21,439,644	1,492,743	**0.7533**	**0.7571**
			***Fish Swarm* Video:**			
GSM	170,096,604	14,821,201	21,153,182	1,289,013	0.5691	0.5721
MoG 1 [Ziv04]	171,894,004	6,339,948	29,026,034	100,014	0.3033	0.3874
MoG 2 [KB02]	172,338,391	1,025,615	33,988,415	7,579	0.0569	0.1556
KNN [ZH06]	171,228,069	14,420,954	21,439,644	271,333	**0.5705**	**0.5904**
			***Marine Snow* Video:**			
GSM	189,927,339	14,378,161	3,209,105	1,918,995	**0.8361**	**0.8487**
MoG 1 [Ziv04]	189,109,247	14,516,541	3,070,725	2,737,087	0.8182	0.8333
MoG 2 [KB02]	192,618,186	7,074,827	7,251,284	415,703	0.6480	0.6682
KNN [ZH06]	190,710,197	13,326,863	4,260,403	1,136,137	0.8224	0.8316
			***small Aquaculture* Video:**			
GSM	152,343,809	34,692,315	16,803,395	3,520,481	0.7734	0.7255
MoG 1 [Ziv04]	154,091,301	34,900,057	17,237,378	1,131,264	0.7383	0.7435
MoG 2 [KB02]	156,568,143	13,975,078	36,795,641	21,138	0.4315	0.4715
KNN [ZH06]	152,381,919	43,446,320	9,336,148	2,195,613	**0.8828**	**0.8504**
			whole Dataset:			
GSM	892,599,998	84,349,046	44,709,358	17,215,198	0.7037	0.69946
MoG 1 [Ziv04]	897,659,412	76,555,440	52,934,753	11,723,995	0.6784	0.6957
MoG 2 [KB02]	912,653,089	36,184,999	86,673,758	1,288,154	0.5137	0.5488
KNN [ZH06]	887,967,097	93,919,051	36,656,258	20,331,194	**0.7472**	**0.7445**

Table 6.1: All four different background subtraction methods evaluated on the UCD dataset. The best *MCC* and F1-Score values for the whole dataset are the sum of the separate values for the videos divided by five. In bold are marked the best *MCC* and F1-Score values for each video.

The most difficult scene for the background subtraction algorithms is the *Fish Swarm* video as it combines a difficult background, color cast and the constant presence of many foreground objects. Especially the many foreground objects which are similar in color to the background make the modeling of the background very difficult. The problem is enhanced by the general loss of color information due to the strong color attenuation in the video, the result of the modeling can

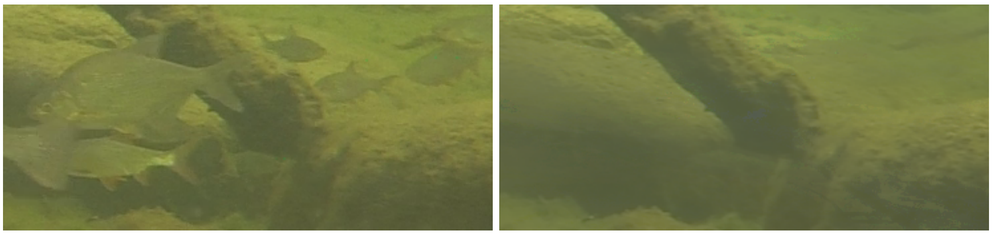


Figure 6.4: A snippet from the *Fish Swarm* video (left) and the trained background model (right) of the GSM algorithm.

be seen in Figure 6.4, the background model is blurred and often a mix between the background and the color of the fish passing by.

Also interesting is that the *two Fish* video, which seems to feature the simplest of all five scenes, does not provide the highest accuracy per se. The results are very stable with regard to the segmentation method but, for example, the KNN algorithm provides better results in the *Caustics*, *Marine Snow* and *small Aquaculture* videos. Overall the KNN algorithm [ZH06] gives the best results although it performed comparatively bad for the *two Fish* video.

Background Subtraction with Enhancement

	two Fish					***Caustics***				
Algorithm	-	+ACE	+AGW	+LbD	+MSR	-	+ACE	+AGW	+LbD	+MSR
GSM	0.7669	0.5278	0.7594	0.7164	0.7669	0.5841	0.3325	0.5786	0.4001	**0.6011**
MoG 1 [Ziv04]	0.7708	0.7297	**0.7969**	0.7633	0.7708	0.7435	0.5520	**0.7466**	0.6618	**0.7512**
MoG 2 [KB02]	0.7555	0.7056	**0.7990**	0.7549	**0.7659**	0.6964	**0.7419**	0.6964	0.6711	**0.6981**
KNN [ZH06]	0.6930	0.5887	**0.7391**	0.6806	0.6930	0.7571	0.4516	0.7557	0.6278	**0.7682**

	Fish Swarm					***small Aquaculture***				
Algorithm	-	+ACE	+AGW	+LbD	+MSR	-	+ACE	+AGW	+LbD	+MSR
GSM	0.5721	0.5014	0.5673	0.5199	0.5654	0.7255	0.6462	0.7232	0.6583	**0.7303**
MoG 1 [Ziv04]	0.3874	**0.7350**	**0.3946**	**0.4500**	0.3767	0.7435	0.4377	**0.7624**	0.7374	**0.7612**
MoG 2 [KB02]	0.1556	**0.3362**	**0.1694**	**0.2423**	0.1395	0.4715	0.3197	**0.4720**	0.4719	0.4699
KNN [ZH06]	0.5904	**0.8402**	0.5503	**0.6160**	0.5812	0.8504	0.4210	**0.8517**	0.8157	**0.8544**

	Marine Snow					***Overall***				
Algorithm	-	+ACE	+AGW	+LbD	+MSR	-	+ACE	+AGW	+LbD	+MSR
GSM	0.8487	0.6316	0.8096	0.6850	**0.8614**	0.6995	0.5279	0.6876	0.5959	**0.7050**
MoG 1 [Ziv04]	0.8333	0.8214	**0.8537**	0.7923	**0.8806**	0.6957	0.6552	**0.7108**	0.6810	**0.7081**
MoG 2 [KB02]	0.6682	**0.8290**	**0.7190**	0.6507	**0.6703**	0.5494	**0.5865**	**0.5712**	**0.5582**	0.5487
KNN [ZH06]	0.8316	0.7975	**0.8625**	0.8115	**0.8792**	0.7445	0.6198	**0.7519**	0.71032	**0.7552**

Table 6.2: Evaluation of the impact of different image enhancement methods on the background subtraction methods. Given are the MCC values and all cases in which an enhancement increased the accuracy are shown in bold.

The impact of the underwater image enhancement methods on the background modeling and subtraction is depicted in Table 6.2. Altogether, the results are very volatile, great improvements could be achieved on some videos but the same method would then also reduce the accuracy on other videos. An example for this is the MoG 2 algorithm ([KB02]) in combination with the *ACE* color correction: in the *Marine Snow* video the accuracy could be improvement from 0.668 to 0.829 but on the *small Aquaculture* video an equally large deterioration from 0.4715 to 0.3197 happened. Therefore, the *ACE* method overall slightly decreased the accuracy and this is true for all different background subtraction methods. Only on videos with a strong color cast (like the *Fish Swarm* video) does it seem to be a viable option.

For *LbD* the results were quite similar to the *ACE* although the impact on the performance was overall less significant. In contrast to that, the *AGW* method showed a relatively small impact on the segmentations of each video but increased the overall results of three out of the four algorithms slightly. It seems to be a suitable method to aid background subtraction in general underwater situations if the additional computational load is unproblematic. An example of the segmentation of the KNN and GSM algorithms in conjunction with the *AGW* is depicted in Figure 6.5.

The last method is the *MSR* approach which is special because normally underwater image enhancement techniques try to make the frames sharper, the color more vivid or increase the contrast. The *MSR* tries none of these and often even blurs the image a bit by trying to remove Marine Snow particles. It could greatly improve the results for the *Marine Snow* and had a varying but little effect on the other results.

In conclusion, the effect of the image enhancement algorithm varies widely from video to video and, therefore, the enhancement method should be chosen based on the degradation of the video. Otherwise, no positive effect or even a deterioration of the segmentation result is very probable.

- Color Cast: The *ACE* showed great results for a prominent color cast. Other approaches could also achieve some improvements but only minor in comparison to the *ACE*.

- Blur and haze: Are in general not that problematic for background subtraction methods. Theoretically, they should be addressed by the *LbD* but these effects usually occur together with a color cast and then the missing color information is the far more substantial problem and should be solved primarily.

- Marine Snow: The especially developed *MSR* algorithm handled this degradation very well. Methods that sharpen the image (like *LbD*) only emphasize the marine snow particles and thereby worsen the problem.

- Caustics: No algorithm especially addresses them. Image restoration methods like *LbD* or *ACE* tend to increase the problem since they make the caustics more prominent. The *MSR* could improve the results for the *Caustics* video slightly, probably because of the small blur effect it has.

- Unknown degradations: If the degradation is not known beforehand the *MSR* seems like the most sensible choice. It has a comparatively small computational load, shows great

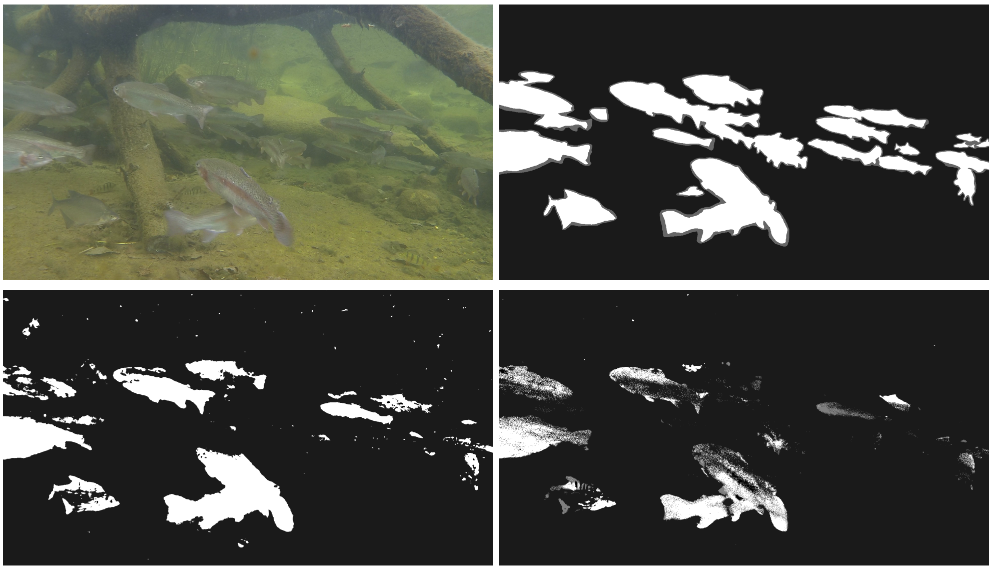


Figure 6.5: Top: the raw image and the ground truth data. Bottom: segmentation results of the GSM algorithm + AGW(left) and on the right the background subtraction of [Ziv04] + AGW.

Video:	-	+ACE	+AGW	+LbD
Marine Snow	0.7669	0.7842	0.6693	0.8548
Fish Swarm	0.4907	0.5936	0.4883	0.6076
Caustics	0.3810	0.5782	0.4588	0.6974

Table 6.3: Results of the motion segmentation from [OMB14] on three of the videos. Also, the effects of three different image enhancement algorithms are shown. The numbers indicate the MCC.

improvements when Marine Snow is present and almost no (negative) effects when other degradations appear.

Run-times are always hard to compare since they not only depend on the image size but also often on the parameters of the algorithm, the specific image data, the parallelization and so on. For the setup used here for a typical image (1920 × 1080) the run-times were as follow:

MSR	~ 0.75 second per frame	*ACE*	~ 1.75 seconds per frame
LbD	~ 4.5 seconds per frame	*AGW*	~ 0.1 seconds per frame

Motion Segmentation on Underwater Videos

The segmentation by motion from [OMB14] has to be treated separately as it did not provide meaningful results for two of the videos. The reason for this is that the videos *small Aquaculture*

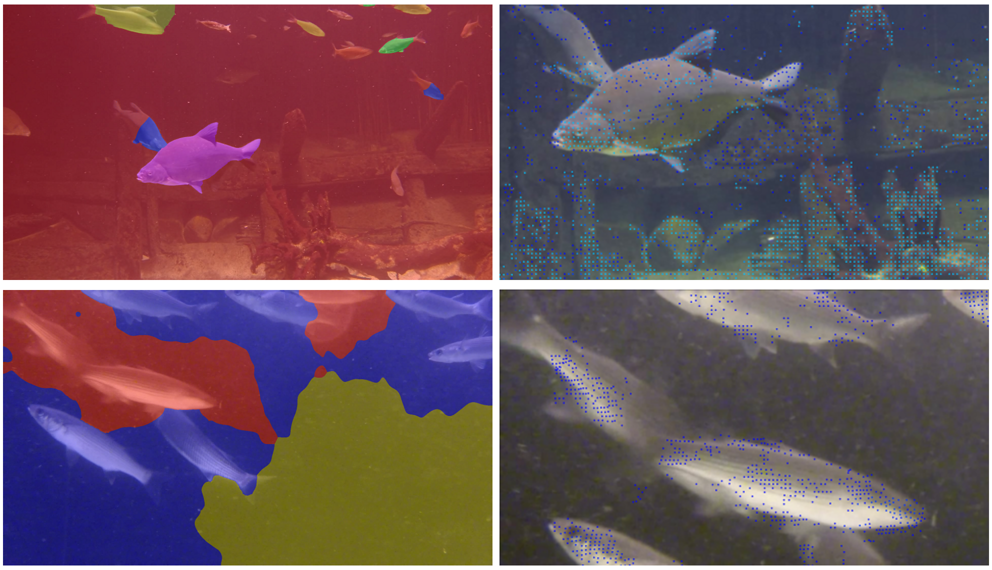


Figure 6.6: On the left side are segmentations shown from the algorithm [OMB14] overlayed over the original frame. The right side displays close-ups of the features which were tracked.

and *two Fish* have a uniform and featureless background. This, which would be considered as an advantage for the background subtraction methods, is an irresolvable problem for this approach since the segmentation is based on the tracking of features and if too few features can be found in the frames the segmentation will fail. An example of this can be seen in Figure 6.6 where for the *small Aquaculture* video no sensible distinction between foreground and background could be made because the black background is completely featureless. This problem does also exist for featureless areas in in-air videos but the degradation effects underwater (e.g. the blur) obscure small features and thereby worsen the problem. On the other three videos, the method provided decent results, although in general not as good as the background subtraction methods. It has to be noted that this approach is slow but has the advantage that it inherently differentiates between different foreground objects and also tracks them.

The results on the UCD dataset and the effects of combining it with image enhancement methods are shown in Table 6.3. The effects of the *AGW* method are ambivalent but *ACE* and *LbD* improve the results consistently through all three videos. The reason for this is that these methods make features in the image more distinct and thereby allow the algorithm to track more features. For example, in the normal *Marine Snow* video 28197 features could be tracked but when the video was first deblurred by *LdD* this number increased to 80225 (in Figure 6.6 on the right side, each small rectangle signifies one found features in the frame).

The results of the Marine Snow Removal (MSR) are not depicted in the Table 6.3 because the method reduced the number of features that could be tracked in the videos drastically and lead to segmentations that could not sensibly be evaluated anymore. Two examples of this are depicted in Figure 6.7, in the *Caustics* video the algorithm would not segment any objects at all because the number of features in motions was too low and in the *Marine Snow* video the

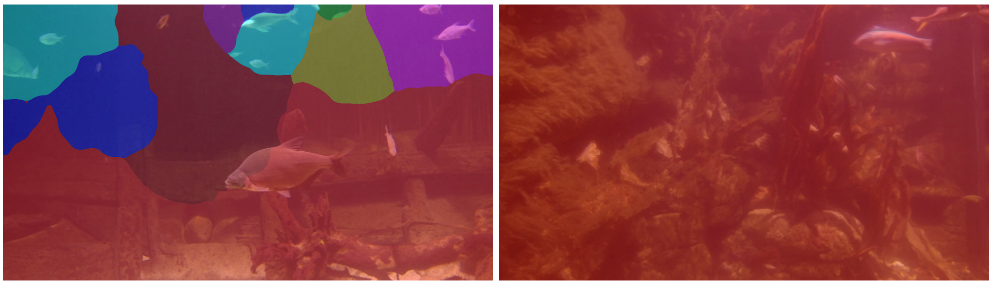


Figure 6.7: Two examples of the segmentations from [OMB14] after the Marine Snow removal from Section 6.1.1 is applied. The smoothing effect of the MSR decreases the number of features drastically and makes the detection of moving objects impossible.

number of background features between the fish was too low, so that vast areas got marked as moving objects. Overall, these examinations show that the optical flow is not competitive with background subtraction for underwater scenarios with static cameras.

6.3 Conclusion

In the previous chapter, the different degradation effects that can influence videos and images in the underwater environment were described. Afterwards, the first change detection dataset was introduced that focuses especially on the underwater context. This data could now be used in this chapter to evaluate five different change detection methods (four background subtraction approaches and one based on the optical flow) in combination with four different underwater image enhancement techniques.

First, the general performance on underwater videos was evaluated (without any image enhancement) and here the background subtraction approaches clearly outperformed the optical flow. The optical flow computation is heavily affected by the degradations in the video since the number of detectable features - which are tracked by this approach - is reduced drastically and the remaining features are less distinguishing and, therefore, harder to match correctly. The performance of the background subtraction is also impaired by the degradations, e.g. when the color information is lost, but considerably less.

By adding image enhancement methods this outcome did not change, however, the effect it had was very different for both segmentation approaches. For the different background subtraction methods, the result was heavily dependent on the degradation effect. The negative effect of color cast could be counteracted very well with the ACE method. The same is true for the Marine Snow particles and the MSR algorithm. Blur and haze are not that problematic for background subtraction and reducing their impact - e.g. with the LbD method - did not show great improvements. The last degradation, caustics, is not addressed specifically by any image enhancement and their effect on the segmentation results could not be significantly mitigated. Methods like ACE or LbD even seem to increase the problem by making the caustics more prominent.

In contrast to that, the results for the optical flow were very consistent for each image enhancement method. It suffered drastically from the MSR since this method reduced the number of detectable features further. However, accuracy could be gained by adding the LbD because this made the features more distinct and easier to detect. Nonetheless, even with image enhancement, the optical flow approach could not produce meaningful results for some videos and overall is not competitive with the background subtraction approach.

In total, these experiments proved that background subtraction should be the favored change detection method for underwater scenes because it can handle the degradations far better than approaches based on the optical flow. Furthermore, it can be stated that the effect of the image enhancement methods is very volatile, strongly depending on the specific underwater scene and the degradations which are present. A poorly chosen image enhancement can often have a negative effect on the segmentation result, e.g. ACE on videos that do not have a color cast. Therefore, they should be chosen and applied very carefully but, in general, a combination of background subtraction and MSR seems to be the most promising approach for an unknown scene since MSR had almost no negative effects when used on videos without Marine Snow.

7 Underwater Change Detection

With this first evaluation of different change detection methods on the UCD dataset in conjunction with four underwater image enhancement methods, it became clear that an image enhancement preprocessing step alone will not be sufficient to counteract the degradation effects in the underwater environment. The degradations are too diverse and the effect of the enhancement methods too unstable to reliably mitigate all of the negative impacts. Therefore, to deal with the difficult underwater scenarios, the next step is the adaption of the proposed change detection methods to this special environment. This will be done by elaborating the GSM background modeling process presented in Section 4.1 and using an improved way to automatically adapt some of the parameters. Also, the special but important case of very crowded underwater scenes – with many fish swimming behind and in front of each other – will be discussed and a method is proposed that at least minimizes the negative effect this has on the background modeling.

7.1 Extension of the GSM - eGSM

The proposed GSM approach is a fast and flexible model which proved that it can handle in-air situations quite well. Now, however, it will be extended by superimposing it with the Mixture of Gaussian (MoG) idea, adding a foreground model and using an enhanced way to define the parameters. This makes the whole approach more complex and slower but also can increase the accuracy of the model further which is especially necessary for difficult scenarios like underwater scenes but can also be beneficially in difficult in-air situations, e.g. when it is snowing or raining.

To achieve this higher model accuracy the two Single Gaussian models that are used in the standard GSM are replaced by MoG models which are then updated in the same way, one partially and one complete. Each MoG consists of a variable number of Gaussians (in the experiments always five are used) and every one of them is described by three values: mean μ, variance σ and weight w. The mean and variance are similar to the Single Gaussian model since they are a description of the probability distribution of a Gaussian. The weight, however, is a new variable needed in the MoG model that reflects the confidence in a specific Gaussian in comparison to the others in the MoG model. If a specific Gaussian has a high weight, this means that it is supported by a lot of data (at least compared to the other Gaussians) and therefore a high confidence can be put into this data. Alternatively, if the weight is low the Gaussian is backed by only a few data points and is very unlikely to be a static background object.

Hence, to be considered as a part of the background a minimum weight is necessary, otherwise, the Gaussian is assumed to belong to the optical flow of a foreground object which only appeared shortly in the video. The minimum weight is defined as a percentage of the sum over all weights of a MoG. As this sum depends on the number of Gaussians in a MoG, the percentage was set to $1/\#\text{gaussians}$. Notation-wise, $M(\mu_2, 100, 100, r)$ would be the mean of the second Gaussian of the red channel of the pixel at location $(100, 100)$ and this gaussian would have the weight $M(w_2, 100, 100, r)$.

The updating process is also slightly different than in the Single Gaussian model, from the mix of different Gaussians only the one that matches the current data the best should get updated. For each pixel the color is modeled by three models, each consisting of two MoGs and representing one color channel. This is similar to the standard GSM where the models for each color channel are independent of each other. However, if these models are treated completely independently they would fail to accurately describe the current scene. To understand the problem, imagine a pixel that switches between black and white. For each channel, there will be one Gaussian that models high intensity (255) and one that models a low intensity (0). But when they are treated independently each combination of them would also be possible, e.g. a high intensity in the red channel and a low intensity in the other two channels. To prevent this, the three channels must be evaluated and updated jointly. Hence, the three Gaussians that match best with the current pixel color are given by

$$g_{bm} = \underset{g=1,\dots,n}{\arg\min} \sum_{c\in\{R,G,B\}} \frac{(I(\bar{v},c) - M(\mu_g,\bar{v},c))^2}{\beta \cdot M(\sigma_g,\bar{v},c)}. \tag{7.1}$$

The g – e.g. in $M(\mu_g,\bar{v},c)$ – stands for the Gaussian of the MoG model that is currently compared against the pixel value and g_{bm} is the best match that was found. The usage of the (L, C_1, C_2) color space from Equation 4.4 is not beneficial anymore in this context since all three channels are evaluated at the same time and therefore a change only in the intensity cannot be identified anymore. Hence, the standard (R, G, B) color space can be applied to avoid a conversion. After the best-matching Gaussians are found they can be updated with the new data from the current frame in the following way

$$\begin{aligned} M(\sigma_{g_{bm}},\bar{v},c) &= \alpha \cdot M(\sigma_{g_{bm}},\bar{v},c) + (1-\alpha)\cdot (M(\mu_{g_{bm}},\bar{v},c) - I(\bar{v},c))^2, \\ M(\mu_{g_{bm}},\bar{v},c) &= \alpha \cdot M(\mu_{g_{bm}},\bar{v},c) + (1-\alpha)\cdot I(\bar{v},c), \\ M(w_{g_{bm}},\bar{v},c) &= M(w_{g_{bm}},\bar{v},c) + 1. \end{aligned} \tag{7.2}$$

This whole updating process is done two times, once for the partially updated MoG and also for the fully updated one. The partial MoG is obviously only updated in the areas classified as background. One constraint for this updating is that there is one Gaussian that matches the current color of the pixel. If even the best match cannot meet a minimum threshold none of the Gaussians are updated and instead, a new Gaussian must be created based on the new value from the current frame. If there exist already the maximum number of Gaussians in the MoG the Gaussian with the lowest weight has to be deleted and replaced with the new Gaussian. The minimum threshold $T_{newGaussian}$ is checked with the following inequality

$$\sum_{c\in\{R,G,B\}} (I(\bar{v},c) - M(\mu_{g_{bm}},\bar{v},c))^2 < T_{newGaussian}, \tag{7.3}$$

where $T_{newGaussian}$ was set to 0.015 for the experiments on the UCD dataset. This threshold prevents the creation of new Gaussians that are very similar to the existing ones but should also not be chosen too large as it would prevent the modeling of different objects in different Gaussians, which is the principle idea of the MoG. If a new Gaussian has to be created it will be initialized with

$$\begin{aligned} M(\sigma_{g_{bm}},\bar{v},c) &= I(\bar{v},c), \\ M(\mu_{g_{bm}},\bar{v},c) &= \zeta, \\ M(w_{g_{bm}},\bar{v},c) &= 1, \end{aligned} \tag{7.4}$$

similar to the initialization of the standard GSM.

To not run into overflow problems after some time (the weight increases by 1 with each update) and, more importantly, to diminish the effect of old data, a decay factor δ is introduced and applied on all weights of the whole model. This is done with every new frame in the following way

$$M(w_g,\bar{v},c) = \delta \cdot M(w_g,\bar{v},c) \quad \forall g\ \forall c\ \forall \bar{v}. \tag{7.5}$$

In the underwater scenes a δ of 0.995 proved to be ideal. Another important change to the Single Gaussian model is that the α value, which is the key parameter for the updating process, is not static anymore but determined dynamically based on the weight of the specific Gaussian. If a Gaussian is selected to be updated the α will be dynamically set to $\frac{1}{M(w_g,\bar{v},c)}$ but it is always capped at 0.5. Hence the value of δ also controls the update rate α and is thereby the most important parameter for the background modeling.

Together, this ensures that Gaussians which until now only got very few data points to back them up or only old data points which are not reliable anymore adapt quickly to new values. At the same time, Gaussians which were updated frequently (and therefore have a high weight) will receive an update with a small α and are not strongly affected by single outliers. Consequently, the decay factor has a strong impact on the update rate, especially in longer videos where the decay rate dominates the update rate. For example, a δ of 0.995 means that the sum of all weights in a MoG model will tend to 200 but never reach it. Therefore, α will always be greater than 0.005.

Furthermore, the parameter β is also dynamically adjusted based on the current video instead of giving it a fixed value. This makes the approach more universally applicable to different scenarios and requires less user interaction. The value of β, e.g. in the Equations 7.1 or 7.8, controls the sensitivity of the whole model to changes in the scene and is therefore very important for the detection of objects. This sensitivity should be dependent on the amount of change present in the scene (e.g. through camera noise, moving objects or a tree waving in the wind) so that in a very noisy scene the sensitivity is reduced and vice versa. For that reason, β is defined as a comparison between the current and previous frame of the video in the following way

$$\beta = (1-\alpha_{avg})\beta + \alpha_{avg}\frac{\sum_{\bar{v}}\sum_{c\in\{R,G,B\}}\sqrt{|I^{old}(\bar{v},c) - I^{new}(\bar{v},c)|}}{\sum_{\bar{v}} 1}. \tag{7.6}$$

This is an updating process similar to a running Gaussian update, with the initialization of $\beta = 1$ and α_{avg} as the average of all α values used in the last updating of the background model. The square root of the absolute value is taken so that large disparities (e.g. due to moving objects) are not overrated and dominate the whole sum. To save processing time, this updating can be done over a smaller version of the original frames or over a randomly sampled subset of pixels instead of using a sum over the whole frame. In experiments 1000 randomly selected pixel proofed to be sufficient and also small enough to not significantly influence the run-time.

