On Outlier Detection in Sequences

—

Finding Anomalies in Mountain Silhouettes

Inaugural-Dissertation

zur

Erlangung des Doktorgrades der
Mathematisch-Naturwissenschaftlichen Fakultät
der Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Michael Singhof

aus Köln

Düsseldorf, März 2019

aus dem Institut für Informatik der
Heinrich-Heine Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

Referent: Prof. Dr. Stefan Conrad

Koreferent: Prof. Dr. Michael Schöttner

Tag der mündlichen Prüfung: 14.05.2019

Erklärung

Ich versichere an Eides Statt, dass die vorliegende Dissertation von mir selbständig und ohne unzulässige fremde Hilfe unter Beachtung der „Grundsätze zur Sicherung guter wissenschaftlicher Praxis an der Heinrich-Heine-Universität Düsseldorf“ erstellt worden ist. Desweiteren erkläre ich, dass ich eine Dissertation in der vorliegenden oder in ähnlicher Form noch bei keiner anderen Institution eingereicht habe. Ich habe keinerlei andere Promotionsversuche unternommen.

Düsseldorf, 1. März 2019 Michael Singhof

Acknowledgements

This thesis would not have the form presented here – or any form at all – without the help of many people to whom I want to express my gratitude. First of all I would like to thank my advisor Stefan Conrad who offered me a chance to stay at the Heinrich-Heine-Universität even though my previous studies weren't primarily at his department. I couldn't have hoped for a warmer welcome to the Data Base working group and I would also like to express my thanks for many discussions, motivation and all the help both regarding the scientific work as well as the teaching work. I would also like to thank Michael Schöttner for agreeing to act as a referee for this thesis.

Also, this thesis could have never succeeded without the great help of Sabine Freese, who was always helping with the many, many forms and regulations, and booking rooms on extremely short term. Also thank you for the many nice talks in the mornings. Guido Königstein, the best administrator in the world, was also crucial for the success of this thesis, since doing all the work on a sheet of paper would have been quite cumbersome. I'll also certainly remember the "film seminar" sessions on Friday afternoons.

Having a nice working environment was one of the foundations of this work. Thus I would like, in hopefully more or less chronological order, Tim Schlüter, the master of garlic and laughs, Thomas Scholz, carrier of the banana protection box, Ludmila Himmelspach, resident rumour generator and caring mother, Magdalena Rischka, the only person in the world to get high on Berliners, Robin Küppers, HHU's premier film critic, Pashutan Modaresi, who didn't sleep for a long time, Janine Golov, the organiser of teaching, Matthias Liebeck, the planner of everything in universe, Alexander Askinadze, distributor of high spirits, Julia Romberg, the sleepiest person I know, Mr database project Kirill Bogomasov, Gerhard Klassen, who definitely was alive every morning in Stuttgart, and Martha Tatusch, who opened the roller blind. Without the many fun hours and fruitful discussions we spent together, the time in university would not have been the same.

I would especially like to thank Daniel Braun, without whom this thesis would have an entirely different topic. Working with you has always been an absolute pleasure, as was and is spending time with you outside of work!

Also I want to thank all of my friends for being there all the time. Especially I would like to mention Anja and Lisa Rey, who happened to work nearly next door, and often walked home with me.

Last but not least I want to thank my family for all the support and patience, and especially my wife Julia for making me laugh at the right times, urging me on when I was lazy, helping out with thinking through many problems at any time, and being there in general. I couldn't have done it without you.

Abstract

The advent and rise of social networks such as Facebook and Instagram and image sharing platforms like Flickr and 500px lead to a huge number of photos publicly shared over the internet. While many of the people that share their photos add appropriate and detailed tags, short keywords that describe the image, many others do not. Those photos without tags are much more difficult to search for than other ones. Another problem arises if the knowledge of the photographer is limited. In these cases, a photo showing the Gran Paradiso could just be tagged with words like "mountain", "snow", "Italy", but for someone who searches for a specific photo of that mountain, those tags are not of much use. Similar examples can of course be found for many other landmarks like skyscrapers in New York City or for more precise knowledge like photos of flowers, when flowers of a certain type are searched for.

The automatic generation of tags is a way to solve these problems. For example of the identification of mountains, this thesis presents an approach to extracting an exact silhouette of the mountain the photo. This is needed to later identify the mountain by a comparison of the silhouettes. As the number of mountains on earth is very large, albeit hard to define, extracting a very precise silhouette is important. This extraction step is in essence a segmentation process, that is similar with sky detection problems. However, identifying the mountain is more difficult than the identification of the sky, as obstacles in front of the mountain can occur and thus conceal parts of it. Also, in many cases, a mountain is much less homogeneous in terms of texture and features than the sky.

In order to overcome those difficulties, this thesis emphasises on an outlier detection framework that is able to detect irregularities in silhouettes computed by a segmentation algorithm. We therefore introduce a baseline outlier detection algorithm and two improvements to it, as well as an approach to silhouette outlier detection by artificial neural networks. Additionally, by the second improvement to the base algorithm, we present a way to classify the detected outliers in order to be able to utilise different correction strategies for different kinds of outliers.

Our evaluation shows that both improvements mentioned before do indeed enhance the quality of the outlier detection, up to an F_1 score of 0.83 for the correct detection of outliers. Of the detected outliers, more than 92% are then correctly classified in regard to the kind of detected outlier.

Zusammenfassung

Durch das Aufkommen von sozialen Netzwerken wie Facebook und Instagram und Plattformen, auf denen Fotos im Internet geteilt werden können, wie Flickr und 500px, wird der Öffentlichkeit eine große Anzahl an Fotos zur Verfügung gestellt. Während diese Fotos häufig mit passenden, kurzen Stichwörtern, sogenannten Tags, versehen werden, gibt es auch eine große Anzahl von Bildern, bei denen dies nicht der Fall ist. Diese unmarkierten Fotos lassen sich besonders schwierig suchen, da die meisten internetbasierten Suchen anhand von textuellen Beschreibungen durchgeführt werden. Ein weiteres Problem, das im Zusammenhang mit dem Tagging auftritt, ist mangelndes Wissen des Fotografen über den Inhalt des Fotos. So könnte beispielsweise ein Foto, das den Gran Paradiso zeigt, nur mit den Schlagworten „Berg“, „Schnee“ und „Italien“ gekennzeichnet sein. Sucht nun ein Benutzer speziell nach Fotos, die den Gran Paradiso zeigen, würde dieses Foto nicht unter den Ergebnissen auftauchen. Ähnliche Beispiele lassen sich auch in vielen anderen Anwendungen, wie etwa bei Sehenswürdigkeiten oder Fotos von Blumen, finden.

Eine Lösung für die oben genannten Probleme besteht darin, die Schlagworte automatisch zu erzeugen. Für die automatische Identifikation von Bergen ist es aufgrund der hohen Anzahl von Bergen nötig, eine sehr präzise Silhouette aus Bergfotos zu extrahieren. Mithilfe einer solchen Silhouette ist es dann möglich, den Berg auf einem Foto durch vergleiche zu einer bekannten Referenzdatenbank zu identifizieren. Diese Arbeit stellt daher ein solches Verfahren vor. Grundsätzlich handelt es sich bei diesem Vorgehen um ein Segmentierungsverfahren, das große Ähnlichkeiten zu Verfahren zum Entdecken des Himmels auf Fotos hat. Die Extraktion der Form eines Berges aus einem Foto ist allerdings schwieriger als die Himmelserkennung, da Teile des Berges durch Hindernisse wie Menschen oder Gebäude verdeckt sein können. Zusätzlich sind die Textur und farbliche Eigenschaften von Bergen in vielen Fällen weniger homogen als beim Himmel.

Diese Arbeit konzentriert sich auf die Einführung eines Modells zur Erkennung von Ausreißern, mit dem es möglich ist, Unregelmäßigkeiten in Silhouetten, die von einem Segmentierungsalgorithmus berechnet werden, zu erkennen. Im Laufe der Arbeit werden ein grundlegender Referenzalgorithmus sowie zwei Verbesserungen vorgestellt. Hierbei ermöglicht die zweite Verbesserung eine Klassifizierung der erkannten Ausreißer, wodurch unterschiedliche Strategien für die Entfernung angewendet werden können. Weiterhin wird ein Ansatz für die Erkennung von Ausreißern mittels künstlicher neuronaler Netze vorgestellt.

In der abschließenden Auswertung wird gezeigt, dass die in der Arbeit eingeführten Verfahren ein F_1 Maß von 0.83 bei der Erkennung von Ausreißern erreichen. Außerdem werden mehr als 92% der erkannten Ausreißer der richtigen Klasse zugeordnet.

Contents

1

Introduction

Today, a huge bandwidth of social media platforms provides easy means for many people to release text or multimedia messages such as images or even videos to the public. Examples for such platforms include Facebook, Instagram, Flickr, Tumblr, DeviantArt, and 500px. This leads to a huge amount of information that is released, but in case of multimedia content is difficult to search for. While texts are relatively easy to index and find, for example by full text searches, for other media types this is much more difficult. This stems from the fact that on one hand the content of a sound file, an image, or a video is in a form that cannot be easily queried for by subsamples of the relevant file by users. On the other hand, descriptional keywords to such media, called labels or tags, are often inconsistent, imprecise, incomplete, or missing altogether since in most cases tags have to be assigned by users.

A possible solution for this problem is the usage of an automatic tag generation for submitted content. In such a case a machine learning algorithm could analyse the input and suggest suitable tags. This could either be a fully automatic, unsupervised process, where tags are added to the media or a supervised process where the suggested tags are presented to the user who then is able to accept, modify, or recline tags. In a general context for image annotation, many entries in challenges such as the ImageNet Large Scale Visual Recognition Challenge [RDS+15] have shown very impressive results in labelling images and videos with a set of known labels and even localising those in the images. Especially the advances made are staggering. In the classification task, in 2014 the best achieved mean average precision was 0.37212 for 1000 classes while in 2017 is has risen to a mean average precision of 0.732227.

However, even such an approach has limitations, for example when a specific instance of a class is searched instead of generic samples. Such searches can arise in many case, like when searching for specific persons, buildings, or natural sights such as mountains. In this cases, the number of possible classes is strongly increased, as every instance of a class needs to be its own class. Therefore, in many such cases, other strategies or specifically trained classifiers are necessary to successfully solve these problems.

For sights and other landmarks a variety of approaches to identifying them on photos exists. Gammeter et.al. [GBQG09] propose a method that generates annotations on holiday pictures. The approach not only generates labels, but attaches those and links to relevant Wikipedia articles to bounding boxes of matching objects in the photos. The paper emphasizes on the creation of a training database. The actual classification of the images is implemented via a bag of visual words of SIFT [Low04] or SURF [BTG06] features that are compared to the training images via tf-idf [Jon72].

In [RC15], first the SIFT features of the photos are computed. Those are then summarised as a bag of visual words that describes the image in a similar fashion to Gammeter's approach. To classify a photo, its bag of visual words is compared to that of a set of labelled training images. A HKM tree [NS06] is used as index structure to provide fast lookups on very large training datasets.

Another similar approach is presented in [PLSP10]. Here, instead of a visual bag of words representation as in the aforementioned works, the authors propose to instead use Fisher vectors [JH99]. The authors argue that the benefits of being able to use the dot product to compare the vectors outweigh the fact that Fisher vectors in contrast to bags of visual words are dense and thus require more memory. Further optimisations applied by the authors includes several binarisation techniques in order to compress the size of the vectors.

1.1 Mountain Identification

For the example of mountain identification, Baatz et al. released a data set in conjunction with [BSKP12], which contains 203 images of mountainous terrain in Switzerland as well as corresponding segmentation ground truth and GPS coordinates of the location the photo was taken in. To our knowledge, this is the first dataset that was publicly released and included a ground truth. The approach presented in this paper does not make use the GPS coordinates as the author's argue that not all photos have those. However, the segmentation part of the algorithm, where an input photo is separated in sky and foreground, is interactive. This means, that the segmentation algorithm computes a preliminary segmentation, which then is presented to the user. Then, the user is able to manually mark regions of the image as either foreground or background and to recompute the segmentation. This technique is similar to relevance feedback techniques. After an arbitrary number of correction steps the user is able to mark the quality of the segmentation as sufficient. The authors report that a manual improvement was needed in 49% of the images of the used data set. The silhouettes are then converted into so called contourlets, i.e. smoothed contours which are then coded in a symbol format. These coded contourlets are then matched with contourlets extracted from a digital elevation map (DEM) in order to find matches.

In contrast to the work presented in [BSKP12] the work presented here describes an approach to extract silhouette from mountain images in a fully automated fashion. Therefore a part that corrects possible errors in the silhouettes that are initially extracted is necessary and the main part of this thesis.

In [NMMI97] Naval et al. try to solve a problem similar to the mountain identification problem, namely the estimation of the point a photography was taken from. In order to extract the silhouette of the mountain of the image, the authors propose to first use a Canny edge detection [Can86] on the red channel of the image. Then,

every pixel that has been recognised as an edge by this step is classified by a multilayer perceptron. This is done by classifying the pixel below and above of the original pixel in their original RGB representations into the two classes of sky and ground. If the pixel below is classified as being a ground pixel and the pixel above is classified as sky the edge pixel has to belong to the silhouette. The actual silhouette or skyline is then extracted by following the edges detected by the Canny edge detection and guided by the classification results. Finally, so called feature points such as peaks are extracted from the silhouette and are used to compare them to artificial silhouettes from a DEM by a k nearest feature point search in order to localise the position from which the photography was taken from. Hereby, as the name suggests, the k nearest feature point search resembles k nearest neighbour classificators.

The approach has only been evaluated on a data set of 32 images of the same mountain which were taken from eleven different camera positions resulting in a mean error of 373 meters. However, there are no results that describe the actual capacity of identifying the mountain in a set of more than one mountains such that the results from the paper are not easily comparable. From today's perspective, however, a classification of single pixels in sky and ground classes based on the pixels colours alone without taking the neighbourhood into account seems to be rather simplistic, keeping in mind the advances made in classification and computing power during the last two decades.

Apart from the approaches described above there have been proposals of mountain identification tasks that require images to be GPS-tagged. Using GPS tags makes the identification of a certain mountain much easier, since the location from which the photography was taken from is known and thus the search space, i.e. the number of mountains one has to take into account as possible matches, is reduced drastically in size. In turn, this leads to less demands on the precision of the feature extraction from the images which simplifies every step in the task. In the following some solutions that make use of GPS tags are introduced.

In [BČES11], Baboud et al. present a mountain identification and annotation algorithm that is based on mapping a photography to a three dimensional model taken from a DEM. Additionally to GPS data the authors also require to know the field of view with which the photography was taken. This can be computed by correct EXIF data regarding focal length and sensor size. However, even if this information is available it is only correct if the images have not been cropped. If all these prerequisites are fulfilled an edge detection is used on the image. As in case of [NMMI97] many of the detected edges in the image are not part of any edges extracted from a DEM, which the authors call silhouette edges. Another possible case is that silhouette edges may not detected completely. The authors observe that edges that follow curves of the terrain usually have T-junctions only, i.e. one edge ends when meeting another one that continues, while crossings do not usually appear on edges following the terrain and thus can be neglected.

The remaining edges are then matched to the silhouette edges extracted from the DEM in the following fashion. Around every silhouette edge e_s an ε-neighbourhood is defined. If an edge e_p extracted from the photo enters the ε-neighbourhood of an edge e_s, the entry and exit points are computed, as well as the distance d of those two points along e_s. If d is greater than a predefined threshold d_{fit} the edges are defined as matching and if it is shorter, they are declared mismatches. By trying to match the image with all possible extracted images in a 360 degree circle around the location

of the photographer the best match is found. Once this happened, peaks and other points of interest can be transferred from information stored with the DEM.

While being a very precise method it is obvious that such an approach is not feasible if the location from which the photograph was taken is not known at least approximately, as the authors name a required accuracy of a few hundred meters. In this regard an adaption of the presented technique might be suitable for the last steps of a global mountain identification system when the possible locations have been reduced to very few.

In [FFT14] an approach with a similar idea to the one above is presented, as it does also propose a peak identification method that requires positional information about the photo as well as knowledge of the field of view the photography was taken with. In contrast to [BČES11], however, this work deals with coarse digital elevation maps specifically as the authors argue that most of the publicly available DEMs have spatial resolutions between 30 and 90 meters per pixel. Thus those DEMs are missing many details regarding shape and existence of features of mountains. As a consequence of this instead of a detailed edge map only the silhouette or skyline is compared to data from a DEM. The actual skyline extraction is executed by applying the algorithm presented in [LLLH05] which uses an edge-detection and threshold approach to detect edges that are candidates for the skyline. The actual silhouette detection is then carried out by a dynamic programming approach where the binary edge map created in the previous steps is interpreted as a weighted graph and the shortest path according to cost from the left to the right is searched. The actual matching process is then carried out by the same approach as in [BČES11], that has been described above.

In essence this work is a more robust version of the approach by Baboud et al. that from our point of view suffers from the same problems in regard of high requirements for the photo. In our experience many photographies without GPS or valid EXIF data exist. For those neither [BČES11] nor [FFT14] present, or claim to present, solutions.

Porzi et al. present an augmented reality application that is able to annotate mountain pictures on smartphones in [PBV+14]. The application uses a client-server architecture where the rendering an edge extraction of the digital elevation map is carried out on a server while the contour detection and actual annotation of the image is processed on the smartphone. While the edges in the images are detected by applying a Sobel filter, they are then classified by a Random Ferns classificator [OFL07] in order to find edges that belong to terrain contours. The edge extraction from the DEM is done by using a ray-casting algorithm in order to compute a two-dimensional representation from the view point. Finally, the actual mapping between the terrain contours extracted from the image and the DEM is performed on the smartphone with help of the smartphone's sensors in regard to positioning as well as by an optimisation in order to minimise the differences between the two edge maps.

As the mapping operation is very similar to the other mentioned approaches it shares the same limitations in respect to global scalability. The main feat of the presented work is the fact that the implementation in this case is fast enough to be carried out on a smartphone from 2014 in nearly real time and thus is a useful tool in circumstances where images are taken with a smartphone or, in this case specifically, to fulfil the information needs of the user in real time.

As already mentioned we share the idea presented in [BSKP12] in that we strive for a mountain identification system that poses no additional requirements on the image

or its meta data other than that it shows a mountain. Additionally, we believe that such a system is much more useful and useable if it works as a fully automatic system. The only step in the system presented by Baatz et al. which makes human intervention necessary is the extraction of a correct silhouette from an input image. Apart from the segmentation of the image and the actual extraction of the silhouette this step also consists of a check of the silhouette in order to note and mark errors. Given such a silhouette the actual identification process described in [BSKP12] does not need further intervention by humans.

The contribution of this thesis is the presentation of mechanisms to automatically detect errors in silhouettes extracted from mountain images. This thesis therefore presents an outlier detection algorithm that works on sequences such as polygonal chains or time series and can be adapted to different features. Also, an approach to outlier detection with artificial neural networks is presented. These approaches have been developed by the author in combination with an adaptive segmentation approach that was developed by Daniel Braun.

Used together, these processes can replace the relevance feedback part of the work of Baatz et al. and thus provide an automated solution to a global mountain identification task. Additionally, this work can be used to improve results of existing segmentation algorithms for example in the context of sky detection.

1.2 Structure of the Thesis

After giving an introduction and motivation to the topic presented in this thesis, we now outline the further structure of this work. Chapter 2 gives a brief introduction to outlier detection in general and outlier detection on sequences in particular. The chapter takes a look at the beginnings of outlier detection in statistics around the middle of the twentieth century in order to give some context to the general problem of outlier detection. Then, machine learning approaches to outlier detection are introduced and finally an overview over existing techniques that are able to detect different kinds of outliers in sequences is given.

Chapter 3 presents the Adaptive Mountain Silhouette (AdaMS) framework. This chapter focuses on the segmentation and silhouette extraction process as well as the refinement step, once errors in the silhouettes have been found. Thus it gives an overview of the work of Daniel Braun which is necessary to understand the general framework. The segmentation algorithm is based on a grid that divides the image into squares. Each of these squares has a local brightness factor that is initialised identically for all grid cells. This brightness factor commits to the threshold that is used to distinguish between ground and sky. During the adaptive refinement of the silhouette it can be changed for single cells in order to improve the segmentation results. Also, the sources of errors that can occur during the segmentation are presented and explanations for the causes of these errors are given.

Chapters 4 and 5, together with the evaluation in Chapter 6, form the contribution of the author of this thesis. The 4th chapter describes the basic outlier detection algorithm and two extensions. First, an outlier on a polygonal chain is formally defined as a subsequence with certain properties and the above average distance is introduced and compared to other histogram distances. After these basics are established we present the original outlier detection algorithm, which works on positional properties

of the silhouette, only. It consists of three steps. The first of those is a silhouette conversion step, where the absolute coordinates of the points in the silhouette are transformed into values relative to the predecessor of a given point. Then the anomaly score computation follows, which utilises a sliding windows approach. Each window is summarised into an histogram whose distance to a reference histogram forms the basis for the anomaly score. In the final step the outliers are found based on those anomaly scores. The first of the two extensions is the multi-reference enhancement. While the basic algorithm uses just one histogram to represent the normal data, this enhancement introduces multiple histograms and describes how these can be aggregated. The second enhancement explains how different feature sets and parameter combinations can be used in parallel to not only improve the outlier detection accuracy but also to implement an implicit classification of different kinds of outliers. The chapter is concluded by some ideas that outline a generalisation of the detection algorithm to generic applications on sequences.