Besides the two MoG models, there is also a foreground model that is created and will be updated simultaneously. It consists of a simple SG model as only the latest visible foreground object should be modeled for each pixel. The update rate α is a fixed value as there is no weight which could control it (compare with equation 4.2). It should be set relatively low so that the model learns a new foreground object fast enough before it disappears again, for the UCD dataset it was set to 0.64. The updating process is in a way the reverse version of the partial updating scheme, only the parts of the scene that were detected as foreground will get updated. The usage of a foreground model is especially promising in underwater scenarios with fish since they often are very similar to each other and therefore the model does not have to change drastically between different foreground objects. This argument does not hold for most in-air videos where the foreground objects usually exhibit radical change, e.g. cars of different colors or humans with different colored clothes. Of course, there are also some underwater scenes with many different colorful fish or other creatures, however, these scenes are usually not in Europe and these fish are often not economically important, e.g. grown in aquacultures.

After the model is created, the segmentation itself is then done by comparing the current frame with the two MoGs. The MoG model is assumed to agree with the current pixel color if the following inequality holds

$$0.5 < \exp\left[-\max_{g\in\{1,\ldots,n\}} \sum_{c\in\{R,G,B\}} \frac{(I(\bar{v},c)-M(\mu_g,\bar{v},c))^2}{\beta\cdot M(\sigma_g,\bar{v},c)}\right]. \tag{7.7}$$

However, not all Gaussians are considered for this check. Only the Gaussians that have a weight that exceeds the minimum weight ($M(w_g,\bar{v},c) > \frac{1}{n}\sum_{k=1}^{n} M(w_g,\bar{v},c)$) are considered part of the background model and therefore qualify for this check.

After this check, there are three possibilities. If the pixel matches with at least one Gaussian in both MoGs it will be classified as background and if it does not match with any Gaussian in the models it will be seen as a foreground object. More difficult is the last situation when the current pixel value agrees with only one of the models. To solve this problem the foreground model for the corresponding pixel is used as a tiebreaker by evaluating the inequality

$$0.5 < \exp\left[-\sum_{c\in\{R,G,B\}} \frac{(I(\bar{v},c)-M(\mu,\bar{v},c))^2}{\beta\cdot M(\sigma,\bar{v},c)}\right], \tag{7.8}$$

where $M(\mu,\bar{v},c)$ and $M(\sigma,\bar{v},c)$ are the values from the foreground model. If the inequality holds the pixel matches the foreground model and therefore should be marked as foreground, otherwise background.

Similar to the original GSM algorithm, there is an adaption of the partially updated MoG to the fully updated MoG to compensate for the weaknesses of partial updating scheme. Such an adaption should occur when there is something in the scene which is static and constantly classified as foreground, because then, with a high probability, an error in the background modeling happened and should be corrected. To detect such an error the first condition is that the fully updated MoG and the foreground model are similar since this indicates that this pixel has been mainly classified as foreground in the recent past. The models are considered similar if

$$\exists g \forall c \qquad M(\mu_g^{BG}, \bar{v}, c) - M(\mu_g^{FG}, \bar{v}, c) < \frac{M(\sigma_g^{FG}, \bar{v}, c)}{2}. \tag{7.9}$$

Here $M(\mu_g^{BG}, \bar{v}, c)$ is the mean value of the partially updated MoG model and $M(\mu^{FG}, \bar{v}, c)$ is accordingly the mean value of the foreground model. This similarity could also occur when there appear many foreground objects in a short period of time. To filter these events out the variance is used as well since foreground objects usually generate higher variations in the pixel color due to their movement. Hence, the second condition is a small variance and the limit is set to the median of all variances of the completely updated MoG to have an automatically adapting threshold which does not require any input parameters. If both conditions are fulfilled (inequality 7.9 and a small variance) an error in the partially updated MoG is very probable and therefore the model is overwritten with the values from the fully updated MoG to restart the model.

Lastly, it can occur that two Gaussians in one MoG get very similar over time. These Gaussians then should be unified as they are modeling the same object. The similarity between Gaussian g_1 and g_2 is checked with

$$\sum_{c \in \{R,G,B\}} (M(\mu_{g_1}, \bar{v}, c) - M(\mu_{g_2}, \bar{v}, c))^2 < \min_{g \in \{g_1, g_2\}} \left[\sum_{c \in \{R,G,B\}} (M(\sigma_g^{FG}, \bar{v}, c))^2 \right] \tag{7.10}$$

and if the inequality holds, the old Gaussians are deleted and a new Gaussian is created with the following values

$$\begin{aligned} M(\mu_{g_{new}}, \bar{v}, c) &= \frac{M(w_{g_1}, \bar{v}, c) \cdot M(\mu_{g_1}, \bar{v}, c) + M(w_{g_2}, \bar{v}, c) \cdot M(\mu_{g_2}, \bar{v}, c)}{M(w_{g_1}, \bar{v}, c) + M(w_{g_2}, \bar{v}, c)}, \\ M(\sigma_{g_{new}}, \bar{v}, c) &= \frac{M(w_{g_1}, \bar{v}, c) \cdot M(\sigma_{g_1}, \bar{v}, c) + M(w_{g_2}, \bar{v}, c) \cdot M(\sigma_{g_2}, \bar{v}, c)}{M(w_{g_1}, \bar{v}, c) + M(w_{g_2}, \bar{v}, c)}, \\ M(w_{g_{new}}, \bar{v}, c) &= M(w_{g_1}, \bar{v}, c) + M(w_{g_2}, \bar{v}, c). \end{aligned} \tag{7.11}$$

Altogether, this extension of the GSM leads to a complex but also more robust and accurate model building process since now several different objects can be represented by the model at the same time and the update rate adapts itself automatically based on the confidence the model has in the data. Especially in the difficult underwater scenes, this proved to be very advantageous compared to the standard GSM. Three examples of modeled backgrounds can be seen in Figure 7.1.

7.2 Segmentation of Crowded Underwater Scenes

In underwater videos, the scene can become very crowded when whole swarms of fish appear and this can cause several problems for the segmentation process. For once it becomes very difficult to separate the individual fish as they swim in front and behind each other, however, if

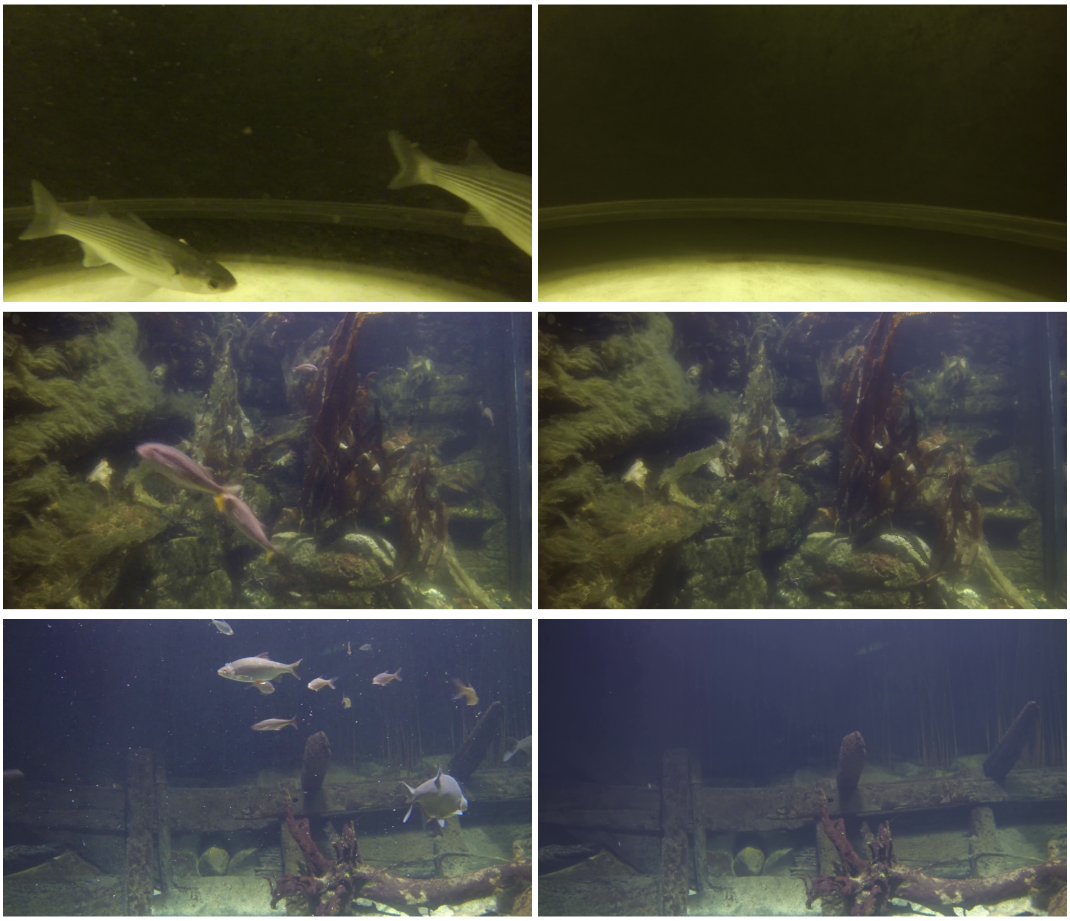


Figure 7.1: The right side depicts three background models created with the eGSM and on the left are the corresponding original frames from the video. The background models are visualized by taking the Gaussian with the highest weight of the partially updated MoG and displaying the mean of it.

the swarm stays permanently in the scene another problem arises with the background modeling. The concept of the background modeling is based on the idea that the objects in the background will be visible most of the time and therefore the model will learn the pixel values that are visible most of the time. If now in more than 50% of the time a fish is visible at a certain location, the model will tend to learn the color of the fish. This is especially true for fish since they tend to be very similar in color and then the model will learn the grayish color of the fish instead of the real background pixel value. A very good example of this is the *Fish Swarm* video of the UCD dataset where there are many fish constantly swimming back and forth in the middle of the scene. An example of the problems normal background modeling algorithms have with this is depicted in Figure 7.2.

7.2.1 Optical Flow as Pre-Segmentation in Crowded Scenes

An improvement of the background modeling is not able to solve this as it will by design model the objects visible most of the time. However, if the time the fish – or more general: moving objects – are visible can be reduced, this would aid the background modeling process immensely. To achieve this, a second segmentation approach will be used to create coarse pre-segmentations which are then applied as a mask to reduce the time the foreground objects are visible for the background modeling. These pre-segmentations are created based on the optical flow method because of two reasons. First, the optical flow can be used for change detection and thereby is as general as background subtraction and should in theory segment the exact same objects. Other possible approaches like machine learning algorithms are always limited to one or two specific kinds of objects and therefore their whole aim is already different than for background subtractions methods. The second reason for using optical flow based methods is that they use a completely different approach to detect change in videos. Therefore, they are less likely to suffer from the same problems and create the same errors than background subtraction algorithms. Also, it does not have any training phase and therefore can be used immediately for the learning phase of the background subtraction. As shown earlier, the accuracy of optical flow approaches is often not competitive in underwater scenarios, however, since the method is only used to create coarse pre-segmentations this accuracy is sufficient here. As long as it classifies more pixel correct than false an improvement of the background modeling should be possible by narrowing down the model learning to background pixels. There exist many different approaches for the computation of an optical flow, in this work the Flux Tensor will be used because it has some special advantages in the creation of binary segmentations.

Flux Tensor

Two-dimensional structure tensors have been widely used for edge and corner detection in images, e.g. in [NP05]. They use the information of derivates of the images and are applied as filters on the image which makes them computationally very efficient. Motion information can be recovered in a similar way, but then there has to be a three-dimensional tensor which is applied to an image volume of a video (see e.g. [NG98; PJB04]).

In an image volume each pixel has not two but three coordinates, additionally to the standard i and j coordinates there is also a t coordinate which signifies which frame in the video/volume is meant. So the location $\bar{v} = (i, j, t)$ in the image volume is the pixel (i, j) in frame t. The optical flow $f(\bar{v}) = (f_x, f_y, f_t)$ for $\bar{v}$ is defined by

$$\frac{\partial I(\bar{v})}{\partial x} f_x + \frac{\partial I(\bar{v})}{\partial y} f_y + \frac{\partial I(\bar{v})}{\partial t} f_t = 0. \tag{7.12}$$

This formula can be derived with the assumption that each pixel in the first frame is present again in the next frame

$$I(i, j, t) = I(i + v f_x, j + f_y, t + f_t), \tag{7.13}$$

and the Taylor series

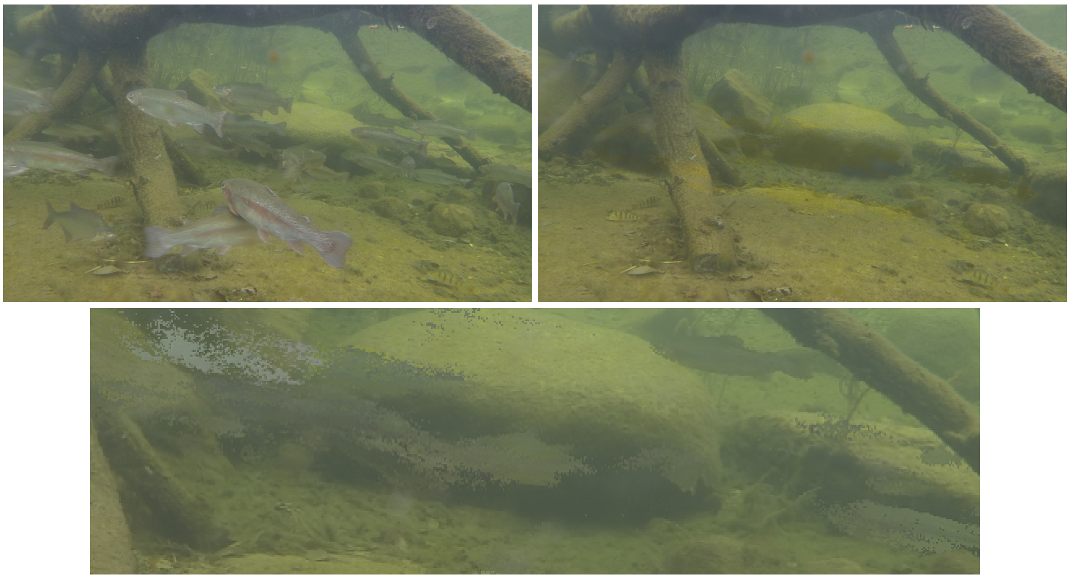


Figure 7.2: On the top left is a frame of the *Fish Swarm* video and the right side the corresponding background model. It can be seen that especially in the middle, where many fish swim by, even the eGSM has difficulties modeling the correct objects. At the bottom is a close-up of this where many fragments of the passing fish are still part of the background model.

$$I(i+f_x, j+f_y, t+f_t) = I(i,j,t) + \frac{\partial I(\bar{v})}{\partial x} f_x + \frac{\partial I(\bar{v})}{\partial y} f_y + \frac{\partial I(\bar{v})}{\partial t} f_t + O(f(\bar{v})^2). \tag{7.14}$$

This, of course, is an idealistic view and for real data an energy function has to be derived and minimized to estimate the theoretical solution. For the *least square measure* the energy function can be written as

$$E(f(\bar{v})) = \int_{\bar{z}\in\Omega} (\nabla I^T(\bar{z}) f(\bar{v}))^2 W(\bar{v},\bar{z}) d\bar{z}. \tag{7.15}$$

The function $W(\bar{v},\bar{z})$ weights the impact of different pixels in the neighborhood Ω based on their distance to the center pixel $\bar{v}$, usually a Gaussian function is used. As $f(\bar{v}) = 0$ would always be an optimal solution for this formulation a normalization factor has to be added so that the final energy function is defined as

$$E(f(\bar{v})) = \int_{\bar{z}\in\Omega} (\nabla I^T_{(\bar{z})} f(\bar{v}))^2 W(\bar{v},\bar{z}) d\bar{z} + \lambda(1 - f(\bar{v})^T f(\bar{v})). \tag{7.16}$$

The impact of the normalization is controlled by λ and for this the minimum can be found by solving the following Eigenvalue problem,

$$A(\bar{v},W) f(\bar{v}) = \lambda f(\bar{v}), \tag{7.17}$$

with

$$A(\bar{v},W) = \begin{pmatrix} \int_\Omega W(\bar{v},\bar{z})\frac{\partial I(\bar{z})}{\partial x}\frac{\partial I(\bar{z})}{\partial x}d\bar{z} & \int_\Omega W(\bar{v},\bar{z})\frac{\partial I(\bar{z})}{\partial x}\frac{\partial I(\bar{z})}{\partial y}d\bar{z} & \int_\Omega W(\bar{v},\bar{z})\frac{\partial I(\bar{z})}{\partial x}\frac{\partial I(\bar{z})}{\partial t}d\bar{z} \\ \int_\Omega W(\bar{v},\bar{z})\frac{\partial I(\bar{z})}{\partial y}\frac{\partial I(\bar{z})}{\partial x}d\bar{z} & \int_\Omega W(\bar{v},\bar{z})\frac{\partial I(\bar{z})}{\partial y}\frac{\partial I(\bar{z})}{\partial y}d\bar{z} & \int_\Omega W(\bar{v},\bar{z})\frac{\partial I(\bar{z})}{\partial y}\frac{\partial I(\bar{z})}{\partial t}d\bar{z} \\ \int_\Omega W(\bar{v},\bar{z})\frac{\partial I(\bar{z})}{\partial t}\frac{\partial I(\bar{z})}{\partial x}d\bar{z} & \int_\Omega W(\bar{v},\bar{z})\frac{\partial I(\bar{z})}{\partial t}\frac{\partial I(\bar{z})}{\partial y}d\bar{z} & \int_\Omega W(\bar{v},\bar{z})\frac{\partial I(\bar{z})}{\partial t}\frac{\partial I(\bar{z})}{\partial t}d\bar{z} \end{pmatrix}. \tag{7.18}$$

The actual computation of the eigenvector $v(\bar{x})$ is very computational expensive and therefore estimation approaches have been suggested, e.g. using $trace(A(\bar{x},W))$. However, this is just a measure of the gradient change in an area and cannot distinguish between spatial and temporal changes. Therefore, a new formulation was proposed in [Bun+07] which gives a better approximation by adding a temporal derivation to every existing derivation in the matrix A,

$$A_F(\bar{v},W) = \begin{pmatrix} \int_\Omega W(\bar{v},\bar{z})\left[\frac{\partial I(\bar{z})}{\partial x}\frac{\partial I(\bar{z})}{\partial t}\right]^2 d\bar{z} & \int_\Omega W(\bar{v},\bar{z})\frac{\partial I^2(\bar{z})}{\partial x\partial t}\frac{\partial I^2(\bar{z})}{\partial y\partial t}d\bar{z} & \int_\Omega W(\bar{v},\bar{z})\frac{\partial I^2(\bar{z})}{\partial x\partial t}\frac{\partial I^2(\bar{z})}{\partial t^2}d\bar{z} \\ \int_\Omega W(\bar{v},\bar{z})\frac{\partial I^2(\bar{z})}{\partial y\partial t}\frac{\partial I^2(\bar{z})}{\partial x\partial t}d\bar{z} & \int_\Omega W(\bar{v},\bar{z})\left[\frac{\partial I(\bar{z})}{\partial y}\frac{\partial I(\bar{z})}{\partial t}\right]^2 d\bar{z} & \int_\Omega W(\bar{v},\bar{z})\frac{\partial I^2(\bar{z})}{\partial y\partial t}\frac{\partial I^2(\bar{z})}{\partial t^2}d\bar{z} \\ \int_\Omega W(\bar{v},\bar{z})\frac{\partial I^2(\bar{z})}{\partial t^2}\frac{\partial I^2(\bar{z})}{\partial x\partial t}d\bar{z} & \int_\Omega W(\bar{v},\bar{z})\frac{\partial I^2(\bar{z})}{\partial t^2}\frac{\partial I^2(\bar{z})}{\partial y\partial t}d\bar{z} & \int_\Omega W(\bar{v},\bar{z})\left[\frac{\partial I(\bar{z})}{\partial t}\frac{\partial I(\bar{z})}{\partial t}\right]^2 d\bar{z} \end{pmatrix}. \tag{7.19}$$

The Flux Tensor is then defined by the trace of this new matrix,

$$\begin{aligned} trace(A_F) &= \int_{\bar{z}\in\Omega} W(\bar{v},\bar{z})\Big[\Big(\frac{\partial^2 I(\bar{z})}{\partial x\partial t}\Big)^2 + \Big(\frac{\partial^2 I(\bar{z})}{\partial y\partial t}\Big)^2 + \Big(\frac{\partial^2 I(\bar{z})}{\partial t\partial t}\Big)^2\Big]d\bar{z} \\ &= \int_{\bar{z}\in\Omega} W(\bar{v},\bar{z})\|\frac{\partial}{\partial t}\nabla I(\bar{z})\|_2^2 d\bar{z}. \end{aligned} \tag{7.20}$$

The additional temporal derivations ensure that only movements (temporal changes) will be detected by this trace.

By computing the Flux Tensor one value per pixel is obtained which represents the magnitude of motion in that area and this can be thresholded to get a binary segmentation. In contrast to most other optical flow approaches the direction of movement is not computed, this information would be useful for the separation of overlapping foreground objects or in tracking tasks. If this additional data is necessary or useful, other optical flow approaches might be favorable (see for example Section 4.2.3), however, for binary segmentations the direction of movement is completely irrelevant. Nonetheless, one problem that occurred was uniform objects, the Flux Tensor does have difficulties segmenting the interior of these objects and often only detects the edges as moving. To cope with this behavior a density-based spatial clustering is applied after the thresholding and the next step is the construction of a convex hull around these clusters of foreground detections. This method can detect most of the moving objects but the created segmentations do not reflect the actual shape of the objects very well. Segmentation methods based on the optical flow, in general, have problems with an accurate shape determination of foreground objects but here this problem is enhanced by the coarse shape estimation of the convex hull. Two examples of both steps of the algorithm can be seen in Figure 7.3.

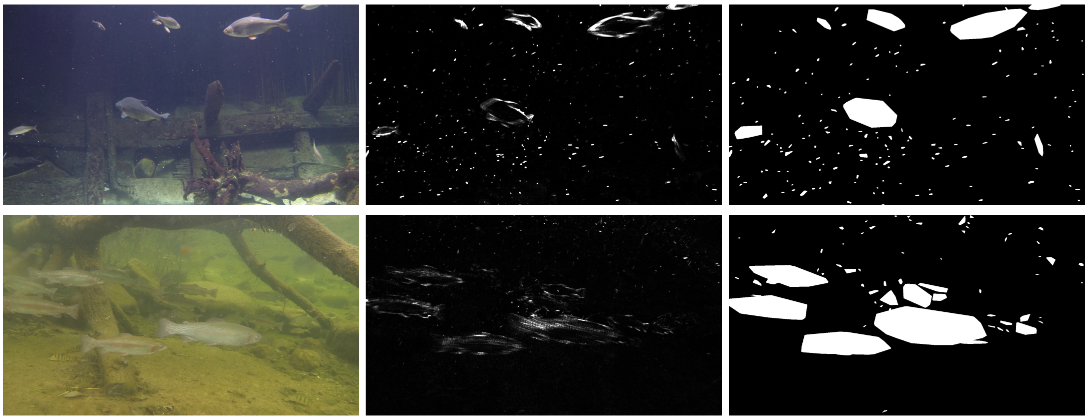


Figure 7.3: The Flux Tensor on two examples with fish as moving objects. The images in the middle show the result of the actual Flux Tensor, higher intensities depict higher detected movement. On the right side is the segmentation after the clustering and building a convex hull around the foreground these clusters. The noise, especially in the upper example, is due to the Marine Snow which get detected very well by optical flow approaches.

To mitigate some of the accuracy problems optical flow approaches have in underwater scenarios more than two frames were used for the computation of the Flux Tensor. For once, this allows the computation of two Optical Flows, one between the current and the previous frame and a second one between the current frame and the next frame. If the results of these two are averaged it becomes less likely to miss movement and thereby the overall result is improved. Furthermore, if more frames are available the computation of the derivates can be more exact, especially the second derivative in the time domain. Therefore, altogether nine frames are used for the computation of the Tensor Flow, namely: the current frame, the last four previous frames and the next four frames of the video. This adds a bit of complexity but mainly increases the memory requirements as all of these frames have to be kept in the main memory and it also makes it necessary to always wait until the next four frames of the video stream were received until the computation of the Flux Tensor can begin. However, these are only small limitations and the overall accuracy can be improved by this usage of more frames. The runtime is even then only around 0.025 seconds for one HD frame on one 3.4 GHz CPU Core and therefore the algorithm can easily be run in real-time.

For a new frame of the video the Flux Tensor segmentation is first computed and then used to exclude all detected foreground areas from the updating process of the background model. This can reduce the number of foreground objects the background model is exposed to drastically and thereby prohibits the inclusion of foreground information into the background model. The principle is similar to that of the partial updating scheme which excluded pixels that are classified as foreground from the updating. However, this does only work if the background model is already accurate and a good segmentation can be provided. In a scene that is constantly crowded no reliable background model can be created and therefore the partial updating scheme fails. Here the pre-segmentations show their strength as they help to build an accurate model in the first

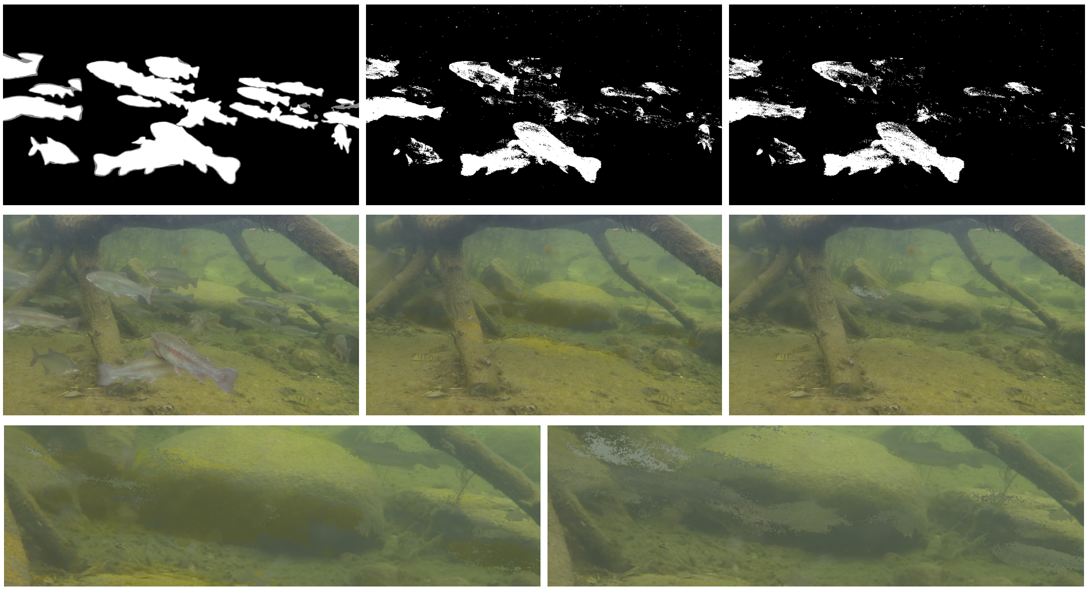


Figure 7.4: Effect of the Flux Tensor pre-segmentations on the background modeling. In the top row are from left to right the ground truth image, the segmentation of the eGSM method with pre-segmentation and the same without pre-segmentation. Below that are the original frame and the corresponding visualizations of the two background models. The last row shows a close-up of the background models in an area where many fish were passing by. The model created with pre-segmentations (left) has fewer artifacts of fish and is also not as blurry.

place. If a reliable model has once been established the additional pre-segmentations can even be deactivated since the partial updating scheme will prevent the corruption of the model, at least for a long time. An example of this effect can be seen in Figure 7.4 where the *Fish Swarm* video from the UCD dataset was segmented with the eGSM, once with and once without the usage of these pre-segmentations. The visualization of the background model is created by taking the Gaussian with the highest weight of the partially updated MoG and displaying the mean of it. When eGSM is used hereafter, the pre-segmentations are always included.

7.3 Spatial Methods

Similar to the in-air case a spatial method should be added here as well. The assumptions of natural images apply equally to underwater videos and, therefore, the same methods that were discussed in Section 4.2 should be useful. However, the degradation effects might have negative (or positive) impacts on the effectiveness of these methods and this will be evaluated in the following on the UCD dataset.

Markov Random Fields

Like for the in-air footage of the Wallflower dataset the MRF model can smooth the segmentations very naturally. Also, small false detections due to noise are removed but larger defects in the segmentation are preserved by the MRF because it does not use the image itself but only the probability map derived from the background subtraction. Therefore, it cannot repair large errors of the background subtraction but only make them spatially more coherent. Examples of this can be seen in Figure 7.5, in the *small Aquaculture* scene some areas at the right side are so noisy that they are not removed and in the *Marine Snow* scenario are some small shadows that are removed after the application of the MRF model but the large shadow in the left bottom of the scene stays since it is too large. Overall, it gave the visually most accurate and pleasant result and does not need any adaption for the underwater scenario since the assumptions for natural images apply equally underwater.

N^{2}Cut

The N^{2}Cut from Section 4.2.2 adapts the segmentations from the background subtraction to edges in the current frame. This can produce great results if there are strong edges between the foreground objects and the background, an example of this are the results on Wallflower dataset (see Figure 4.14). Unfortunately, underwater scenes are usually quite blurry and this makes the adaption to existing edges in the frame difficult. This can lead to a situation where the adaption process does not stop, since it does not find any suitable edge, and degenerates the segmented objects to simple rectangles or lets them vanish completely. The impact of the N^{2}Cut on the segmentation is regulated by the different sizes of the frame it is run on. If the image is downscaled the effect and range of the N^{2}Cut grow drastically with each additional layer in the image pyramid (see Section 4.2.2) and similarly the effect is locally very limited if it is only used on high-resolution images without downscaling.

To prevent this degeneration in underwater images without sharp edges it is necessary to limit the effect of the N^{2}Cut and only run it on the largest scale. With this restriction, slight adaptions of the borders are still possible but the deletion of small false detection (e.g. Marine Snow) can often not be done anymore by the N^{2}Cut as the effect is too locally limited. Hence, it was necessary to introduce a minimum object size so that all detected objects below a certain size are counted as noise and will be deleted. This minimum size takes care of larger errors like Marine Snow detections or noisy areas while the N^{2}Cut can still adapt the borders of the larger objects slightly and closes small holes inside of them. In Figure 7.5 are two examples of the effect of the N^{2}Cut depicted, it was only run on the original size of the frame and the minimum size of objects was 1000 pixels.

Temporal Trajectories

The positive effect of the temporal trajectories is mostly limited by the accuracy of the dense optical flow (DOF) which is used to compute the trajectories over several frames of the video. As already discussed, Optical Flows tend to lose accuracy in underwater videos because there are less detectable features and less contrast between moving and static objects. Therefore, the impact on the segmentations, although overall positive, is lower than in the previously presented methods and the segmentations often stay fragmented. An example can be seen in Figure 7.5 where it was run on the *two fish* video and the *Fish Swarm* scene of the UCD dataset. The problem, especially on the *Fish Swarm* video, is that many fish are not detected by the optical flow as they are too similar to the background. Therefore, many areas that were correctly detected at first are afterwards classified falsely as background. Overall, this method seemed to cope the worst with the special situation in underwater scenes and one of the two previous methods (N^2Cut or MRF) should be preferred.

7.4 Evaluation

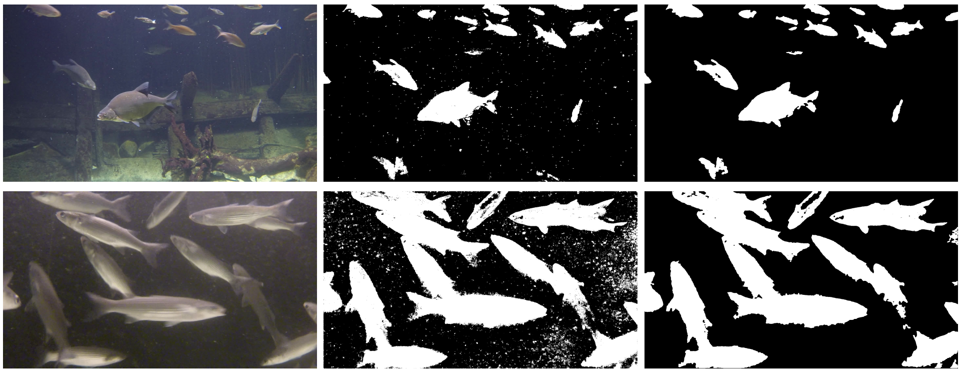
For the evaluation on the UCD dataset the three background subtraction methods which have implementations in OpenCV are used again, similarly to the evaluation of the underwater image enhancement. The implementation of the algorithm from [KB02] is done in CUDA and hence runs on the graphics card and is faster than the other two who are executed on the normal CPU, but all of them are fast and capable of real-time segmentation of HD videos. An optimization of the parameters of these algorithms was done, but no real positive effect could be achieved for the implementations of [Ziv04] and [ZH06], it appears they have an automatic parameter adaption that works very well. For [KB02] the optimization had a significant effect, the overall F1-Score went from 0.4513 to 0.7527.

A comparison between these methods, the original GSM and the proposed algorithms, is given in Table 7.1 and 7.2. It shows that the eGSM (including the pre-segmentations) is a substantial improvement to the original GSM on each of the five videos and also outperforms the other methods on the whole dataset. In the *Fish Swarm* video the largest improvement could be achieved, mainly because of the pre-segmentations which made it possible to create a better background model of that scene. Nonetheless, not all fish in the *Fish Swarm* video could be detected since some of them barely move or are almost indistinguishable from the background. In the other four videos of the dataset the fish could be detected very reliable by the eGSM approach and the problems there mostly consist of false detections of shadows caused by the fish or caustics on the water surface. It is a complicated task to avoid these errors since the algorithm needs to be very sensitive to detect fish even when they are similar to the background which then causes these false detections. Also, the best indicator to distinguish shadows from real objects is the hue because shadows darken the scene but the hue should stay almost constant when a shadow appears. In contrast to that, a real moving object will change the brightness and the hue. However, this color information is deteriorated in underwater scenes which makes the detection of shadows even more complicated.

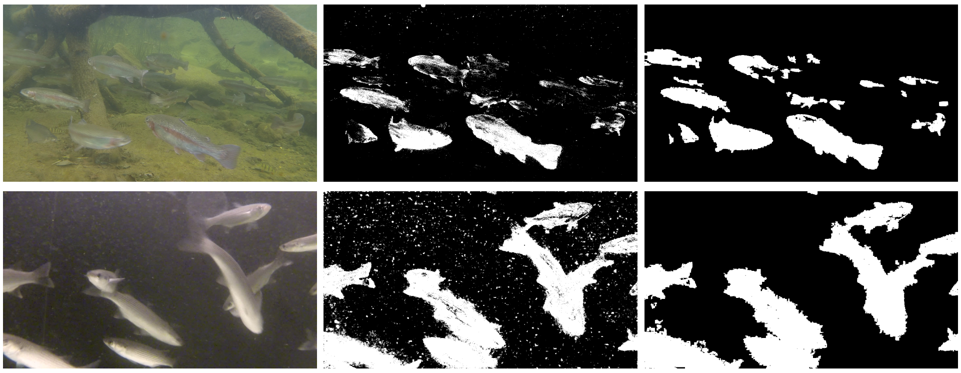
The usage of this eGSM approach on in-air scenes did not provide any substantial improvement. For the Wallflower dataset the number of falsely classified pixels decreased slightly from 9718 (normal GSM) to 9600 (eGSM). The advancement here is mainly because of the advanced background model updating algorithm, the pre-segmentations have almost no influence.

The addition of any of the three presented spatial methods could further improve the results on the underwater videos, not only visually – as already shown – but also in a concrete comparison against the ground truth data. In Table 7.1 it can be seen that the temporal trajectories performed worst, as already assumed, with only a minor improvement overall, they reduce the false positives

MRF



N^2Cut



Temporal Trajectories

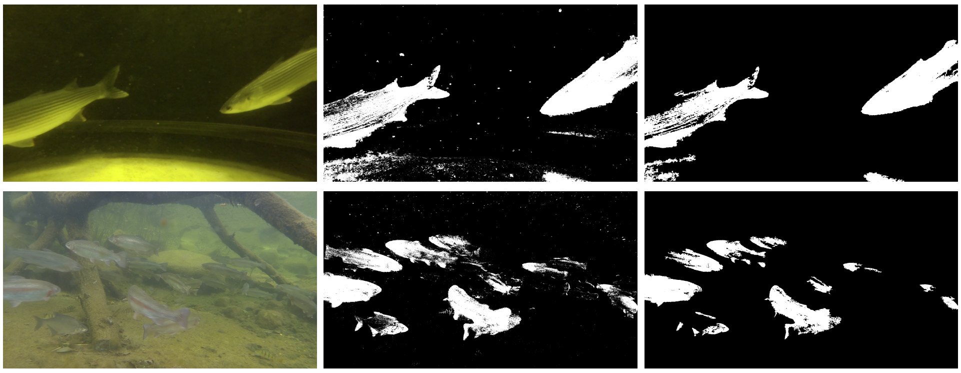


Figure 7.5: The effect of the three different spatial methods on the UCD datasets. On the left is always the original image, in the middle the segmentation with the eGSM background subtraction and on the right side is the result after the corresponding spatial method was included.

drastically but at the same time increase the number of false negative classifications. The MRF model achieved a drastic reduction in false positive classifications without affecting the other categories negatively and therefore showed a more substantial improvement in the aggregated results. Nonetheless, the best segmentations by far were created by the N^2Cut which – in contrast to the other two methods – was able to increase the number of true positive results. This is a bit surprising as the lack of sharp edges should obstruct the N^2Cut but the limitation of the effect of the N^2Cut in combination with the minimum object size seem to work together very effectively.

It should be noted that the results for each video differ drastically, e.g. the temporal trajectories can achieve the best results of all algorithms on the *Caustics* video and the greatest improvement among the spatial methods in the *two fish* video. In the *Fish Swarm* video, however, it decreased the results significantly and therefore could not compete in the overall results with the other approaches. In Figure 7.6 some results of the eGSM in combination with the N^2Cut are depicted and compared with the ground truth images.

The addition of underwater image enhancement to this advanced method has similar effects as discussed in Chapter 5 with the standard GSM. It would be too much and confusing to now test all different background subtraction methods with all different spatial methods and all image enhancement approaches, especially since the effect is very similar, independently from the specific background subtraction of spatial method. As an example only the combination of the most promising methods – eGSM+N^2Cut with a previous MSR – shall be presented, it could achieve an overall F1-Score of 0.8532, which is a noticeable improvement over the 0.8371 without MSR. The improvement was, as expected, mainly in the videos with strong Marine Snow (Two fish video: from 0.8245 to 0.8977 or Marine Snow video: from 0.9100 to 0.9252) and in other videos there was hardly an effect or even a decrease in accuracy (Fish Swarm video: from 0.8459 to 0.7821).

Algorithm	True Negative	True Positive	False Negative	False Positive	F1-Score
[Ziv04]	897,659,412	76,555,440	52,934,753	11,723,995	0.6713
[ZH06]	887,967,097	93,919,051	36,656,258	20,331,194	0.7515
[KB02]	872,346,370	106,827,320	40,266,356	13,682,950	0.7527
GSM	892,599,998	84,349,046	44,709,358	17,215,198	0.7036
eGSM	879,524,562	111,071,329	20,976,510	25,227,599	0.7965
eGSM+Trajectories	888,905,557	98,456,774	29,168,633	9,901,036	0.8159
eGSM+MRF	887,107,298	110,810,067	20,803,490	8,079,145	0.8194
eGSM+N^{2}Cut	872,360,637	116,926,034	30,167,642	17,345,687	0.8371

Table 7.1: The results of the proposed algorithms and three different background subtraction methods on the UCD dataset. The result of the normal GSM is amongst these three approaches and only by extending the GSM can a considerable improvement over existing methods be achieved. These results can be further enhanced by adding a spatial method. Here, the N^{2}Cut showed the most significant improvement, similar to the in-air scenes of the Wallflower dataset. The amount of foreground (TP+FN) is not constant because of the small uncertainty area (gray) in the dataset. The F1-Score here is the average of the F1-Scores of the five videos (see Table 7.2), the other values are the sums.

Video Algorithm	Fish Swarm	Marine Snow	small Aquaculture	Caustics	two Fish
[Ziv04]	0.3033	0.8182	0.7030	0.7383	0.7938
[ZH06]	0.5904	0.8244	0.8828	0.7533	0.7068
[KB02]	0.6320	0.8162	0.8600	0.6048	0.8507
GSM	0.5691	0.8361	0.7734	0.5499	0.7898
eGSM	0.7537	0.8908	0.8684	0.6524	0.8174
eGSM+Trajectories	0.6660	0.9104	0.9021	0.7553	0.8456
eGSM+MRF	0.7514	0.9053	0.9133	0.7046	0.8223
eGSM+N^{2}Cut	0.8459	0.9100	0.9332	0.6719	0.8245

Table 7.2: The F1-Scores of the algorithms from Table 7.1 for each video of the dataset separately.

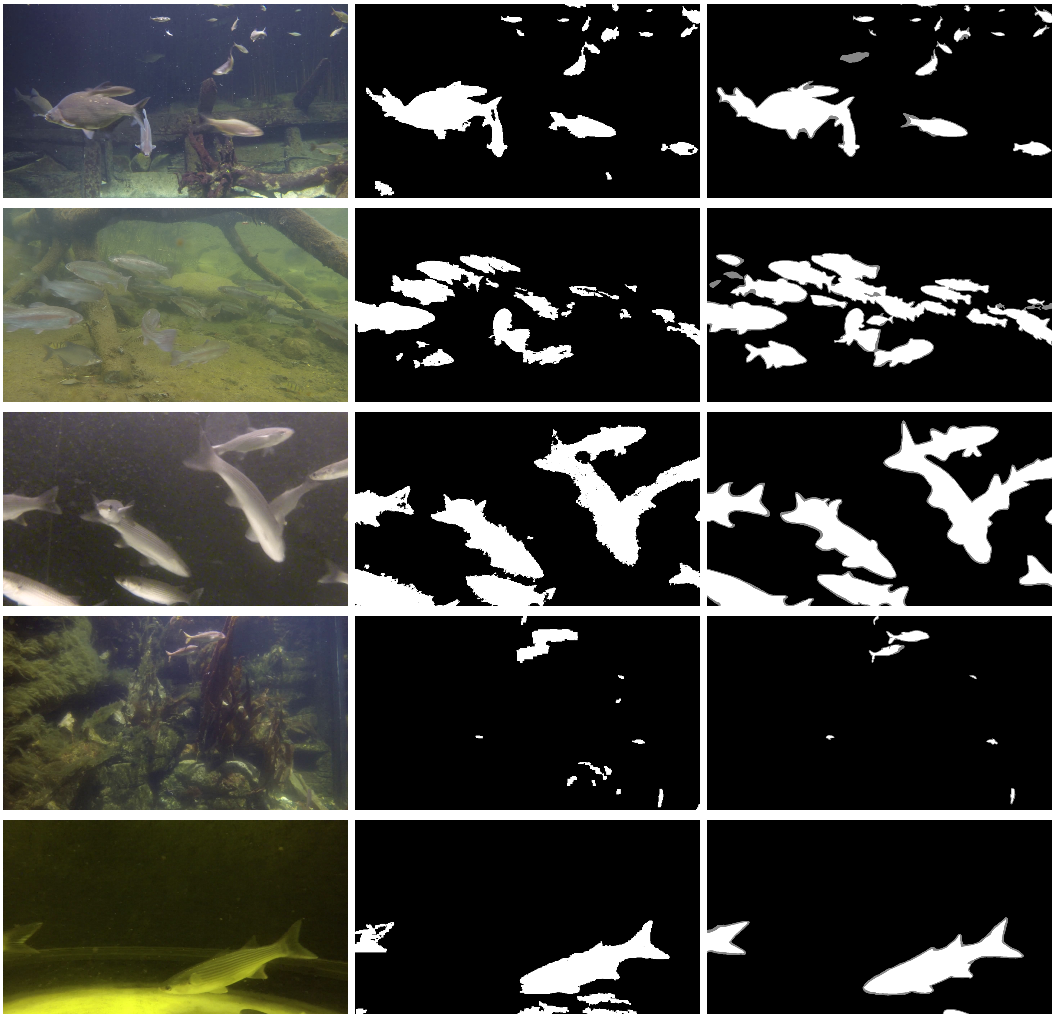


Figure 7.6: One frame of each of the five videos of the UCD dataset. From top to bottom are shown the videos: *Marine Snow*, *Fish Swarm*, *small Aquaculture*, *Caustics* and *two fish*. In the middle column are the segmentations of the proposed approach and on the right the hand-segmented ground truth images.

7.5 Conclusion

After dealing with the general change detection in the first part of this thesis and then introducing the UCD dataset and evaluating different image enhancement methods on it in the second part; this chapter dealt with the adaption of the presented background modeling approach to this new and difficult environment. The first steps were the merging of the GSM with the MoG idea, adding a foreground model and making the parameters more adaptive to the current scene. This made the whole process slower and more complex but showed great improvements in accuracy on the UCD dataset. Nonetheless, for simpler in-air footage like the Town Center video [BR11], usually, the normal GSM is usually still sufficient and preferable because of its lower runtime and complexity. Another special difficulty of underwater scenes is crowded scenes with many fish due to the swarm behavior of them. For this reason, a pre-segmentation based on the Flux Tensor computation of the optical flow was introduced that improved the background modeling drastically by filtering moving objects out of the modeling process at the very beginning. In the *Fish Swarm* scene of the UCD dataset this problem was most prominent and, consequently, the greatest improvement could be achieved on it with the new eGSM in combination with pre-segmentations (from 0.5681 to 0.7537 F1-Score). However, also in the other four videos these changes could improve the results significantly.

In a next step, the spatial methods of the previous section were evaluated. They showed similar results to the in-air footage of the Wallflower dataset. All of them could improve the results of the pure eGSM approach and their rankings, as well as weaknesses and strengths, were almost the same as for the in-air dataset. One difference was that the temporal trajectories were overall a bit worse than the MRF method because the computation of the dense optical flow in underwater videos is not as reliable. Because of this unsteadiness in the computation of the optical flow, the results of the temporal trajectory method were quite variable; in some videos it excelled (e.g. *Caustics* where it could produce the most accurate results) and in others it made the eGSM results worse (*Fish Swarm*). Overall, the N^{2}Cut proved to be by far the best method as it consistently increased the results in all five videos, achieved the best overall accuracy and is also the method with the lowest runtime.

In the end, the results of the spatial methods for the underwater videos were quite similar to the in-air case discussed in Chapter 4, especially for the N^{2}Cut approach. This still holds if any of the four image enhancement methods from Chapter 6 is added. Therefore, the combination of eGSM + N^{2}Cut with MSR as a preprocessing step is the overall preferred method for underwater change detection.

Consequently, the eGSM + N^{2}Cut combination will be used in the next chapter to create the segmentations for the tracking approach. From the runtime perspective, the eGSM without pre-segmentations takes about three times as long as the normal GSM because of the advanced updating mechanism. When pre-segmentations are used (and when not mentioned otherwise they always are), this increases slightly to around four times of the GSM runtime. With a parallelization similar to the one described in Figure 4.10 the segmentation of around 10 HD frames per second is possible with an eGSM + N^{2}Cut combination. The memory consumption, in this case, is around 700 MB. For the tracking in underwater scenes the MSR is added to this combination. It needs about 1 second per HD frame (Single Core on a Desktop PC), and therefore, a real-time capability is hard to achieve, even with parallelization. However, the memory consumption is very low with only 40 MB.

8 Underwater Blob Tracking

To extract valuable information from the segmentations derived from change detection, e.g. about the behavior of fish, it is necessary to associate the detected objects with found objects in the previous (and succeeding) frames. With these associations, a tracking of individual objects can be realized and allows the computation of movement speeds, paths and other valuable higher-level features. Change detection is a very general approach and not limited to a specific object type, nor has it the need for a learning phase for every new object class. Therefore, the tracking approach used should be also very general and not based on any object-specific features (e.g. eyes, faces, colors, etc.). Furthermore, the similarity of the detected fish in a swarm can be a great problem. When humans or cars are tracked there are usually quite notable differences between objects, e.g. in the size (car or truck), color (blue jacket or green jacket) or shape (thin or big). These differences result in distinct features for each object which can then be matched between different frames [XLL12; Cam+16]. For different fish of the same species – or even a swarm – these features have to be expected to be very similar and additionally degraded because of the underwater scenario. Another distinguishing attribute which is often used in in-air tracking is depth, however, it is very hard to obtain depth information in underwater scenes because of the refraction and absorption properties of water.

Based on this, the here presented tracking approach will solely rely on the information contained in the segmentation provided by the change detection. This allows a general usage of the approach in underwater scenarios as well as in-air videos to track fish, cars, humans or any other moving object. The strategy used here is in sharp contrast with most other tracking methods where a specific object detector is trained, e.g. [Shu+12] uses Haar-like features and a SVM classifier to detect humans. The tracking of any arbitrary foreground detections made by a change detection approach can become extremely difficult, especially when many objects are present in the scene at the same time and overlap each other. However, this strategy preserves the generality of the overall approach and can distinguish even between very similar objects. The first step in the building of this tracking method is a matching function that evaluates how similar two foreground detections are.

8.1 Matching Function

At first, all components in the binary segmentations must be extracted so that they can then be matched against each other. For this, a random foreground pixel is chosen in the segmentation as the start of a new component. Then all neighboring foreground pixels are added to this component and this is repeated for all newly added foreground pixels until no further foreground pixels can be added and then a new, so far unused, pixel is chosen to start the next component. These components will be called connected components since each pixel in it can be connected to each other pixel in that component by a path, a result of this extraction of connected components can be seen in Figure 8.1.

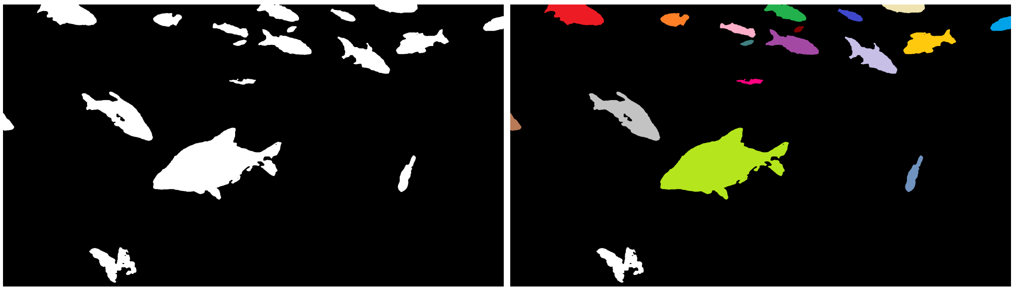


Figure 8.1: On the left is a segmentation result of the previously described change detection approaches and on the right side is each connected component colored differently.

Each of these connected components has a list of properties that is used to compare them against each other, these properties are:

Properties of connected components

- A unique identifier
- Height and width of the bounding box in pixels – BBh, BBw
- Center point of the bounding box – BBx, BBy
- Number of pixels that belong to that connected component – NoP
- Growth rate: change of the NoP in pixels per frame – GR
- Coordinates of the centroid of the connected component – (Cx, Cy)
- Direction of movement (of the centroid) – (Dx, Dy)
- Velocity in pixels per frame (of the centroid) – Vel
- A vector that includes all pixel positions of that component

All properties that rely on a trend/history are initialized as zero (velocity, growth rate) except the direction of movement which is initialized as $Dx = \sqrt{\frac{1}{2}}$ and $Dy = \sqrt{\frac{1}{2}}$ since it is always normalized to 1. These properties require two matched components to be computed since they are defined as a change over time, e.g. the GR is defined as the change in the NoP between two components. With this information the properties get updated in the fashion of a running Gaussian with each new match in the following ways

$$\begin{aligned}
GR_n &= \alpha \cdot GR_{n-1} + (1-\alpha) \cdot NoP_n - NoP_{n-1}, \\
Vel_n &= \alpha \cdot Vel_{n-1} + (1-\alpha) \cdot \sqrt{(Cx_n - Cx_{n-1})^2 + (Cy_n - Cy_{n-1})^2}, \\
\widetilde{Dx_n} &= \alpha \cdot Dx_{n-1} + (1-\alpha) \cdot \frac{|Cx_n - Cx_{n-1}|}{|Cx_n - Cx_{n-1}| + |Cy_n - Cy_{n-1}|}, \\
\widetilde{Dy_n} &= \alpha \cdot Dy_{n-1} + (1-\alpha) \cdot \frac{|Cy_n - Cy_{n-1}|}{|Cx_n - Cx_{n-1}| + |Cy_n - Cy_{n-1}|}.
\end{aligned} \tag{8.1}$$

The Direction of movement is afterwards normalized

$$\begin{aligned}
Dx_n &= \frac{\widetilde{Dx_n}}{\widetilde{Dx_n} + \widetilde{Dy_n}}, \\
Dy_n &= \frac{\widetilde{Dx_n}}{\widetilde{Dx_n} + \widetilde{Dy_n}}.
\end{aligned} \tag{8.2}$$

The α was chosen as 0.9 in the equations above. In this way, one wrong detection of a component does not completely change these parameters, but they can still quickly adapt to changes (in around 5 to 10 frames).