We present an approach to outlier detection with the help of artificial neural networks in Chapter 5. As classifiers need training data for each class, but in outlier detection problems usually only samples for one class – the normal data – exist, we first describe how training data that represents outliers can be generated. The basic idea is to deviate from the correct silhouette of the training data in an image and label the data that emerges from this process as outliers. This is possible as we are sure that the silhouettes created in such a fashion do not overlap with the normal silhouettes from the images. Subsequently, we describe the two convolutional neural network architectures that were used to evaluate this approach and explain the mechanisms employed by the utilised layers. Finally, we explain how the output of the network can be used to identify outliers.

In Chapter 6 the methods from the previous chapters are evaluated. After presenting the training and test sets, we concentrate on the algorithms from Chapter 4 and discuss how changes to the single parameters affect the quality of the outlier detection. This is first carried out for the basic algorithm and after that, for both extensions. We also evaluate the results achieved by the artificial neural network introduces in Chapter 5, which shows promising results that are close to those achieved by the base algorithm.

Chapter 7 concludes the thesis by summarising the ideas and results. We also point out future work both in regard to the outlier detection algorithms as well as providing basic approaches for a silhouette based mountain identification tool.

2

Outlier Detection

The general topic of outlier detection emerged as a subfield of statistics in the middle of the last century [Gru50, Ans60, Fer61]. In this context, an outlier is usually regarded as an observation in a series of measurements that does not fit the general distribution of the data. This methods fit Hawkins [Haw80] famous, if much later, description of an outlier, stating:

> "An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism."

A popular implementation of this idea is to use the expectation-maximisation (EM) algorithm [DLR77, Wu83] on the measurements to estimate the most probable distribution or distributions in multivariate cases. The EM algorithm essentially assigns a probability of fitting the computed distribution or distributions to each data point in the data set. Given these probabilities, data points with very low fitting probability can be considered as outliers.

Another early approach to outlier detection, although with a slightly different interpretation of what an outlier is, is extreme value analysis [Güm58]. In this case, rather than assuming a certain distribution for a data set the most extreme points, i.e. maximum and minimum points or points that are outside of certain quantiles, are treated as outliers.

In the field of data mining anomaly detection emerged with algorithms such as DBSCAN [EKSX96] and OPTICS [ABKS99] in the context of clustering algorithms as well as the local outlier factor (LOF) [BKNS00] and the local correlation integral (LOCI) [PKGF03] in terms of local density. Hereby, DBSCAN and OPTICS are density based clustering algorithm that define points that do not belong to any cluster as noise or outliers. In this regard these approaches are somewhat limited in respect to anomaly detection since no differentiation between noise – which often fits known distributions – and true anomalies, as in points that are generated by a different mechanism altogether,

is made. It is important to notice, though, that both algorithms are primarily clustering algorithms and thus do not concentrate on anomaly detection in the first place.

In contrast, LOF and LOCI are dedicated spatial outlier detection algorithms. Both algorithms consider a point as outlier if its local density varies from the average local density of its neighbours strongly. While both approaches share the same basic idea, the definitions of local density vary severely. With the local outlier factor, a point p's neighbourhood in a dataset D is defined as the set of points $L_k(p) = \{q \in D | \text{dist}(p,q) \leq \text{dist}_k(p)\}$ where $\text{dist}_k(p)$ is the distance between p and its k-nearest neighbour. Then, the average reachability distance of p is given as

$$AR_k(p) = \frac{1}{|L_k(p)|} \sum_{q \in L_k(p)} \max\{\text{dist}(p,q), dist_k(q)\}.$$

Finally, the name-giving local outlier factor of a point p is defined as

$$LOF_k(p) = \frac{1}{|L_k(p)|} \sum_{q \in L_k(p)} \frac{AR_k(p)}{AR_k(q)}.$$

With this, points that behave normal have a local outlier factor of about one while outliers have a much higher local outlier factor. In general, LOF is a relative distance-based approach rather than a local density based approach as it does not heed a traditional definition of density.

As has already been mentioned the definition of density is the main difference between LOF and LOCI. In LOCI the local density around a data point p in a data set D is given by $M(p,\varepsilon) = |\{q \in D | \text{dist}(p,q) \leq \varepsilon\}|$ for a distance threshold $\varepsilon > 0$. Then the average density can be computed as

$$AM(p,\varepsilon,\delta) = \frac{1}{M(p,\delta)} \sum_{q \in D : \text{dist}(p,q) \leq \delta} M(q,\varepsilon)$$

for $\delta > \varepsilon$. The basic outlier score of LOCI is called multi-granularity deviation factor

$$MDEF(p,\varepsilon,\delta) = 1 - \frac{M(p,\varepsilon)}{AM(p,\varepsilon,\delta)},$$

which resembles the local outlier factor in the sense that it considers the local density of a given point relative to the average local density of its surroundings. Additionally, the authors suggest to treat a point as an outlier if $MDEF(p,\varepsilon,\delta) \geq 3\sigma(p,\varepsilon,\delta)$ where $\sigma(p,\varepsilon,\delta)$ is the standard deviation of the local densities $M(q,\varepsilon)$ for all points q in p's δ-neighbourhood normalised by $AM(p,\varepsilon,\delta)$. This proposal shows that $MDEF$ should be normally distributed.

All approaches mentioned until now share the assumption that an outlier is a single point in a data set where each dimension is expected to have the same importance in regard to its behaviour. However, there exist many cases in data mining where either outliers do not consist of single points or where one or more dimensions are deemed more important in respect to behaviour than others. In respect to sequences such as time series both those problems may arise.

In general, a sequence establishes some kind of order over the points in the sequence in the sense that a point usually has a predecessor if it is not the first point in the

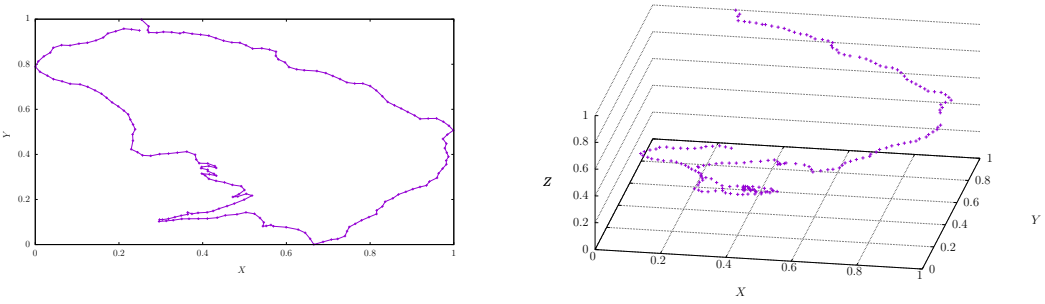


Figure 2.1: Example for a trace in form of a sequence and as spatial data.

sequence as well as a successor if it is not the last point in the sequence. Usually it is assumed that points that are close to each other in regard to that order behave similarly while it is not unusual for points that are far from each other to be vastly different. Consider for example a GPS trace of a vehicle where every second the position is noted. In this case, as illustrated by the example in Figure 2.1 the vehicle is able to cover huge distances over time but every point is expected to be relatively close to its predecessor and successor. If one looks at such a data set as consisting of three dimensional points only, those jumps are much more difficult to detect as illustrated by the section with low z values in the spatial plot on the right where the order of points cannot be seen clearly.

Common cases of outliers in sequences, apart from single points, are unusual subsequences, that are sometimes called discords, and change points. Discords or anomalous subsequences fit the general outlier definition from above in that they are subsequences that do not fit the general behaviour of the sequence they are a part of. This is a generalisation of the outlier definition given above in the sense that an outlier consisting of a single point can be interpreted as a subsequence of length one. One popular approach to outlier detection on time series is HOT SAX [KLF04, KLF05], which is an algorithm that is able to find the most unusual discord of a fixed length in a given time series. The approach is based on the symbolic aggregate approximation (SAX) technique [LKLC03], that first discretises the time dimensions in fixed length parts. Each such part gets the average value as its discreet value. Those are then divided into equal depth bins and represented by a symbol. By this approach, comparisons between subsequences of a fixed length can be carried out relatively fast, due to the possibility to utilise indexing functions on the symbolic representations. In context of HOT SAX this is used in order to compute lower and upper bounds on the distances. Thus many distances do not have to be computed in detail. Other works expand on the HOT SAX approach. In [PLD10] an extension that ensures equal probability of the symbols for non-normal distributions by utilising a clustering of the values is introduced. With iSAX [SK08] SAX was expanded on in a way to allow for distance computations of different lengths, which was then applied to discord detection in single time series in [BA11, KA12].

In contrast to this, a change point is a point in sequence that is not necessarily an outlier in the sense of the definition above, but a point that indicates a change of certain properties of the subsequence before the change point and after the change point. In

this context a change point can be seen as an outlier in the sense that it is a point that does not fit into the behavioural structure of one of the resulting subsequences. Also, it is possible to find outlying subsequences by looking at sequences that are framed by two change points. As with any outlier detection theme, there are statistical approaches [Pic85], that can be used in quality control where the statistical properties of the base distribution or distributions are known or can be estimated, as well as machine learning approaches such as [KYM07] in which a subspace method is utilised.

Another related approach to outlier detection is the so-called novelty detection. The basic idea is very similar to outlier detection, as are used techniques, though the interpretation is different. In novelty detection or novel class detection in streams, parts that are unusual up to a point in a stream, are treated as new classes in a classification problem. The idea behind this is that they might occur again later on. In general this behaviour of newly arising classes and vanishing classes is called concept-drift. One example for such an approach is [MAKK$^+$11]. Here, the authors argue that in such a case new, formerly unknown classes cannot be classified correctly by a method that assumes a fixed number of classes. This is because it is not possible to predict how many classes will arise during a stream. The paper also presents a possible solution to the recurring class problem. A recurring class is a class that has not occurred for a long time, such that there is a high probability that it is classified as a new class. In order to solve those problems a two stepped approach is presented, using two classification ensembles. The primary ensemble M, contains classifiers that are able to classify frequent classes. If the classification result for all classes has a low confidence, the query subsequence q is called an outlier. If this is the case it is again classified by a second ensemble M^A which contains the less frequent classes. If q is classified as an outlier by M^A, again, a new class for q may be added if during a giving time frame similar examples arise. In contrast to classical outlier detection methods, the approach presented in [MAKK$^+$11] needs human annotators in order to define the classes on the training data as well as on the stream data, while most outlier detection algorithms only need a model for the normal data which is prepared before the actual outlier detection. Therefore such an approach is not suitable to automated outlier detection on sequences.

Other related approaches such as [MP03a, MP03b, GKVL06] use one-class support vector machines to facilitate the detection process for novel classes. These approaches can be used for outlier detection in the same way as [MAKK$^+$11]. Alternately, a static model can be used to train the support vector machines such that these approaches can be used for automated outlier detection.

The last kind of anomaly detection on sequences to be mentioned here takes a whole set of sequences and searches for unusual series among this set. In such a case, a single sequence is often treated as a high dimensional data point such that outlier detection techniques that are usually deployed for this kind of data can be utilised. Examples for this contain evolutionary algorithms [AY01] to determine lower dimensional projections of the original space that are locally sparse, or angle-based techniques [KSZ08] where the properties of a point are described by the angle of the connection between different points rather than a distance to these points in order to handle the curse of dimensionality. One approach that explicitly targets sequences in the form of trajectories is presented by Lee et.al. in [LHL08] and is able to also find outlying sub-trajectories in a database of trajectories. In order to accomplish this,

all trajectories are split in sub-trajectories. In those, outliers are then searched via a combination of the ideas of the distance-based outliers from DBSCAN [EKSX96] and the LOF [BKNS00]. This finds outlying sub-trajectories in all trajectories. Then, a trajectory is defined as an outlier if it contains sub-trajectories that are outliers. The limitations of this approach resemble those of HOT SAX [KLF04] in that the length of the sub-trajectories and thus of possible outliers has to be known beforehand. Also, computation time is in $\mathcal{O}(n^2)$ for n the number of sub-trajectories, though in this time outliers in all trajectories are detected. There is, however, no shortcut to just detect outliers in one trajectory, only.

Papadimitriou, Sun, and Faloutsos present an interesting approach in [PSF05] to detect patterns in parallel data streams. Here, a data point is defined as a n-dimensional point containing all values of the different streams at the same point in time. These points are then reduced in dimensionality with help of a principal component analysis to chose a lower dimensional base with only a small deviation from the original data. Trends in the stream are then monitored by a hidden variable model in order to adapt the model to new data from the streams. This model can be used on one hand to complement missing data in data points, as well as to discover formerly unknown patterns since they show huge deviations from the model. However, as a downside, this approach only works well if there are correlations between the original streams. Otherwise neither the PCA nor the hidden variable model are able to reduce the dimensionality in a meaningful way and thus the approach degrades into independent predictions of the single streams.

3

AdaMS and Extensions

AdaMS is a framework for adaptive mountain silhouette extraction. The goal of the framework is to find a silhouette of a mountain in a photography that is as exact as possible in order to allow a precise mapping to a silhouette of a known mountain extracted from either a labelled image or a digital elevation map. This chapter describes the AdaMS framework that has been introduced in [BS15, BSC16]. The silhouette extraction methodology consists of five steps, namely 1) Grid and Parameter Initialisation, 2) Image Segmentation, 3) Silhouette Extraction, 4) Outlier Detection, and 5) Silhouette Refinement. Hereby, steps two to five can be repeated multiple times in order to improve the accuracy of the silhouette in multiple steps. Figure 3.1 gives an overview over the general structure of the AdaMS framework. This chapter mainly concentrates on the steps one to three, that have been developed by Daniel Braun. On their own, these three steps implement a complete segmentation and silhouette extraction algorithm, while the outlier detection and subsequent silhouette refinement steps implement the adaptive improvement of the silhouette.

The remainder of this chapter is structured as follows: Section 3.1 describes the preliminaries of the the extraction process. Section 3.2 describes the bisection of the

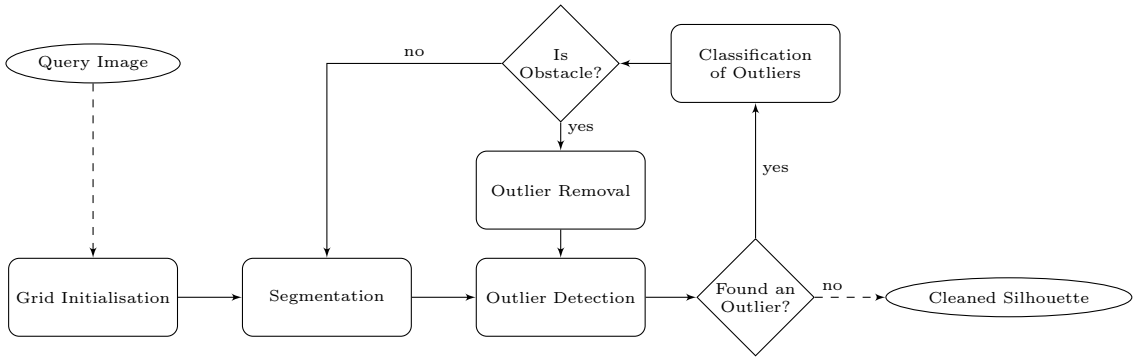


Figure 3.1: Flow diagram of the AdaMS framework

image in the two classes sky and ground, and Section 3.3 show how to get the silhouette from this. Section 3.4 gives a brief overview over possible errors in the silhouette and Section 3.5 shows how these can be adaptively removed.

3.1 Grid and Parameter Initialisation

Before the actual segmentation and silhouette extraction steps can begin, some preparations have to be made. The main point of these is the computation of a grid structure that partitions the image into several squares. This grid is mainly used later on, during the local adaptive refinement, where the segmentation process is limited to a number of affected grid cells. Additionally to this, some other global parameters are computed as well.

In the following, when talking about image we mean a digital representation of an image, as defined as follow:

Definition 1 (Image)**.** An *image* is a matrix

$$I = \begin{pmatrix} p_{1,1} & \cdots & p_{w,1} \\ \vdots & \ddots & \vdots \\ p_{1,h} & \cdots & p_{w,h} \end{pmatrix} \in \mathcal{C}^{w \times h}$$

where $\mathcal{C} \subseteq \mathbb{R}^c$ is called the channel or the channels of I if $c = 1$ or $c > 1$, respectively.

An entry $p_{i,j} \in \mathcal{C}$ of an image is called the *pixel* at position (i, j). For sake of an easier notation for every image I we define $I(i, j) = p_{i,j}$.

Usually, digital photos are 8 bit RGB coloured images. That means they have three channels with an integer range of $[0, 255]$, each. In this notation, the first channel gives the red intensity, the second the green intensity, and the third the intensity of blue for a given pixel. Since the RGB colour space is an additive colour space, higher value represent brighter colours and especially $(0, 0, 0)$ is black and $(255, 255, 255)$ is white. In some applications, the channels are normalised to a range $[0, 1] \in \mathbb{R}$ in order to make algorithms work more easily on images of different channel depth. Popular examples for this behaviour are Matlab[1] or scikit image [VSNI+14].

However, many segmentation algorithms work on brightness values, only. Usually, the brightness of a RGB pixel $p = (r_p, g_p, b_p)$ is defined as $b(p) = \frac{r_p + g_p + b_p}{3}$ or the rounded value of $b(p)$ such that $b(p) \in [0, 255]$.

Consequently, for a given input image I we compute the brightness image B as

$$B = \begin{pmatrix} b(I(1,1)) & \cdots & b(I(w,1)) \\ \vdots & \ddots & \vdots \\ b(I(1,h)) & \cdots & b(I(w,h)) \end{pmatrix}$$

where by Definition 1, $B(i, j) = b(I(i, j))$. Also, we compute σ_B, the standard deviation of all brightness values in B.

The next step during this initialisation phase is the actual grid initialisation. The idea of the grid is to divide an image – in this case the brightness image B – into

[1] `http://www.mathworks.com/products/matlab.html`

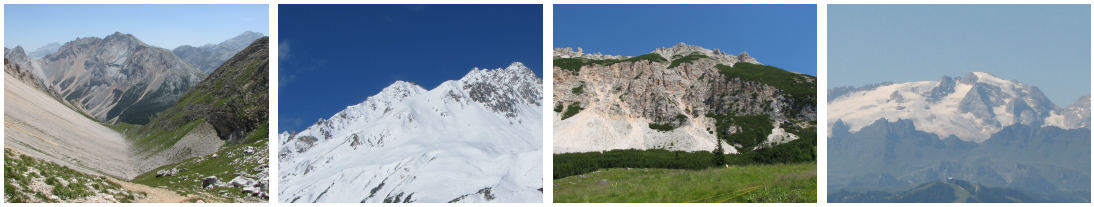


Figure 3.2: Examples for brightness differences between different kinds of sky and ground.

disjunct squares of equal size. As a first step, the centre points of the grid cells are computed. After that, every pixel of the image is said to belong to the cell, whose centre point is the closest based on its coordinates in the image:

Definition 2 (Grid Centre Points)**.** Let $I \in \mathcal{C}^{w \times h}$ be an image with width w and height h. Let $d_G \geq 2$ be an even integer. Then the grid with step size d_G has

$$G_I = \left\{ \left(\frac{d_G}{2} + i \cdot d_G, \frac{d_G}{2} + j \cdot d_G \right) \middle| \, 0 \leq i \leq \frac{w}{d_G} - \frac{1}{2} \wedge 0 \leq j \leq \frac{h}{d_G} - \frac{1}{2} \right\}$$

as cell centre points.

A pixel $I(k, l)$ belongs to the grid cell of the closest cell centre point, i.e. the $g \in G_I$ where $\|(k, l) - g\|_2$ is minimal.

Finally, for every grid cell $g \in G_B$ we store a local brightness factor γ_g that we initially set to the fixed value $\gamma = 0.1 \in \mathbb{R}_{\geq 0}$.

3.2 Image Segmentation

After all parameters have been initialised, we can now commence with the first segmentation. The goal of this is the segmentation of the image into two uniform segments, namely sky and foreground. In order to reach that goal, we utilise a region growing algorithm in order to identify the sky. In the beginning, we give all pixels the class "ground" except for a few seed pixels that we are fairly sure belong to the "sky". Then, for every pixel that is a neighbour of a "sky" pixel, we check whether that pixel belongs to the sky.

More specifically, given a brightness image $B \in [0, 255]^{w \times h}$ we initially set the class $C(p)$ of the pixels of the top row of the image as "sky", i.e. the pixels $B(1, 1), \ldots, B(w, 1)$. This is because in most cases, the sky is at the top such that we have the highest probability for "sky" pixels, here. All the other pixels in the image get the class "ground" and we initialise a look-up queue $Q = \{B(1, 2), \ldots, B(w, 2)\}$ as the second row of the brightness image.

Now, while Q is not empty, we take one of the pixels $p \in Q$ and remove that from Q. We set the class $C(p) =$ "sky" if

$$|p - \mu(p, r)| < \gamma \cdot \sigma_B,$$

with $\mu(p, r)$ the mean brightness of radius r around p and γ and σ_B as in Section 3.1. Also, if the above inequality holds, we add every pixel of the 8 neighbourhood of p to

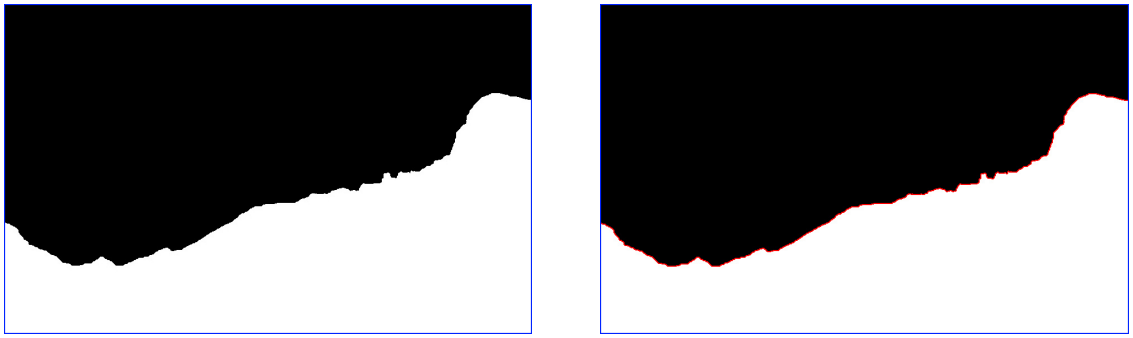


Figure 3.3: Segmentation on the left and added silhouette in red on the right.

Q if is not already of the class "sky". The idea behind this approach is that inside the sky, brightness does only change gradually. In contrast to this, when the ground is hit, that is usually either much brighter, if the sky has a dark blue colour, or much darker, in case the sky's colour is bright. Figure 3.2 gives some examples of this. In the first image we see an example of a light sky with darker foreground. The second and third image show a darker sky with a lighter foreground, although there is a part in the third image where the foreground is darker than the sky at the silhouette where the bushes reach the top of the mountain. Finally, the fourth images shows and example where both cases occur. The sky is lighter than the rocky parts of the mountain while the glacier is brighter than the sky.

If Q is empty, there are no further pixels that can be added to the sky segment, thus the algorithm terminates. At this point the sky segment is connected, due to the growing algorithm, however this is not necessarily the case for the ground segment. Therefore, in order to enhance comparability with other segmentation algorithm and to make the silhouette extraction easier, patches that are classified as ground and that have no connection to either the right or the left border of the image, are removed by setting the class of those pixels as "sky".

After this process, we have a class for every pixel in the brightness image. This can be interpreted as a binary image

$$S \in \{0,1\}^{w \times h}$$

by the mapping

$$S(i,j) = \begin{cases} 1 & \text{if } C(B(i,j)) = \text{ "sky"} \\ 0 & \text{else} \end{cases}$$

for all $i \in [1,w]$ and $j \in [1,h]$.

3.3 Silhouette Extraction

Once we have a segmentation image, such as the result from the process described in the previous section, we have to extract a silhouette from it. Hereby, figuratively speaking, we mean the transition between the mountain in the foreground, and the sky in the background. Figure 3.3 shows an illustration of this. On the left hand, a segmentation image is shown, while on the right hand, the same image with added

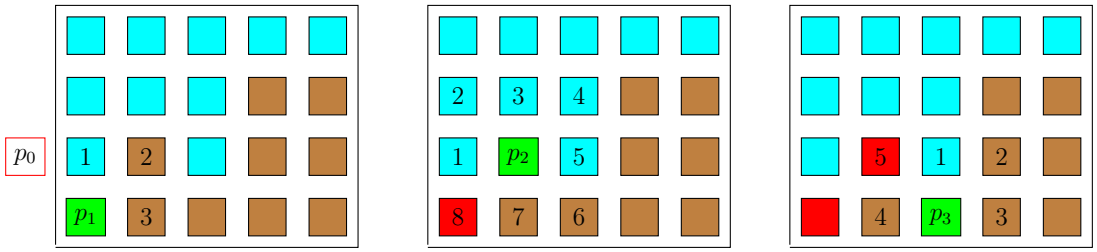


Figure 3.4: An example for the silhouette extraction process.

silhouette is shown. In order to be able to distinguish between foreground and the rest of the page, both images are framed in blue. This frame is not part of the actual images. As can be seen from this figure, the silhouette is the set of ground pixels that are adjacent to sky pixels. For easier comparison, our silhouettes run from the left to the right of an image.

In order to get a continuous silhouette, we first set the bottom line of pixels in the segmentation image $S \in \{0,1\}^{w \times h}$ to 0, i.e. define them as ground pixels. Thus, for all $i \in [1, w]$, set $S(i, h) = 0$. In most cases, this is already the case since most photographs of mountains do not only show a peak but rather surrounding landscape or a mountain whose base is wider than the image. Now we are able to commence with the actual silhouette extraction. In our case, a silhouette is represented as a polygonal chain, that is, an ordered sequence of pixel coordinates from an image. In the following, let P denote that polygonal chain. The goal of the silhouette extraction process is to assemble

$$P = (p_i = (x_i, y_i))_{i=1}^{n},$$

$n \in \mathbb{N}$, such that all $p_i \in [1, w] \times [1, h]$, $x_1 = 1$, and $x_n = w$. By this we get a silhouette that spans the whole width of the input image from left to right. Since we now have a segmentation image where the uppermost row is filled completely with ones, and the lowermost row is filled completely with zeros, and we know that there is one sky segment only, we can guarantee that such a polygonal chain always exists.

As first silhouette point we set p_1 to the pixel in the left column of the pixel, that is closest to the top and marked as ground, i.e. we set $p_1 = (x_1, 1)$ with

$$x_1 = \min\{k \in [1, h] \mid S(1, k) = 0\}.$$

For the algorithm to extract the next silhouette point, we need two silhouette points. For this we set $p_0 = (0, y_1 - 1)$, however we do *not* add p_0 to the actual silhouette but treat it as a helper point.

Now, given the silhouette points p_{i-1} and p_i, we look at the 8 neighbourhood of p_i and enumerate those pixels in clockwise fashion starting with the point that follows p_{i-1} in clockwise direction. By this, we set p_{i+1} as the first of those pixels we encounter, that has the class "ground". Thus p_{i+1} is added to the silhouette and we search for the next silhouette pixel. This process finishes, once we have reached the right border of the image, i.e. if we found $p_n = (x_n, y_n)$ with $x_n = w$.

Figure 3.4 gives an example of this process. Here, pixels with class "sky" are marked by a blue filling and pixels with class "ground" are pictured in brown. Pixels that are

part of the silhouette are shown in red and the last silhouette pixel is shown in green. In Figure 3.4a the first point of the silhouette p_1 is found as the highest – and in this case only – ground pixel in the first row. The helper pixel p_0 is included as a pixel outside the image and the numbers in the other pixels depict the order in that these are examined. The pixel marked by "2" is obviously the first ground pixel we encounter, so this becomes p_2 in Figure 3.4b. Here, we see, that p_1 becomes the last pixel in p_2's neighbourhood that will be examined. At this point, it is important to note, that one pixel can be included in a silhouette at different positions if it is the only connection to the remainder of the ground segment. However, in this case we find the pixel "6" as the next ground pixel, so this becomes p_3 and in 3.4c we look at its neighbourhood. Here, it is noteworthy, that apart from p_0 in the very first step of the silhouette extraction, only pixels inside the image are considered.

Once by the approach described above, we have a complete silhouette, we are able to compress it by removing pixels that hold no additional information. These are pixels that lie on the line segment between their predecessor and successor. Thus, if

$$p_i \in \{p_{i-1} + t \cdot p_{i+1} | t \in [0, 1]\}$$

holds, p_i can be removed from P without loss of information.

3.4 Possible Errors in Silhouettes

The previous section describes a technique to extract silhouettes from segmentation images. In our case, segmentation images are binary images denoting portions of the image that are recognised as sky with a pixel value of one. Pixels that are recognised as ground are denoted by a value of zero. Given a segmentation image S, the algorithm described in Section 3.3 finds the border between these sections, and therefore the silhouette of the segmentation image, perfectly. However, for the silhouettes being free or errors, the segmentation S has to be fault free, too. Unfortunately, this is not the case for most images. In this section, we explain what errors can occur and name possible causes for those.

On an abstract level, there are two error sources that we have to consider: First, we want to find the silhouette of a mountain. However, the previous parts of this chapter describe a way to extract the silhouette of the complete foreground of an image. Since images can obviously contain objects, that are neither mountains nor sky, this is not always the same. Second, even the segmentation into sky and ground, or foreground, is not always free of faults.

In the first case, objects that are not part of the mountain are a problem if they actually hide a part of the mountain's silhouette. This may be the case, when there are trees on parts of the mountain, or buildings, persons, animals or other objects such as clouds between the mountain and the camera. In the following, we call such objects *obstacles*. Some of these obstacles can be removed in a separate preprocessing step, such as described in [Kla17]. This approach works especially well for obstacles that contain a high degree of parallel lines, such as many human made structures like houses or cable car towers and the cables themselves. Once such structures are identified they can be retouched by techniques such as inpainting [BSCB00, CS01].

As mentioned above, the second reason for errors in the silhouettes are *segmentation errors*. Since the segmentation algorithm described in Section 3.2 assumes, that there

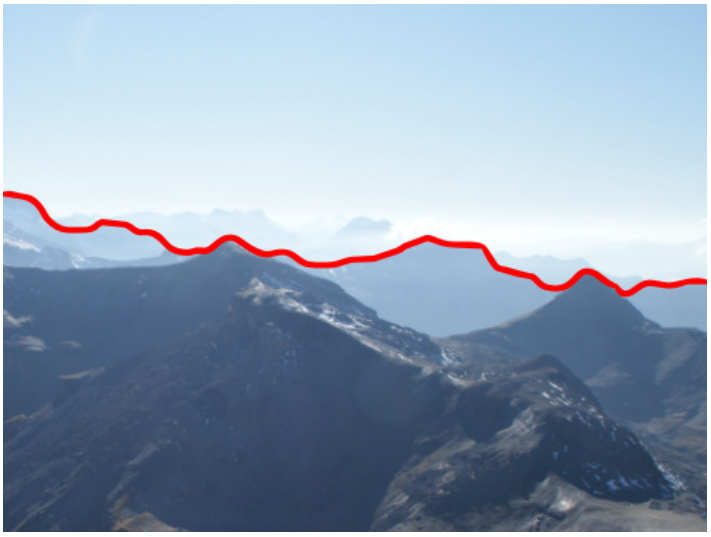


Figure 3.5: Low contrast between sky and foreground leads to foreground classified as sky.

are no huge contrasts in the sky, there are two cases where errors can happen. Parts of the sky, such as clouds, are not classified as sky because there is a high contrast between them and the surrounding sky. Due to the refinement step that is the last part of the segmentation algorithm and the silhouette extraction process, that is based on finding the upper-most ground pixels that are connected to the borders of the image, most of these errors are not problematic. In order to affect the silhouette, they have to be in the proximity of the silhouette, such as clouds that are directly above the mountain.

Another class of errors occurs, if the contrast between sky and ground is low in certain parts. In such a case, a portion of the actual foreground gets classified as sky. Figure 3.5 shows an example for this behaviour. Low contrast between sky and foreground can have several reasons, such as a low image quality in terms of noise, blur, or reduced contrast due to wrong exposure. Also, low contrast can be caused by natural phenomenons such as fog, light coloured rocks, snow, or clouds.

In order to detect such errors in the silhouette, AdaMS utilises an outlier detection step. This is described in depth in Chapter 4. At this point, we only need to know, that we pass the silhouette P to the outlier detection step and in return get a set of outliers in the form of subsequences of P and corresponding classes. The next section describes how we correct these errors.

3.5 Silhouette Refinement

As discussed in Section 3.4, there are essentially two kinds of outliers: Segmentation errors and obstacles. In order to fix these different kinds of errors, we utilise different strategies. In respect to the mountain silhouette extraction, all obstacles have in common that we do not know what the actual mountain silhouette behind them looks like. Obstacles are non-transparent, so there is no way to get the actual mountain silhouette by means of other segmentation parameters. In contrast to this, we can fix segmentation errors by a resegmentation of either the whole image or parts of the

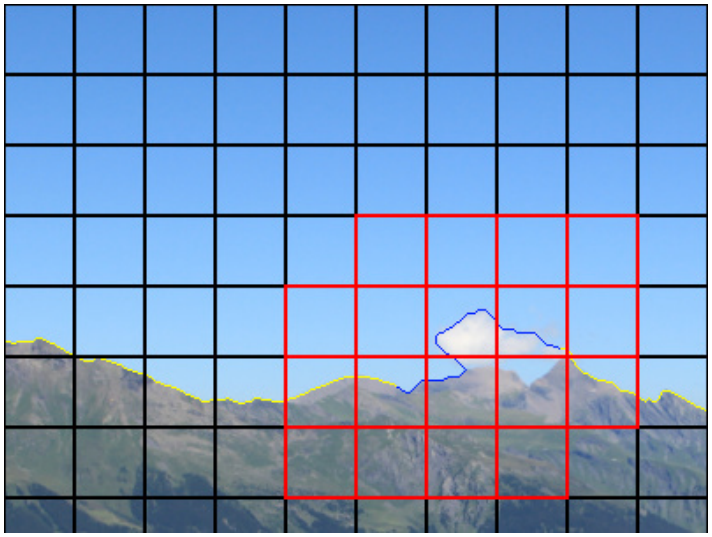


Figure 3.6: Grid cells in the proximity of an outlier.

image.

In the following, let us assume that the outlier detection step returned a set $\mathcal{O}$ of outliers. An outlier $O = (v_s, \ldots, v_e) \in \mathcal{O}$ is a continuous subsequence of the silhouette P that gets extracted in the silhouette extraction step as described in Section 3.3. Also, let $C(O)$ denote the class of the outlier O. Depending on the outlier detection algorithm, the set of possible classes can either be {segmentation error, obstacle} or {segmentation high, segmentation low, obstacle}. In the second case, both the classes "segmentation high" and "segmentation low" are segmentation errors. However, in this case, for an obstacle of the class "segmentation high" the outlier detection sees that outlier as being segmented too high. This means, that a part of the sky has been segmented as ground. In case of an obstacle that has been classified as "segmentation low" we assume that a part of the ground has been classified as sky. When using an outlier detection algorithm that returns three classes of outliers, and we say an outlier O is a segmentation error, we mean by this, that its class

$$C(O) \in \{\text{segmentation high, segmentation low}\}.$$

The first thing we do during the silhouette refinement step is to calculate the ratio of the length of segmentation errors to the total length of the silhouette

$$sr = \frac{1}{|P|} \sum_{O \in \mathcal{O}_{seg}} |O|,$$

with $\mathcal{O}_{seg} \subseteq \mathcal{O}$ the set of segmentation errors in $\mathcal{O}$.

If $sr > \tau$, where $\tau \in [0, 1]$ is called the resegmentation threshold, a large portion of the silhouette has been classified as segmentation errors. Thus it seems, that the global brightness factor γ was not suited to this particular image. In this case we recompute the segmentation for a number of precomputed different values of γ and then keep the segmentation for that the segmentation error ratio sr is the lowest.

If $sr \leq \tau$ we assume that the silhouette in general has been extracted correctly. In this case, we try to fix the segmentation errors locally. In order to do this, we execute

a local resegmentation in the proximity of the outlier. In detail, for a segmentation error O we search the set of grid cells G_O, that either contain points of the outlier or are neighbours of such cells. Figure 3.6 illustrates this. There, the silhouette is shown in yellow, the detected outlier is blue. The black grid shows the unaffected cells and the cells that are coloured in red are the cells belonging to G_O. The grid cells in G_O are the part of the image, that we use for the resegmentation. Our assumption is, that the global brightness factor is not ideal. It is either too high if ground is segmented as sky, or too low if sky is segmented as ground.

In order to change this, we adjust the brightness factor γ_g for each of the affected grid cells $g \in G_O$ by setting the new brightness factor

$$\gamma_g^* = \gamma_g + \alpha \cdot \theta,$$

where $\alpha \in \{-1, 1\}$ is the direction of the change and $\theta > 0$ is the step width. In case of an outlier detection method with three classes, we know in which direction we have to move the brightness factor. Here, we set

$$\alpha = \begin{cases} 1 & \text{if } C(O) = \text{“segmentation high”} \\ -1 & \text{else.} \end{cases}$$

However, if we only get two classes we do not know, in which direction we have to move the brightness factor and such have to try both variants and then select the one with a lower sr.

In all cases, we execute the local refinement of the segmentation as a local region growing of the sky. Therefore, as with the global segmentation described in Section 3.2, we add every pixel $p \in G_O$ that has a neighbour that is classified as sky, to the queue Q. Note here, that those neighbouring pixels can be outside of G_O, however we only add pixels that are included in G_O to the queue at any point during the local refinement. Now, as long as Q is not empty, if

$$|p - \mu(p, r)| < \gamma_g^* \cdot \sigma_B$$

we classify a pixel p as sky and add all non-sky pixels in the 8 neighbourhood of p to Q.

After each change – i.e. the removal of segmentation outliers – we extract a new silhouette P from the changed segmentation image and repeat the refinement process until no more segmentation errors occur or a maximum number of refinement iterations has been reached after which segmentation errors are ignored.

At this point we are left with obstacles only. As we have argued above, obstacles hide the real silhouette of the mountain such that refining the silhouette with resegmentations is not a possibility. Instead, we replace parts of the outliers, that show high deviations from the connecting line segment of start and end of the outlier with a straight segment. The idea behind this approach is to get rid of distinctive features that are not part of the real silhouette in order to not distort the silhouette too much.

In detail, let $O = (v_s, \ldots, v_e)$ be an obstacle and let $seg = \overline{v_s v_e}$ be the segment between v_s and v_e. Also, let $d(v, seg)$ be the minimal distance between a pixel v and the segment seg. Then we search the first pixel $v_f \in O$ for which

$$d(v_f, seg) > \delta \wedge \forall s \leq i < f : d(v_i, seg) \leq \delta$$

hold, for a deviation threshold $\delta > 0$. By this, v_f is the first pixel that deviates farther than δ from seg. In the same manner, we search the last such pixel v_l with

$$d(v_l, seg) > \delta \wedge \forall l < i \leq e : d(v_i, seg) \leq \delta.$$

If we do not find such v_f and v_l the obstacle does not deviate farther from seg than δ so we do not replace parts of it. This is because in this case, O does not introduce any outstanding features into the silhouette p. Thus, a replacement does not improve the silhouette. If we find v_f and v_l, we replace the portion between these pixels by the straight segment $\overline{v_f v_l}$, thus getting rid of distinctive features that are not part of the actual mountain silhouette.

4

Outlier Detection in AdaMS

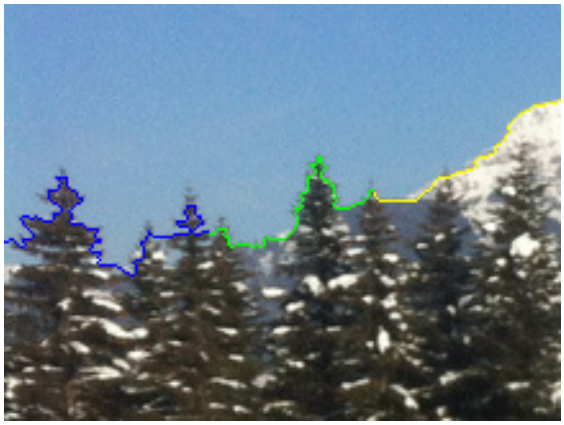
As described in Section 3.3, the AdaMS segmentation process yields a polygonal chain consisting of points in the image that describes the edge of the mountain silhouette. This chapter introduces various variations of the outlier detection approach. First, a definition of outliers on polygonal chains is given and we introduce the Above Average Distance for histograms. Then the basic outlier detection algorithm is presented, followed by an enhancement with multiple reference silhouettes and a differentiation for different kinds of outliers. We then point out other variants of the algorithms and give a generalisation of the algorithm.

Parts of this chapter have already been published: In [BS15] a first draft of the whole AdaMS algorithm including the basic idea of the outlier detection algorithm has been presented. In [BSC16] the first finished version of AdaMS was presented. This includes the basic definitions of outliers as in Section 4.1 and the basic algorithm as presented in Section 4.3, albeit with the polar coordinate representation for the converted silhouettes, only. In [SBC16] we introduced the multi-reference enhancement as in Section 4.4 and the proof of the above average distance being a pseudometric that is shown in Section 4.2. In [SKBC17], the outlier-type distinction was introduced. This is described in Section 4.4.

4.1 Outlier Definition on Polygonal Chains

This section gives a general definition of an outlier in a polygonal chain.

Let $P = (p_1, \ldots, p_n)$ be a polygonal chain, with points $p_1, \ldots, p_n \in \mathcal{P}$. In case of the outlier detection in mountain silhouettes the points p_i correspond to pixels of an input image I as described in Chapter 3. Now, let $SP = (p_s, \ldots, p_e)$, $1 \leq s < e \leq n$ be a continuous part of P, that itself is a polygonal chain. Generally speaking, we call SP an outlier in P if SP's properties are significantly different to the properties of P. This reflects the discussion from Section 3.4, that showed that errors in a silhouette consist not only of one but several points.



(a) Weak anomalies (green) next to strong anomalies (blue) form an outlier.

(b) Weak anomalies (green) on their own in a normal part of a mountain.

Figure 4.1: Different contexts for weak anomalies.

In order to measure the properties of a polygonal chain, let us assume a function

$$\text{anom} : \mathcal{P} \to \mathbb{R}_{\geq 0}$$

to be the anomaly score of the points in P. Our only assumption on anom is, that if $\text{anom}(p_i) > \text{anom}(p_j)$, then p_i is more unusual than p_j. We are now able to define two types of anomalies. Note here, that an anomaly is a single point $p \in P$ while an outlier is a continuous subsequence of P.