The other values can be directly inferred from the connected component itself. With these properties, a measure of similarity between two components can be derived which will be called Connected Component Similarity Measure ($CCSM$) in the following. To compute the CCSM it is assumed that two connected components (CC_1 and CC_2) are from consecutive frames. The lower the $CCSM$ value, the more likely it is that these components represent the same object of the scene and vice versa. The first part of the $CCSM$ is defined as

$$\begin{aligned}
CCSM_a = & |BBh_1 - BBh_2| + |BBw_1 - BBw_2| && (8.3) \\
& + \sqrt{|(NoP_1 + GR_1) - NoP_2|} && (8.4) \\
& + |(Cx_1 + Dx_1 \cdot Vel_1) - Cx_2| + |(Cy_1 + Dy_1 \cdot Vel_1) - Cy_2| && (8.5) \\
& + |(BBx_1 + Dx_1 \cdot Vel_1) - BBx_2| && (8.6) \\
& + |(BBy_1 + Dy_1 \cdot Vel_1) - BBy_2|. && (8.7)
\end{aligned}$$

In 8.3 the sizes of the different bounding boxes are compared under the assumption that they should be very similar if the connected components are representing the same object in consecutive frames. In the next line, the number of pixels is evaluated where $NoP_1 + GR_1$ is the estimated number of pixels in the next frame and should, therefore, be close to NoP_2. Then the 2nd root is taken of that value because it describes a two-dimensional property (the pixels can lie in x- and y-direction) whereas the other values describe one-dimensional values (e.g. length of the bounding box in x-direction). In 8.5 the positions of the Centroids are compared and $Cx_1 + Dx_1 \cdot Vel_1$ is the estimated x-coordinate in the next frame which is compared to Cx_2. It should be noted that the direction of movement is always scaled so that $\|(Dx, Dy)\|_2 = 1$ and therefore $Dx \cdot Vel$ is the

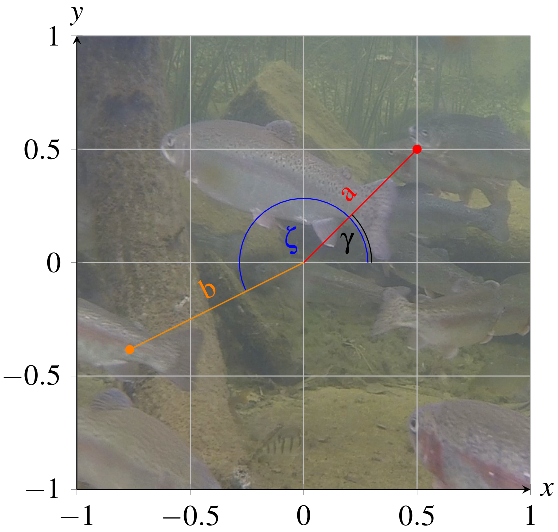


Figure 8.2: In the standard coordinate system the pixel marked with a red dot is at position $(0.5, 0.5)$. If the coordinate system is converted to polar coordinates it would be at (γ, a) with an angle γ of 45° and a length $a = \sqrt{(0.5^2 + 0.5^2)}$. The same applies to the pixel marked in orange, which has the polar coordinates (ζ, b) in the image.

estimated movement in x-direction in pixels per frame. A similar comparison is done in 8.6 and 8.7 with the middle point of the bounding box.

One important aspect that is missing from the CCSM so far is the shape of the detected foreground object. For this, the bounding box of the connected component is taken and its coordinate system is transferred to polar coordinates in a new frame with a fixed size of 360×360. For this, the center of the bounding box is taken and from there the image is sampled in 360 different angles with a sampling rate of 360. In Figure 8.2 the conversion of Euclidean to polar coordinates is illustrated and in Figure 8.3 an example is given of a connected component in Euclidean and in polar coordinates.

In order to compare the shapes of two connected components, both are transferred to polar coordinates and then five different scores are computed. These scores work under the assumption that the same detected foreground object in two consecutive frames has a barely changed 3D rotation and pixel size, that means, for example, that the distance and angle to the camera are similar in both frames. Therefore, the aim of this shape comparison is not to be rotational and size invariant in contrast to most of the recent work in shape recognition where exactly this is the objective [WG16; Abu+12]. In this scenario, since fish have very similar shapes, especially if they are of the same species, the rotation of the fish in the scene can be an important indicator to differentiate between fish in an underwater scene and should be used.

The first shape feature that is computed is the percentage of detections (or non-detections) that do not match in both polar images and it can be seen as a general measure of the similarity of the two detected objects. Let PI^1 and PI^2 be the binary polar images of two connected components and $PI_{(\xi,l)}$ the value at angle ξ and length l, the shape feature is then defined as

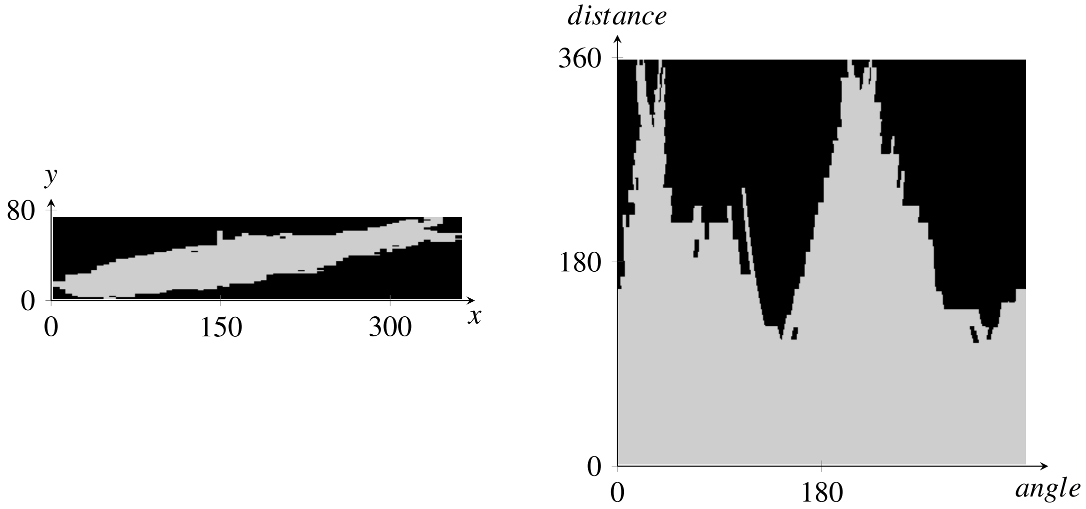


Figure 8.3: On the left side is the bounding box of a detected fish from the eGSM approach that has the dimensions $(364, 73)$. This bounding box is converted to polar coordinates to compare the shapes of different foreground objects, the result of the transformation is depicted on the right side. After the conversion, the detections have always the size of $(360, 360)$ to simplify the comparisons between them.

$$Sh_a = 1 - \left[\sum_{\xi=1}^{360}\sum_{l=1}^{360} g(PI^1_{(\xi,l)}, PI^2_{(\xi,l)})\right] / (360 \cdot 360), \tag{8.8}$$

with

$$g(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{else} \end{cases}. \tag{8.9}$$

The 360 in the equation comes from the size of the polar images which is fixed to 360×360 because of the sampling rate. The second feature is based on a comparison of the number of foreground pixels per angle of the two frames. Since no rotation is assumed the same object should have a similar extension in each angle in both frames, this is checked with:

$$Sh_b = \left[\sum_{\xi=1}^{360}\left|\left(\sum_{l=1}^{360} g(PI^1_{(\xi,l)}, 1)\right) - \left(\sum_{l=1}^{360} g(PI^2_{(\xi,l)}, 1)\right)\right|\right] / (360 \cdot 360) \tag{8.10}$$

The third and fourth features are similar but evaluate each pixel separately and give them different weights based on their distance to the center of the original detection. The third feature focuses on pixels close to the center and the fourth on pixels at the outside. Furthermore, the standard Euclidean metric is applied so that outliers have a stronger impact.

$$Sh_c = \left[\sum_{\xi=1}^{360} \sum_{l=1}^{360} \sqrt{ \left(\frac{l}{360} \cdot g(PI^1_{(\xi,l)},1) - \frac{l}{360} \cdot g(PI^2_{(\xi,l)},1) \right)^2 } \right] / (360 \cdot \sum_{l=1}^{360} \frac{l}{360}) \tag{8.11}$$

$$Sh_d = \left[\sum_{\xi=1}^{360} \sum_{l=1}^{360} \sqrt{ \left(\frac{360-l}{360} \cdot g(PI^1_{(\xi,l)},1) - \frac{360-l}{360} \cdot g(PI^2_{(\xi,l)},1) \right)^2 } \right] / (360 \cdot \sum_{l=1}^{360} \frac{l}{360}) \tag{8.12}$$

The last shape feature compares the contours of both connected components. First, both polar images are reduced to their contours by scanning each angle separately and only keeping the pixels as foreground that have a background neighbor along that angle. The result of this are the binary contour images $Cont^1$ and $Cont^2$ with the same dimensions as PI^1, an example of this can be seen in Figure 8.4. Afterwards, the contours are compared by computing for each angle the minimal distance in pixels from a contour pixel in one of the connected components to any contour pixel in the same angle in the other connected component. It is computed in the following way

$$\begin{aligned} Sh_e = \sum_{\xi=1}^{360} \sum_{l=1}^{360} \Bigg[& g(Cont^1_{(\xi,l)},1) \cdot \min_{\substack{l_2=-360...360 \\ g(Cont^2_{(\xi,l_2)},1)>0}} |l-l_2| \\ & + g(Cont^2_{(\xi,l)},1) \cdot \min_{\substack{l_2=-360...360 \\ g(Cont^1_{(\xi,l_2)},1)>0}} |l-l_2| \Bigg] / (2 \cdot 360), \end{aligned} \tag{8.13}$$

where first $Cont^1$ is compared with $Cont^2$ and then the other way around. The term $g(Cont^1_{(\xi,l_2)},1)$ is just a check whether the binary contour image is true or false at that position. Furthermore, there appear negative length values in this expression and they imply that the original connected component is scanned in the opposite angle and therefore:

$$Cont_{(\xi,-l)} = Cont_{(\beta,l)} \quad \text{with} \quad \beta = (\xi + 180) \mod 360. \tag{8.14}$$

Finally, with these five shape features the shape component of the CCSM can be easily derived,

$$CCSM_b = \sqrt{ \sum_{k \in \{a,b,c,d,e\}} Sh_k^2 }. \tag{8.15}$$

After the first two components that compared general properties and the shapes, the third and last component of the CCSM focusses on outlier detection. It prevents the matching of connected components that are e.g. in completely different parts of the frame or have very different sizes but are otherwise very similar. This is done by using the exponential function so that a large disparity in one parameter dominates the whole CCSM value. Otherwise the computation is resembling calculation of the first component ($CCSM_a$) with the Equations 8.3 to 8.7. It is defined as

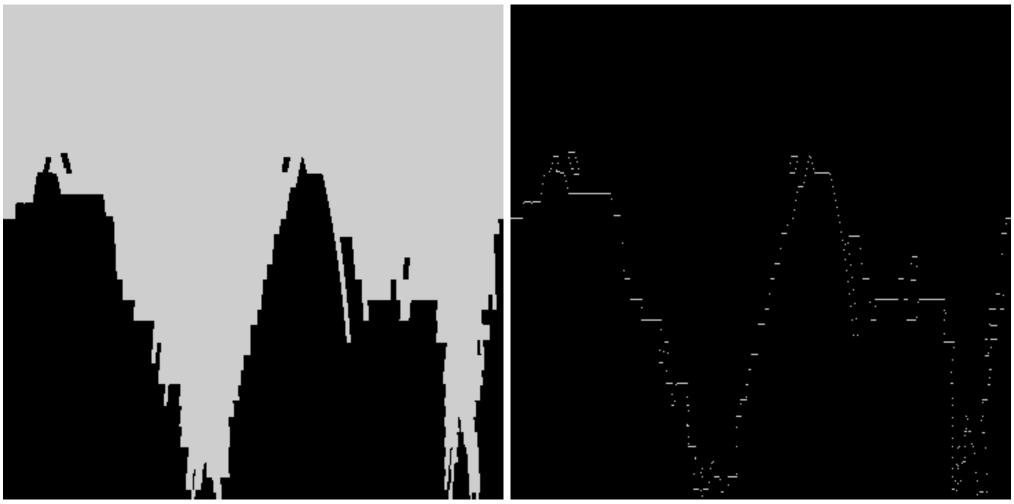


Figure 8.4: On the left side is a polar image of a fish and on the right the contour of that polar image is displayed.

$$
\begin{aligned}
CCSM_c = \exp\left(|Cx_1 + Dx_1 \cdot Vel_1 - Cx_2| - z\right) && (8.16)\\
+ \exp\left(|Cy_1 + Dy_1 \cdot Vel_1 - Cy_2| - z\right) && (8.17)\\
+ \exp\left(|BBh_1 - BBh_2| - z\right) && (8.18)\\
+ \exp\left(|BBw_1 - BBw_2| - z\right) && (8.19)\\
+ \exp\left(|(BBx_1 + Dx_1 \cdot Vel_1) - BBx_2| - z\right) && (8.20)\\
+ \exp\left(|(BBy_1 + Dy_1 \cdot Vel_1) - BBy_2| - z\right) && (8.21)\\
+ \exp\left(|(NoP_1 + GR_1) - NoP_2| - 0.15 \cdot (NoP_1 + NoP_2)\right), && (8.22)
\end{aligned}
$$

where z is a parameter that was set empirically and is based on the image size

$$z = \frac{\sqrt{FrameHeight \cdot FrameWidth}}{25}. \tag{8.23}$$

The subtraction of z in conjunction with the exponential function ensures that only properties which are very different (outliers) contribute substantially to the $CCSM_c$ component. For example, if the heights of the bounding boxes are less than z pixels apart, the element 8.18 will contribute less than 1 to the overall sum. However, if the disparity is larger than z it will increase very quickly so that basically all connected components where the difference is greater than z will be rejected. The same applies to all the components of the $CCSM_c$ in the lines 8.16 to 8.21 and in 8.22 it is checked whether the overall number of pixel is not changing by more than 15% in consecutive frames. With these three components, the final $CCSM$ is computed in the following way

$$CCSM = CCSM_b \cdot CCSM_a + CCSM_c. \tag{8.24}$$

The multiplication of the values $CCSM_a$ and $CCSM_b$ ensures that both values are important for the matching, even though the $CCSM_b$ is usually much smaller than the $CCSM_a$. The $CCSM_c$ should always be very small for correct matches and its only purpose is the rejection of wrong matches.

8.2 Optimal Match Finding

To now compute the best matches between the connected components of two frames according to the $CCSM$, the first step is the creation of a list of all connected components for each of the two frames. However, it has to be considered that two connected components in one frame can merge into one larger component in the next or vice versa. This can be caused by two or more fish swimming in front or behind each other but also by segmentation errors of the change detection approach, two examples of this can be seen in Figure 8.5. To take this into account, not only the single connected components are considered but also the unifications of two or more connected components that are close to each other. This means that if three connected components are near to each other, that the three single components would be considered for matching but also the three possible different unions of two of them as well as the union of all three of them.

There exist methods for the task of computing the best overall matchings between two frames, e.g. the Hungarian Method which was originally introduced by Kuhn in [Kuh55] and is still often used today [LZF12]. However, since the matching method applied here should also take unions and splits of connected components into consideration, it cannot be used. To find the globally best matching between two frames gradually matching components (e.g. greedy approach) will not be enough, the algorithm must look at all objects and their possible matches at the same time. When splits and unions are allowed this becomes extremely difficult since each component can be a part of many splits and/or unions. Therefore, an adaption of this method that takes unions and splits into account is probably not feasible. The here presented method focuses on finding a local optimum in a shorter runtime.

Considering the possibility of unions of separate connected components makes the handling of splits and merges more accurate but can also be very computational intensive when many smaller components are close to each other. For n components the number of possible combinations is

$$\sum_{k=1}^{n} \binom{n}{k} = \sum_{k=1}^{n} \frac{n!}{(n-k)! \cdot k!}, \tag{8.25}$$

which means e.g. 15 combinations for $n = 4$ or 31 for $n = 5$. If many small objects are expected in a scene and time is a concern, it is, therefore, advisable to limit the number of possible unions, e.g. by only considering unions of at most two or three components. To be considered for a union the connected components must be close enough together. Here the minimal distance between the borders is important since that is the region where a split or merge would most likely happen. Therefore, to be considered close, two pixels have to exist (one from each connected component) that have a distance of less than T_{CCdist} pixels between them. The threshold is dependent on the resolution of the video and, therefore, it is set to $T_{CCdist} = z$ (compare with equation 8.23).

A union of two or more connected components consists of all the pixels that belong to each of the connected components which are combined. Therefore, the same properties can be derived

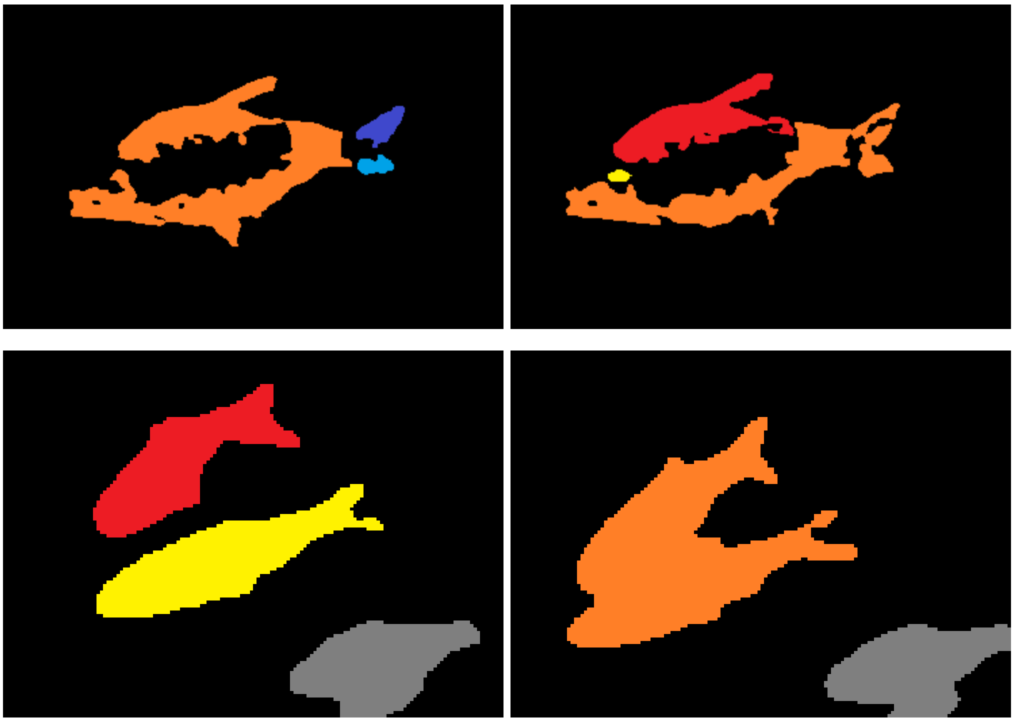


Figure 8.5: In the top row are two segmentations of a fish with a low accuracy. This causes the connected components to merge and split frequently although they all belong to the same foreground object. In the bottom row are two fish which are accurately segmented but their paths cross which causes the connected components to merge into one single component. Both situations are very difficult for the tracking approach and have to be handled with care.

as for a connected component, e.g. bounding box, number of pixels and so on. Values like the growth rate, direction of movement or velocity are averaged over all connected components weighted by their size. For example, the velocity of a union of t connected components would be

$$Vel_{union} = \frac{1}{\sum_{s=1}^{t} NoP_s} \cdot \sum_{k=1}^{t} NoP_k \cdot Vel_k. \tag{8.26}$$

With these properties defined, the unions can be compared to connected components or other unions by using the *CCSM*. The only new thing about the unions is that they also have a list of all old connected components they are composed of. This is important in case the union splits later again so that the old connected components can be matched correctly again. An example of the merging and splitting of components can be seen in Figure 8.12.

When for each of the two frames (between which the matches have to be found) a list of all connected components and possible unions is created the actual matching can start. For this, the *CCSM* is computed for every possible match. That means if the list for the first frame has length t and for the second frame length s, a $t \times s$ Matrix of CCSM values is created. Now, let the entry (k,u) be the lowest value in the whole matrix, this means that the connected components k (from frame one) and u (from frame two) are the best possible match at the moment and will be assigned to each other. These two values and all unions that include one of them – or, if they are unions already themselves, all connected components that belong to them and unions that include at least one of them – have to be deleted from the matrix and then the process can be repeated. This is depicted in Figure 8.6 where the row and column corresponding to the element (k,u) get deleted and two other elements – indicated in dashed lines – also get deleted because the corresponding elements contain the just matched elements k and l and therefore a matching in the next step is not possible.

By repeating this process all connected components from frame one can be matched with corresponding components in frame two and then the elements in frame two can be similarly matched with the components found in the next frame (frame three) and so on. By choosing the relatively simple greedy approach the time for the optimization, after computing the CCSM values, is neglectable. More elaborated optimization approaches would have to take into account the overall matching costs for all connected components between several frames at the same time. This optimization would be computationally very expensive, require the processing of whole batches of frames (which prohibits any online usage) and promises little accuracy improvement since parameters that include combined information from several frames are already included in the CCSM (e.g. velocity or direction of movement).

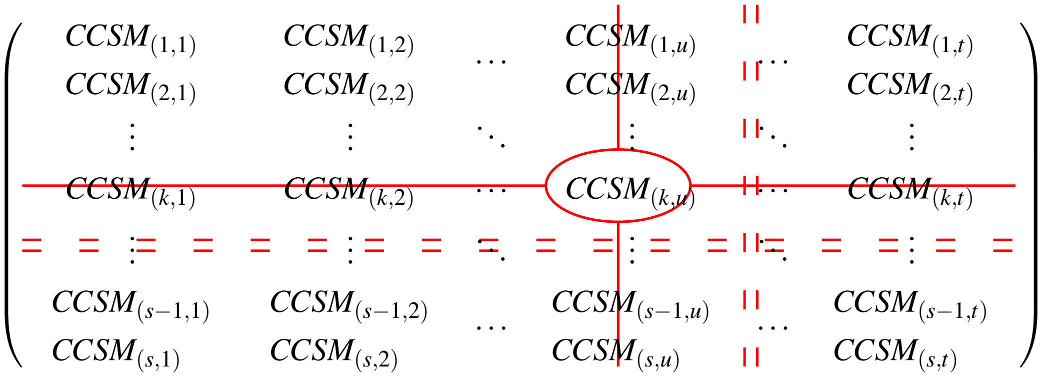


Figure 8.6: A correlation matrix between the s elements in frame one and t elements in frame two. The element (k,u) has the lowest value (is the best match) and for any further matching all components that include elements from k or u have to be deleted.

In normal situations, this matching works very well but specific situations can be problematic, e.g. if a new fish appears it cannot be matched since there are no prior detections and hence the connected component of that fish has to be treated differently. In the following, these special cases and their handling are discussed.

8.2.1 Special Cases

A different number of connected components in two consecutive frames does not yet mean that a new fish appeared or an old one disappeared; sometimes a bad segmentation splits a fish in two and thereby generates two connected components instead of one or two fish swim close to each other so that their connected components merge to one. These cases can usually be handled by the algorithm already because it tries to match also unions of connected components that are close by or expects a split of a connected component that was the result of a merge in the past. However, if at the end there are still elements left over which cannot be matched, they are handled as follows:

Leftover Elements in Frame One - Disappearance of an Existing Fish

If an element in frame one is left over it will most likely correspond to a fish that disappeared, e.g. by swimming out of the scene or swimming so far away from the camera that it cannot be distinguished anymore from the background. To avoid errors (e.g. by a missed detection), these elements will not immediately be forgotten but especially tagged and added to the list of connected components for frame 2. As it could not be matched it will not be shown as an element in the result of frame two, but in the next step, there will be a trial to match this connected component to a component in the following frame. Only if the connected component could not be matched in five consecutive frames will it be completely forgotten. This ensures that the disappearance of a fish is not just a short time error in the segmentation but consistent over several frames.

Leftover Elements in Frame Two - Appearance of a new Fish

If a connected component in frame 2 is left over, it will most likely correspond to the appearance of a new fish but could also be a misdetection (e.g. of a shadow) or an unfortunate split of an existing connected component. The last aspect will be checked first by going through all found matches between frame one and two and checking if the CCSM score can be improved by adding the leftover component to any of the matches. That means uniting the matched component in frame two with the leftover component and then computing the CCSM value with this new component. Most of these unions are already evaluated in the first step of the matching but sometimes the split of a connected component can happen so awkwardly or the objects move so fast that the resulting new components are already far away from each other and therefore a union of these two elements was not checked in the previous step (because they were farther away than T_{CCdist}).

If an existing match could be improved by adding the leftover component, the component will be added to that match and everything is done. Otherwise, a new detection has to be assumed and will be added to the list of components in frame 2. However, similar to the previous case, this component will be tagged as new and not shown in the results at first. It will only be fully accepted as an object that has to be tracked if the connected components could be matched for five consecutive frames. This makes the whole approach more resistant to errors in the segmentation which is especially necessary since the wrong addition of only one new component can strongly affect the tracking accuracy of the already tracked objects in the scene.

Leftover Elements in both Frames

If both events happen simultaneously, an old connected component disappears and a new one appears, completely false matches can occur since at the end a connected component in frame one that has no true match is left over and also one component in frame two that has no true match. To avoid that these two are matched falsely a threshold for the CCSM has to be set so that all matches above this threshold will be rejected. Since the CCSM score already entails an effective outlier detection (see equation 8.16 to 8.22) this threshold can be set quite high. The CCSM value, in this case, is dominated by the exponential functions in the outlier detection which are depending on the image size and, therefore, the threshold is set to

$$T_{CCSMmax} = \exp z, \tag{8.27}$$

where z is also depending on the resolution of the frame (compare with equation 8.23). If the CCSM score exceeds $T_{CCSMmax}$ no matching will occur and instead the remaining components will be handled separately as discussed in the previous two paragraphs.