Definition 3 (Strong Anomaly)**.** Let $P = (p_1, \ldots, p_n)$ be a polygonal chain, anom be an anomaly score and a threshold $\tau_s \in \mathbb{R}_{\geq 0}$. We call $p \in P$ a *strong anomaly* if

$$\text{anom}(p) \geq \tau_s.$$

The problem here is choosing a sensible value for the threshold τ_s such that a strong anomaly indeed has unusual properties. Normal values for such a threshold in outlier detection applications as a rule of thumb suggest to set such a threshold to $\mu + 3\sigma$ for μ the mean of the anomaly score distribution and σ the standard deviation of that distribution. However, in our experiments we found out that in case of mountain silhouettes much smaller values in the range of $\mu + 1.5\sigma$ yielded much better results.

The following definition of a weak anomaly is more or less the same as for a strong anomaly, only with a lower anomaly score as a threshold.

Definition 4 (Weak Anomaly)**.** Let P, anom, and τ_s as in Definition 3. Let $\tau_w \leq \tau_s \in \mathbb{R}_{\geq 0}$. A point $p \in P$ is called a *weak anomaly* if

$$\text{anom}(p) \geq \tau_w.$$

As a consequence of this, every strong anomaly also is a weak anomaly. The reasoning behind the disambiguation between strong and weak anomalies is, that in most cases, an outlier does not have to have the points with highest anomaly scores at the beginning and end of the outlier but rather in the middle. The border area between

outlier and normal parts of a silhouette often consists of weak anomalies. These can also occur on their own in more or less normal parts of the silhouette. In such cases we do not want to mark them as outliers. Figure 4.1 gives an impression of this in the case of mountain silhouettes.

Based on the application, the characteristics of outliers vary, especially in regard to how many consecutive strong anomalies are necessary to form a meaningful outlier. In order to incorporate this thought, we define an outlier with a minimum number of consecutive strong anomalies.

Definition 5 (l Outlier)**.** Let $P = (p_1, \ldots, p_n)$ be a polygonal chain, anom be an anomaly score and $\tau_w \leq \tau_s \in \mathbb{R}_{\geq 0}$ be thresholds for weak and strong anomalies.

We call a subsequence $O = (p_s, \ldots, p_e)$ of P a l *outlier* if the following holds:

1. All points in O are weak anomalies, that is, $\text{anom}(p) \geq \tau_w$ for all $p \in O$.

2. A subsequence $SO \subseteq O$ with a length of l exists, that consists of strong anomalies only, i.e. $\forall p \in SO : \text{anom}(p) \geq \tau_s$.

Note here, that the length l does not correspond to the total length of the outlier, as is the case for example with discords in Hot Sax [KLF05]. Instead l is the minimal number of consecutive strong anomalies required for O to be considered as outlier. Obviously, the fact that every $l + 1$ outlier also is a l outlier follows directly from the definition.

In most cases, we are not only interested in finding outliers of a given severity, but wants to find the largest possible outliers of that severity. We therefore introduce the following definition:

Definition 6 (Maximum l outlier)**.** Let $P = (p_1, \ldots, p_n)$ be a polygonal chain, anom be an anomaly score and $\tau_w \leq \tau_s \in \mathbb{R}_{\geq 0}$ be thresholds for weak and strong anomalies.

We call a l outlier $O = (p_s, \ldots, p_e)$ of P a *maximum l outlier* if neither $(p_{s-1}, \ldots, p_e)$ nor $(p_s, \ldots, p_{e+1})$ is a l outlier.

From this definition it follows that a maximum l outlier O is the outlier with the maximal number of points in a certain region of a polygonal chain in the sense that any longer subsequence intersecting with O cannot be an l outlier. It does not mean, that there cannot be another l outlier $\tilde{O} \in P$ that consists of more points than O. We can however state the following theorem:

Theorem 7. *Let $P = (p_1, \ldots, p_n)$ be a polygonal chain, anom be an anomaly score, $l \in \mathbb{N}$ and $\tau_w \leq \tau_s \in \mathbb{R}_{\geq 0}$ be threshold for weak and strong anomalies. Then the set $\mathcal{O}$ of all maximum l outliers is unique.*

Proof. Let $\mathcal{O}_1 \neq \mathcal{O}_2$ be two different sets of all maximum l outliers of P. This is equivalent to $\exists O \in \mathcal{O}_1 : O \notin \mathcal{O}_2$ or $\exists O \in \mathcal{O}_2 : O \notin \mathcal{O}_1$.

Let, without loss of generality, $O \in \mathcal{O}_1$ and $O \notin \mathcal{O}_2$. It follows that either $\mathcal{O}_2 \subset \mathcal{O}_1$ or there exists $Q \in \mathcal{O}_2$ with $Q \notin \mathcal{O}_1$.

Case 1 Let $\mathcal{O}_2 \subset \mathcal{O}_1$. Then obviously $\mathcal{O}_2$ is not the set of all maximum l outliers of P and thus a contradiction to the statement.

Case 2 Let $Q = (q_1, \ldots, q_i) \in \mathcal{O}_2$ with $Q \notin \mathcal{O}_1$. Let the intersection of the outliers $O \cap Q := \{o | o \in O \wedge o \in Q\}$.

Case 2.1 Let $O \cap Q = \varnothing$. Then neither $\mathcal{O}_1$ nor $\mathcal{O}_2$ are sets of all maximum l outliers.

Case 2.2 Let $O \cap Q \neq \varnothing$. Then, without loss of generality, there exists $o \in O$ with $o \notin Q$ and o is either adjacent to q_1 or q_i in P. Also, since O is an outlier, o has to be a weak anomaly, such that either $(o, q_1, \ldots, q_i)$ or $(q_1, \ldots, q_i, o)$ is an l outlier. Due to Definition 6, Q is not a maximum l outlier and thus $\mathcal{O}_2$ cannot be a set of all maximum l outliers which, again, is a contradiction to the statement. □

4.2 Above Average Distance

For the outlier detection in AdaMS we assume, that outliers do not consist of single points, but of subsequences of the polygonal chain that forms the silhouette. Hence, the anomaly scores are not based on the properties of single points, but on histograms that summarise the properties of consecutive points in the silhouette. In order to compare these histograms, specialised histogram distance functions are necessary, since normal distances for Euclidean spaces usually do not yield good results with histograms. This is mostly due to the fact, that in an Euclidean space the dimensions are not ordered, i.e. the distance between two values in different dimensions does not change by the order of the dimensions:

$$\left| \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} - \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \right|_p = \left| \begin{pmatrix} a_1 \\ a_3 \\ a_2 \end{pmatrix} - \begin{pmatrix} b_1 \\ b_3 \\ b_2 \end{pmatrix} \right|_p .$$

For histograms, this behaviour is usually not desirable since it ignores the semantics of the bins. Consider, for example, a histogram that counts the length of segments, where the first bin or dimension contains the number of samples of a length in $[0, 1)$, the second bin contains the number of samples of a length in $[1, 3)$ and the third bin the number of samples with a length greater or equal than 3. In such a case, intuitively we would want $d((1, 0, 0), (0, 0, 1))$ to be greater than $d((1, 0, 0), (0, 1, 0))$ since the difference between the samples is larger in the first example.

A related problem is the fact, that, for example, with the Euclidean norm,

$$||(1, 0, 0) - (2, 0, 0)||_2 = ||(1, 0, 0) - (1, 1, 0)||_2.$$

Consider these vectors a histograms as in the paragraph above and intuitively one would say that a set of one segment with a length in $[0, 1)$ is closer related to a set of two segments with a length in $[0, 1)$ than to a set of one segment with a length in $[0, 1)$ and the second segment with a length in $[1, 3)$. We can formally introduce histograms as follows:

Definition 8 (Histogram, Normalised Histogram)**.** We call $H = (h_1, \ldots, h_n) \in \mathbb{R}^n_{\geq 0}$ a *histogram* with n bins.

We call $H = (h_1, \ldots, h_n) \in [0,1]^n$ a *normalised histogram* with n bins if

$$\sum_{i=1}^{n} h_i = 1.$$

In this section, the *above average distance* is introduced, that considers the second of the problems mentioned above. It is defined as follows:

Definition 9 (Above Average Distance)**.** Let $G = (g_1, \ldots, g_n)$ and $H = (h_1, \ldots, h_n)$ be histograms with n bins.

We call

$$\mathrm{aad}(G, H) := \max\{|\mathrm{aab}(G)|, |\mathrm{aab}(H)|\} - |\mathrm{aab}(G) \cap \mathrm{aab}(H)|,$$

with

$$\mathrm{aab}(I) := \left\{ i \in \{1, \ldots, n\} \,\middle|\, f_i \geq \frac{1}{n} \sum_{j=1}^{n} f_j \right\}$$

for a histogram I with n bins, the *above average distance* of G and H.

As mentioned above, a histogram with n bins in this work is a n dimensional vector with an underlying semantics as to what a dimension means. Consequently, in the following, a normalised histogram $H = (h_1, \ldots, h_n)$ with n bins is a normalised vector, i.e. $\sum_{i=1}^{n} h_i = 1$.

Theorem 10. *The above average distance from Definition 9 is a pseudometric but in general not a metric.*

This theorem means, that for the set of histograms with n bins $\mathcal{H}_n$, $(\mathcal{H}_n, \mathrm{aad})$ is a pseudometric space, i.e. a generalised metric space in which two distinct points can have a distance of zero.

Proof. Let G, H, I be histograms with n bins. First, we show, that aad is a pseudometric, that means, that aad is non-negative, that it fulfils the identity of indiscernibles, that it is symmetric and that it is subadditiv.

Non-negativity We have to show that $\mathrm{aad}(G, H) \geq 0$. Since

$$|\mathrm{aab}(G)| \geq |\mathrm{aab}(G) \cap \mathrm{aab}(H)| \text{ and } |\mathrm{aab}(H)| \geq |\mathrm{aab}(G) \cap \mathrm{aab}(H)|$$

it holds that

$$\max\{|\mathrm{aab}(G)|, |\mathrm{aab}(H)|\} \geq |\mathrm{aab}(G) \cap \mathrm{aab}(H)|.$$

Thus

$$0 \leq \max\{|\mathrm{aab}(G)|, |\mathrm{aab}(H)|\} - |\mathrm{aab}(G) \cap \mathrm{aab}(H)| = \mathrm{aad}(G, H)$$

follows.

Identity of Indiscernibles It holds that

$$\mathrm{aad}(H, H) = \max\{|\mathrm{aab}(H)|, |\mathrm{aab}(H)|\} - |\mathrm{aab}(H) \cap \mathrm{aab}(H)| = |\mathrm{aab}(H)| - |\mathrm{aab}(H)| = 0.$$

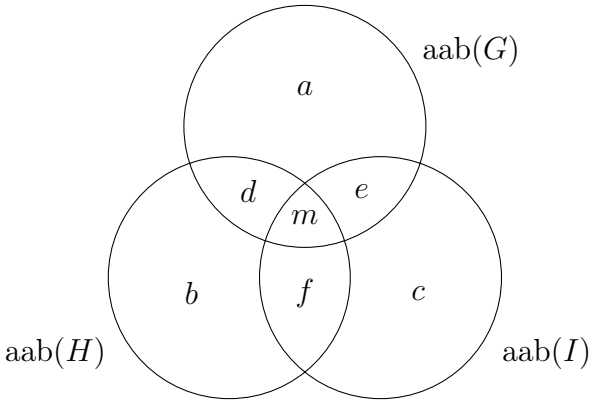


Figure 4.2: Sets for proof of subadditivity.

Symmetry The intersection of sets is symmetric and sets are not ordered, such that

$$\begin{aligned}\operatorname{aad}(G,H) &= \max\{|\operatorname{aab}(G)|, |\operatorname{aab}(H)|\} - |\operatorname{aab}(G) \cap \operatorname{aab}(H)| \\ &= \max\{|\operatorname{aab}(H)|, |\operatorname{aab}(G)|\} - |\operatorname{aab}(H) \cap \operatorname{aab}(G)| = \operatorname{aad}(H,G).\end{aligned}$$

Subadditivity For the proof of subadditivity we split the sets $\operatorname{aab}(G)$, $\operatorname{aab}(H)$, and $\operatorname{aab}(I)$ in seven disjoint sets

$$\begin{aligned}a &= \operatorname{aab}(G) \setminus (\operatorname{aab}(H) \cup \operatorname{aab}(I)) \\ b &= \operatorname{aab}(H) \setminus (\operatorname{aab}(G) \cup \operatorname{aab}(I)) \\ c &= \operatorname{aab}(I) \setminus (\operatorname{aab}(G) \cup \operatorname{aab}(H)) \\ d &= (\operatorname{aab}(G) \cap \operatorname{aab}(H)) \setminus \operatorname{aab}(I) \\ e &= (\operatorname{aab}(G) \cap \operatorname{aab}(I)) \setminus \operatorname{aab}(H) \\ f &= (\operatorname{aab}(H) \cap \operatorname{aab}(I)) \setminus \operatorname{aab}(G) \\ m &= \operatorname{aab}(G) \cap \operatorname{aab}(H) \cap \operatorname{aab}(I)\end{aligned}$$

as shown in figure 4.2.

Without loss of generality, let $|\operatorname{aab}(G)| \geq |\operatorname{aab}(H)|$. Then

$$\operatorname{aad}(G,H) = |\operatorname{aab}(G)| - |\operatorname{aab}(G) \cap \operatorname{aab}(H)| = |a| + |e|.$$

Case 1 Let $|\operatorname{aab}(H)| \geq |\operatorname{aab}(I)|$. Then $\operatorname{aad}(H,I) = |b| + |d|$ and $\operatorname{aad}(G,I) = |a| + |d|$, because $\operatorname{aad}(G) \geq \operatorname{aad}(H) \geq \operatorname{aad}(I)$. From this it follows that

$$\begin{aligned}\operatorname{aad}(G,I) + \operatorname{aad}(H,I) - \operatorname{aad}(G,H) &= |a| + |d| + |b| + |d| - |a| - |e| \\ &= |b| + 2|d| - |e| \geq 0,\end{aligned}$$

because

$$\begin{aligned}&|\operatorname{aab}(H)| \geq |\operatorname{aab}(I)| \\ \Rightarrow\ &|b| + |d| + |f| + |m| \geq |c| + |e| + |f| + |m| \\ \Rightarrow\ &|b| + |d| \geq |c| + |e| \geq |e|.\end{aligned}$$

and $|b| + 2|d| \geq |b| + |d|$.

Case 2 Assume now, that $\text{aab}(H) < \text{aab}(I)$, so $\text{aad}(H, I) = |c| + |e|$.

Case 2.1 Let $|\text{aab}(G)| \geq |\text{aab}(I)|$. It follows that

$$\begin{aligned}\text{aad}(G, I) + \text{aad}(H, I) - \text{aad}(G, H) &= |a| + |d| + |c| + |e| - |a| - |e| \\ &= |d| + |c| \geq 0.\end{aligned}$$

Case 2.2 Now consider $|\text{aab}(G)| < |\text{aab}(I)|$, so that $\text{aad}(G, I) = |c| + |d|$. Then

$$\begin{aligned}\text{aad}(G, I) + \text{aad}(H, I) - \text{aad}(G, H) &= |c| + |f| + |c| + |e| - |a| - |e| \\ &= 2|c| + |f| - |a| \geq 0,\end{aligned}$$

because, similar to Case 1,

$$|\text{aab}(I)| \geq |\text{aab}(G)| \Rightarrow |c| + |f| \geq |a| + |d| \geq |a|.$$

We have now shown, that aad is a pseudometric. However, since

$$\text{aad}((0.5, 0.5, 0), (0.4, 0.6, 0)) = 2 - 2 = 0$$

and $(0.5, 0.5, 0) \neq (0.4, 0.6, 0)$, aad is not a metric. □

Comparison to Other Histogram Distances

In this section we compare the above average distance to other distances that are often used with histograms. These are the histogram intersection distance [SB91] and the earth mover's distance [RTG98, RTG00] or Wasserstein metric [Vas69].

The histograms intersection distance is a distance function based on the histogram intersection. For two normalised histograms $G = (g_1, \ldots, g_n), H = (h_1, \ldots, h_n)$ with n bins each, the intersection of those histograms is defined as

$$G \cap H = \sum_{i=1}^{n} \min\{g_i, h_i\}.$$

Obviously it holds that $0 \leq G \cap H \leq 1$, and G and H are more similar the larger $G \cap H$ is. Consequently, in order to derive a distance function from the histogram intersection, we define

$$\text{hid}(G, H) = 1 - G \cap H.$$

Theorem 11. *Let $G = (g_1, \ldots, g_n)$ and $H = (h_1 \ldots, h_n)$ be normalised histograms with n buckets. Then hid(G, H) is a metric.*

Proof. In the following $G = (g_1, \ldots, g_n)$, $H = (h_1 \ldots, h_n)$, and $J = (j_1, \ldots, j_n)$ be normalised histograms with n buckets.

Non-negativity $\text{hid}(G, H) \geq 0$ for all G, H, since

$$\text{hid}(G, H) = 1 - G \cap H = 1 - \sum_{i=1}^{n} \min\{g_i, h_i\} \geq 1 - \sum_{i=1}^{n} g_i = 1 - 1 = 0.$$

Identity of Indiscernibles $\mathrm{hid}(G,H)=0$ is equivalent to $G\cap H=1$. Since both G and H are normalised, this is equivalent to $g_i=h_i=\min\{g_i,h_i\}$ for all $i\in[1,n]$ and thus $G=H$.

Symmetry It holds that

$$\mathrm{hid}(G,H)=1-\sum_{i=1}^{n}\min\{g_i,h_i\}=1-\sum_{i=1}^{n}\min\{h_i,g_i\}=\mathrm{hid}(H,G).$$

Subadditivity First, note that

$$g_i=\begin{cases}\min(g_i,h_i)+|g_i-h_i| & \text{if } g_i>h_i\\ \min(g_i,h_i) & \text{else}\end{cases}$$

and

$$h_i=\begin{cases}\min\{g_i,h_i\}+|g_i-h_i| & \text{if } h_i\geq g_i\\ \min\{g_i,h_i\} & \text{else.}\end{cases}$$

From this it follows that in any case

$$2\cdot\min(g_i,h_i)=g_i+h_i-|g_i-h_i|$$

because

$$g_i+j_i=2\cdot\min\{g_i,h_i\}+|g_i-h_i|.$$

We have to prove that $\mathrm{hid}(G,J)\leq\mathrm{hid}(G,H)+\mathrm{hid}(H,J)$ which is equivalent to $\mathrm{hid}(G,H)+\mathrm{hid}(H,J)-\mathrm{hid}(G,J)\geq 0$. It holds, that

$$\begin{aligned}
&\mathrm{hid}(G,H)+\mathrm{hid}(H,J)-\mathrm{hid}(G,J)\\
=&1-G\cap H+1-H\cap J-1+G\cap J\\
=&1+(G\cap J-G\cap H-H\cap J)\\
=&1+\left(\sum_{i=i}^{n}\min\{g_i,j_i\}-\sum_{i=i}^{n}\min\{g_i,h_i\}-\sum_{i=i}^{n}\min\{h_i,j_i\}\right)\\
=&1+\frac{1}{2}\sum_{i=1}^{n}\left(2\cdot\min\{g_i,j_i\}-2\cdot\min\{g_i,h_i\}-2\cdot\min\{h_i,j_i\}\right)\\
=&1+\frac{1}{2}\sum_{i=1}^{n}(g_i+j_i-|g_i-j_i|-(g_i+h_i-|g_i-h_i|)-(h_i+j_i-|h_i-j_i|))\\
=&1+\frac{1}{2}\sum_{i=1}^{n}(-2h_i+|g_i-h_i|+|h_i-j_i|-|g_i-j_i|)\\
\geq&1+\frac{1}{2}\sum_{i=1}^{n}(-2h_i)=1-\sum_{i=1}^{n}h_i=0,
\end{aligned}$$

since $|g_i-h_i|+|h_i-j_i|-|g_i-j_i|\geq 0$, because $|\cdot|$ is a metric in $\mathbb{R}$. □

It can be seen from the definition of the histogram intersection distance, that it treads every bin the same. In contrast to this, with the above average distance the distance computation is focused on the outstanding bins of a histogram. Thus, the above average distance exaggerates the difference between histograms that differ greatly while the distance of histograms with similarly filled bins is understated in comparison with the histogram cut distance. The computational complexity of both distance functions is identical, with both being in linear time in respect to the number of histogram bins.

The earth mover's distance is a metric that has its roots in the first Wasserstein distance for normalised data, and was introduced first as a means to compare probability distributions. For this, assume that each distribution D has a so called signature S_D consisting of a number n of tuples (p_i, w_i), $i \in [1, n]$, where $p_i \in \mathbb{R}^d$ is the position of a representative of the distribution D and $w_i \in [0, 1]$ is its weight. A popular way of creating the signature of a distribution is by clustering it into n clusters. Then, the cluster representatives form the p_i and the portion of points that are assigned to a cluster i give its weight w_i. In general, the earth mover's distance can be defined between distributions with a varying number of representatives. However, for the sake of simplicity in the following we only consider the case for a fixed number n of representatives as this case suffices for histograms with a fixed number of bins.

Given two probability distributions D and E and their respective signatures

$$S_D = \{(p_1, w_1), \ldots, (p_n, w_n)\}$$
$$S_E = \{(q_1, v_1), \ldots, (q_n, v_n)\}$$

with

$$\sum_{i=1}^{n} w_i = \sum_{i=1}^{n} v_i = 1,$$

the earth mover's distance is the minimal solution of the transportation problem to turn D into E. Hereby we use a metric $dist$ to compute the distances between the cluster representatives. Most commonly used are L^P-norms. It can be shown, that the earth mover's distance is a metric if, and only if $dist$ is a metric.

To solve the transportation problem, we want to find a flow

$$F = \begin{pmatrix} f_{1,1} & \cdots & f_{n,1} \\ \vdots & \ddots & \vdots \\ f_{1,n} & \cdots & f_{n,n} \end{pmatrix} \in [0, 1]^{n \times n}$$

where $f_{i,j}$ is the flow from p_i to q_j, such that

$$\sum_{i=1}^{n} \sum_{j=1}^{n} f_{i,j} \cdot dist(p_i, q_j)$$

is minimised with respect to the following constraints:

$$\sum_{i=1}^{n} f_{i,j} \leq v_j \text{ for all } j \in [1, n]$$
$$\sum_{j=1}^{n} f_{i,j} \leq w_i \text{ for all } i \in [1, n].$$

These constraints ensure that the flow never exceeds the weight at one of the representatives. Speaking in terms of transportation theory, this means, that we ensure, that we never take more goods than are available from p_i and never deliver more goods than needed to q_j.

The above minimisation is usually solved by algorithms that solve the minimum-cost flow problem, such as the polynomial time network simplex algorithm [Orl97]. By this we get

$$\text{emd}(S_D, S_E) = \min \sum_{i=1}^{n} \sum_{j=1}^{n} f_{i,j} \cdot dist(p_i, q_j).$$

Since normalised histograms are another way to express signatures of distributions, the earth mover's distance can be applied to those, too. In this case the position of a bin is used as a representative and the bin's entry gives the weight at that point.

It is obvious that the earth mover's distance is a more complicated metric than the above average distance and the histogram intersection distance both in terms of computational cost as in interpretation. This is because in contrast to the aforementioned distances, the earth mover's distance does take the positions of bins in the histograms into account and sees bins that are closer together as more similar. As an example to underline this point, consider three normalised histograms $H_1 = (1, 0, 0), H_2 = (0, 1, 0)$, and $H_3 = (0, 0, 1)$ over the bin values $\{1, 2, 3\}$. Then,

$$\text{aad}(H_1, H_2) = \text{aad}(H_1, H_3) = 1$$

and

$$\text{hid}(H_1, H_2) = \text{hid}(H_1, H_3) = 1$$

while

$$\text{emd}(H_1, H_2) = 1 \text{ but } \text{emd}(H_1, H_3) = 2,$$

with *dist* the L^2 norm, since the distance between 1 and 2 is smaller than the distance between 1 and 3.

The classic earth mover's distance in our case has a computational time in $\mathcal{O}(n^3 \log(n)^2)$ due to the network simplex algorithm [Tar97]. A number of faster implementations of the earth mover's distance exist, that imposes additional constraints on the ground distance, such as [PW08, PW09] and has been used in this work. Also, approximated earth mover's distances exist, for example [GD04, SJ08].

4.3 Basic Algorithm

With the definitions and outlier properties from Section 4.1 and the histogram distances introduced in Section 4.2, the basic outlier detection algorithm from AdaMS can now be presented. In the following, we show the different steps of the outlier detection in its basic form as depicted by Figure 4.3. In that figure, it can be seen, that the outlier detection process takes place in three steps, namely the silhouette conversion that yields a relative silhouette, followed by the anomaly score computation, in which each point in the relative silhouette gets an anomaly score. The pipeline is concluded by the actual outlier detection, where the maximum l outliers are computed.

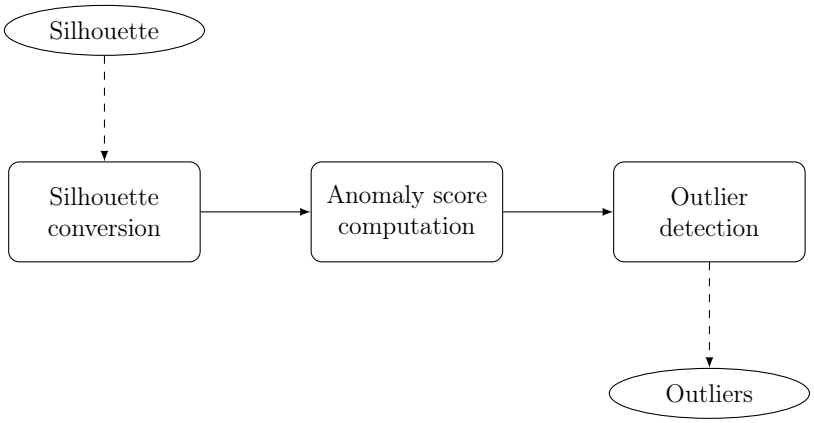


Figure 4.3: Flow diagram of the outlier detection pipeline.

4.3.1 Silhouette Conversion

We get a silhouette in the form of a polygonal chain $P = (p_1, \ldots, p_n)$ out of the segmentation part of AdaMS. In the basic case, P consists of the pixels coordinates, i.e. for every $p_i \in P : p_i = (x_i, y_i)$. This representation has the disadvantage that patterns in the silhouette are hard to recognise since the coordinates are absolute. For example, the subsequence $((1,1),(2,1),(2,3))$ and the subsequence $((5,5),(6,5),(6,7))$ have completely different absolute values, however, the form of the subsequences is the same. We therefore convert P to a so-called relative silhouette. We currently have two approaches to this, both of which give the position of a point p_i relative to its predecessor p_{i-1}.

The first of these methods has been used in [BSC16] and uses a representation based on polar coordinates. That means, that we use the angle of the x-axis and the segment between p_{i-1} and p_i together with the length of that segment for all points $p_i \in P$ except for p_1. Essentially, the conversion to such a relative silhouette is a mapping

$$P = (p_1, \ldots, p_n) \to alrel(P) = (v_1, \ldots, v_n)$$

where we set $v_1 = (0,0)$ and for all $i \in [2,n], v_i = (l_i, a_i)$ with $l_i = |p_i - p_{i-1}|$ and $a_i = \text{atan2}(y_i - y_{i-1}, x_i - x_{i-1})$. Here, $\text{atan2}(y,x)$ is the function implemented in nearly every programming language, which returns $\arctan(x/y)$ but, since the signum of both x and y are known, the right quadrant for the vector (x,y) can be given. It is defined by

$$\text{atan2}(y,x) = \begin{cases} 2\arctan\left(\frac{y}{\sqrt{x^2+y^2}+x}\right) & \text{if } x > 0 \text{ or } y \neq 0, \\ \pi & \text{if } x < 0 \text{ and } y = 0, \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0. \end{cases}$$

Another variant for computing a relative silhouette, that is both easier to compute as well as less prone to rounding errors, is given by the mapping

$$P = (p_1, \ldots, p_n) \to vrel(P) = (v_1, \ldots, v_n)$$

where, again, we set $v_1 = (0,0)$ and $v_i = p_i - p_{i-1}$ for all $i \in [2,n]$. This obviously has the advantage, that, if the points in P only contain integers, as is the case in

silhouettes extracted from pixel images, then $vrel(P)$ also contains integers, only. Thus, no rounding errors can occur. On the negative side, we can in general give no boundaries on the values of points in a relative silhouette computed by the *vrel* mapping, in contrast to the *alrel* mapping, where we know that the lengths are greater or equal than 0 and the angles are in a range of $[0°, 360°]$.

In terms of computational complexity, both mappings can obviously be implemented in linear time depending on the number n of points in the silhouette, since each point has to be looked at at most two times.

4.3.2 Anomaly Score Computation

Irrespective of the silhouette conversion algorithm from the section above, the anomaly score computation is based on a reference histogram. Essentially, the reference histogram is a histogram that is computed in the same way as normal histograms from one or more relative silhouettes that themselves are marked by a human being as not containing outliers.

In the following, let either $P_r = alrel(P)$ or $P_r = vrel(P)$ for a given silhouette P, such that $P_r = (v_1, \ldots, v_n)$. As has already been mentioned, we use histograms of reference data as well as histograms of parts of P. In Section 4.2 we introduced histograms as vectors with entries greater or equal than zero. However, we have now data with more than one dimension or feature, namely length and angle in case of the *alrel* relative silhouette or relative x and y coordinates if the *vrel* mapping is used. In order to accommodate this, we first define corresponding histograms.

Definition 12 (Histogram with f Features)**.** We call

$$H \in \mathbb{R}_{\geq 0}^{n_1 \times \ldots \times n_f}$$

a *histogram with f features* and $n_1, \ldots, n_f$ bins per feature.

Lemma 13. *A bijective mapping $red : \mathbb{R}_{\geq 0}^{n_1 \times \ldots \times n_f} \to \mathbb{R}_{\geq 0}^{n_1 \cdot \ldots \cdot n_f}$ exists.*

Proof. The index sets of histograms in $\mathbb{R}_{\geq 0}^{n_1 \times \ldots \times n_f}$ and $\mathbb{R}_{\geq 0}^{n_1 \cdot \ldots \cdot n_f}$ are finite and have the same cardinality. □

Let $H_1 \in \mathbb{R}_{\geq 0}^{n_1 \times \ldots \times n_f}$ and let $H_1(i_1, \ldots, i_f) = h_{i_1, \ldots, i_f}$ denote the element at position $(i_1, \ldots, i_f)$ in H_1. We set $red : \mathbb{R}_{\geq 0}^{n_1 \times \ldots \times n_f} \to \mathbb{R}_{\geq 0}^{n_1 \cdot \ldots \cdot n_f}$ as the mapping $H_1 \mapsto H_2$ where $H_2 = (h_1, \ldots, h_{n_1 \cdots n_f}) \in \mathbb{R}_{\geq 0}^{n_1 \cdot \ldots \cdot n_f}$ and under which $H_1(i_1, \ldots, i_f)$ is mapped onto

$$H_2 \left(\sum_{j=1}^{f} \left((i_j - 1) \prod_{k=j+1}^{f} n_k \right) + 1 \right).$$

Here, we use $H_2(i) = h_i$, for a similar notation as above.

For the inverse mapping $red^{-1} : \mathbb{R}_{\geq 0}^{n_1 \cdot \ldots \cdot n_f} \to \mathbb{R}_{\geq 0}^{n_1 \times \ldots \times n_f}$ as the mapping

$$H_2(i) \mapsto H_1(i_1, \ldots, i_f)$$

with

$$i_j = \left\lceil \frac{\operatorname{nmod}\left(i, \prod_{k=j}^{f} n_k\right)}{\prod_{k=j+1}^{f} n_k} \right\rceil, j \in \{1, \ldots, f\}$$

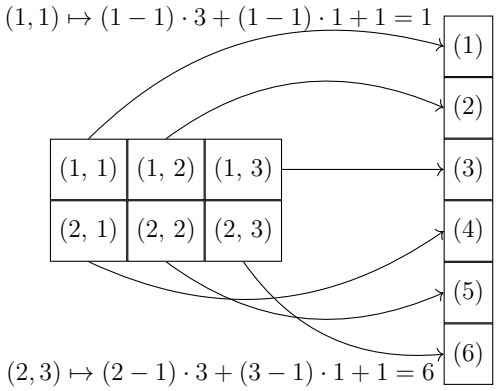


Figure 4.4: Example for the mapping of a two dimensional histogram onto a one dimensional histogram.

and

$$\operatorname{nmod}(i,n) := \begin{cases} n & \text{if } n|i, \\ i \mod n & \text{else.} \end{cases}$$

This definition makes $\operatorname{nmod}(i,n)$ a normal modulo function, only that the result is in $[1,n]$ instead of $[0,n)$. Figure 4.4 gives an example where a two dimensional histogram is mapped onto a one dimensional histogram.

With these functions we are able to use histograms with multiple features and the histograms under *red* equivalently in the following. The representation with multiple features usually is more natural if the semantics of the histograms are used, while for other computations, such as the above average distance, the reduced version of the histogram is more handy.

Definition 14 (Normalised Histogram with f Features)**.** We call

$$H \in [0,1]^{n_1 \times \ldots \times n_f}$$

a *normalised histogram with f features* and $n_1, \ldots, n_f$ bins per feature if

$$\sum_{i=1}^{n_1 \cdots n_f} h_i = 1$$

for $red(H) = (h_1, \ldots, h_{n_1,\cdots,n_f})$.

Given the reference histogram H_{ref} and a number of window lengths and weights we want to consider, the actual outlier score computation is carried out as follows:

For all window lengths wl:

For all windows w of length wl:

Compute the histogram H of w.

Compute the above average distance $d_{wl} = \operatorname{dist}(H, H_{ref})$, where dist is one of the histogram distances from Section 4.2.

Store d_{wl} with the corresponding weight at every point in w.

For all points in P_r compute the anomaly score as the weighted mean of all stored distances for that point.

In this case, the window length is a positive integer that gives the length of the subsequences of the relative polygonal chain that are used to compute the histograms. Such a subsequence $w = (v_i, \ldots, v_{i+wl})$ is called a window.

In order to compute a histogram from a window we use a sequence of borders for each of the two dimensions, i.e. $B_1 = (b_1^1, \ldots, b_1^{n_1})$ and $B_2 = (b_2^1, \ldots, b_2^{n_2})$ where $b_j^i < b_j^{i+1}$ for $j \in \{1, 2\}$ and $i \in \{1, \ldots, n_j\}$. Let $v = (f_1, f_2)$ be a point in P_r. We say, that a feature f_j falls into the i-th bin if

$$b_j^i \leq f_j < b_j^{i+1}.$$

Consequently, for a two dimensional point v we get a pair of bin coordinates. We then increase the counter of the bin in the two dimensional histogram. Thus, for each window of length wl we get a histogram H with

$$\sum_{h \in H} h = wl.$$

We then compute $d_{wl} = \text{dist}(red(H), red(H_{ref}))$ and store that distance and the corresponding weight w_{wl} for every point v that contributed to H. So, for every point we get multiple distances for every window length, so that finally we get a set

$$\text{dists}(v) = \{(d_{wl_1}^1, w_{wl_1}), \ldots, (d_{wl_1}^{k_1}, w_{wl_1}), (d_{wl_2}^1, w_{wl_2}), \ldots)\}.$$

From this, we can compute a point's anomaly score as

$$\text{anom}(v) = \frac{\sum_{(d,w) \in \text{dists}(v)} d \cdot w}{\sum_{(d,w) \in \text{dists}(v)} w}.$$

For the sake of a shorter notation, in the future we use

$$\text{anom}(P) = \text{anom}(P_r) = (\text{anom}(v_1), \ldots, \text{anom}(v_n))$$

for $P = (p_1, \ldots, p_n)$ and $P_r = (v_1, \ldots, v_n)$.

Obviously, since every point is looked at at most window length times per window and for a fixed number of windows, the anomaly score computation, again, is in linear time complexity in respect to the number of point in the silhouette.

4.3.3 Outlier Detection

The previous sections introduced all means necessary to now establish the actual outlier detection algorithm as used in [BSC16]. The goal of this step is to find the set of maximum l outliers of a given silhouette in as little steps as possible.

Let $\text{anom}(P_r) = (\text{anom}(v_1), \ldots, \text{anom}(v_n))$ be the sequence of anomaly scores of a silhouette $P = (p_1, \ldots, p_n)$. Also, let $l > 0$ be an integer that gives the minimum inner length of an outlier and let $0 < \tau_w \leq \tau_s$ as in Definitions 3 and 4 be thresholds for weak and strong anomalies.

In order for the algorithm to work in linear time, we need three variables that store information. These are s_{out} for the start index of an outer outlier, s_{in} for the start of

Condition \ Point type	no anomaly	weak anomaly	strong anomaly
No outlier start	—	set $s_{out} = i$	set $s_{out} = s_{in} = i$
Outer start	set s_{out} to nothing	—	set $s_{in} = i$
Inner start, $i - s_{in} \leq l$	set s_{out} and s_{in} to nothing	set s_{in} to nothing	—
Inner start, $i - s_{in} > l$	store outlier, reset variables	set $e_{in} = i - 1$	—
Inner end found	store outlier, reset variables	—	—

Table 4.1: Effects of point types and conditions during the outlier detection.

an inner outlier, and e_{in} for the end of an inner outlier. At the beginning all of these variables are initialised to nothing.

Now, for every $i \in [1, n]$ we check whether v_i qualifies as a weak or strong anomaly or no anomaly at all and the consequences this have in the current conditions. Table 4.1 gives an overview over this. Note, that in this table, the column "weak anomaly" only gives the effects for weak anomalies, that are not strong anomalies as well.

If v_i is no anomaly we have the following cases to consider: If we do not have an outer outlier start yet, i.e. s_{out} is set to nothing, we do nothing. If we do have a possible outer outlier start, that is s_{out} has a value, but no inner start has been found yet – i.e. s_{in} is nothing – we set s_{out} back to nothing, because there is no inner part to this potential outlier. If we have found an inner outlier start, such that s_{in} has a value, we have to have an outer outlier start as well, since every strong anomaly is a weak anomaly, too. We then have to distinguish between two cases. First, let $i - s_{in} \leq l$. Then the outlier is not an l outlier, since the subsequence of strong anomalies is shorter than l. Thus we reset s_{in} and s_{out} both to nothing and proceed with the next point. Second, let $i - s_{in} > l$. In this case, the outlier from position s_{out} to position $i - 1$ is a maximum l outlier, since it cannot be extended in any direction and the sequence of strong anomalies has a length of at least l. Thus we add $(p_{s_out}, \ldots, p_{i-1})$ to the list of outliers and afterwards reset both s_{in} and s_{out} to nothing. If an inner end has already been found, i.e. e_{in} has a value, we also store the outlier and reset all three outlier variables.

Next, we consider the case, that p_i is a weak anomaly but not a strong anomaly. Again, we have to consider the same cases as above. If s_{out} is nothing, we have not found the start of a possible outlier yet. Thus, we set $s_{out} = i$. If s_{out} already has a value we have to do nothing, since a weak anomaly can be part of the outer part of an outlier. If we have already a value for s_{in} and $i - s_{in} \leq l$, that means, we have found a beginning of an inner outlier but the inner portion is shorter than l, we reset s_{in}. In this case, we do not reset s_{out} since the possibility to find a sequence of strong anomalies that has a length of l or more in this sequence of weak anomalies still exists. If we have a value for s_{in} and $i - s_{in} > l$ but no value for e_{in} yet, we set $e_{in} = i - 1$ since we now have a sequence of l or more strong anomalies in the current outlier. However, we do not save the outlier yet, since further points could be weak anomalies. Thus, we are not sure yet, whether we already have a maximum l outlier. If there already is a value for e_{in}, we have to do nothing.

Finally, we have to consider the case, that p_i is a strong anomaly. If we do not have a value for s_{out} yet, we set $s_{out} = s_{in} = i$, since we have found the beginning of a possible outlier that starts with a strong anomaly. Also, since we did not have a value for s_{out}, it is not possible to have a value for s_{in}. Next, assume the case, that s_{out}

already has a value. In this case we set $s_{in} = i$ since we found the start of a sequence of strong anomalies. If s_{in} already has a value, we do nothing, due to the following consideration: Either, we do not have an value for e_{in} yet, i.e. we did not find the end of the sequence of strong anomalies, but then we still have not found the end of that sequence because p_i is a strong anomaly as well, or we already have a value for e_{in}. In this case, we already found a sequence of strong anomalies with a length of at least l, thus we can handle p_i as a weak anomaly in this case. Thus, we further extend the outlier in order to find the maximum l outlier.

It is obvious, that the detected outliers have to be maximum l outliers: No outliers, that are not l outliers are reported, since in this case, the information describing the outlier is discarded. Also, if an outlier $O = (p_m, \ldots, p_k)$, $m, k \in [1, n]$, is detected, we are sure, that neither p_{m-1} nor p_{k+1} are weak anomalies. Because of this neither $(p_{m-1}, \ldots, p_k)$ nor $(p_m, \ldots, p_{k+1})$ are l outliers. This makes O an maximum l outlier by definition.

Also, we know that maximum l outliers do not overlap. For every weak anomaly, that is not already part of a possible outlier, we check whether an l outlier starts at that position, we are sure to find every possible maximum l outlier. From Theorem 7 it follows, that the reported set is the unique set of all maximum l outliers of P.

As can be seen from the description above, we look at every anomaly score only once. Since the number of anomaly scores equals the number of points in the polygonal chain that we get as input, it is obvious that the outlier detection step is linear to the length of the polygonal chain, i.e. the time complexity of the algorithm is in $\mathcal{O}(n)$ for n the number of points in the polygonal chain P.

4.3.4 Outlier Classification

In the Sections 3.4 and 3.5 it has been mentioned that the adaptive refinement steps of AdaMS expects not only outliers but also the classes these outliers belong to. Due to the nature of outliers, usually not many samples exist. Therefore, we opt for a simple k nearest neighbour algorithm [Alt92] for the classification due to its robustness to small training samples.