8.3 Results

Evaluating and comparing tracking algorithms is more difficult than the evaluation of change detection methods. First of all, the creation of ground truth data can become very cumbersome and ambiguous even for humans – especially in crowded scenes – and, in addition to that, the

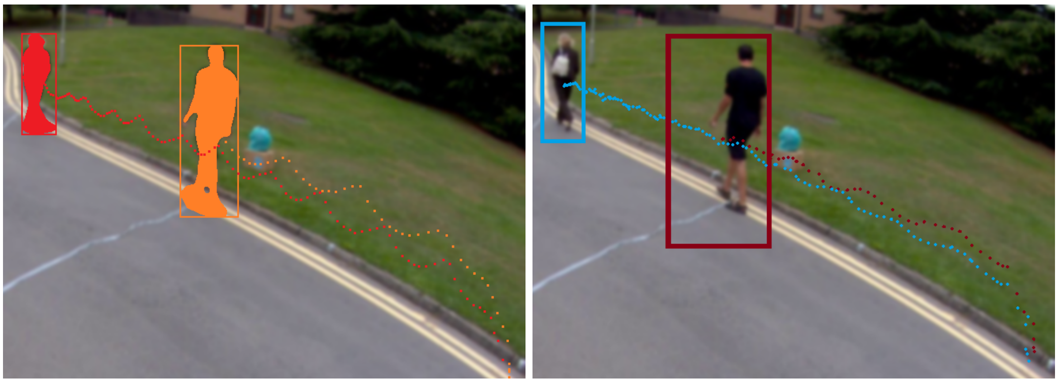


Figure 8.7: Comparison of the tracking method from [PF14] (right) with the proposed algorithm (left). In both cases a bounding box is drawn around the detected objects and the path tracked so far is shown with a small dot for each previous detection. Each tracked object has a specific ID and a color which is assigned to that ID. On the left side is also the background subtraction result overlayed over the actual image.

comparison of the results with the ground truth data is not as well defined as for change detection, e.g. when does an object counts as detected? which objects should be detected? and so on. This is further complicated by the very different approaches to this problem. There exist single camera systems, stereo camera systems but also multi-camera systems with many cameras from different perspectives which are later combined. Furthermore, some approaches use neural networks or similarly trained detectors which then only detect specific objects (mostly humans) and have no consideration for other moving objects in the scene like cars or animals. More general approaches, like the here presented, detect all moving objects and therefore would detect cars also. This makes a direct comparison very difficult and often unfair since the algorithms pursue different aims.

Although there exist several datasets for evaluating tracking algorithms, there is so far, again, no dataset with underwater videos. Therefore, an evaluation on in-air data will be done and the first dataset is the one presented in [PF14]. The aim of the Arena dataset is the detection of unusual (criminal) behavior and not the comparison of tracking algorithms. However, they provide the detection and tracking data of their own algorithm, so it can be used for the higher level tasks of behavior analysis. The proposed approach will be compared with their data on the video with the ID `06_01`[1]. Since no ground truth data for the tracking is provided only a visual comparison of the results is possible but it allows the underlining of some of the specialties of the proposed approach compared to most other tracking methods.

A first impression of the results can be seen in Figure 8.7 where two humans were tracked. The situation is quite easy and therefore both methods could track the humans without problems but it can be seen that the approach from [PF14] only gives bounding boxes of the found detections whereas the proposed approach gives more exact sets of foreground pixels that belong to the detected object. Therefore, in the tracked path every step of the human can be seen clearly since the body goes up and down slightly with each step. In the method of [PF14] these nuances are

---

[1] `http://www.cvg.reading.ac.uk/PETS2014/a.html`

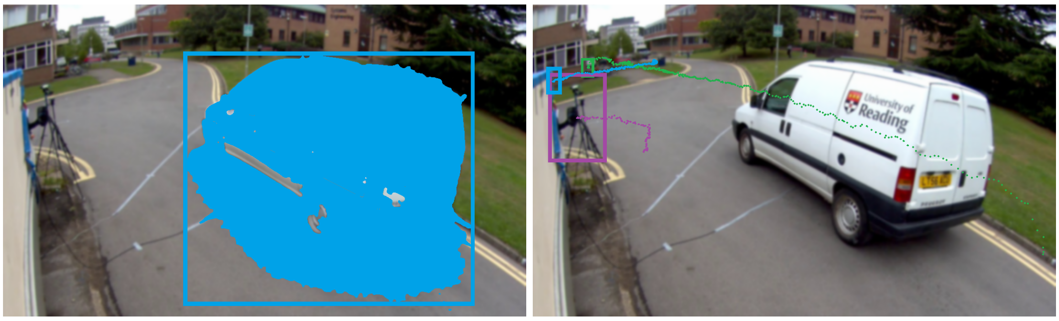


Figure 8.8: On the left side is the proposed approach where the car is tracked since it is a moving object. On the right side is the data from [PF14], there the car is not tracked since the method focuses on humans. There are also some wrong detections at places where humans previously left the field of view of the camera.

lost and the paths are almost straight lines.

Detection: Humans vs all Moving Objects

In Figure 8.8 the proposed approach detects all moving objects, even the car, whereas the method of [PF14] tries to detect only the humans and does not track the car. Either can be advantageous, depending on the situation and task, however, one problem with the detection of specific objects is that they depend on pre-trained data and cannot correct errors later. For example, in the Figure 8.8 the tripod is mistakenly recognized as a human and this detection stays there constantly for the rest of the video. Of course, the background subtraction also makes errors but these errors vanish over time as the background model adapts to the new situation and therefore this method is overall more stable, especially for longer periods.

Occlusion Handling

Normally, tracking algorithms try to track the different objects through occlusions which can be seen on the right side of Figure 8.9. The proposed method, however, only looks at the detected foreground blobs and therefore cannot differentiate between them during the occlusion but tries to match the objects correctly after the occlusion based on their expected movement. A good example for this in an underwater scene can be seen in Figure 8.12.

Fragmentation

Since the proposed approach does not try to detect specific objects but any movement, in general, it is more prone to fragmentation errors. An example of this can be seen in Figure 8.10 where the human is divided into three smaller detections. Normally, the handling of splits and merges would unite these three components and treat them as one, but in this case, the human just appeared from the left side and, therefore, it is unknown yet if these three detections belong together or if they represent three different objects. The approach from [PF14] handles this situation better and detects the human as a whole, although the bounding box is quite inaccurate.

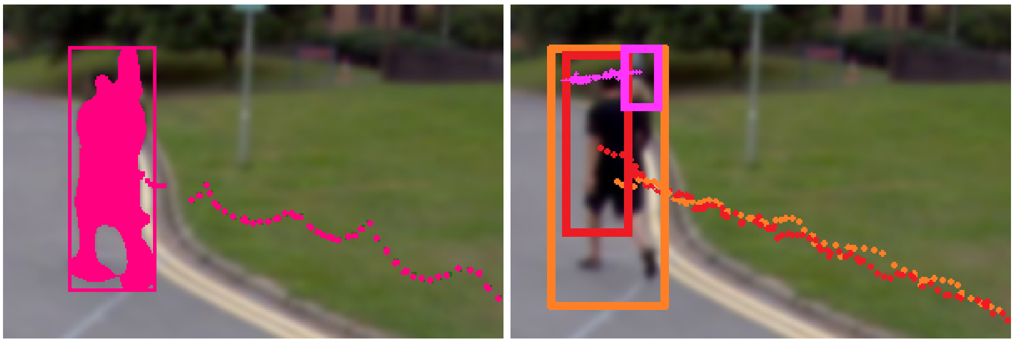


Figure 8.9: The proposed algorithm (left) sees the objects as one blob during the occlusion whereas the approach of [PF14] and most other tracking approaches try to track the different objects during the occlusion.

Different Detection and Tracking Errors

The tracking accuracy of the proposed approach seems to be higher overall, an example of a difficult situation is given at the top of Figure 8.11. The detection of the human becomes difficult for both approaches at some point; that can be seen very well in the left image where the track is at first accurate and stable but becomes messy and confusing at some point because of a very low detection accuracy. However, the proposed approach tracks that human constantly and does not lose the track whereas in the data of [PF14] the track is lost and a new ID created so that the human seems to appear out of the middle of the street. In the lower part of that Figure other errors are shown, the background subtraction produces some false detection because of the tarpaulin that is moving in the wind. The approach of [PF14] is not affected by this but has the previously discussed misdetection of the tripod as a human in that same frame.

Lastly, a good example of how the proposed tracking method deals with occlusion (merging and splitting of blobs) can be seen in Figure 8.12 where a part of the Marine Snow video from the UCD dataset is shown.

8.3.1 Quantitative Evaluation

For a meaningful assertion of the quality of different tracking algorithms a quantitative evaluation of the errors against ground truth data created by experts is necessary. The scoring of the results against such ground truth data is not as simple as for the change detection task because many different kinds of errors are possible and have to be reflected all in the metrics. This cannot be done by one measure alone and therefore several metrics will be used here that focus on different aspects. In [BS08] two measures especially created for the evaluation of tracking algorithms are presented.

MOTP - Multiple Object Tracking Precision

The definition of the original MOTP is: Let $D_{k,t}^{cor}$ represent the coordinates of the centroid of the detection with label k in frame t and $G_{k,t}^{cor}$ the coordinates of the centroid of the ground truth

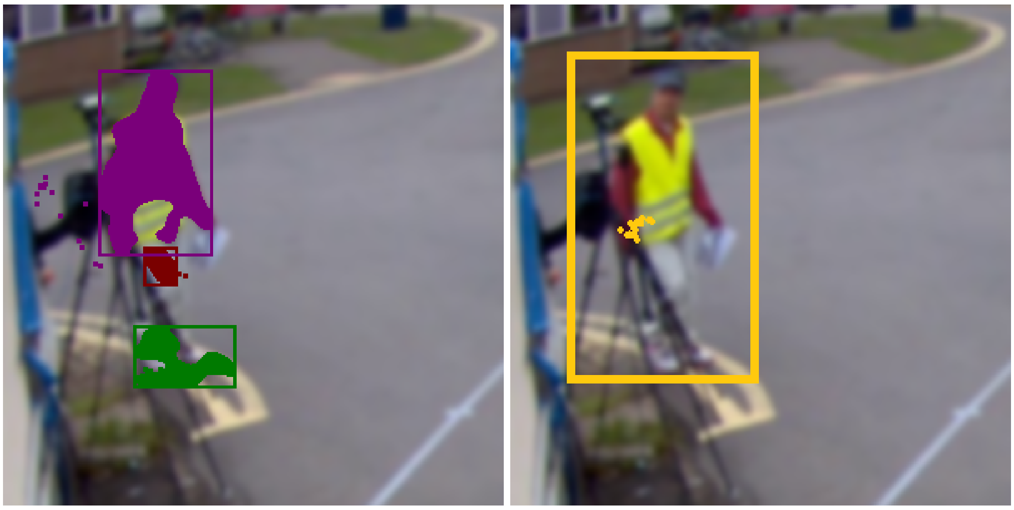


Figure 8.10: The just appearing human is fragmented due to partial occlusion from the tripod. In the algorithm of [PF14] the human gets detected as a whole and can hence be tracked better.

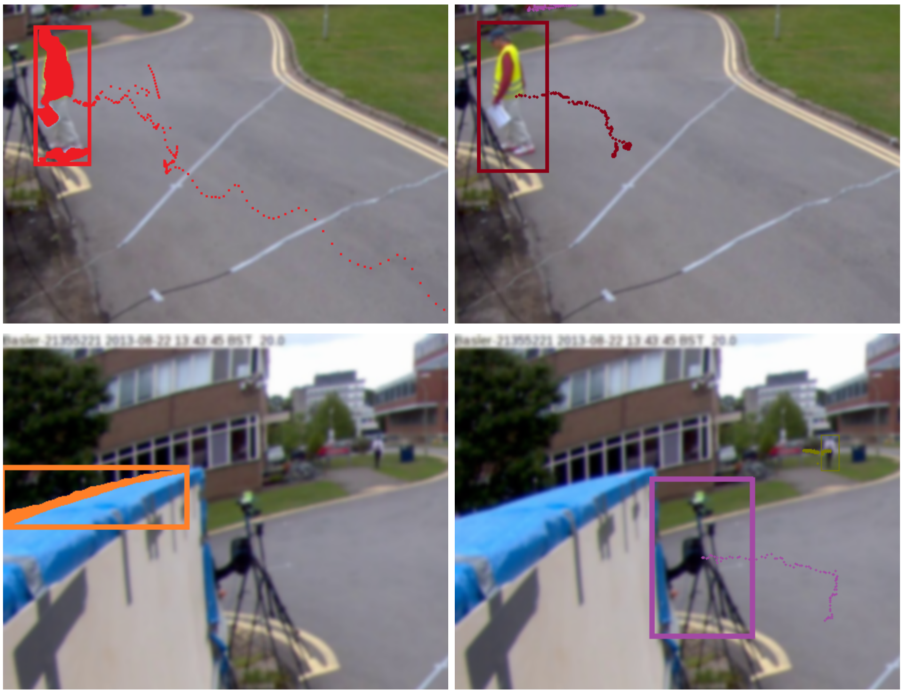


Figure 8.11: The top row shows the loss of a track in the [PF14] data (right) and in the bottom row different detection errors are displayed.

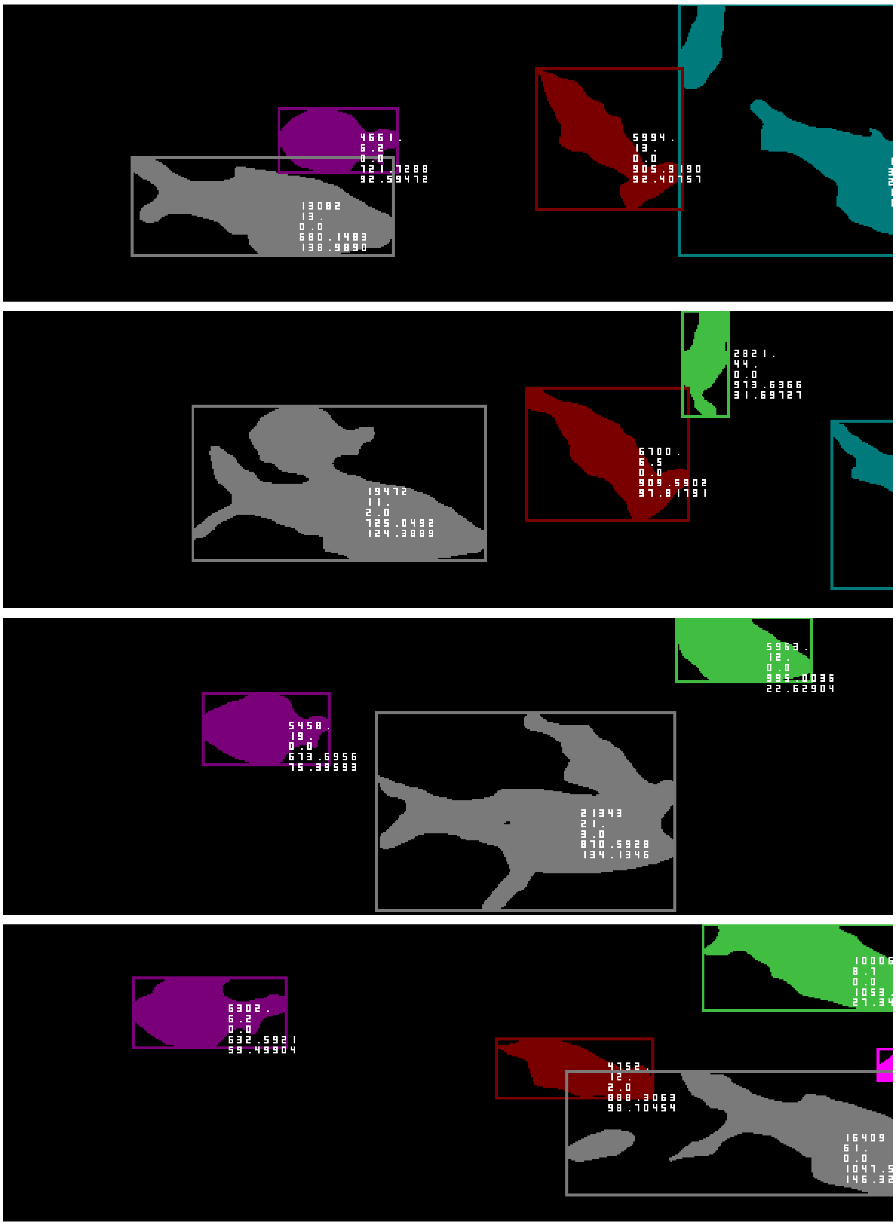


Figure 8.12: An example of the tracking in a crowded scene where components split and merge constantly. The numbers in white for each tracked component give some of the properties, from top to bottom they are number of pixels, velocity, number of old components, x- and y-coordinate of the centroid. In the first two images the dark green object splits because its different connected components drifted too far apart. Also, the gray and purple one are unified. It can be seen that the purple object is not forgotten but kept as an old component and was matched correctly in the third frame after they split again. The same is true for the gray and red component, they are first unified and later matched correctly again after they split.

annotation with label k in frame t. The MOTP is then defined as

$$\frac{\sum_{k,t} \|D_{k,t}^{cor} - G_{k,t}^{cor}\|_2^2}{\sum_{k,t} 1}. \tag{8.28}$$

Objects that could not be detected or matched are ignored as only the precision of the tracked objects is measured. Matching and detection errors are evaluated separately in the next metric. The MOTP is the average distance in pixels between the centroids of correctly matched detections and the ground truth data. It is a value greater or equal to 0 and 0 would mean a perfect accuracy, all found detections would be completely similar to the ground truth annotations.

In the literature, however, often a slightly different definition is used under this name. It will be called MOTP2 here and is the ratio of the intersection and union of the bounding boxes of a correct detection and its ground truth counterpart,

$$MOTP2 = \frac{\sum_{k,t} \frac{|D_{k,t}^{bb} \cap G_{k,t}^{bb}|}{|D_{k,t}^{bb} \cup G_{k,t}^{bb}|}}{\sum_{k,t} 1}. \tag{8.29}$$

Here $D_{k,t}^{bb}$ is the set of all pixels which belong to the bounding box of the correctly matched detection k in frame t and $G_{k,t}^{bb}$ is the corresponding set for the ground truth data. Hence, the MOTP2 is a value between 0 and 1, where an optimal tracking algorithm would receive a score of 1 since that would mean $D_{k,t}^{bb} = G_{k,t}^{bb}$ for all k and t. The MOTP2 measure is used more often in the literature and is also more expressive since it takes the whole bounding box into account and not only the centroid, e.g. can it differentiate between two bounding boxes with the same centroid but different sizes which the original MOTP cannot do.

MOTA - Multiple Object Tracking Accuracy

The MOTA metric is a measure of the detection and tracking errors and consists of three components:

- m_t – Number of misses in frame t
- fp_t – Number of false positive detections in frame t
- mme_t – Number of mismatches in frame t

These numbers added together and divided by the total number of objects gives a ratio of errors to true objects. The MOTA is then defined as

$$1 - \frac{\sum_t m_t + fp_t + mme_t}{\sum_t c_t}, \tag{8.30}$$

where c_t is the number of objects present in frame t. It is a value smaller or equal to 1 and a rating of 1 would signify a perfect accuracy. The metric is not limited to the range of $[1,0]$, values lower than 0 are possible if there are e.g. more false positives than actual objects.

Algorithm	MOTP2	MOTA
Jiang et al. [JRD12]	**0.788**	0.608
Breitenstein et al. [Bre+11]	0.563	0.797
Yang et al. [Yan+09]	0.538	0.759
Berclaz et al. [BFF06]	0.600	0.660
Andriyenko et al. [AS11]	0.761	0.814
proposed approach	0.550	**0.885**

Table 8.1: Results of different tracking approaches on the S2.L1 Video of the PETS 2009 dataset. The data is taken from [JRD12]. The results show that the proposed approach is very good at accurately detecting and matching objects in the scene (MOTA). However, it fails in precision (MOTP2) since objects that are close together are often not separated and treated as only one object.

PETS 2009 Dataset

To compare various methods based on these metrics the video S2.L1 of the PETS 2009 Benchmark[2] [FS09] is chosen. The ground truth data for this video is available[3] and several different algorithms have already been tested on this video so that a meaningful evaluation is possible. A comparison of five different algorithms with the proposed algorithm can be seen in Table 8.1. The metrics MOTP2 and MOTA have been used (since the data for these two metrics were available) and they show that the proposed algorithm has a high tracking accuracy and a below average precision. Both of these results come at least partially from the combination of change detection with a blob tracker. The extended GSM background subtraction delivers very accurate detection results so that hardly any misses occur. Only when a person was mainly occluded by the sign in the middle of the scene did the algorithm fail to detect this person occasionally (compare Figure 8.14). However, this detection accuracy comes with the price that no single humans are detected but only moving foreground blobs which consequently makes the tracking more difficult (splitting and merging of blobs) and also reduces the accuracy of the detection since often humans that are close together are detected and tracked together (see Figure 8.13).

TownCenter Video

A second evaluation was done on the TownCenter video presented in [BR11] where tracking results of many different algorithms are available at `motchallenge.net`[4] and even videos of these results can be downloaded. The ground truth data is freely available[5] and focuses on the heads of the humans so that a human where the head is outside of the field of view of the camera

---

[2] `http://www.cvg.reading.ac.uk/PETS2009/a.html`

[3] `http://www.milanton.de/data.html`

[4] `https://motchallenge.net/results/3D_MOT_2015/`

[5] `http://www.robots.ox.ac.uk/ActiveVision/Research/Projects/2009bbenfold_headpose/project.html`

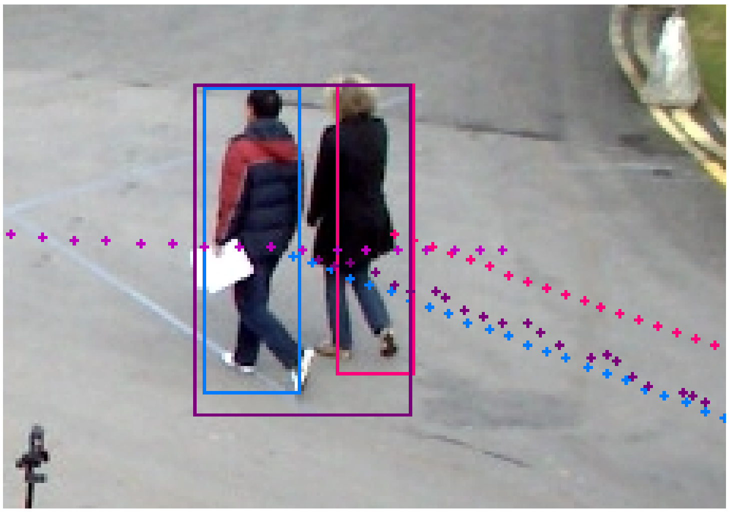


Figure 8.13: Example of the S2.L1 Video of the PETS 2009 dataset. The bounding box of the detection of the proposed algorithm is shown in purple and in blue and pink are the provided ground truth bounding boxes. Both humans were accurately detected but considered as one blob and therefore the detection accuracy (MOTP2 value) for both human is each time below 40%.

is usually not marked in the ground truth data although the rest of the body might still be visible.

A comparison with six other methods can be seen in Table 8.2 where the two previous metrics (MOTP2 and MOTA) are used as well as False Alarms per Frame (FAF). FAF is the number of falsely detected objects per frame. It can be seen again that the accuracy is very high and the precision quite low, in general for the same reasons as before. Although the accuracy is so high the FAF is still quite large with 3.3. These false alarms are mainly detections of humans where the head is not visible but (parts of) the rest of the body or inaccurate detections so that one actual object is split into two or more in the segmentation. These false detections are the main error source that contributes to the MOTA measure and other errors rarely happen (e.g. missed detections are around 0.3 per frame).

Results of the proposed algorithm and [KRH17] can be seen in Figure 8.15 where they are compared against the ground truth data. The proposed method detects many partly occluded persons which are not marked in the ground truth data because their heads are not visible. In contrast to the proposed change detection, the approach of [KRH17] uses a specific human model for the detection and tracking so that the head position (or the lack of it) can be estimated, this leads to far fewer false detection. However, the detector often fails to recognize persons so that the miss rate is higher than with the GSM background subtraction; for example the couple in the bottom-left corner or the man with the baby stroller in the bottom-right corner.

Hungarian Method

Lastly, the proposed matching method (greedy matching but with consideration of possible unions or splits) is compared against the Hungarian method (best overall match but without considering

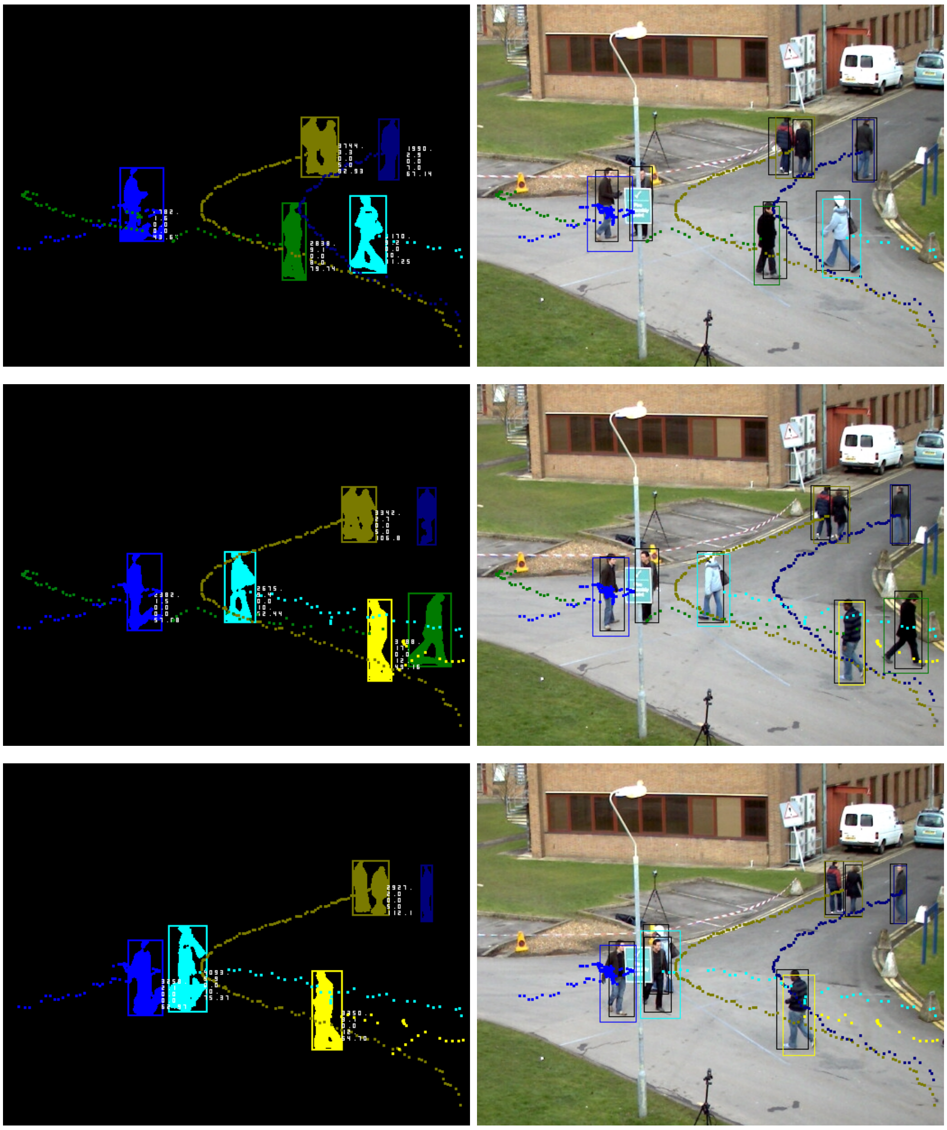


Figure 8.14: Shown are results of the proposed tracking algorithm in combination with the eGSM background subtraction on the S2.L1 Video of the PETS 2009 dataset. On the left side are the pure detection and tracking results of the proposed method and on the right side the results are interposed with the current frame and the ground truth bounding boxes which are shown in black. In the top frame it can be noticed that the person in the middle behind the pole is not detected correctly because he is mainly occluded. In the other images it can be seen how the persons tracked as light blue, green and yellow are tracked correctly although the pass and occlude each other.