The silhouette refinement step can handle two sets of classes as described in Section 3.5, namely either {segmentation error, obstacle} or {segmentation high, segmentation low, obstacle}. In order to improve the results, during the classification we introduce the additional class of false positives to improve the result of the outlier detection. Outliers, that are classified as false positives are simply not reported back to the refinement step. By this, we end up with a set of classes

$$\mathcal{C} = \begin{cases} \{\text{segmentation error, obstacle, false positive}\} & \text{or} \\ \{\text{segmentation high, segmentation low, obstacle. false positive}\}. \end{cases}$$

The actual classification is carried out on the histograms of detected outliers using a weighted k nearest neighbour approach. Given a set of labelled outliers $\{(O_i, C(O_i))\}$ we compute the corresponding histograms $H(O_i)$ for every O_i such that our training set consists of labelled histograms $\mathcal{H}_T = \{(H(O_i), C(O_i)\}$. Now, in order to classify a new outlier Q, we also compute its histogram $H(Q)$ and, under the same histogram distance function as used in the previous steps, we compute the distances between

$H(Q)$ and all of the $H(O_i)$ resulting in a set of tuples

$$D = \{(d_i, C(O_i)) \,|\, d_i = \text{dist}(H(Q), H(O_i)) \wedge (H(O_i), C(O_i)) \in \mathcal{H}_T\}.$$

Then, for every class $C \in \mathcal{C}$, we compute the class weight

$$w_C = \sum_{C(O_i)=C} \frac{1}{d_i + \varepsilon}$$

for $(d_i, C(O_i)) \in D$ and assign Q to the class $C(Q)$ with the maximum class weight w_C, i.e.

$$C(Q) = \underset{c \in C}{\text{argmax}}\{w_C\}.$$

As mentioned before, if $C(Q) =$ false positive, we do not report Q to the silhouette refinement step.

4.4 Multi-Reference Enhancement

In Section 4.3.2 the reference histogram creation has been laid out very briefly: A set of silhouettes, that contain no errors, is chosen by human annotators. We call this set the reference silhouettes. For the basic algorithm, we just compute a histogram for every single reference silhouette and then take the bin-wise average of these histograms. We call this histogram the reference histogram H_{ref}.

Based on this reference histogram, we execute the anomaly score computation as described in Section 4.3.2 on the reference silhouette in order to compute the mean μ and standard deviation σ of the anomaly score distribution. These values are then used to set the anomaly thresholds

$$\tau_w = \sigma + t_w \mu$$

and

$$\tau_s = \sigma + t_s \mu$$

with $t_w \leq t_s$.

Obviously, this approach uses a strong aggregation of the reference silhouettes by representing all of the reference silhouettes as a single histogram. On the one hand this makes the anomaly score computations very fast and straight forward, but on the other hand this aggregation leads to a high standard deviation thus obscuring anomalies. The problem here is, that the combined histogram H_{ref} represents an average of all reference silhouettes. However, histograms of single reference silhouettes or even of parts of a single reference silhouette can be quite different from this average histogram. As a result, the distance of such an histogram to H_{ref} can be large.

A solution to this problem is to use a weaker aggregation than before. This can be achieved by using multiple reference histograms instead of only one. An intuitive solution for this is the usage of one reference histogram per silhouette. However, when using more than one reference histogram, we cannot just compute the distance between the histogram of a query silhouette window and the reference histogram any more. Instead, we use

$$\text{dist}_{mr}(H, \mathcal{H}_{ref}) = \min\{\text{dist}(H, H_{ref}) | H_{ref} \in \mathcal{H}_{ref}\}$$

with $\mathcal{H}_{ref}$ the set of all reference histograms. By using the minimum distance to any of the reference histograms, we ensure that we use the distance to the most similar silhouette and therefore the distances of histograms containing true anomalies should differ more strongly from those that do not contain anomalies.

One disadvantage of using one reference histogram per reference silhouette is the fact, that the distance computation during the outlier search grows with the number of reference silhouettes. With a growing number of reference silhouettes this can hamper the performance. Also, some mountain silhouettes resemble each other, so an aggregation of those can even be beneficial, by cancelling out small irregularities in the single silhouettes. In order to overcome both the problem of using a fixed number of reference silhouettes as well as making use of the strengths of a local aggregation we use the k means clustering approach [Mac67] in order to group the reference histograms.

Let $\mathcal{P} = \{P^1_{ref}, \ldots, P^n_{ref}\}$ be the set of reference silhouettes. For every P^i_{ref} in $\mathcal{P}$ we compute the corresponding reference histogram HP^i_{ref} resulting in a set $\mathcal{HP}_{ref} = \{HP^1_{ref}, \ldots, HP^n_{ref}\}$ that we then use as the input data set for the k means algorithm. Since the data set consists of histograms, we use the above average distance as distance function for the clustering. For the computation of the cluster representatives we use centroids by taking the bin-wise average of all histograms in the cluster, which results in a normalised histogram. As usually, the clustering stops, if the variance cannot be minimised any more by changing the cluster assignments.

By this, we get k centroids as a result which we then use a reference histograms. Since the initial assignment of the cluster centroids is random, we start the clustering process multiple times for every k and then choose the best clustering based on the smallest sum of quadratic distance data points to their respective cluster centroids.

As the results of the evaluation in Chapter 6.3 show, by this approach we get better outlier detection results without using more reference silhouettes or even information about reference silhouettes, since the complete reference histogram computation process is unsupervised except for the choice of k, the number of resulting reference histograms. By this, even for a varying number of reference silhouettes, we manage to keep the outlier detection process in linear time complexity.

4.5 Outlier-type Distinction

The previous sections described an approach to outlier detection in silhouettes that treats the silhouettes as generic polygonal chains. That means, that we only use the positional properties of the points that make up the silhouette in order to detect outliers. In this section, we introduce some additional image specific properties in order to enhance the outlier detection. We also point out, that different mechanisms are better suited to detect different kinds of outliers.

As Hawkins noted in [Haw80], an outlier is an observation which derivates so much from the other observations as to arouse suspicion that it was generated by a different mechanism. In the case of mountain silhouettes, we do however have a notion, what causes outliers. In fact, we can divide outliers into two classes, namely *obstacles* and *segmentation errors*, that have different properties. Segmentation errors hereby are outliers that are caused by the segmentation algorithm not being able to follow the correct silhouette. This can be caused by low contrast in certain regions of the image, where the global set of parameters for the segmentation algorithm is not working.

Figure 4.5: Different errors in a silhouette.

Obstacles, in contrast, are objects in the image, that are between the camera and the mountain, such as trees, persons, buildings or cable cars. In this cases, the segmentation algorithm is usually able to follow the borders between the obstacles and the sky, thus working correctly from an intrinsic perspective. However, these obstacles usually have silhouettes that do not resemble mountain silhouettes very much. Figure 4.5 shows examples of different kinds of outliers. The trees on the left of the silhouette are obstacles, whose borders with the sky gets correctly detected. Since they are not part of the actual mountain silhouette, we treat them as outliers. The cut in the right below the lower peak is a segmentation error caused by low contrast between the blue of the sky and the grey of the rock at this part of the mountain.

Following those arguments, there are two methods to introduce additional properties to the outlier detection. First, we can add the additional information to the outlier detection approach as described before. By this, we get five dimensional histograms instead of histograms with two dimensions, by adding the contrast in x direction, the contrast in y direction and the gradient direction of the contrast values. Here, the latter describes the direction in which an edge determined by the directional contrast values is supposed to go. Second, we can split the attributes in two sets of attributes, for example the relative coordinates in one set and the contrast based attributes in another set and execute two separate outlier detection steps.

Both of these approaches have advantages and drawbacks. The first approach is obviously easier to implement since we only need to add three extra dimensions. Since all our methods work regardless of the number of dimensions this can be done very easily. However, adding dimensions to a histograms enlarges the number of bins in the histogram exponentially for a fixed number of values per dimension. If we use five bins per dimension, instead of $5^2 = 25$ bins, we end up with $5^5 = 3125$ bins. While this is not a problem in respect to computational resources, we have to remember, that for the creation of live histograms we use sliding windows that are rather short, i.e. in a range of three to ten data points. This leads to very sparse live histograms, that have very few bins that have values greater or above the average which in turn leads to large distances under the above average distance to the reference histograms that are denser because they consist of many more data points.

The second approach does not have the sparsity problem like the first one. For the positional attribute set, i.e. the relative coordinates, the density of the histograms is exactly on the level of the basic approach, while the contrast set consists of histograms of $5^3 = 125$ bins. However, since in this case two independent outlier detection runs are

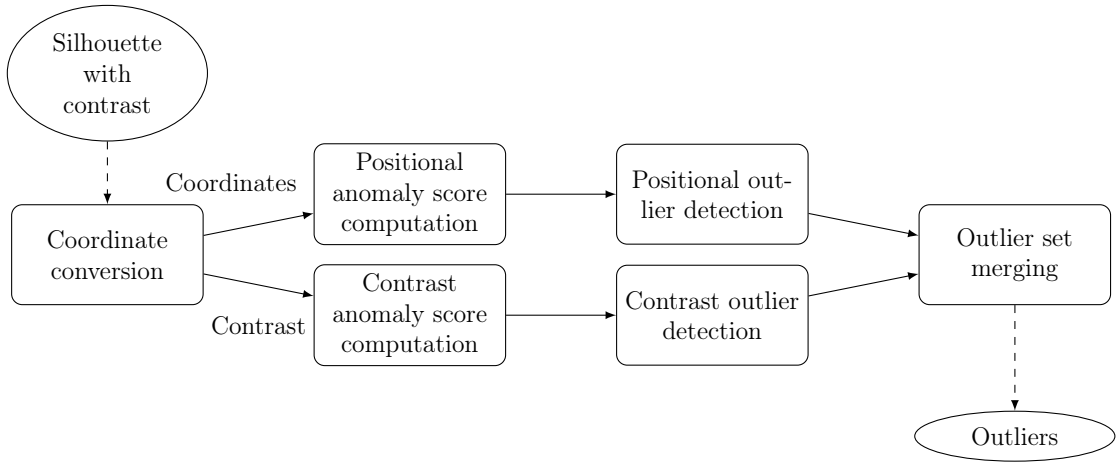


Figure 4.6: Flow diagram of the outlier detection pipeline with type distinction.

executed, that yield an independent set of outliers, each, we have to have a solution for merging these outlier sets. A further advantage of the second approach is, that it is able to give a hint of the type of the detected outliers based on the outlier detection instance that reported the outlier. This can be used for a better adaption of the segmentation process.

Regarding the properties of both approaches, we decided to go with the second approach, since it has the intrinsic outlier classification as advantage and the negative properties, i.e. the merging of two sets of outliers, can be handled easier than the histogram sparseness of the first approach. In order to implement an outlier classification to the other variants of the outlier detection, one has to rely on classic classification techniques, such as k nearest neighbours classification [Alt92] or decision trees [Qui86]. Such an approach is hampered by the low number of outlier examples, which also makes more elaborate techniques like artificial neural networks [Ros61] or support vector machines [VC64, BGV92, CV95] unfeasible.

Figure 4.6 gives an overview of the architecture of the outlier detection framework that utilises two distinct outlier detection techniques. Note here, that, in theory, further outlier detection algorithms could be introduced. This can be sensible in other domains where a greater number of causes for outliers is possible. In contrast to the basic outlier detection approach as depicted in Figure 4.3, we now expect the contrast information to be passed additionally. The silhouette or coordinate conversion, as it is called in Figure 4.6, is the same as before, since the contrast features do not have to be converted to a relative representation. In contrast to the coordinates, this information does not depend on the information of previous points in order to be meaningful. Then, the relevant information is passed to the corresponding anomaly score computation method. Apart from using a different set of attributes, the actual anomaly score computation works exactly as described in Section 4.3.2. The same is true for the actual outlier detection steps, that work as described in Section 4.3.3 and can be combined with the multi-reference enhancement from Section 4.4. This means, the main difference to the previous approaches lies in the outlier set merging step, that was not necessary before since we worked with one set of detected outliers, only.

In contrast to the approaches with a single outlier detection, we now get multiple

sets of outliers. Considering the architecture from Figure 4.6, we obtain one set of outliers from the positional outlier detection and another set from the contrast outlier detection. Both sets contain the set of maximum l outliers for the extended relative silhouette in respect to the different anomaly scores. However, two or more outliers from the different sets can overlap with one another, due to the different anomaly scores, as in the following definition.

Definition 15 (Overlapping Outliers)**.** We say that two outliers O_1, O_2 overlap if

$$O_1 \cap O_2 \neq \emptyset,$$

i.e. if there are points in the silhouette P that are both part of O_1 and O_2.

Let $\mathcal{O}_1, \ldots, \mathcal{O}_n$ be sets of outliers returned by different outlier detection approaches. Then we say that a set

$$\mathcal{O}_{over} \subseteq \bigcup_{i=1}^{n} \mathcal{O}_i$$

overlaps, if for all $O, Q \in \mathcal{O}_{over}$ one of the following constraints holds:

1. O and Q overlap.
2. There exist $O_1, \ldots, O_k \in \mathcal{O}_{over}$ such that O and O_1 overlap, O_j and O_{j+1} overlap for $j \in [1, k-1]$, and O_k and Q overlap.

If we have such a set of overlapping outliers, we need to dissolve the overlaps in order to support the adaptive segmentation method by giving a single outlier type per outlier. This is due to the fact, that outlier removal strategies on part of the segmentation are different. In order to solve this problem, three splitting strategies have been developed and are introduced in the following. Since all of these strategies need information on the strong anomalies in an outlier, in the following let

$$I(O) = \{o \in O | \text{anom}(o) \geq \tau_s\}$$

be the set of strong anomalies included in an outlier O and let $C(O)$ be the class of the outlier O determined by the algorithm that reported O, i.e. one of "positional" or "contrast".

The first strategy we developed is called *Merge*. The basic idea of this approach is to merge all outliers in an overlapping set $\mathcal{O}_{over}$ into a single outlier and to determine that outlier's class by a majority voting of the strong anomalies of each class. Formally, let $O = (p_s, \ldots, p_e)$ an outlier. Then let $\text{indmin}(O) = \min\{i \,|\, p_i \in O\}$ and $\text{indmax}(O) = \max\{i \,|\, p_i \in O\}$ be the minimum and maximum index in that outlier, respectively. For a set $\mathcal{O}_{over}$ of overlapping outliers, let

$$\text{pmin}(\mathcal{O}_{over}) = p_{mins} \text{ with } mins = \min\{\text{indmin}(O) \,|\, O \in \mathcal{O}_{over}\}$$

be the point with the smallest index in $\mathcal{O}_{over}$ and

$$\text{pmax}(\mathcal{O}_{over}) = p_{maxs} \text{ with } maxs = \max\{\text{indmax}(O) \,|\, O \in \mathcal{O}_{over}\}$$

be the point with the largest index in $\mathcal{O}_{over}$.

Now, we are able to merge a set of overlapping outliers $\mathcal{O}_{over}$ into a single outlier $O_{merge} = (\min(\mathcal{O}_{over}), \ldots, \max(\mathcal{O}_{over}))$. In respect to the class of O_{merge}, let $\mathrm{Ic}(\mathcal{O}_{over}, c) = \{p \in I(O) \,|\, C(O) = c \wedge O \in \mathcal{O}_{over}\}$. Note here, that outliers of a single class c do not overlap, since every outlier detection method returns a set of maximum l outliers. We then set the class of O_{merge}

$$C(O_{merge}) = \underset{c \in C}{\operatorname{argmax}}\{|\,\mathrm{Ic}(\mathcal{O}_{over})|\},$$

for the set C of possible classes.

The reasoning behind this strategy is, that corrections of one outlier often influence the further form of the silhouette. Trying to correct one large outlier might be beneficial, because a larger portion of the silhouette is taken into account.

The second strategy is called *Merge to Contrast*[1]. As the name suggests, the overlapping outliers set is merged in the same way as with the Merge strategy, however in this case, if at least one outlier is of the contrast class, the merge outlier is given the contrast class as well. Note here, that with using two strategies as depicted in Figure 4.6, all overlapping outlier sets that contain at least two outlier have to contain one outlier of each class.

This strategy is based on the fact, that the positional algorithm is also able to find segmentation errors, as the evaluation results of the basic approach show, so treading all outliers found by this method as obstacles does not seem sensible. This is especially the case, because obstacles are removed by replacing them by a straight line while segmentation errors get resegmented and then rechecked for outliers. Thus classifying an outlier as obstacle is a more definite choice.

The last strategy is called *Split and Merge*. The idea here is to separate outliers, whose overlap only consists of weak anomalies since this might be a sign, that actually multiple outliers exist but are connected because of their proximity. This kind of overlap is formalised in the following definitions.

Definition 16 (Strongly Overlapping Outliers)**.** Two outliers O, Q *overlap strongly*, if

$$I(O) \cap I(Q) \neq \emptyset.$$

A set $\mathcal{O}_{sover}$ is called strongly overlapping if for all $O, Q \in \mathcal{O}_{sover}$ one of the following constraints holds:

1. O and Q overlap strongly.
2. There exist $O_1, \ldots, O_k \in \mathcal{O}_{over}$ such that O and O_1 overlap strongly, O_j and O_{j+1} overlap strongly for $j \in [1, k-1]$, and O_k and Q overlap strongly.

Given the strongly overlapping outlier definitions we can now define weakly overlapping outliers.

Definition 17 (Weakly Overlapping Outliers)**.** Two outliers O, Q *overlap weakly*, if O, Q overlap but do not strongly overlap, i.e.

$$O \cap Q \neq \emptyset \wedge I(O) \cap I(Q) = \emptyset.$$

[1] In [SKBC17] this strategy is called Merge to Segmentation, because outliers with low contrast are mostly due to segmentation errors.

Now, for a set $\mathcal{O}_{over}$ of overlapping outliers, we search all maximum sets of strongly overlapping outliers. By a maximum set of strongly overlapping outliers, we mean a set $\mathcal{O}_{sover} \subseteq \mathcal{O}_{over}$ such that for all $O \in \mathcal{O}_{over} \setminus \mathcal{O}_{sover}$, $\mathcal{O}_{sover} \cup \{O\}$ is *not* a set of strongly overlapping outliers. We can now use either the Merge or Merge to Contrast strategy on each of the maximum sets of strongly overlapping outliers. By this we get a set $\mathcal{O}_{wover}$ of outliers, where for each pair $O, Q \in \mathcal{O}_{wover}$ either O and Q overlap weakly or O and Q do not overlap at all.

We now order the outliers in $\mathcal{O}_{wover}$ by the smallest index of a strong anomaly of each outlier in ascending order, such that $\mathcal{O}_{wover} = \{\tilde{O}_1, \ldots, \tilde{O}_m\}$ with

$$\text{indmin}(I(\tilde{O}_1)) < \text{indmin}(I(\tilde{O}_2)) < \ldots < \text{indmin}(I(\tilde{O}_m)).$$

Since the outliers in $\mathcal{O}_{wover}$ cannot be strongly overlapping, this ordering is distinct.

Now, for $i = 1, \ldots, m-1$, $j = 1+1, \ldots, m$ we inspect the outliers $\tilde{O}_i = (p_{s_i}, \ldots, p_{e_i})$ and $\tilde{O}_j = (p_{s_j}, \ldots, p_{e_j})$. If the outliers do not overlap, we have to do nothing. If $\tilde{O}_i$ and $\tilde{O}_j$ overlap, we know that they can only overlap weakly. We next compute the middle between the two outliers inner parts as

$$m = \left\lfloor \frac{1}{2} \left(\text{indmax}(I(\tilde{O}_i)) + \text{indmin}(I(\tilde{O}_j)) \right) \right\rfloor$$

and the practical middle index as

$$\tilde{m} = \underset{i \in [1,\ldots,n]}{\text{argmin}} \left\{ |i - m| \;\middle|\; p_i \in \tilde{O}_i \cap \tilde{O}_j \right\}.$$

We can now set $\tilde{O}_i = (p_{s_i}, \ldots, p_{\tilde{m}})$ and $\tilde{O}_j = (p_{\tilde{m}+1}, \ldots, p_{e_j})$. By this, we transform $\mathcal{O}_{wover}$ to a set of outliers, that do not overlap pairwise.

Finally, we sort the outliers in $\mathcal{O}_{wover}$ again, this time by the smallest index in all of the outlier and by this we get $\mathcal{O}_{wover} = \{\bar{O}_1, \ldots \bar{O}_m\}$ with

$$\text{indmin}(\bar{O}_1) < \ldots < \text{indmin}(\bar{O}_m).$$

For $i = 1, \ldots, m-1$, if $C(\bar{O}_i) = C(\bar{O}_{i+1}$ and $\text{indmax}(\bar{O}_i) + 1 = \text{indmin}(\bar{(O)}_{i+1})$, i.e. if $\bar{O}_i$ and $\bar{O}_{i+1}$ are outliers of the same class that are direct neighbours, we merge those two outliers.

4.6 Generalisation of the Algorithm

The previous parts of this chapter described an outlier detection algorithm that focuses on the detection of outliers in mountain silhouettes. In the remainder we will call this algorithm and its variants AdaMS outlier detection or AdaMS OD for short. However, almost all parts of that algorithm and its variants can be used to detect outliers in all kind of sequences. In this case, though, the general understanding of what an outlier is has to be similar, i.e. an outlier still has to be a subsequence of unknown length greater than a minimum length l whose properties deviate from the norm.

While l and window length w can be set to 1 and thus single points can be found as outliers, our algorithm cannot, for example, explicitly find change points, which are another view of being outliers on sequences, especially in time series. A change point

is a point, at which the behaviour of a sequence changes, but then stays stable for a longer period of time in that behaviour. While it is possible for AdaMS OD to find such a change point as an outlier, it is not specialised on this type of outliers and thus other outliers will most probably found as well.

Also, the AdaMS outlier detection does not support finding outliers of a fixed length. Due to this, one can argue that it is not as well suited to outlier detection in cyclic time series, such as outputs of electrocardiography, as specialised algorithm like those of the HOT SAX family [KLF04, KLF05, PLD10]. If the length of the cycle and therefore the outliers has to be known however, one could as well declare a cycle an outlier, if AdaMS OD finds an outlier in it. One of the main advantages of AdaMS OD over most time series based outlier detection algorithms is the fact, that it is designed from scratch to support an arbitrary number of features in one sequence. By this, it is able to detect outliers that do seem normal if one looks at the different features as separate time series.