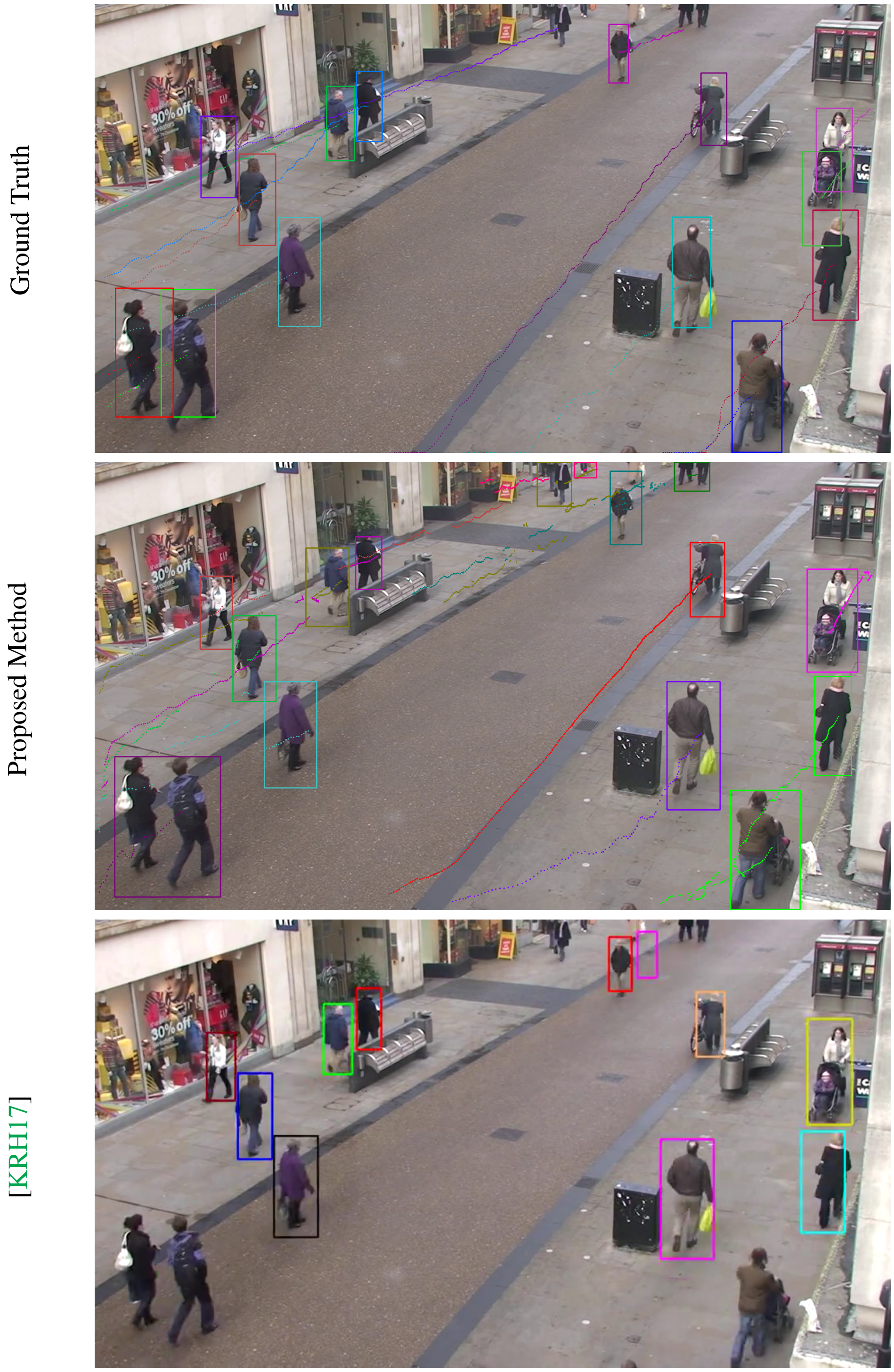


Figure 8.15: Comparison of two approaches with the provided ground truth data on the Town-Center video.

Algorithm	MOTP2	MOTA	FAF
Klinger et al. [KRH15]	**0.610**	0.511	**2.3**
Klinger et al. [KRH17]	0.574	0.422	2.6
Leal-Taixé et al. [LPR11]	0.519	0.287	3.1
Wen et al. [Wen+17]	0.542	0.168	4.3
Sadeghian et al. [SAS17]	0.546	0.153	2.5
Pellegrini et al. [Pel+09]	0.514	0.152	3.6
proposed approach	0.410	**0.780**	3.3

Table 8.2: Evaluation on the TownCenter video. The data for the other approaches is taken from `motchallenge.net` and the results are overall very similar to the previous comparison with a high accuracy but low precision because objects that are close together are not tracked separately.

unions or splits). The greedy approach does not ensure an overall optimal solution between two frames since sometimes it can be preferable to not choose the best match for a connected component because this opens better matching options later for other components. The Hungarian method, on the other hand, finds the optimal matches for all found foreground objects but, to do this, it must consider all objects and their possible matches at the same time which prohibits the consideration of unions and splits. Hence, only individual blobs will be considered for the Hungarian method.

To evaluate which approach performs better, a greedy approach with consideration of unions and splits or the overall optimal solution without this, an evaluation on 1000 frames of the Marine Snow video is done. This video has a high density of fish in various shapes and many occlusions happening, therefore it is ideal for an evaluation. Occlusions among fish and the following confusion or loss of labels are the main challenge for the tracking task and an example of this can be seen in Figure 8.16. The tracking of fish when no occlusion is happening was very good for both approaches and the only reason for errors during these times was a poor detection quality of the GSM background subtraction, e.g. the detection of shadows.

For a fair comparison both methods use the same detection results and, therefore, errors based on bad detections of fish are not important since they are similar in both cases. The only errors that are interesting are the cases where a label is lost or switched for a fish (foreground blob) that was permanently detected by the eGSM. These errors are solely based on the performance and accuracy of the tracking algorithms and therefore a good measure. For the first 1000 frames of the Marine Snow video, these errors were counted by hand and occurred 33 times for the proposed method and 81 times for the Hungarian method which clearly shows that a consideration of unions and splits is necessary for this kind of task and scenario. Also, both methods used the same energy function, the CCSM, so that only the matching quality is evaluated.

For an evaluation with the previous measures (MOTP or MOTA), ground truth tracking data would be necessary but is not available for underwater videos. However, a test on the TownCenter video could validate the previous results. A change of the matching method from the proposed

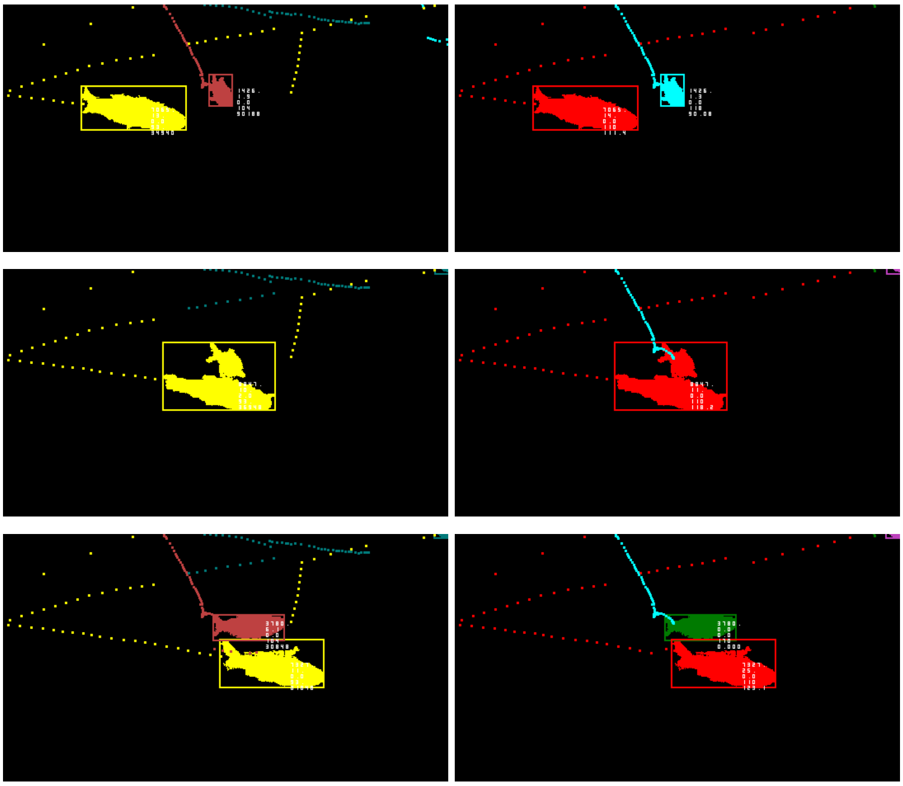


Figure 8.16: A comparison of the proposed tracking approach (left) with the Hungarian method (right). At first, the two fish are detected and tracked separately and accurately (top row) but then they overlap slightly (middle row) and therefore only one foreground blob is detected. In the last frame (bottom row) the fish separate again and the proposed method can assign the correct labels from the first frame again whereas the Hungarian method has lost the smaller fish (light blue) and gave him a new label (green).

approach to the Hungarian method decreased the accuracy (MOTA) from 0.78 to 0.745 and increased the precision (MOTP2) slightly from 0.41 to 0.43. The accuracy deteriorated because more tracks were lost, like in the Marine Snow video, and the precision improved because each blob was always handled separately and never united with others that move in a similar direction, this could in some situations produce higher precision results.

Runtime

The matching based on the CCSM is computationally expensive as the measure incorporates many different properties which have to be computed and compared. Moreover, the runtime is highly dependent on the number of objects present in the scene. First of all, the CCSM between every object in frame 1 and every object in frame 2 is computed which is quite time-consuming. But more importantly are the possible unions and splits that are considered because the number

Algorithm	Proposed Approach	Hungarian Method
TownCenter Full Size (1920 × 1080)	8.97	1.25
TownCenter Half Size (960 × 540)	3.65	0.70
PETS S2.L1 (769 × 576)	0.11	0.10
Marine Snow (1920 × 1080)	15.92	0.18

Table 8.3: Shown are the average times (in seconds) necessary to process one frame for the proposed algorithm (eGSM + Tracking) and the Hungarian method on different videos.

of possible unions increases very fast with the number of foreground blobs that are close together (factorial!). Therefore, in frames with many foreground blobs close together the runtime increases drastically. Different mean runtimes of the algorithm on the previously shown videos are shown in Table 8.3 and compared to the Hungarian method. As a comparison: in [KRH17] it is stated that their algorithm takes in general around 10 seconds per frame.

8.3.2 Conclusion

In this chapter, the previous change detection results were used to further extract information out of the (underwater) videos by not only detecting the different foreground objects in the scene but also tracking them during the whole video. Since the proposed tracking is based solely on the detected foreground blobs – so that the whole approach stays very general and can easily be applied on different objects in other scenarios – a new matching measure, the CCSM, is introduced first. It uses several distinct features of the foreground blobs to evaluate their similarity, e.g. shape, direction of movement or size. The measure includes an outlier detection to avoid matching blobs that are far away or very different in size but otherwise similar. This was especially necessary since fish of the same species tend to be extremely similar in their appearance (e.g. shape) and behavior (e.g. direction of movement in a swarm). In these cases, a small inaccuracy of the detection could lead to false matchings without the discussed outlier detections.

Afterwards, the actual matching is done based on the CCSM scores between all foreground blobs in two consecutive frames. In this process, the unions and splits of blobs are also considered so that detection inaccuracies or occlusions can be handled better. Then, in a greedy approach, always the two most similar blobs are matched successively. If elements are leftover at the end of this process they most probably signify an appearance or disappearance of a foreground object in the scene. Nonetheless, these cases are handled with special care to avoid falsely adding or deleting tracked objects since this could lead to a cascade of wrong matchings.

The accuracy of this method was evaluated on two different datasets against several other approaches. These videos feature in-air scenes since no underwater videos or ground truth data are available and all existing methods were only tested on in-air scenes. In general, the comparison of the proposed tracking algorithm is complicated since – by focusing on the detected foreground blobs instead of trying to detect specific objects – the whole tracking approach is handled slightly different here than in most previous methods. Therefore, the accuracy is higher than in the other approaches but, at the same time, the precision with which single objects are tracked is lower than in most approaches. The main reason for the high accuracy is the accurate detections of the eGSM background subtraction which barely misses any moving objects. The cause for the low precision is the foreground blobs which can consist of several actual objects (fish, humans, etc.) that are close together or occlude each other. Since these blobs are tracked as a whole, this consequently lowers the detection and tracking accuracy of each individual object drastically.

9 Conclusion and Future Work

Advancements in technology and the production process have made video cameras cheap and readily available to a large group of customers, this includes end-users as well as businesses. These advances were originally mainly developed for in-air equipment but often paved the way for improvements for underwater imaging tools as well, and this process of ever decreasing costs with increasing imaging quality is very likely to continue. This opens many new possibilities for automation, e.g. aquacultures in the open sea which are monitored constantly by cameras or cameras which are mounted on cruise ships to constantly study the natural sea life. These cameras, however, produce immense amounts of data which cannot be watched and interpreted by humans manually anymore but need computer vision algorithms so that they can be processed semi- or fully-automatic. A crucial problem for these algorithms is the great variety of tasks that have to be handled, e.g. the objects that should be detected (humans, plants, fish, ships ...) or the environment they are in (indoor or outdoor, different lighting conditions, ...).

To this account, a new change detection algorithm – the Gaussian Switch Model – was proposed, which can adjust and correct itself by comparing two different models. Nonetheless, the approach is still fairly simple, fast and can be applied to detect any moving object in almost all environments with the only necessity being a single static camera. To make the change detection results easier usable in later stages of a computer vision pipeline, the use of a method to increase spatial coherency is advisable since the GSM results are completely pixel-wise and do not incorporate any spatial model of the real world. Three different methods were proposed for this task, one adjusts the segmentations to the edges in the frame (N^2cut), one uses a spatial model based on natural objects (MRF) and the last one combines the information from several frames via an optical flow (temporal trajectories). Each of these methods has specific advantages and problems in certain scenarios but overall the N^2cut delivered the highest accuracy in a fraction of the runtime of the other approaches.

Although these algorithms can be in principle applied to almost all scenarios, an adaption to special difficulties of specific videos can be beneficial. Underwater scenarios pose such special difficulties with increased blur, color cast, Marine Snow and many other image degradation effects. However, no one has so far investigated the possible adaptions and their potential to increase the segmentation accuracy underwater since there does not exist any change detection dataset with underwater scenes (although many for in-air scenes are available). Therefore, the first step in adapting the presented algorithms to underwater videos was the creation of a new dataset, the Underwater Change Detection dataset. On these videos, the proposed algorithms were compared to state of the art change detection approaches and combined with several underwater image enhancement techniques to investigate if they can have a positive impact on the segmentation quality. On the one hand, the impact of these enhancement methods was fairly consistent throughout the different change detection approaches. However, on the other hand, the results were heavily dependent on the specific scene they were applied on and often even decreased the accuracy instead of increasing it. The overall most promising and usable enhancement method seems to be Marine Snow removal since it combines an overall positive impact with a low runtime (0.75s per frame), other approaches took up to several seconds per frame without delivering better results.

Since underwater image enhancement alone was not enough to tackle the special problems of the medium water, an adaption of the GSM change detection algorithm to these special conditions was necessary. Because it is very common that moving objects underwater have a high similarity between them (e.g. a swarm of fish) a foreground model was added that could help differentiate between foreground and background objects. Furthermore, the GSM model was combined with the Mixture of Gaussian idea and important parameters were updated automatically and adaptively. This made the whole model more complex and increased the runtime, but also could enhance the accuracy notable. A special problem in underwater scenarios is crowded scenes. Fish often appears in whole swarms – especially the ones which are interesting for fishers and aquacultures – which makes it very difficult to create and sustain a model of the background since the actual background is often hardly visible anymore. To account for this, pre-segmentations were created with an optical flow approach (Flux Tensor). This allowed to exclude most foreground objects from the background modeling process in the first place and, thereby, increased the quality of the model and the accuracy of the resulting segmentations significantly in crowded scenes.

Lastly, these segmentations were combined with new a blob tracking approach so that important information about the movement of the detected objects could be extracted. To keep the generality of the overall approach, the tracking was restricted to the detected foreground blobs and no object-specific properties or models are used. Therefore, it can be used to track fish, divers, boats or any other moving object. To compare and match the different blobs a new metric was introduced, the Connected Component Similarity Measure, which combines information about the shape, size, location, movement vector and so on. Furthermore, an outlier detection is part of the score to avoid matches of blobs that are very far away or too different in size but otherwise very similar. This was integrated into a matching algorithm that takes into account possible splits or unions of blobs which are close to each other. Taking these into account allowed a better handling of occlusions or inaccurate segmentations and provided more accurate results than the Hungarian algorithm. Because of the neglection of underwater videos in segmentation and tracking datasets, the evaluation had to be done mainly on in-air footage and showed a very good accuracy (MOTA value) but low precision (MOTP) in comparison to state of the art methods. The reason for this is the focus on general foreground blobs instead of specific objects which generates a high detection accuracy but does not allow a separation of several occluding objects.

Overall, the here presented algorithms can be combined efficiently to a computer vision pipeline that detects accurately all moving objects in an underwater scene and also tracks them as long as they are inside the field of view of the camera. The proposed adaption of the GSM and possible additions of image enhancement techniques like Marine Snow removal make it especially valuable for underwater scenes but, nonetheless, the algorithms work also accurately on in-air scenarios. The results are important and useful information about the scene, e.g. the number fish and their movement patterns in an aquaculture. Also, it is possible to easily adapt or extend this pipeline by changing single algorithms like the spatial model or by adding further parts to gain even more high-level information.

To summarize, in this work were presented:

1. A novel background modeling approach that is lightweight and achieves State of the Art results for in-air foreground-background detection
2. An adaption to the common challenges of underwater footage
    - The background modeling in crowded scenes was improved with pre-segmentations.
    - Several underwater image enhancement methods were explored to counter the effects of image degradation.
3. Three novel methods that increase the spatial coherence in binary segmentations
    - The MRF model was enhanced by enlarging the neighborhood in a way that still allowed the computation in a feasible time.
    - N^{2}cut is an adaption of the Ncut to the video segmentation problem. It provides the most accurate results and has the shortest runtime.
    - Temporal trajectories used a dense optical flow to combine the segmentations from several consecutive frames.
4. The first underwater dataset for foreground-background segmentation
    - It contains all of the common problems of underwater video footage.
    - This made it possible to evaluate different underwater image enhancement methods in combination with segmentation methods.
    - The evaluation also showed that background subtraction methods are in general superior to optical flow based approaches in underwater conditions.
5. A tracking approach that only uses the binary segmentation results
    - A very general method that does not rely on any prior information about the objects.
    - To still obtain accurate tracking results the splitting and merging of the detected foreground blobs had to be considered and handled carefully.

Future Work

To address the low precision of the tracking approach it would be beneficial to have the possibility to add a model of the specific object class that should be tracked at the moment. This would allow dividing foreground blobs into the individual objects they consist of and improve the precision during occlusions. Furthermore, the creation of tracking ground truth data on underwater videos, especially with fish swarms and aquacultures, would be interesting and allow a better and more complete comparison. The next step of the computer vision pipeline would be the integration in a stereo camera setup and the addition of an algorithm that allows the extraction of accurate 3D data even in underwater scenarios where refraction is a great problem. This data would allow e.g. the observation of the growth process of fish in an aquaculture and could also be used to aid and verify the segmentation and tracking results.

Another interesting aspect would be the addition of machine learning approaches, e.g. in an aquaculture to find illnesses among the detected fish or even to recognize individual fish based on features like their scale patterns. Furthermore, over longer periods the behavior and patterns in the entire scene could be learned and abnormalities detected, e.g. if the whole fish swarm swims slower and/or less than before. Furthermore, to make the system overall more flexible and usable on ROVs and AUVs, an adaption of the background model to small/slow camera movements like described in [RFL17a] could be further developed for underwater scenarios which could potentially eliminate the necessity of a static camera.

Abbreviations and Symbols

Abbreviations

ACE	Automatic Color Equalization
AGW	Adaptive Gray World
AUV	Autonomous Underwater Vehicle
BP	Belief Propagation
B	Bootstrap (Video from the Wallflower Dataset)
CCSM	Connected Component Similarity Measure
CLAHE	Contrast Limited daptive Histogram Equalization
CNN	Convolutional Neural Network
CV	Computer Vision
C	Camouflage (Video from the Wallflower Dataset)
DOF	Dense Optical Flow
e.g.	For example
FA	Foreground Aperture (Video from the Wallflower Dataset)
FN	False Negative (e.g. false detection of background pixels)
FoE	Focus of Expansion
FP	False Positive (e.g. false detection of foreground pixels)
GB	Gigabyte
GMN	Generalized Moore Neighbourhood
GSM	Gaussian Switch Model
HD	High Definition (1920 × 1280)
ICA	Independent Component Analysis
IHLS	Improved HLS Color Space (Hue, Lightning Saturation)
INMF	Incremental Non-Negative Matrix Factorization
K-SVD	Combination of K-Means Clustering and Singular Valud Decomposition (SVD)
KNN	K-Nearest Neighbors
LbD	Learning-based Deblurring
LBP	Local Binary Pattern
LBSP	Local Binary Similarity Patterns
LDA	Linear Discriminant Analysis
LED	Light-Emitting Diode
LS	Light Switch (Video from the Wallflower Dataset)
MAP	Maximum A posteriori Probability
MCA	Morphological Component Analysis
MCC	Matthews Correlation Coefficient

MoG	Mixture of Gaussians
MO	Moving Object (Video from the Wallflower Dataset)
MRF	Markov Random Field
NCut	Normalized Cut
OCR	Optical Character Recognition
PCA	Principal Component Analysis
PSO	Particle Swarm Optimization
RANSAC	Random Sample Consensus
ROV	Remotely Operated underwater Vehicle
SURF	Speeded Up Robust Features
TN	True Negative (e.g. correct detection of background pixels)
TP	True Negative (e.g. correct detection of foreground pixels)
UCD	Underwater Change Detection
WT	Waving Tree (Video from the Wallflower Dataset)

Symbols

#	Quantity of something
α	GSM parameter: update rate
$Assoc(FG)$	Measures the association in a set FG
β	GSM parameter: controls the sensitivity of the Background Subtraction
BBh	Height of the Bounding Box of a connected component
BBw	Width of the Bounding Box of a connected component
BBh	Position of the centroid of the bounding box, x-coordinate
BBw	Position of the centroid of the bounding box, y-coordinate
C_k	Cost function in the MRF model for cliques of size k
Cx	Position of the centroid of a connected component, x-coordinate
Cy	Position of the centroid of a connected component, y-coordinate
$D(x)$	Cost function in the MRF model based on background subtraction data
$DOF_{m-1,m}(\bar{v})$	Optical Flow of pixel $(\bar{v})$ from frame $m-1$ to frame m.
$DOF^{x}_{m-1,m}(\bar{v})$	Vertical component of the Optical Flow of pixel $(\bar{v})$.
$DOF^{y}_{m-1,m}(\bar{v})$	Horizontal component of the Optical Flow of pixel $(\bar{v})$.
$d_{m-1,m}(\bar{v})$	Distance the pixel $\bar{v}$ of frame $m-1$ moved to its estimated position in frame m
Dx	Direction of Movement of a connected component, x-coordinate
Dy	Direction of Movement of a connected component, y-coordinate
η	Belief Propagation parameter: controls the influence of Otsu's Method
$\|\cdot\|_2$	Euclidean Norm
e	Euler's number $e = \sum_{n=0}^{\inf} \frac{1}{n!} \approx 2.71828$
γ	GSM parameter: Minimum value of the Variance for a pixel
GR	Growth rate of a connected component (in respect to the NoP)
I	Image - matrix and set of pixel values
$I(\bar{v})$	Pixel-value at the location $I(\bar{v})$

$I(\bar{v},c)$	Value of the channel c of the pixel $\bar{v}$
$K(v)$	Gaussian kernel function: $K(v) = e^{-c\|v\|^2}$
M	Gaussian model of the background of a video
$M(\mu_g,\bar{v},c)$	Mean of the g-th Gaussian of channel c of the pixel $\bar{v}$
$M(\mu^p)$	Mean of the partially updated Gaussian
$M(\sigma_g,\bar{v},c)$	Variance of the g-th Gaussian of the channel c and pixel $\bar{v}$
$M(\sigma^p)$	Variance of the partially updated Gaussian
$M(w_g,\bar{v},c)$	Weight of the g-th Gaussian of channel c of the pixel $\bar{v}$
$max(a,b)$	Returns the maximum of the two values a and b
$min(a,b)$	Returns the minimum of the two values a and b
$N(\bar{v})$	Set of all pixels in the neighborhood of $\bar{v}$
$N^2Cut(FG,BG)$	Similar to NCut but with $nAssoc(FG)$ instead of the normal association
$nAssoc(FG)$	Normalized association of the set FG
$NCut(FG,BG)$	NCut value of a graph which is partitioned into the sets FG and BG
NoP	Number of Pixels of a connected component
ω_{xy}	Weight of the edge between the nodes representing the pixels x and y
$Cut(FG,BG)$	Cut value between the sets FG and BG in a graph
$p_{BS}(\bar{v})$	Probability that pixel $\bar{v}$ is background, based on the background subtraction
$\propto$	is proportional to
$PI_{(\xi,l)}$	a polar image at the coordinates ξ (angle) and l (radius)
s_x	State of node x
Sh	a shape feature of a connected component
T_{CCdist}	Controls when the union of two components is also considered for matching
$T_{CCSMmax}$	Maximal value of the CCSM score, above that not matching will occur
$T_{newGaussian}$	Threshold to control the creation of new Gaussians in the MoG model
$trace(A)$	Sum of all diagonal elements of the square matrix A
v	Pixel in an image, usually with the RGB value (r,g,b)
$\bar{v}$	Position of pixel v in an image ($\bar{v} = (i,j)$) or video ($\bar{v} = (i,j,t)$)
V	Three dimensional image volume / Video
$V(t,\bar{v},c)$	Value at time-point t, position $\bar{v}$ and of channel c of the video V
Vel	Velocity of a connected component
$f(\bar{v})$	Optical flow at the position $\bar{v}$ of an image or image volume
Q_{v_j,f_i}	Message from node n_j to the factor vertex f_i in the Belief Propagation model

References

[Abu+12] J. Abukhait, I. Abdel-Qader, J. S. Oh und O. Abudayyeh. “Road sign detection and shape recognition invariant to sign defects”. In: *2012 IEEE International Conference on Electro/Information Technology*. Mai 2012, S. 1–6. DOI: `10.1109/EIT.2012.6220774`.