In general, the foremost question to answer when trying to use AdaMS OD as an algorithm is, whether outlier free training or reference data is available. If this is the case, AdaMS OD with all of its variants can be applied to the problem. If, however, no training data is available, the whole sequence that is to be inspected can be used as reference data. In this case, we would advise to use a single reference histogram, since this minimises the probability for a reference histogram consisting of abnormal parts, only, and it also minimises the influence of outliers on the whole of the reference data.

Lets say, $S = (v_1, \ldots, v_n)$ is a sequence that we want to examine and there is no reference data available. Then, in order to use the multi-reference variant, we had to split S in $s \geq k$ parts if we wanted to use k reference histograms where each part has an average length of $\frac{n}{s}$. In fact, since we have no knowledge about the properties of S, it is sensible to make the lengths of the parts as equal as possible. In this case, the larger s becomes, the smaller the length $\frac{n}{s}$ of the reference sequences becomes and thus the larger the probability that one or multiple of these sequences consist of outliers to a large part or even exclusively gets.

The outlier-type distinction presented in Section 4.5 can be used whenever there is knowledge of different sources of outliers that affect different characteristics. It is noteworthy here, that multiple variants do not have to work on different features. It is also possible to use varying parameters on the single instances of the outlier detection. This can be useful, if for example there is knowledge about very short outliers, that deviate greatly from normal data on one the hand and on the other hand outliers that are very long but otherwise not as remarkable as the short ones exist.

5

Outlier Detection with Neural Networks

The previous chapter concentrated on partly unsupervised techniques for outlier detection. While the properties of outlier-free data is learned from reference silhouettes, that have been selected by humans, and some parameters have to be selected, the definition of outliers itself and their behaviour is based on assumptions and not on training data.

On the one hand, artificial neural networks, especially since the rise of multi-layered deep learning architectures [HOT06, BLPL07, BL+07, PCC+07], yield great results in many knowledge discovery and information retrieval tasks, such as classification [CMS12, KSH12], image segmentation [GDDM14, Gir15, LSD15, RHGS15], and many natural language processing problems [BDVJ03, MSC+13, LM14]. On the other hand, in all these cases there are samples of all classes that are involved or, if this is not the case, results suffer.

When trying to adopt neural network solutions to outlier detection, the main problem arises from the fact, that in many cases, there are not many or in many cases even no examples of outliers available for training. One possible solution for this is to artificially create outliers. However, as in Chapter 4, we only have a set of outlier-free silhouettes, such that we have to make assumptions on the nature of outliers in order to simulate them based on that training data. Once we have created an appropriate amount of training samples, we can then train a neural network on both the outlier and the outlier-free data in order to be able to classify parts of the silhouette. In this case, we will use convolutional neural networks since these show good results in the classification of images.

The remainder of this chapter is structured as follows: Section 5.1 describes the process of the training data creation of both normal and anomalous data from the reference silhouettes and corresponding images. Section 5.2 introduces the network architecture used for the classification task and give a short overview over the employed methods.

5.1 Generation of Training Data

In order to train a convolutional neural network we need to have training data for all classes. In case of an outlier detection problem, however, we only have training data for outlier free silhouettes. Therefore, in order to be able to train the network, we have to artificially create outliers.

The basis for this is a set of reference images $\mathcal{I} = \{I^1_{ref}, \dots, I^m_{ref}\}$ and their corresponding outlier-free reference silhouettes $\mathcal{P} = \{P^1_{ref}, \dots, P^m_{ref}\}$. In this context, let $P^i_{ref} \in \mathcal{P}$ be the reference silhouette of the image $I^i_{ref} \in \mathcal{I}$. In order to have a balanced training set, our goal is to create one anomalous silhouette for every outlier-free silhouette. This means, we want to create a set of silhouettes $\overline{\mathcal{P}} = \{P^1_{out}, \dots, P^m_{out}\}$ where $P^i_{out} \in \overline{\mathcal{P}}$ is an anomalous silhouette for the image $I^i_{ref} \in \mathcal{I}$.

Furthermore, as we discussed in the previous chapter, outliers are essentially deviations from the correct silhouette. Therefore, our goal is to create a silhouette, that only consists of outliers. In order to do so, we create an anomalous silhouette in the following way. Let $P_{ref} = (v_1, \dots, v_n)$ with $v_i = (x_i, y_i) \in P_{ref}$ be a reference silhouette. Then we create the anomalous silhouette $P_{out} = (w_1, \dots, w_n)$ by setting

$$w_1 = (x_1, y_1 + \rho_1)$$

where $\rho_1 \in \{-1, 1\}$ is a random number. Let $o_1 = \rho_1$. We then recursively set

$$w_i = (x_i, y_i + o_i)$$

with

$$o_i = \begin{cases} \rho_i, & \text{if } o_{i-1} + \rho_i = 0 \\ o_{i-1} + \rho_i & \text{else.} \end{cases}$$

Here, like ρ_1, all $\rho_i \in \{-1, 1\}$ are random numbers as well.

By this, we create a silhouette P_{out} where nearly no point $w_i \in P_{out}$ is also a point in the corresponding reference silhouette P_{ref}. In fact, the only case when this can happen, is, if there exist $v_i, v_j \in P_{ref}$ with $x_i = x_j$ and we coincidentally choose either $o_i = y_j - y_i$ or $o_j = y_i - y_j$. Thus, P_{out} only consists of outliers, apart from highly unlikely cases.

Having generated the silhouettes alone is not sufficient in order to get training data because we do not want to classify whole silhouettes but parts of silhouettes in order to find outliers. At the same time, we do not want to preprocess the data to much. Preprocessing would induce certain assumptions about the importance of features in respect to the classification of parts of the silhouettes. However, in this we want to explore the feature extraction qualities and as such, keep human assumptions about features and therefore the amount of preprocessing as low as possible.

Again, as in the previous chapter, we use a sliding window approach with a fixed window length over the points of the silhouette. For each window of a silhouette we create an image patch out of the corresponding image that shows the silhouette pixels vertically centred. Therefore, let $P = (v_1, \dots, v_n)$ be a silhouette for the image I. For this, P can either be a reference silhouette, a generated anomalous silhouette, or, in case of the actual outlier detection after the training is finished, an extracted silhouette.

In the following, let w_p be the width of the image patches in pixels and let h_p be half the height of the image patches. In order to create the i-th image patch I^P_i in

Figure 5.1: Image on the top with first image patches below.

respect to the silhouette P, let $p_i = (x_i, y_i), \ldots, p_{i+w_p-1} = (x_{i+w_p-1}, y_{i+w_p-1}) \in P$ be the w_p silhouette pixels that contribute to I_i^P. We then construct the image patch as

$$I_i^P = \begin{pmatrix} I(x_i, y_i - h_p) & \cdots & I(x_{i+w_p-1}, y_{i+w_p-1} - h_p) \\ \vdots & & \vdots \\ I(x_i, y_i) & \cdots & I(x_{i+w_p-1}, y_{i+w_p-1}) \\ \vdots & & \vdots \\ I(x_i, y_i + h_p - 1) & \cdots & I(x_{i+w_p-1}, y_{i+w_p-1} + h_p - 1) \end{pmatrix} \in \mathcal{C}^{w_p \times 2h_p}.$$

By this, the pixels that belong to the silhouette, form the h_p-th row of I_i^P and their horizontal neighbourhood is shown in the same column as that pixel. It is worth mentioning here, that I_i^P is not a rectangle taken out of I. A pixel $p \in I$ can be included multiple times in I_i^P if the silhouette P contains multiple points that have the same x-coordinate and are close enough together. Figure 5.1 illustrates this behaviour. Here, the area marked in red is the area covered by the first 16 image patches that are shown below the silhouette.

For the training data, we know that all patches created from reference silhouettes do not contain outliers and therefore can be labelled as normal data and all patches created from anomalous data contain anomalies only. Thus, those get labelled as outliers.

5.2 Network Architecture

With the training data computed in the previous section, we are now able to train a supervised classifier such as a *Convolutional Neural Network* (CNN). CNNs are multilayered or deep artificial neural networks that are often used for image classification tasks, as first described in [LBD+90] for the identification of handwritten digits. Essentially, they consist of three different kinds of layers: convolutional layers, that give the whole architecture their name, pooling layers, and fully connected or dense layers.

Layers of neural networks are essential matrix operations. In the following we briefly explain the three basic layers of CNNs. Convolutional and pooling layers take

the dimensionality of the input into account. Here, we will only introduce them in their two dimensional cases. This is because we only use two dimensional images. However, variants of these layers in other dimensions exist and work basically analogue.

Let $x \in \mathbb{R}^n$ be the input data. Then, a *fully connected layer* is a function

$$f : \mathbb{R}^n \to \mathbb{R}^m \text{ with } x \mapsto Wx + b,$$

with $W \in \mathbb{R}^{n \times m}$ the weight matrix and $b \in \mathbb{R}^m$ the bias vector. The values in W and b are usually initialised randomly and then trained. Training is normally done by a stochastic gradient descent or a variation such as the adam optimiser [KB14].

Obviously, a fully connected layer is linear and as such not useful in many scenarios. Therefore, similarly to the usage of the kernel trick [Aiz64, BGV92] for support vector machines [CV95], non-linear activation functions are combined with layers. An activation function is usually an element wise function $\phi : \mathbb{R} \to \mathbb{R}$ that is applied to the output of a layer of an artificial neural network. For sake of convenience, for a vector $y = (y_1, \dots, y_m)^\top \in \mathbb{R}^m$ let us define

$$\phi(y) = \begin{pmatrix} \phi(y_1) \\ \vdots \\ \phi(y_m) \end{pmatrix}$$

and likewise, for a matrix

$$A = \begin{pmatrix} a_{1,1} & \cdots & a_{n,1} \\ \vdots & \ddots & \vdots \\ a_{1,m} & \cdots & a_{n,m} \end{pmatrix} \in \mathbb{R}^{n \times m}$$

let

$$\phi(A) = \begin{pmatrix} \phi(a_{1,1}) & \cdots & \phi(a_{n,1}) \\ \vdots & \ddots & \vdots \\ \phi(a_{1,m}) & \cdots & \phi(a_{n,m}) \end{pmatrix}.$$

By this, if we use a non-linear activation function ϕ together with a fully connected layer f, we get a non-linear function

$$f_\phi(x) = \phi(Wx + y).$$

Often used activation functions are rectified linear units (ReLU) [NH10], where

$$\phi(x) = \max\{0, x\}$$

or the logistic sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Convolutional layers are based on the mathematical convolution operation in the same way that fully connected layers are based on linear functions. In general, for two functions $f, g : \mathbb{R}^n \to \mathbb{C}$ the convolution of f and g is given by

$$(f * g)(x) = \int_{\mathbb{R}^n} f(\tau) g(x - \tau) d\tau$$

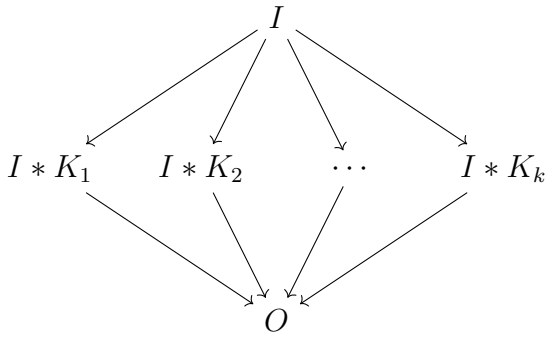


Figure 5.2: Parallel execution of multiple convolutions in a convolutional layer.

for a well-defined integral. In computer science, data is often discrete. For discrete functions $f, g \in Z \subseteq \mathbb{Z} \to \mathbb{C}$ the discrete convolution is defined as

$$(f * g)(x) = \sum_{k \in Z} f(k)g(x - k).$$

The convolution $f * g$ can be interpreted as the weighted mean of f where g is the weight function. Due to the fact, that g is parametrised by x, the weight is dependant on x.

Convolutions are often used in the image processing domain in the form of a *kernel*, *filter*, or *mask* in applications like edge detection [Rob63, Sob90, Pre70, Can86, Sch00] or in smoothing [HA91]. In such a case, let $I \in \mathcal{C}^{w \times h}$ be an image. Also, let $K \in \mathcal{C}^{w_K \times h_K}$ for odd $w_K, h_K \in \mathbb{N}$ be the kernel matrix and let

$$(a_w, a_h) = \left(\left\lceil \frac{w_K}{2} \right\rceil, \left\lceil \frac{h_K}{2} \right\rceil \right)$$

be the centre coordinates of K. Then the filtered image $(I * K) \in \mathbb{R}^{w \times h}$ is given by

$$(I * K)(x, y) = \sum_{i=1}^{w_K} \sum_{i=1}^{h_K} I(x - a_w + i, y - a_h + j) \cdot K(i, j).$$

Note here, that in case of matrices with more than one channel, the product in this sum is the dot product, such that the resulting matrix has only one channel. Also, the formula above uses indices that are not in I. In order to solve this problem, I has to be padded. Usually this is done by treating the values outside of I as zeros, but other strategies such as padding with the closest existing value exist, too.

Convolutional layers use adaptive masks. That means, that the single weights in the kernel matrix or in multiple matrices, if more than one filter is used, are learned during the training. In case of more than one filter, the convolutions are executed in parallel as depicted in Figure 5.2. For a convolutional layer with k filters $K_1, \ldots, K_k$ and an input $I \in \mathcal{C}^{w \times h}$, we independently compute the convolutions $I * K_1, \ldots I * K_k \in \mathbb{R}^{w \times h}$. Finally, we join these into a single matrix $O \in (\mathbb{R}^k)^{w \times h}$ where each entry

$$O(i, j) = ((I * K_1)(i, j), \ldots, (I * K_k)(i_j)) \in \mathbb{R}^k$$

has k channels.

In the same manner as in the fully connected layers, a bias can be applied to a convolutional layer c^k with k kernels. In this case, a bias matrix B with the same dimensions as O is used, so that $c^k(I) = O + B$. Also, an activation function ϕ is used on the output of a convolutional layer, resulting in

$$c^k_\phi(I) = \phi(O + B)$$

as output of the layer. Single channels of the output of $c^k_\phi(I)$ are often referred to as feature maps.

The weights in both the kernels and the bias matrix are usually randomly initialised. Thus, the different kernels are able to retain different values due to different initialisations.

The last kind of layers, that are regularly used in CNNs are *pooling layers* [LBOM98]. In contrast to fully connected and convolutional layers, pooling layers are static in the sense, that they are not trained. As a result of this, they are not used in combination with activation functions as well. The idea of pooling layers is to aggregate parts of the input, i.e. pools, in order to amplify important characteristics.

Let $I \in \mathbb{R}^{w_I \times h_I}$ be the input and let $w_P \times h_P$ be the pool size. Then we get an output $O \in \mathbb{R}^{w_O \times h_O}$ where

$$w_O = \left\lceil \frac{w_I}{w_P} \right\rceil \text{ and } h_O = \left\lceil \frac{h_I}{h_P} \right\rceil$$

with

$$O(i,j) = aggr\{I(x,y) \,|\, (i-1)w_P < x \leq iw_P \wedge (j-1)h_P < y \leq jh_P\},$$

where $aggr$ is an aggregation function such as $\max, \min$, or avg. As with convolutional layers, I gets padded if necessary. Obviously, the padding strategy depends on the aggregation function. While padding with zeros is sensible for max pooling, it is counterproductive for min pooling in many cases.

If the input of a pooling layer has more than one channel, the pooling operation is executed on single channels. Let us assume that the values of the input I have k channels, i.e. $I(i,j) \in \mathbb{R}^k$. Then we execute k pooling operations in parallel, by applying the aggregation function as an element-wise operation. For this, let $S \subseteq \mathbb{R}^k$ with $v = (v_1, \ldots, v_k)$ for all $v \in S$. Then we define

$$aggr\{S\} = (aggr\{v_1 \,|\, v \in S\}, \ldots, aggr\{v_k \,|\, v \in S\}) \in \mathbb{R}^k.$$

Figure 5.3 shows an example for a max pooling layer. In CNNs, max pooling layers are the most frequently used pooling layers. On the one hand this has to do with the fact, that often input data has values in $\mathbb{R}_{\geq 0}$. Thus, max pooling minimises the probability of getting zero values in the data. Too many zeros in the matrices should be avoided, since they are also used as null values. This is because, zeros are the absorbing element in respect to multiplications. On the other hand, max pooling highlights the strongest features in the input if this is a feature map output by a convolutional layer.

For the outlier detection we tested two approaches, that differ in depth. Both are loosely based on the very deep ConvNets architecture of the Oxford University's Visual Geometry Group's (VGG) presented in [SZ14]. The basic building blocks of

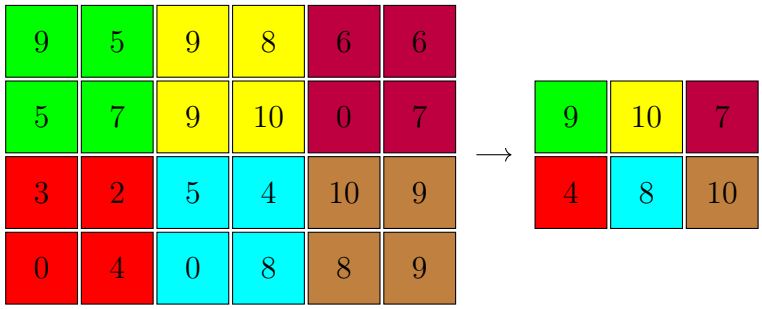


Figure 5.3: Max pooling example.

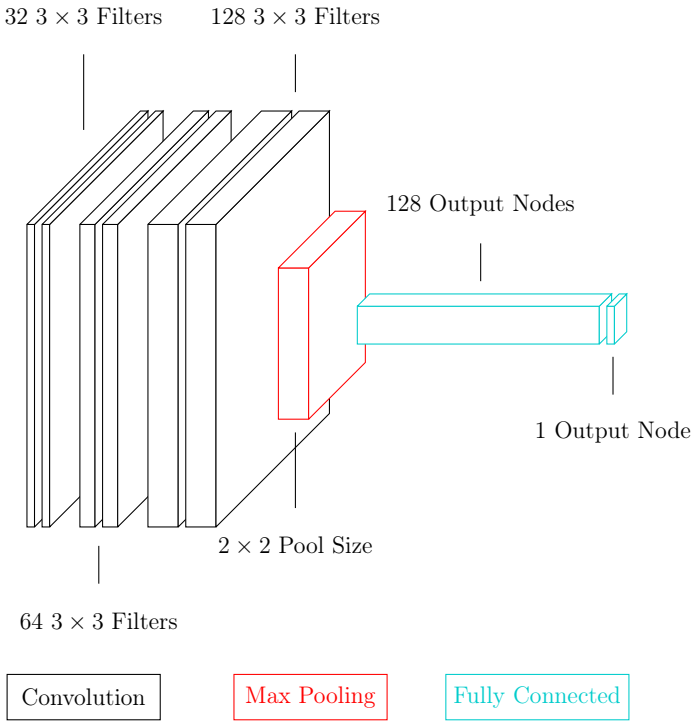


Figure 5.4: Smaller proposed network architecture.

these networks are blocks of two to four convolutional layers with the same number of 3×3 kernels that are followed by a 2×2 max pooling layer. The next such block doubles the number of filters of the previous block, while the sizes of the input data are halved in every direction. After a certain number of such blocks, three fully connected layers follow.

As can be seen in Figures 5.4 and 5.5, our network architectures follow the VGG architecture in the sense, that we use blocks of two convolutional layers with the same parameters. The next block then doubles the number of filters. However, we do not utilise a max pooling layer after each of the blocks, since our input is too small for this, with a width of 16 pixels and a height of 32 pixels. Thus, using more than one max pooling layer results in too strong an aggregation and thus meaningless results. Also, since we only have a two class problem, i.e. a single output, we reduce the number of fully connected layers to two.

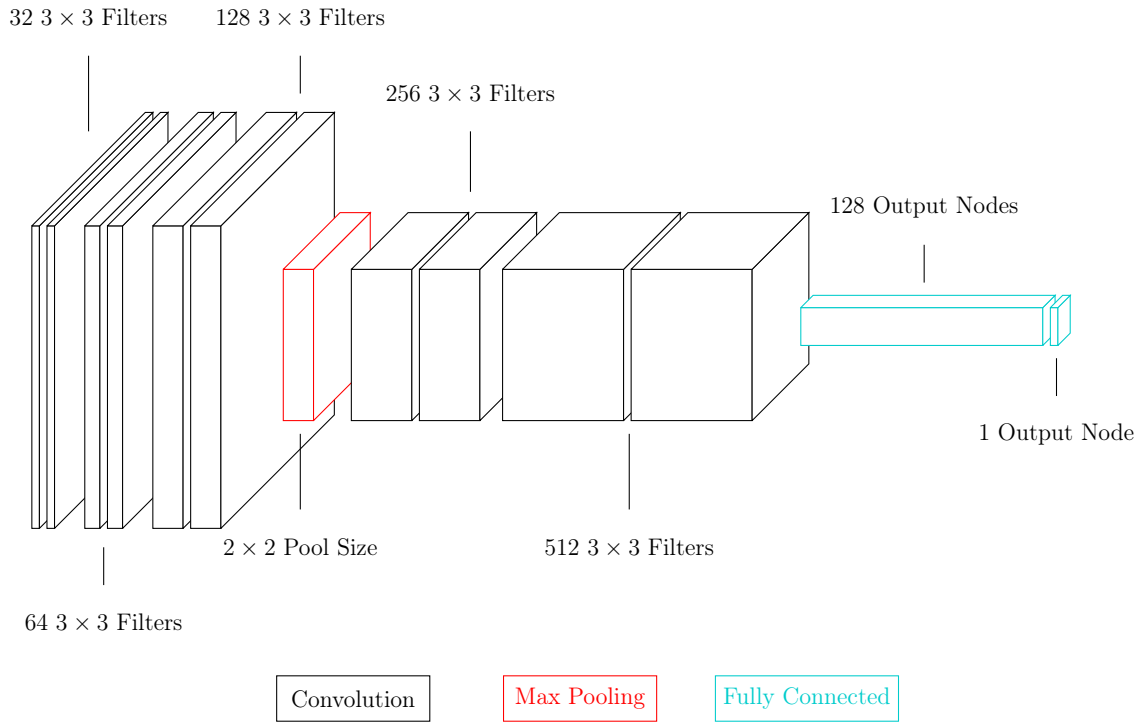


Figure 5.5: Larger proposed network architecture.