[AEB06] M. Aharon, M. Elad und A. Bruckstein. “K -SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation”. In: *IEEE Transactions on Signal Processing* 54.11 (Nov. 2006), S. 4311–4322. ISSN: 1053-587X. DOI: `10.1109/TSP.2006.881199`.

[AM16] J. D. Adarve und R. Mahony. “A Filter Formulation for Computing Real Time Optical Flow”. In: *IEEE Robotics and Automation Letters* 1.2 (Juli 2016), S. 1192–1199. ISSN: 2377-3766. DOI: `10.1109/LRA.2016.2532928`.

[AS11] A. Andriyenko und K. Schindler. “Multi-target tracking by continuous energy minimization”. In: *CVPR 2011*. Juni 2011, S. 1265–1272. DOI: `10.1109/CVPR.2011.5995311`.

[Bai+16] Jisong Bai, Yongjie Pang, Yinghao Zhang, Qiang Zhang und Zhen Li. “Underwater image segmentation method based on MCA and fuzzy clustering with variational level set”. In: *OCEANS 2016 MTS/IEEE Monterey*. Sep. 2016, S. 1–6. DOI: `10.1109/OCEANS.2016.7761205`.

[Ban+14] S. Banerjee, G. Sanyal, S. Ghosh, R. Ray und S. N. Shome. “Elimination of Marine Snow effect from underwater image - An adaptive probabilistic approach”. In: *Electrical, Electronics and Computer Science (SCEECS), 2014 IEEE Students’ Conference on*. März 2014, S. 1–4. DOI: `10.1109/SCEECS.2014.6804438`.

[BBN14] M. Boudhane, S. Badri-Hoeher und B. Nsiri. “Optical fish estimation and detection in noisy environment”. In: *2014 Oceans - St. John’s*. Sep. 2014, S. 1–6. DOI: `10.1109/OCEANS.2014.7003238`.

[BCS15] Simone Bianco, Gianluigi Ciocca und Raimondo Schettini. “How Far Can You Get By Combining Change Detection Algorithms?” In: *CoRR* abs/1505.02921 (2015). URL: `http://arxiv.org/abs/1505.02921`.

[BD11] O. Barnich und M. Van Droogenbroeck. “ViBe: A Universal Background Subtraction Algorithm for Video Sequences”. In: *IEEE Transactions on Image Processing* 20.6 (Juni 2011), S. 1709–1724. ISSN: 1057-7149. DOI: `10.1109/TIP.2010.2101613`.

[Bes+12] Frederic Besse, Carsten Rother, Andrew Fitzgibbon und Jan Kautz. “PMBP: PatchMatch Belief Propagation for Correspondence Field Estimation”. In: *BMVC - Best Industrial Impact Prize award*. Jan. 2012. URL: `https://www.microsoft.com/en-us/research/publication/pmbp-patchmatch-belief-propagation-for-correspondence-field-estimation/`.

[BFF06] J. Berclaz, F. Fleuret und P. Fua. “Robust People Tracking with Global Trajectory Optimization”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Bd. 1. Juni 2006, S. 744–750. DOI: `10.1109/CVPR.2006.258`.

[BG09] Serhat S. Bucak und Bilge Gunsel. “Incremental Subspace Learning via Non-negative Matrix Factorization”. In: *Pattern Recogn.* 42.5 (Mai 2009), S. 788–797. ISSN: 0031-3203. DOI: `10.1016/j.patcog.2008.09.002`. URL: `http://dx.doi.org/10.1016/j.patcog.2008.09.002`.

[BHH11] Sebastian Brutzer, Benjamin Höferlin und Gunther Heidemann. “Evaluation of Background Subtraction Techniques for Video Surveillance”. In: *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2011, S. 1937–1944.

[BM11] T. Brox und J. Malik. “Large displacement optical flow: descriptor matching in variational motion estimation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.3 (2011), S. 500–513. URL: `http://lmb.informatik.uni-freiburg.de//Publications/2011/Bro11a`.

[BR11] Ben Benfold und Ian Reid. “Stable Multi-Target Tracking in Real-Time Surveillance Video”. In: *CVPR*. Juni 2011, S. 3457–3464.

[Bre+11] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier und L. Van Gool. “Online Multiperson Tracking-by-Detection from a Single, Uncalibrated Camera”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.9 (Sep. 2011), S. 1820–1833. ISSN: 0162-8828. DOI: `10.1109/TPAMI.2010.232`.

[BS08] Keni Bernardin und Rainer Stiefelhagen. “Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics”. In: *J. Image Video Process.* 2008 (Jan. 2008), 1:1–1:10. ISSN: 1687-5176. DOI: `10.1155/2008/246309`. URL: `http://dx.doi.org/10.1155/2008/246309`.

[Bun+07] F. Bunyak, K. Palaniappan, S. K. Nath und G. Seetharaman. “Flux tensor constrained geodesic active contours with sensor fusion for persistent object tracking”. In: *J. Multimedia* 2.4 (Aug. 2007), S. 20–33. URL: `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2605095/`.

[Cam+16] M. Camplani, A. Paiement, M. Mirmehdi, D. Damen, S. Hannuna, T. Burghardt und L. Tao. “Multiple human tracking in RGB-depth data: a survey”. In: *IET Computer Vision* 11.4 (2016), S. 265–285. ISSN: 1751-9632. DOI: `10.1049/iet-cvi.2016.0178`.

[Car+10] M.A.G. de Carvalho, A.L. da Costa, A.C.B. Ferreira und R. Marcondes Cesar Junior. “Image Segmentation Using Component Tree and Normalized Cut”. In: *Graphics, Patterns and Images (SIBGRAPI), 2010 23rd SIBGRAPI Conference on*. Aug. 2010, S. 317–322. DOI: `10.1109/SIBGRAPI.2010.49`.

[Car+12] João Carreira, Rui Caseiro, Jorge Batista und Cristian Sminchisescu. “Semantic Segmentation with Second-Order Pooling”. In: *Computer Vision – ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VII*. Hrsg. von Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato und Cordelia Schmid. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 430–443. ISBN: 978-3-642-33786-4. DOI: `10.1007/978-3-642-33786-4_32`. URL: `http://dx.doi.org/10.1007/978-3-642-33786-4_32`.

[Chu+15] M. C. Chuang, J. N. Hwang, K. Williams und R. Towler. “Tracking Live Fish From Low-Contrast and Low-Frame-Rate Stereo Videos”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 25.1 (Jan. 2015), S. 167–179. ISSN: 1051-8215. DOI: `10.1109/TCSVT.2014.2357093`.

[Cir+11] Dan C. Cireşan, Ueli Meier, Jonathan Masci, Luca M. Gambardella und Jürgen Schmidhuber. “Flexible, High Performance Convolutional Neural Networks for Image Classification”. In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*. IJCAI’11. Barcelona, Catalonia, Spain: AAAI Press, 2011, S. 1237–1242. ISBN: 978-1-57735-514-4. DOI: `10.5591/978-1-57735-516-8/IJCAI11-210`. URL: `http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-210`.

[Cri+06] A. Criminisi, G. Cross, A. Blake und V. Kolmogorov. “Bilayer Segmentation of Live Video”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*. Bd. 1. Juni 2006, S. 53–60. DOI: `10.1109/CVPR.2006.69`.

[Dan+14] Alexandru Dancu, Mickaël Fourgeaud, Zlatko Franjcic und Razmik Avetisyan. “Underwater Reconstruction Using Depth Sensors”. In: *SIGGRAPH Asia 2014 Technical Briefs*. SA ’14. Shenzhen, China: ACM, 2014, 2:1–2:4. ISBN: 978-1-4503-2895-1. DOI: `10.1145/2669024.2669042`. URL: `http://doi.acm.org/10.1145/2669024.2669042`.

[DK14] T. Dolereit und A. Kuijper. “Converting underwater imaging into imaging in air”. In: *2014 International Conference on Computer Vision Theory and Applications (VISAPP)*. Bd. 1. Jan. 2014, S. 96–103.

[DLK15] T. Dolereit, U. F. von Lukas und A. Kuijper. “New constraints for underwater stereo calibration”. In: *2015 9th International Symposium on Image and Signal Processing and Analysis (ISPA)*. Sep. 2015, S. 176–181. DOI: `10.1109/ISPA.2015.7306054`.

[DP16] B. U. Dhaware und A. C. Pise. “Lung cancer detection using Bayasein classifier and FCM segmentation”. In: *2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*. Sep. 2016, S. 170–174. DOI: 10.1109/ICACDOT.2016.7877572.

[Dun73] J. C. Dunn. “A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters”. In: *Journal of Cybernetics* 3.3 (1973), S. 32–57. DOI: 10.1080/01969727308546046. URL: http://dx.doi.org/10.1080/01969727308546046.

[EHD00] Ahmed M. Elgammal, David Harwood und Larry S. Davis. “Non-parametric Model for Background Subtraction”. In: *Proceedings of the 6th European Conference on Computer Vision-Part II*. ECCV ’00. London, UK, UK: Springer-Verlag, 2000, S. 751–767. ISBN: 3-540-67686-4. URL: http://dl.acm.org/citation.cfm?id=645314.649432.

[FAH14] R. Fier, A. B. Albu und M. Hoeberechts. “Automatic fish counting system for noisy deep-sea videos”. In: *2014 Oceans - St. John’s*. Sep. 2014, S. 1–6. DOI: 10.1109/OCEANS.2014.7003118.

[Far03] Gunnar Farnebäck. “Two-Frame Motion Estimation Based on Polynomial Expansion”. English. In: *Image Analysis*. Hrsg. von Josef Bigun und Tomas Gustavsson. Bd. 2749. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, S. 363–370. ISBN: 978-3-540-40601-3. DOI: 10.1007/3-540-45103-X_50. URL: http://dx.doi.org/10.1007/3-540-45103-X_50.

[FH04] P.F. Felzenszwalb und D.P. Huttenlocher. “Efficient belief propagation for early vision”. In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Bd. 1. Juni 2004, I-261-I-268 Vol.1.

[FH75] K. Fukunaga und L. Hostetler. “The estimation of the gradient of a density function, with applications in pattern recognition”. In: *IEEE Transactions on Information Theory* 21.1 (Jan. 1975), S. 32–40. ISSN: 0018-9448. DOI: 10.1109/TIT.1975.1055330.

[FMB12] Diana Farcas, Cristina Marghes und Thierry Bouwmans. “Background subtraction via incremental maximum margin criterion: a discriminative subspace approach”. In: *Machine Vision and Applications* 23.6 (2012), S. 1083–1101. ISSN: 1432-1769. DOI: 10.1007/s00138-012-0421-9. URL: http://dx.doi.org/10.1007/s00138-012-0421-9.

[FS09] J. Ferryman und A. Shahrokni. “PETS2009: Dataset and challenge”. In: *2009 Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*. Dez. 2009, S. 1–6. DOI: 10.1109/PETS-WINTER.2009.5399556.

[FZL15] F. Farhadifard, Z. Zhou und U. F. von Lukas. “Learning-based underwater image enhancement with adaptive color mapping”. In: *2015 9th International Symposium on Image and Signal Processing and Analysis (ISPA)*. 2015, S. 48–53.

[Get12] Pascal Getreuer. “Automatic Color Enhancement (ACE) and its Fast Implementation”. In: *Image Processing On Line* 2 (2012), S. 266–277.

[GM99] T.R. Gardos und J. Monaco. *Encoding video images using foreground/background segmentation*. US Patent 5,915,044. Juni 1999. URL: `http://www.google.de/patents/US5915044`.

[Goy+12] N. Goyette, P. Jodoin, F. Porikli, J. Konrad und P. Ishwar. “Changedetection.net: A new change detection benchmark dataset”. In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. Juni 2012, S. 1–8. DOI: `10.1109/CVPRW.2012.6238919`.

[GRM02] C. Gatta, A. Rizzi und D. Marini. “ACE: An automatic color equalization algorithm”. In: *Proceedings of the First European Conference on color in Grpahics Image and Vision (CGIV02)*. 2002, S. 266–277.

[Guo13] Y. Guo. “3D underwater topography rebuilding based on single beam sonar”. In: *Signal Processing, Communication and Computing (ICSPCC), 2013 IEEE International Conference on*. Aug. 2013, S. 1–5. DOI: `10.1109/ICSPCC.2013.6664031`.

[GZS11] A. Geiger, J. Ziegler und C. Stiller. “StereoScan: Dense 3d reconstruction in real-time”. In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. Juni 2011, S. 963–968. DOI: `10.1109/IVS.2011.5940405`.

[Hag+16] M. Haghighat, Xiuying Li, Zicheng Fang, Yang Zhang und S. Negahdaripour. “Segmentation, classification and modeling of two-dimensional forward-scan sonar imagery for efficient coding and synthesis”. In: *OCEANS 2016 MTS/IEEE Monterey*. Sep. 2016, S. 1–8. DOI: `10.1109/OCEANS.2016.7761408`.

[He+15] Wei He u. a. “An improved multilevel banded graph cuts method for extracting colon tissue”. In: *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*. Aug. 2015, S. 369–374. DOI: `10.1109/FSKD.2015.7381970`.

[HHD00] Thanarat Horprasert, David Harwood und Larry S. Davis. “A robust background subtraction and shadow detection”. In: *In Proceedings of the Asian Conference on Computer Vision*. 2000.

[HMY99] H. Hase, K. Miyake und M. Yoneda. “Real-time snowfall noise elimination”. In: *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)*. Bd. 2. Okt. 1999, 406–409 vol.2. DOI: `10.1109/ICIP.1999.822927`.

[HR16] Junhwa Hur und Stefan Roth. “Joint Optical Flow and Temporally Consistent Semantic Segmentation”. In: *CoRR* abs/1607.07716 (2016). URL: `http://arxiv.org/abs/1607.07716`.

[HS80] Berthold K.P. Horn und Brian G. Schunck. *Determining Optical Flow*. Techn. Ber. Cambridge, MA, USA, 1980.

[Hu+11] Weiming Hu, Xi Li, Xiaoqin Zhang, Xinchu Shi, Stephen Maybank und Zhongfei Zhang. “Incremental Tensor Subspace Learning and Its Applications to Foreground Segmentation and Tracking”. In: *International Journal of Computer Vision* 91.3 (2011), S. 303–327. ISSN: 1573-1405. DOI: `10.1007/s11263-010-0399-6`. URL: `http://dx.doi.org/10.1007/s11263-010-0399-6`.

[Huo+16] G. Huo, S. X. Yang, Q. Li und Y. Zhou. “A Robust and Fast Method for Sidescan Sonar Image Segmentation Using Nonlocal Despeckling and Active Contour Model”. In: *IEEE Transactions on Cybernetics* PP.99 (2016), S. 1–18. ISSN: 2168-2267. DOI: `10.1109/TCYB.2016.2530786`.

[HW79] JA Hartigan und MA Wong. “Algorithm AS 136: A K-means clustering algorithm”. In: *Applied Statistics* (1979), S. 100–108.

[Iid+04] K. Iida, Yong Tang, T. Mukai und Y. Nishimori. “Measurement of fish school volume by multi-beam sonar. Directional resolution and estimation error”. In: *OCEANS ’04. MTTS/IEEE TECHNO-OCEAN ’04*. Bd. 1. Nov. 2004, 401–408 Vol.1. DOI: `10.1109/OCEANS.2004.1402950`.

[Isi25] Ernst Ising. “Beitrag zur Theorie des Ferromagnetismus”. German. In: *Zeitschrift für Physik* 31.1 (1925), S. 253–258. ISSN: 0044-3328. DOI: `10.1007/BF02980577`. URL: `http://dx.doi.org/10.1007/BF02980577`.

[JRD12] Xiaoyan Jiang, Erik Rodner und Joachim Denzler. “Multi-person Tracking-by-Detection Based on Calibrated Multi-camera Systems”. In: *Computer Vision and Graphics: International Conference, ICCVG 2012, Warsaw, Poland, September 24-26, 2012. Proceedings*. Hrsg. von Leonard Bolc, Ryszard Tadeusiewicz, Leszek J. Chmielewski und Konrad Wojciechowski. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, S. 743–751. ISBN: 978-3-642-33564-8. DOI: `10.1007/978-3-642-33564-8_89`. URL: `https://doi.org/10.1007/978-3-642-33564-8_89`.

[JWY91] Liu Jianzhuang, Li Wenqing und Tian Yupeng. “Automatic thresholding of gray-level pictures using two-dimension Otsu method”. In: *China., 1991 International Conference on Circuits and Systems*. Juni 1991, 325–327 vol.1. DOI: `10.1109/CICCAS.1991.184351`.

[Kav+14] Isaak Kavasidis, Simone Palazzo, Roberto Di Salvo, Daniela Giordano und Concetto Spampinato. “An innovative web-based collaborative platform for video annotation”. In: *Multimedia Tools and Applications* 70.1 (2014), S. 413–432. ISSN: 1573-7721. DOI: `10.1007/s11042-013-1419-7`. URL: `http://dx.doi.org/10.1007/s11042-013-1419-7`.

[KB02] Pakorn KaewTraKulPong und Richard Bowden. “An improved adaptive background mixture model for real-time tracking with shadow detection”. In: *Video-based surveillance systems*. Springer, 2002, S. 135–144.

[KBV15] B. B. Kumar, M. Bansal und P. Verma. “Designing of Licensed Number Plate Recognition system using hybrid technique from neural network template matching”. In: *2015 International Conference on Computing, Communication and Security (ICCCS)*. Dez. 2015, S. 1–6. DOI: `10.1109/CCCS.2015.7374170`.

[KF10] J. Karlekar und A. Fang. “Underwater swimmer segmentation”. In: *2010 IEEE International Conference on Multimedia and Expo*. Juli 2010, S. 619–624. DOI: `10.1109/ICME.2010.5582608`.

[KFL01] F. R. Kschischang, B. J. Frey und H. A. Loeliger. “Factor graphs and the sum-product algorithm”. In: *IEEE Transactions on Information Theory* 47.2 (Feb. 2001), S. 498–519.

[KRH15] T. Klinger, F. Rottensteiner und C. Heipke. “PROBABILISTIC MULTI-PERSON TRACKING USING DYNAMIC BAYES NETWORKS”. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* II-3/W5 (2015), S. 435–442. DOI: `10.5194/isprsannals-II-3-W5-435-2015`. URL: `https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/II-3-W5/435/2015/`.

[KRH17] T. Klinger, F. Rottensteiner und C. Heipke. “Probabilistic multi-person localisation and tracking in image sequences”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 127 (2017). Geospatial Week 2015, S. 73–88. ISSN: 0924-2716. DOI: `http://dx.doi.org/10.1016/j.isprsjprs.2016.11.006`. URL: `http://www.sciencedirect.com/science/article/pii/S0924271616305299`.

[KSH12] Alex Krizhevsky, Ilya Sutskever und Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems 25*. Hrsg. von F. Pereira, C. J. C. Burges, L. Bottou und K. Q. Weinberger. Curran Associates, Inc., 2012, S. 1097–1105. URL: `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`.

[Kuh55] H. W. Kuhn. “The Hungarian method for the assignment problem”. In: *Naval Research Logistics Quarterly* 2.1-2 (1955), S. 83–97. ISSN: 1931-9193. DOI: `10.1002/nav.3800020109`. URL: `http://dx.doi.org/10.1002/nav.3800020109`.

[KWT88] Michael Kass, Andrew Witkin und Demetri Terzopoulos. “Snakes: Active contour models”. In: *International Journal of Computer Vision* 1.4 (1988), S. 321–331. ISSN: 1573-1405. DOI: `10.1007/BF00133570`. URL: `http://dx.doi.org/10.1007/BF00133570`.

[Lab+12] R. T. Labuguen u. a. “Automated fish fry counting and schooling behavior analysis using computer vision”. In: *2012 IEEE 8th International Colloquium on Signal Processing and its Applications*. März 2012, S. 255–260. DOI: `10.1109/CSPA.2012.6194729`.

[Lau+12] P. Y. Lau, P. L. Correia, P. Fonseca und A. Campos. “Estimating Norway lobster abundance from deep-water videos: an automatic approach”. In: *IET Image Processing* 6.1 (Feb. 2012), S. 22–30. ISSN: 1751-9659. DOI: `10.1049/iet-ipr.2009.0426`.

[LBH15] Yann LeCun, Yoshua Bengio und Geoffrey Hinton. “Deep learning”. In: *Nature* 521 (2015), S. 436–444. URL: `http://dx.doi.org/10.1038/nature14539`.

[Len+11] P. Lenz, J. Ziegler, A. Geiger und M. Roser. “Sparse scene flow segmentation for moving object detection in urban environments”. In: *2011 IEEE Intelligent Vehicles Symposium (IV)*. Juni 2011, S. 926–932. DOI: `10.1109/IVS.2011.5940558`.

[LH14] Jongwoo Lim und Bohyung Han. “Generalized Background Subtraction Using Superpixels with Label Integrated Motion Estimation”. In: *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*. Hrsg. von David Fleet, Tomas Pajdla, Bernt Schiele und Tinne Tuytelaars. Cham: Springer International Publishing, 2014, S. 173–187. ISBN: 978-3-319-10602-1. DOI: `10.1007/978-3-319-10602-1_12`. URL: `http://dx.doi.org/10.1007/978-3-319-10602-1_12`.

[Li+08] Xi Li, Weiming Hu, Zhongfei Zhang und Xiaoqin Zhang. “Robust Foreground Segmentation Based on Two Effective Background Models”. In: *Proceedings of the 1st ACM International Conference on Multimedia Information Retrieval*. MIR ’08. Vancouver, British Columbia, Canada: ACM, 2008, S. 223–228. ISBN: 978-1-60558-312-9. DOI: `10.1145/1460096.1460133`. URL: `http://doi.acm.org/10.1145/1460096.1460133`.

[Li+16] X. Li, M. Shang, J. Hao und Z. Yang. “Accelerating fish detection and recognition by sharing CNNs with objectness learning”. In: *OCEANS 2016 - Shanghai*. Apr. 2016, S. 1–5. DOI: `10.1109/OCEANSAP.2016.7485476`.

[Liu+10] S. Liu, H. Zhang, J. Jia und B. Li. “Feature recognition for underwater weld images”. In: *Proceedings of the 29th Chinese Control Conference*. Juli 2010, S. 2729–2734.

[Liu+16] H. Liu, J. Dai, R. Wang, H. Zheng und B. Zheng. “Combining background subtraction and three-frame difference to detect moving object from underwater video”. In: *OCEANS 2016 - Shanghai*. Apr. 2016, S. 1–5. DOI: `10.1109/OCEANSAP.2016.7485613`.

[LPR11] L. Leal-Taixé, G. Pons-Moll und B. Rosenhahn. “Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker”. In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. Nov. 2011, S. 120–127. DOI: `10.1109/ICCVW.2011.6130233`.

[LSD15] Jonathan Long, Evan Shelhamer und Trevor Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *CVPR (to appear)* (Nov. 2015). arXiv: `1411.4038 [cs.CV]`.

[Lyo14] Aidan Lyon. “Why are Normal Distributions Normal?” In: *The British Journal for the Philosophy of Science* 65.3 (2014), S. 621. DOI: `10.1093/bjps/axs046`. eprint: `/oup/backfile/content_public/journal/bjps/65/3/10.1093_bjps_axs046/3/axs046.pdf`. URL: `+%20http://dx.doi.org/10.1093/bjps/axs046`.

[LZF12] F. Luetteke, X. Zhang und J. Franke. “Implementation of the Hungarian Method for object tracking on a camera monitored transportation system”. In: *ROBOTIK 2012; 7th German Conference on Robotics*. Mai 2012, S. 1–6.

[Mar+13] A. Martins u. a. “Groundtruth system for underwater benchmarking”. In: *2013 OCEANS - San Diego*. Sep. 2013, S. 1–5.

[Mat+07] I. Matsuo, T. Imaizumi, M. Furusawa, T. Akamatsu, Y. Nishimori und S. Ogawa. “Analysis of the Echo for Identifying the Temporal Structure of the Fish by Using the Broadband Sonar Signal of Dolphin”. In: *2007 Symposium on Underwater Technology and Workshop on Scientific Use of Submarine Cables and Related Technologies*. Apr. 2007, S. 540–545. DOI: `10.1109/UT.2007.370758`.

[Mat75] B.W. Matthews. “Comparison of the predicted and observed secondary structure of T4 phage lysozyme”. In: *Biochimica et Biophysica Acta (BBA) - Protein Structure* 405.2 (1975), S. 442–451.

[MB16] A. Marburg und K. Bigham. “Deep learning for benthic fauna identification”. In: *OCEANS 2016 MTS/IEEE Monterey*. Sep. 2016, S. 1–5. DOI: `10.1109/OCEANS.2016.7761146`.

[MHG18] Moritz Menze, Christian Heipke und Andreas Geiger. “Object Scene Flow”. In: *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)* (2018).

[Mig10] M. Mignotte. “A Label Field Fusion Bayesian Model and Its Penalized Maximum Rand Estimator for Image Segmentation”. In: *IEEE Transactions on Image Processing* 19.6 (Juni 2010), S. 1610–1624.

[MOH00] T. Matsuyama, T. Ohya und H. Habe. “Background Subtraction for Non-Stationary Scenes”. In: *Proceedings of Asian Conference on Computer Vision*. 2000, S. 662–667.

[Mor+05] E. F. Morais, M. F. M. Campos, F. L. C. Padua und R. L. Carceroni. “Particle Filter-Based Predictive Tracking for Robust Fish Counting”. In: *XVIII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI’05)*. Okt. 2005, S. 367–374. DOI: `10.1109/SIBGRAPI.2005.36`.

[MS08] M. N. Mokti und R. A. Salam. “Hybrid of Mean-shift and median-cut algorithm for fish segmentation”. In: *Electronic Design, 2008. ICED 2008. International Conference on*. Dez. 2008, S. 1–5. DOI: `10.1109/ICED.2008.4786645`.

[MTR12] Marghes, Bouwmans T. und Vasiu R. “Background Modeling and Foreground Detection via a Reconstructive and Discriminative Subspace Learning Approach”. In: *Proceedings of the 2012 International Conferecne on Image Processing, Computer Vision and Patternrecognition*. 2012, S. 106–113. ISBN: 1-60132-225-9.

[Nan+16] K. Nandy, R. Chellappa, A. Kumar und S. J. Lockett. “Segmentation of Nuclei From 3D Microscopy Images of Tissue via Graphcut Optimization”. In: *IEEE Journal of Selected Topics in Signal Processing* 10.1 (Feb. 2016), S. 140–150. ISSN: 1932-4553. DOI: `10.1109/JSTSP.2015.2505148`.

[NG98] H. -H. Nagel und A. Gehrke. “Spatiotemporally adaptive estimation and segmentation of OF-fields”. In: *Computer Vision — ECCV’98: 5th European Conference on Computer Vision Freiburg, Germany, June 2–6, 1998 Proceedings, Volume II*. Hrsg. von Hans Burkhardt und Bernd Neumann. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, S. 86–102. ISBN: 978-3-540-69235-5. DOI: `10.1007/BFb0054735`. URL: `http://dx.doi.org/10.1007/BFb0054735`.

[NP05] S. Nath und K. Palaniappan. “Adaptive robust structure tensors for orientation estimation and image segmentation”. In: *Lecture Notes in Computer Science (ISVC)* 3804 (2005), S. 445–453. DOI: `10.1007/11595755_54`.

[OK15] A. Opitz und A. Kriechbaum-Zabini. “Evaluation of face recognition technologies for identity verification in an eGate based on operational data of an airport”. In: *2015 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. Aug. 2015, S. 1–5. DOI: `10.1109/AVSS.2015.7301747`.

[OMB14] P. Ochs, J. Malik und T. Brox. “Segmentation of Moving Objects by Long Term Video Analysis”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.6 (Juni 2014), S. 1187–1200. ISSN: 0162-8828. DOI: `10.1109/TPAMI.2013.242`.

[ONN16] J. Osterloff, I. Nilssen und T. W. Nattkemper. “Computational coral feature monitoring for the fixed underwater observatory LoVe”. In: *OCEANS 2016 MTS/IEEE Monterey*. Sep. 2016, S. 1–5. DOI: `10.1109/OCEANS.2016.7761417`.

[ORP00] N.M. Oliver, B. Rosario und AP. Pentland. “A Bayesian computer vision system for modeling human interactions”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.8 (Aug. 2000), S. 831–843. ISSN: 0162-8828. DOI: `10.1109/34.868684`.

[Ota+11] T. Otaki, Y. Miyamoto, K. Amakasu, K. Uchida und T. Sasakura. “Development of a sonar-responding acoustic tag for fish behavior observation”. In: *Underwater Technology (UT), 2011 IEEE Symposium on and 2011 Workshop on Scientific Use of Submarine Cables and Related Technologies (SSC)*. Apr. 2011, S. 1–4. DOI: `10.1109/UT.2011.5774085`.

[Ots79] Nobuyuki Otsu. “A threshold selection method from gray-level histograms”. In: *Systems, Man and Cybernetics,IEEE Transactions on* 9.1 (1979), S. 62–66.

[Pel+09] S. Pellegrini, A. Ess, K. Schindler und L. van Gool. “You’ll never walk alone: Modeling social behavior for multi-target tracking”. In: *2009 IEEE 12th International Conference on Computer Vision*. Sep. 2009, S. 261–268. DOI: `10.1109/ICCV.2009.5459260`.

[PF14] L. Patino und J. Ferryman. “PETS 2014: Dataset and challenge”. In: *2014 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. Aug. 2014, S. 355–360. DOI: `10.1109/AVSS.2014.6918694`.

[PJB04] K. Palaniappan, H. S. Jiang und T. I. Baskin. “Non-rigid motion estimation using the robust tensor method”. In: *2004 Conference on Computer Vision and Pattern Recognition Workshop*. Juni 2004, S. 25–25. DOI: `10.1109/CVPR.2004.405`.

[PL10] Yanmin Peng und Rong Liu. “Object segmentation based on watershed and graph cut”. In: *Image and Signal Processing (CISP), 2010 3rd International Congress on*. Bd. 3. Okt. 2010, S. 1431–1435. DOI: `10.1109/CISP.2010.5647066`.

[PM16] D. K. Panda und S. Meher. “Detection of Moving Objects Using Fuzzy Color Difference Histogram Based Background Subtraction”. In: *IEEE Signal Processing Letters* 23.1 (Jan. 2016), S. 45–49. ISSN: 1070-9908. DOI: `10.1109/LSP.2015.2498839`.

[Pow11] D. M. W. Powers. “Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation”. In: *Journal of Machine Learning Technologies* 2.1 (2011), S. 37–63.

[Pra+03] Andrea Prati, Ivana Mikic, Mohan M. Trivedi und Rita Cucchiara. “Detecting moving shadows: Formulation, algorithms and evaluation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25 (2003), S. 2003.

[Pur+09] Autun Purser, Melanie Bergmann, JÃ¶rg Ontrup und Tim Wilhelm Nattkemper. “Use of machine-learning algorithms for the automated detection of cold-water coral habitats: a pilot study”. In: *Marine Ecology Progress Series* 397 (2009), S. 241–251. DOI: `10.3354/meps08154`.

[QPL14] H. Qin, Y. Peng und X. Li. “Foreground Extraction of Underwater Videos via Sparse and Low-Rank Matrix Decomposition”. In: *2014 ICPR Workshop on Computer Vision for Analysis of Underwater Imagery*. Aug. 2014, S. 65–72. DOI: `10.1109/CVAUI.2014.16`.

[RC14] G. Ramírez-Alonso und M. I. Chacón-Murguía. “Segmentation of dynamic objects in video sequences fusing the strengths of a background subtraction model, optical flow and matting algorithms”. In: *Image Analysis and Interpretation (SSIAI), 2014 IEEE Southwest Symposium on*. Apr. 2014, S. 33–36. DOI: `10.1109/SSIAI.2014.6806022`.

[RGS12] R. Kumar Rai, P. Gour und B. Singh. “Underwater Image Segmentation using CLAHE Enhancement and Thresholding”. In: *International Journal of Emerging Technology and Advanced Engineering, vol. 2*. 2012, S. 118–123.

[RMK95] Christof Ridder, Olaf Munkelt und Harald Kirchner. “Adaptive background estimation and foreground detection using kalman-filtering”. In: *Proceedings of International Conference on recent Advances in Mechatronics*. 1995, S. 193–199.

[SAS17] Amir Sadeghian, Alexandre Alahi und Silvio Savarese. “Tracking The Untrackable: Learning To Track Multiple Cues with Long-Term Dependencies”. In: *CoRR* abs/1701.01909 (2017). URL: `http://arxiv.org/abs/1701.01909`.

[SBB15] P. L. St-Charles, G. A. Bilodeau und R. Bergevin. “SuBSENSE: A Universal Change Detection Method With Local Adaptive Sensitivity”. In: *IEEE Transactions on Image Processing* 24.1 (Jan. 2015), S. 359–373. ISSN: 1057-7149. DOI: `10.1109/TIP.2014.2378053`.

[SCL12] D. T. Swain, I. D. Couzin und N. Ehrich Leonard. “Real-Time Feedback-Controlled Robotic Fish for Behavioral Experiments With Fish Schools”. In: *Proceedings of the IEEE* 100.1 (Jan. 2012), S. 150–163. ISSN: 0018-9219. DOI: `10.1109/JPROC.2011.2165449`.

[Set+06] N. Setiawan, S. Hong, J. Kim und C. Lee. “Gaussian mixture model in improved IHLS color space for human silhouette extraction”. In: *16th Int Conf on Artificial Reality and Telexistence (ICAT 2006)*. 2006, S. 732–741.

[Sey+16a] K. Seyid, A. Richaud, R. Capoccia und Y. Leblebici. “Block matching based real-time optical flow hardware implementation”. In: *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. Mai 2016, S. 2206–2209. DOI: `10.1109/ISCAS.2016.7539020`.

[Sey+16b] K. Seyid, A. Richaud, R. Capoccia und Y. Leblebici. “FPGA Based Hardware Implementation of Real-Time Optical Flow Calculation”. In: *IEEE Transactions on Circuits and Systems for Video Technology* PP.99 (2016), S. 1–1. ISSN: 1051-8215. DOI: `10.1109/TCSVT.2016.2598703`.

[SG00] C. Stauffer und W. E. L. Grimson. “Learning patterns of activity using real-time tracking”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (Aug. 2000), S. 747–757. ISSN: 0162-8828. DOI: `10.1109/34.868677`.

[SG99] Chris Stauffer und W.E.L. Grimson. “Adaptive background mixture models for real-time tracking”. In: *Proceedings 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Vol. Two*. IEEE Computer Society Press, Juni 1999, S. 246–252.

[Sha+15] M. H. Sharif, F. Galip, A. Guler und S. Uyaver. “A simple approach to count and track underwater fishes from videos”. In: *2015 18th International Conference on Computer and Information Technology (ICCIT)*. Dez. 2015, S. 347–352. DOI: `10.1109/ICCITechn.2015.7488094`.

[SHP12] M. A. Soeleman, M. Hariadi und M. H. Purnomo. “Adaptive threshold for background subtraction in moving object detection using Fuzzy C-Means clustering”. In: *TENCON 2012 IEEE Region 10 Conference*. Nov. 2012, S. 1–5. DOI: `10.1109/TENCON.2012.6412265`.

[Shu+12] G. Shu, A. Dehghan, O. Oreifej, E. Hand und M. Shah. “Part-based multiple-person tracking with partial occlusion handling”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. Juni 2012, S. 1815–1821. DOI: `10.1109/CVPR.2012.6247879`.

[SJ15] K. A. Skinner und M. Johnson-Roberson. “Detection and segmentation of underwater archaeological sites surveyed with stereo-vision platforms”. In: *OCEANS 2015 - MTS/IEEE Washington*. Okt. 2015, S. 1–7.

[SM00] Jianbo Shi und Jitendra Malik. “Normalized cuts and image segmentation”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22.8 (Aug. 2000), S. 888–905. ISSN: 0162-8828. DOI: `10.1109/34.868688`.

[Son+16] S. Song, B. Si, X. Feng und K. Liu. “Label field initialization for MRF-based sonar image segmentation by selective autoencoding”. In: *OCEANS 2016 - Shanghai*. Apr. 2016, S. 1–5. DOI: `10.1109/OCEANSAP.2016.7485644`.

[Soo+14] K. Sooknanan, J. Doyle, C. Lordan, J. Wilson, A. Kokaram und D. Corrigan. “Mosaics for Nephrops detection in underwater survey videos”. In: *2014 Oceans - St. John’s*. Sep. 2014, S. 1–6. DOI: `10.1109/OCEANS.2014.7003142`.

[Spa+08] Concetto Spampinato, Yun-Heh Chen-Burger, Gayathri Nadarajan und Robert B Fisher. “Detecting, Tracking and Counting Fish in Low Quality Unconstrained Underwater Videos.” In: *VISAPP (2)* 2008 (2008), S. 514–519.

[SS93] A. J. Shelley und N. L. Seed. “Approaches to static background identification and removal”. In: *Image Processing for Transport Applications, IEE Colloquium on*. Dez. 1993, S. 6/1–6/4.

[SW06] Konrad Schindler und Hanzi Wang. “Smooth Foreground-background Segmentation for Video Processing”. In: *Proceedings of the 7th Asian Conference on Computer Vision - Volume Part II*. ACCV’06. Hyderabad, India: Springer-Verlag, 2006, S. 581–590. ISBN: 3-540-31244-7, 978-3-540-31244-4. DOI: `10.1007/11612704_58`. URL: `http://dx.doi.org/10.1007/11612704_58`.

[TCM04] L. Tao, U. Castellani und V. Murino. “Robust 3D segmentation for underwater acoustic images”. In: *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*. Sep. 2004, S. 813–819. DOI: `10.1109/TDPVT.2004.1335399`.

[Tia+11] Z. Tian, J. Xue, N. Zheng, X. Lan und C. Li. “3D spatio-temporal graph cuts for video objects segmentation”. In: *2011 18th IEEE International Conference on Image Processing*. Sep. 2011, S. 2393–2396. DOI: `10.1109/ICIP.2011.6116124`.

[TL09] D. Tsai und C. Lai. “Independent component analysis-based background subtraction for indoor surveillance”. In: *IEEE Trans Image Proc IP 2009*. Bd. 18. 2009, S. 158–167.

[TNL09] Y. H. Toh, T. M. Ng und B. K. Liew. “Automated Fish Counting Using Image Processing”. In: *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*. Dez. 2009, S. 1–5. DOI: `10.1109/CISE.2009.5365104`.

[Toy+99] Kentaro Toyama, John Krumm, Barry Brumitt und Brian Meyers. “Wallflower: Principles and Practice of Background Maintenance”. In: *Seventh International Conference on Computer Vision*. IEEE Computer Society Press, Sep. 1999, S. 255–261.

[Tru+00] E. Trucco, Y.R. Petillot, I.Tena Ruiz, K. Plakas und D.M. Lane. “Feature Tracking in Video and Sonar Subsea Sequences with Applications”. In: *Computer Vision and Image Understanding* 79.1 (2000), S. 92–122. ISSN: 1077-3142. DOI: `http://dx.doi.org/10.1006/cviu.2000.0846`. URL: `http://www.sciencedirect.com/science/article/pii/S1077314200908464`.

[Tsu+14] C. L. Tsui, D. Schipf, K. R. Lin, J. Leang, F. J. Hsieh und W. C. Wang. “Using a Time of Flight method for underwater 3-dimensional depth measurements and point cloud imaging”. In: *OCEANS 2014 - TAIPEI*. Apr. 2014, S. 1–6. DOI: `10.1109/OCEANS-TAIPEI.2014.6964471`.

[Vit67] A. Viterbi. “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm”. In: *IEEE Transactions on Information Theory* 13.2 (Apr. 1967), S. 260–269. ISSN: 0018-9448. DOI: `10.1109/TIT.1967.1054010`.

[VJ04] Paul Viola und MichaelJ. Jones. “Robust Real-Time Face Detection”. English. In: *International Journal of Computer Vision* 57.2 (2004), S. 137–154. ISSN: 0920-5691. DOI: `10.1023/B:VISI.0000013087.49260.fb`.

[Wan+11] S. l. Wang, Y. r. Xu, L. Wan und X. d. Tang. “Marine Images Segmentation Using Adaptive Fuzzy c-Means Algorithm Based on Spatial Neighborhood”. In: *2011 Third Pacific-Asia Conference on Circuits, Communications and System (PACCS)*. Juli 2011, S. 1–6. DOI: `10.1109/PACCS.2011.5990237`.

[Wan+14a] R. Wang, F. Bunyak, G. Seetharaman und K. Palaniappan. “Static and Moving Object Detection Using Flux Tensor with Split Gaussian Models”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*. Juni 2014, S. 420–424. DOI: `10.1109/CVPRW.2014.68`.

[Wan+14b] Yi Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth und P. Ishwar. “CDnet 2014: An Expanded Change Detection Benchmark Dataset”. In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*. Juni 2014, S. 393–400. DOI: `10.1109/CVPRW.2014.126`.

[Wan+15] Yiru Wang, Liqin Fu, Kexin Liu, Rui Nian, Tianhong Yan und A. Lendasse. “Stable underwater image segmentation in high quality via MRF model”. In: *OCEANS 2015 - MTS/IEEE Washington*. Okt. 2015, S. 1–4.

[WB14] L. M. Wolff und S. Badri-Hoeher. “Imaging sonar-based fish detection in shallow waters”. In: *2014 Oceans - St. John’s*. Sep. 2014, S. 1–6. DOI: `10.1109/OCEANS.2014.7003213`.

[WB15] J. Wulff und M. J. Black. “Efficient sparse-to-dense optical flow estimation using a learned basis and layers”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Juni 2015, S. 120–130. DOI: `10.1109/CVPR.2015.7298607`.

[Wen+17] Longyin Wen, Zhen Lei, Ming-Ching Chang, Honggang Qi und Siwei Lyu. “Multi-Camera Multi-Target Tracking with Space-Time-View Hyper-graph”. In: *International Journal of Computer Vision* 122.2 (Apr. 2017), S. 313–333.

[WG16] B. Wang und Y. Gao. “Structure Integral Transform Versus Radon Transform: A 2D Mathematical Tool for Invariant Shape Recognition”. In: *IEEE Transactions on Image Processing* 25.12 (Dez. 2016), S. 5635–5648. ISSN: 1057-7149. DOI: `10.1109/TIP.2016.2609816`.

[Wie64] Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964. ISBN: 0262730057.

[Wre+97] Christopher Wren, Ali Azarbayejani, Trevor Darrell und Alex Pentland. "Pfinder: Real-Time Tracking of the Human Body". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19 (1997), S. 780–785.

[WS07] B. White und M. Shah. "Automatically Tuning Background Subtraction Parameters using Particle Swarm Optimization". In: *Multimedia and Expo, 2007 IEEE International Conference on*. Juli 2007, S. 1826–1829. DOI: `10.1109/ICME.2007.4285028`.

[WWW11] Xiufen Wang, Huiyuan Wang und Song Wang. "Jellyfish detection based on K-FOE residual map and ring segmentation". In: *2011 IEEE 13th International Conference on Communication Technology*. Sep. 2011, S. 762–766. DOI: `10.1109/ICCT.2011.6157979`.

[WZW04] S. K. Warfield, K. H. Zou und W. M. Wells. "Simultaneous Truth and Performance Level Estimation (STAPLE): An Algorithm for the Validation of Image Segmentation". In: *Ieee Transactions on Medical Imaging* 23 (2004), S. 903–921.

[XLL12] Q. Xiao, X. Liu und M. Liu. "Object Tracking Based on Local Feature Matching". In: *2012 Fifth International Symposium on Computational Intelligence and Design*. Bd. 1. Okt. 2012, S. 399–402. DOI: `10.1109/ISCID.2012.106`.

[Yan+09] J. Yang, P. A. Vela, Z. Shi und J. Teizer. "Probabilistic multiple people tracking through complex situations". In: *Performance Evaluation of Tracking and Surveillance workshop at CVPR 2009*. Miami, Florida, 2009, S. 79–86.

[YFW03] Jonathan S. Yedidia, William T. Freeman und Yair Weiss. "Exploring Artificial Intelligence in the New Millennium". In: Hrsg. von Gerhard Lakemeyer und Bernhard Nebel. 2003. Kap. Understanding Belief Propagation and Its Generalizations, S. 239–269. ISBN: 1-55860-811-7. URL: `http://dl.acm.org/citation.cfm?id=779343.779352`.

[Yin+07] P. Yin, A. Criminisi, J. Winn und I. Essa. "Tree-based Classifiers for Bilayer Video Segmentation". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. Juni 2007, S. 1–8. DOI: `10.1109/CVPR.2007.383008`.

[ZH06] Zoran Zivkovic und Ferdinand Heijden. "Efficient Adaptive Density Estimation Per Image Pixel for the Task of Background Subtraction". In: *Pattern Recogn. Lett.* 27.7 (Mai 2006), S. 773–780. ISSN: 0167-8655.

[Zha+15a] Wei Zhang, Detao Meng, Zhicheng Liang, Yi Guo, Jiajia Zhou und Yunfeng Han. "Research on recognition method of UUV vision based on PSO-BP network". In: *OCEANS 2015 - MTS/IEEE Washington*. Okt. 2015, S. 1–6.

[Zha+15b] Y. Zhao, G. Lv, T. Ma, H. Ji und H. Zheng. "A novel method of surveillance video Summarization based On clustering and background subtraction". In: *2015 8th International Congress on Image and Signal Processing (CISP)*. Okt. 2015, S. 131–136. DOI: `10.1109/CISP.2015.7407863`.

[Zha+16] L. Zhang, B. He, Y. Song und T. Yan. "Consistent target tracking via multiple underwater cameras". In: *OCEANS 2016 - Shanghai*. Apr. 2016, S. 1–4. DOI: `10.1109/OCEANSAP.2016.7485371`.

[Ziv04] Zoran Zivkovic. “Improved Adaptive Gaussian Mixture Model for Background Subtraction”. In: *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR’04), Volume 2 - Volume 02*. ICPR ’04. IEEE Computer Society, 2004, S. 28–31.

[ZWH16] P. Zhang, L. Wang und S. Han. “Research on segmentation of sonar image based on features learning”. In: *2016 IEEE International Conference on Mechatronics and Automation*. Aug. 2016, S. 1897–1901. DOI: `10.1109/ICMA.2016.7558855`.

[ZYD14] D. Zamalieva, A. Yilmaz und J. W. Davis. “Exploiting Temporal Geometry for Moving Camera Background Subtraction”. In: *Pattern Recognition (ICPR), 2014 22nd International Conference on*. Aug. 2014, S. 1200–1205.

[ZZY15] Xiaochun Zou, Xinbo Zhao und Yongjia Yang. “Automatic segmentation of moving objects considering single moving camera”. In: *2015 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. Juli 2015, S. 1–7. DOI: `10.1109/ICCCNT.2015.7395233`.

Own Publications

[FR16] Fahimeh Farhadifard und Martin Radolko. “Adaptive UW Image Deblurring via Sparse Representation”. In: *EG 2016 - Short Papers*. Hrsg. von T. Bashford-Rogers und L. P. Santos. The Eurographics Association, 2016. DOI: `10.2312/egsh.20161010`.

[FRL17a] Fahimeh Farhadifard, Martin Radolko und Uwe Freiherr von Lukas. “Marine-Snow Detection and Removal: Underwater Image Restoration using Background Modeling”. In: *In Proceedings of the 25th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, WSCG 2017 - Full Papers Proceedings*. 2017, S. 81–90.

[FRL17b] Fahimeh Farhadifard, Martin Radolko und Uwe Freiherr von Lukas. “Single Image Marine Snow Removal based on a Supervised Median Filtering Scheme”. In: *In Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017) - Volume 4: VISAPP*. 2017, S. 280–287. ISBN: 978-989-758-225-7.

[Gut+15] Enrico Gutzeit, Martin Radolko, Arjan Kuijper und Uwe von Lukas. “Optimization-based Automatic Segmentation of Organic Objects of Similar Types”. In: *Proceedings of the 10th International Conference on Computer Vision Theory and Applications (VISIGRAPP 2015)*. 2015, S. 591–598. ISBN: 978-989-758-089-5. DOI: `10.5220/0005314905910598`.

[Rad+15] Martin Radolko, Fahimeh Farhadifard, Enrico Gutzeit und Uwe Freiherr von Lukas. “Real time video segmentation optimization with a modified Normalized Cut”. In: *Image and Signal Processing and Analysis (ISPA), 2015 9th International Symposium on*. Sep. 2015, S. 31–36.

[Rad+16] Martin Radolko, Fahimeh Farhadifard, Enrico Gutzeit und Uwe Freiherr von Lukas. “Dataset on Underwater Change Detection”. In: *OCEANS 2016 - MONTEREY*. 2016, S. 1–8.

[Rad15] Martin Radolko. “Comparison of Spatial Models for Foreground-Background Segmentation in Underwater Videos”. In: *Proceedings of the International Summer School on Visual Computing 2015*. Hrsg. von Hans-Jörg Schulz. Aug. 2015, S. 91–100.

[RF16] Martin Radolko und Fahimeh Farhadifard. “Using trajectories derived by dense optical flows as a spatial component in background subtraction”. In: *24th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, WSCG 2016 - Full Papers Proceedings*. 2016, S. 1–8.

[RFL17a] Martin Radolko, Fahimeh Farhadifard und Uwe Freiherr von Lukas. “Background Modeling: Dealing with Pan, Tilt or Zoom in Videos”. In: *Proceedings of the 25th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, WSCG 2017 - Poster Papers Proceedings*. 2017, S. 53–58.

[RFL17b] Martin Radolko, Fahimeh Farhadifard und Uwe Freiherr von Lukas. “Change Detection in Crowded Underwater Scenes - Via an Extended Gaussian Switch Model Combined with a Flux Tensor Pre-segmentation”. In: *Proceedings of the 12th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2017) - Volume 4: VISAPP*. 2017, S. 405–415. ISBN: 978-989-758-225-7.

[RFL18] Martin Radolko, Fahimeh Farhadifard und Uwe Freiherr von Lukas. “Change Detection and Blob Tracking of Fish in Underwater Scenarios”. submitted to Communications in Computer and Information Science. 2018.

[RG15] Martin Radolko und Enrico Gutzeit. “Video Segmentation via a Gaussian Switch Background-Model and Higher Order Markov Random Fields”. In: *Proceedings of the 10th International Conference on Computer Vision Theory and Applications Volume 1*. 2015, S. 537–544.

Thesis Statements

1. Change in a scene is an important cue – for humans as well as computers – that helps in quickly understanding the most important aspects of a given situation and making important decisions. Therefore, the detection of change is a vital aspect in the field of computer vision and often the first step in a whole pipeline of different algorithms to extract information from a video. Furthermore, its generality allows a broad usage in almost any scenario with a static camera.

2. There already exists a wide range of literature about change detection and it has been applied successfully in many different scenarios. However, some aspects have not been sufficiently addressed in their entirety. Examples for this are the lack of spatial coherency in background subtraction methods or the segmentation of crowded scenes. Other aspects that are more special, like the handling of underwater scenarios, have been neglected almost completely so far.

3. Since many change detection approaches are based on a pixel-wise approach, where each pixel is handled completely separately, there is no spatial coherence inherent in these methods. Nonetheless, spatially coherent detections are necessary to facilitate subsequent steps in the computer vision pipeline like tracking or classification. Hence, the addition of a spatial model is mandatory for the production of accurate and useful results.

4. The objects which shall be detected in natural images – in water or in air – can be expected to be smooth, coherent and without holes. This is also true for most synthetic images as they usually imitate natural scenes. A spatial model must consider these properties and adapt the change detection results accordingly.

5. Underwater videos suffer from various additional degradations in contrast to in-air videos. These effects mainly emerge from the different physical properties of the medium water and can heavily impair the quality of the segmentations from change detection methods and computer vision algorithms in general. Hence, special consideration of these degradations is necessary when dealing with underwater videos.

6. Change detection approaches based on the optical flow suffer more from underwater degradations than methods which rely on background modeling. The computation of the optical flow relies heavily on the detection and matching of features in the images, and the degradation effects of underwater scenes result in fewer features and a reduced distinctiveness of the remaining features. In general, background modeling should be the preferred method for change detection in underwater videos.

7. Today's underwater image enhancement algorithms are designed with the perception of humans in mind and not optimized for change detection methods. Therefore, their effect

on the accuracy of change detection results can be positive but is extremely depending on the scene. For the general case, only methods that deal with specific effects (like Marine Snow or color cast removal) can be recommended as they have no negative impact on the segmentations on videos without these effects but usually increase the accuracy when the effect is present. If a specific scene is given, the enhancement method has to be chosen with extreme care to achieve measurable accuracy improvements and no deteriorations.

8. Fishes of a specific species are very similar among themselves and often also to the background of the scene, especially the fishes with swarm behavior which are used for livestock. Their similarity to the background makes change detection by background subtraction difficult. However, at the same time, the similarity among themselves can be used to build a reliable and consistent foreground model which can be very beneficial for the separation of foreground and background.

9. When whole swarms of fishes appear they pose a special problem since the view to the background can be completely obstructed. This aggravates the background modeling immensely since it works under the assumption that the background is visible most of the time. This problem can be relieved with the prior exclusion of foreground areas by using pre-segmentations based on the optical flow.

10. Most tracking approaches use object detectors which are especially trained and adjusted for a particular object type (often humans or cars). General detectors do not have this information and, therefore, cannot detect only one specific object class and have problems separating different objects which are close to each other. Tracking based on such a general detection approach, blob tracking, has to consider these problems and deal with blobs that represent several actual objects and blobs that are merging and splitting constantly while not losing track of the original detections.