The smaller of our architectures is shown in Figure 5.4. As can be seen there, it consists of eight weight layers, namely six convolutional layers and the two fully connected layers at the end. The first two convolutional layers use 32 kernels, the second two use 64 kernels, and the final two convolutional layers utilise 128 kernels. The first of the fully connected layers has 128 output neurons, while the second fully connected layer reduces this to a single output. Before the output data from the max pooling layer can be used as input to the first convolutional layer, it has to be reshaped as a vector in a manner similar to the conversion of higher dimensional histograms into one dimensional histograms introduced in Section 4.3. Altogether, this network architecture has 2 097 409 trainable parameters.

Figure 5.5 shows a larger network architecture that we tested in order to investigate the effect of having more trainable parameters. Essentially this architecture is the direct result of adding four more convolutional layers after the max pooling layer from the smaller architecture. These layers are organised in two blocks, where the layers of the first block use 256 kernels, each, and those of the second block use 512 kernels, each. As a result of this, the input vector of the first fully connected layer now consists of 65 536 single values instead of 16 384 in the smaller net. Also, we have now a total of 8 388 865 training weights.

5.3 Network Output Interpretation

After the previous sections showed how training data for a neural network for outlier detection can be generated and what the used network architectures look like, we now discuss how the predictions the network outputs can be interpreted. We first briefly

describe the prediction data preparation and then explain the outputs of the network. Finally, we discuss how these outputs can be used in order to find the outliers.

In order to prepare an image with a given silhouette, we compute image patches of the silhouettes as introduced in Section 5.1. These have the same dimensions as those of the training data, i.e. they have a width of 16 pixels and a height of 32 pixels, with the silhouette being horizontally in the middle. As we have already explained, for every point in the silhouette that has 15 successors such an image patch is computed, thus implementing a sliding window approach similar to the algorithms presented in Chapter 4. Every image patch is the used as an input for the prediction function of the network. The prediction output for a single image patch I_i^P is a floating point number $\text{pred}(I_i^P) \in [0, 1]$ that can be interpreted as the probability of I_i^P being an outlier.

To summarise the procedure until now, given an image I and a silhouette $P = (v_1, \ldots, v_n)$ we first compute the images patches $(I_1^P, \ldots, I_{n-15}^P)$. We stop at I_{n-15}^P because every image patch has to have a width of 16 pixels, so it must contain 16 silhouette points. Then, we compute the prediction scores $(\text{pred}(I_1^P), \ldots, \text{pred}(I_{n-15}^P))$ by applying the network. At this point, we have a probability for each image patch being an outlier. However, as has been discussed in the previous chapters, outliers in silhouette can be of arbitrary length. Additionally, as each point $v \in P$, except for v_1 and v_n is part of more than one image patch, we have multiple probabilities for the single points of belonging to an outlier.

A way to solve both problems lies in computing a single outlier probability for each silhouette point v_i. The easiest way to do this is by computing a prediction value $\text{pred}(v_i)$ as an aggregate of the prediction scores of all images patches I_j^P that contain v_i, i.e.

$$\text{pred}(v_i) = \underset{j \in \{1,\ldots,n-15\} \cap \{i-15,\ldots,i\}}{\text{aggr}} \{I_j^P\}.$$

Possible aggregates aggr are the maximum, the minimum, or the average. Given these values we can compute the outliers by finding subsequences that consists of points v_i that have an outlier prediction score higher than a given threshold. Due to the training process described above, the natural threshold is $\frac{1}{2}$, as the training sets for outlier and normal data are balanced in size.

By choosing such an approach we have the advantage of a method with very few parameters. Apart from the meta-parameters of the network, the only real parameters are the width w_p and height parameter h_p of the image patches. If we set $w_p = h_p$ and thus the size of the image patches to $w_p \times 2w_p$ we can reduce this to a single parameter. Additionally this parameter has a lower boundary of two since otherwise the max pooling layer cannot be applied. Additionally, we have to choose w_p in a size that does not make the image patches too small to include visual information about the silhouette. It also has to be even and should not be chosen too large, in order to be able to recognise smaller outliers.

Another possible interpretation of the prediction scores $\text{pred}(v_i)$ is to treat them as anomaly scores as introduced Chapter 4. This approach on the one hand includes an additional training step as we have to compute the anomaly scores of all points in all of the training images in order to determine the mean and standard deviation of their probability distribution. On the other hand it also includes much more parameters than the direct interpretation of the prediction scores, because we have to chose all of the parameters introduced in the previous chapter, i.e. both threshold parameters t_s

and t_w, as well as the minimal inner outlier length l.

In comparison the first mentioned approach is much more usual in regards to the treatment of output by an artificial neural network in classification problems. In contrast to the algorithms in the previous chapter, the convolutional networks introduced here, have the advantage of being able to not only work on the features of the silhouette pixels themselves, but also on the vertical surroundings given by the height of the image patches. We therefore feel, that the advantages of fewer parameters that have to be tuned exceed the flexibility of the second approach. This is also due to the fact that the second approach was designed for a method that uses much less information of the surroundings on the one hand while being much more generalisable on the other hand. As described in Section 4.6, the algorithms presented in the previous chapter can be used for outlier detection on any sequence with little or no modifications, while the convolutional networks introduced in this chapter depend strictly on images as input.

6

Experiments with AdaMS Outlier Detection

In this chapter we evaluate the performance of the algorithms presented in the previous parts of the thesis. In order to do so, we have used two data sets. The first data set consists of photos by Michael Singhof, that have been manually segmented by Daniel Braun. We refer to this data set as *our data set* or *Test Set 1*. The second data set is the data set published with [BSKP12], that contains images of mountains in Switzerland. In the remainder of this chapter we refer to this data set as the *Switzerland data set* or *Test Set 2*.

First, an evaluation of the AdaMS OD algorithm is given, starting with the training of all variants. Then, we investigate the different parameters and their effects for the base algorithm. This is followed by similar examinations for the multi-reference enhancement and the outlier-type distinction variant. Then the OutlierNet approach from the previous chapter is benchmarked. Finally, we round the chapter off by comparing the results of the different approaches.

6.1 Training

The data set of the Swiss mountains consists of 203 photos from all over Switzerland that show mountainous terrain. Additionally, for all photos a segmentation ground truth is given. We used 61 of the images of the Switzerland data set as training for the outlier detection algorithm in the following fashion: First, the segmentations of all the images in the data set were computed and then the silhouettes were extracted from those as described in Chapter 3. We then selected 61 outlier free silhouettes for the reference data computation. This training portion of the Switzerland data set can be seen in full in Appendix B.1.

Based on the 61 reference silhouettes we computed reference histogram clusterings for all numbers of clusters from one cluster to 61 clusters. Note here, that the clustering with one cluster is equivalent to the basic approach without any clustering whereas with

Distance	μ	σ	σ/μ
aad	0.1574	0.0138	0.0879
hid	0.5388	0.0717	0.1332
emd	2.7651	0.1698	0.0614

Table 6.1: Mean μ and standard deviation σ for the different distance functions without clustering.

61 clusters every reference silhouette histogram becomes a reference histogram on its own. For every k the k means clustering has been started 1000 times with randomly initialised cluster centroids and the best clustering in respect to the mean squared distance between each histogram and the centroid histogram has been chosen. As a distance function in this case we used the same distance function that is used during the algorithm itself.

Table 6.1 gives the basic statistical properties of the outlier score distributions for the three different distance functions. It can be seen here, that all the functions give very different mean distances, with the mean distance of the earth mover's distance being more than 17 times larger than that of the above average distance. Equally, the standard deviations vary widely. In order to assess whether these differences are caused by the different scaling of the distributions the last column of the table gives the standard deviation σ normalised by the mean distance μ for each distance function. By this we see, that apart from scale the three distance functions yield similar anomaly score distributions. The earth mover's distance has the lowest relative standard deviation, while the histogram intersection distance has the largest relative standard deviation, being a little more than double the relative standard deviation of the emd.

Figure 6.1 gives the normalised mean and standard deviation for the different clusterings. In order to normalise the data, all values have been divided by the mean distance without clustering (i.e. one cluster centroid) for the respective distance function as given in Table 6.1. The points connected by the lines give the normalised mean distance for each number of clusters while the whiskers show the normalised standard deviation. Obviously, clustering has the strongest effect on the above average distance. The mean distance drops rapidly to a relative mean of about 0.7 for $k = 3$. From there on, there is a phase where the relative mean jumps between 0.7 and about 0.55 between $k = 14$ and $k = 38$. After that, the relative mean settles to about 0.55.

The histogram intersection distance is much less influenced by the number of clusters and converges relatively smoothly to a relative mean of about 0.8. This value is reached first for $k = 13$. After that, the slope gets much flatter with only small jitters in comparison to the above average distance. Finally, the earth mover's distance seems to be least affected by the number of clusters. The relative mean even in the beginning is dropping very slowly compared with the other two distances and stays relatively close to the value without clustering. The only unusual point occurs for $k = 21$, where the mean distance is higher than for the surrounding values. However, even this deviation is only small compared to the effect of the number of clusters on the other distances means. An explanation for this behaviour is given below.

An interesting observation is the fact, that for all three distance functions, the standard deviation does not seem to be greatly affected by the number of clusters. In

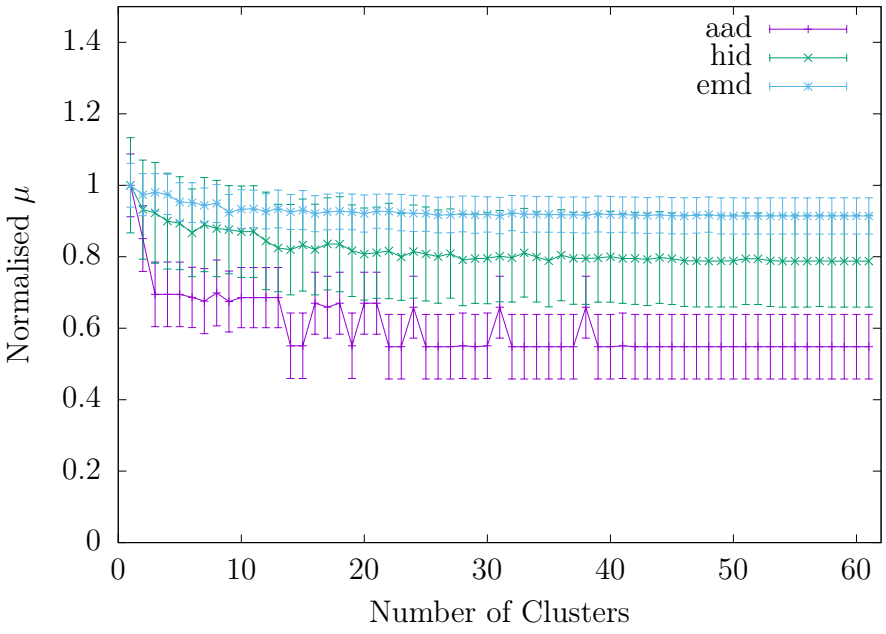


Figure 6.1: Mean normalised window distance μ on reference data over number of clusters.

order to further investigate this, Figure 6.2 shows the relative standard deviation for each distance function. For this plot the actual standard deviation for the number of clusters is divided by the corresponding mean value for the same distance function. We see from this plot that both the aad as well as the hid converge to a normalised standard deviation of about 0.165 with a rising number of clusters. Again, as with the relative mean, the behaviour of the hid is much smoother than that of the above average distance that has jumps in the same places as above.

As a result of its somewhat erratic behaviour, the first time the above average distance reaches its final result is for $k = 14$ clusters, exactly the same number of clusters at which the aad first reaches its minimum μ as seen in Figure 6.1. In both cases, the final values without larger deviations are reached for a number of clusters of $k = 39$. An explanation for this behaviour of the above average distance is the fact, that relatively small values to the bins can have a fairly large influence on the distance between histograms. This happens especially for more or less equally filled histograms, where changes of just one occurrence from one bin to another can lead to one bin falling below the average filling threshold and another one rising above it. Thus, a change of a single data point inside the histogram may lead to a difference in distance of $\frac{2}{|H|}$ for $|H|$ the number of bins in the histogram H. As a consequence, a very small number of changes in the histograms due to different clusterings or different number of clusters can lead to slightly different cluster representatives. As described before, in certain cases these small differences can lead to relatively large differences in the distances.

In contrast to this, both the histogram intersection distance and the earth mover's distance do not have a threshold like the above average distance that decides whether a bin will be considered or not. This results in the much smoother behaviour regarding the number of clusters we see in the figures, since under both distances, small changes in

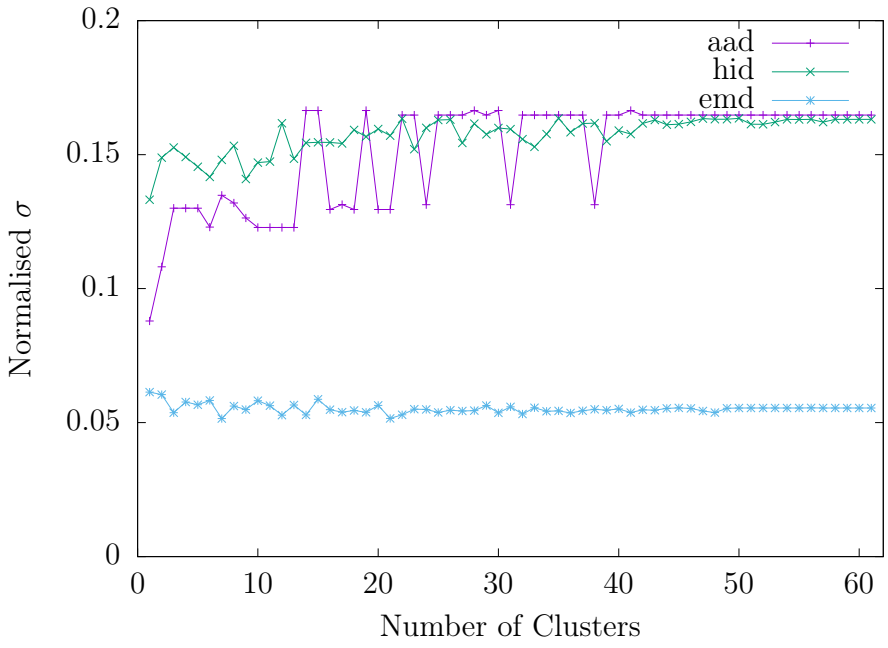


Figure 6.2: Normalised standard deviation σ/μ on reference data over number of clusters.

the data lead to small changes in the distances, only. Under the earth mover's distance, this effect is further amplified by the fact, that histograms are not only compared bin-wise but also between different bins with the distance between those bins taken into account.

Interestingly in contrast to the other two distances, the overall behaviour of the earth mover's normalised standard deviation rather than growing towards the threshold value is staying even if not getting a little lower over time. Starting at a value of 0.0614 for one cluster it gradually sinks towards its lowest value of 0.0516 at $k = 21$ and then again rises to a value of 0.0555 at $k = 58$, the last known value in this case. The behaviour between $k = 19$ and $k = 21$ looks a bit unusual due to the low dip at $k = 21$ and a comparably high value at $k = 20$. However, the hid shows a somewhat related behaviour in that region, thus this seems data related. Comparing this to Figure 6.1, both distance function show a dip for $k = 20$ which seems to be a good number of clusters to fit the data under both distance functions. The overall behaviour of the normalised standard deviation of the emd can be explained by the fact, that – other than both aad and hid – the normalised μ does not change very much at all. Thus, we see the overall decrease of the standard deviation, that all three distance functions show, in Figure 6.2, because in case of the earth mover's distance alone, it is not diminished by the stronger decline of μ.

We suspect that an explanation for the unusual behaviour of the values for μ of the earth mover's distance stems from the fact, that the distances between the single reference silhouettes' histograms are much more uniform under the earth mover's distance than under the other two distances. This is due to the case, that the emd considers the spatial distance of the bins in the histograms. Due to this uniformity in distances, the minimal distance to the centroids does not decrease in the same way as under the

Length	Test Set 1	Test Set 2
1 – 4	1213	9
5 – 9	135	41
10 – 19	65	24
20 – 99	29	31
≥ 100	7	6
Total used	101	61

Table 6.2: Number of outliers by outlier length.

other histogram distances, rendering the clustering possibly less useful. However, due to the lower standard deviation in comparison with the other two distances, it could also be possible, that anomalous histograms are more visible under the earth mover's distance. If this is the case, it is probable that visibility then is additionally enhanced by raising the number of clusters.

6.2 Testing of the Base Algorithm

As the first test data set, Test Set 1, we used a set of 46 images from our data set that we segmented using the AdaMS segmentation. We then classified every point on these silhouettes as an anomaly, that is farther than three pixels away from the nearest point of the reference silhouette extracted from the manual segmentation. Finally, we set all sequences of anomalies of a minimal length of 10 as outliers.

Test Set 2 is the data set that we created for [SBC16]. It consists of 111 manually annotated outliers on 14 silhouettes extracted by AdaMS from the Switzerland data set. In contrast to Test Set 1 which only contains segmentation errors, Test Set 2 contains obstacles, too. Again, we only use outliers of a minimal length of 10. This results in 61 used outliers that are divided in 11 obstacles and 50 segmentation errors.

Table 6.2 shows the distributions of outlier lengths in the test data sets. Due to the minimal length restrained of 10 pixels, only 101 outliers are used. It is obvious here, that the vast majority of outliers is shorter than 10 points. However, shorter anomalies are not as meaningful to the general appearance of the silhouette since they cannot deviate much from the correct silhouette. Most of these shorter outliers are just jitters in the silhouette due to image artefacts and thus can be easily ignored in order to get more meaningful results in respect to the quality of the outlier detection.

We start by evaluating Test Set 1 in order to better understand the effects the parameters of the outlier detection algorithm have on the detection of segmentation errors. Test Set 2 will be used later in order verify these results.

First we want to investigate the influence of the threshold parameters t_s and t_w on the results of the basic outlier detection without clustering and multi-reference enhancements. Hereby, precision is the ratio of true positive pixels to all pixels that are detected as parts of outliers, recall is the ratio of true positive pixels to all pixels that are part of real outliers, and F_1 score is the harmonic mean of precision and recall, i.e.

$$F_1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

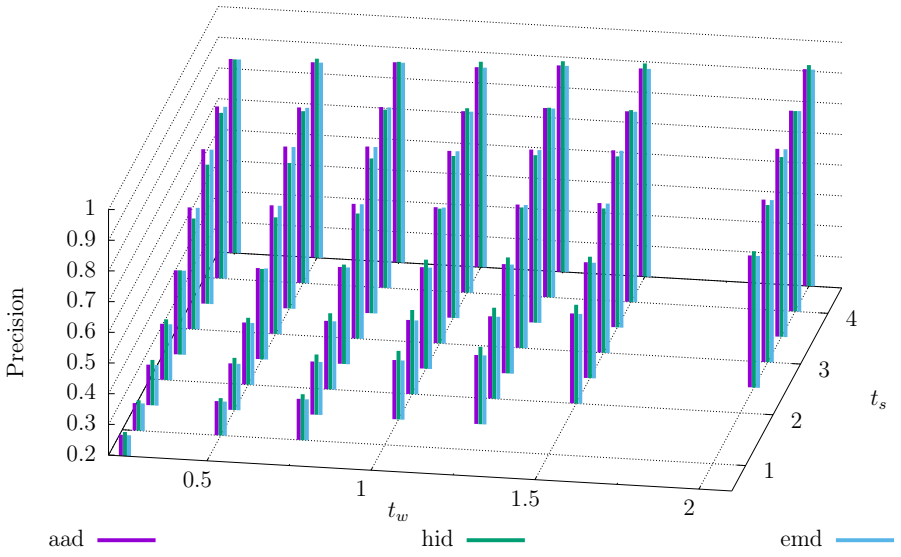


Figure 6.3: Influence of t_w and t_s on precision without clustering and $l = 1$ on Test Set 1.

The parameters t_s and t_w directly influence the strong anomaly border τ_s and the weak anomaly border τ_w via

$$\tau_s = \mu + t_s \cdot \sigma$$

and

$$\tau_w = \mu + t_w \cdot \sigma.$$

Figure 6.3 shows the influence of t_s and t_w on the precision of the algorithm for all three distance functions. Note, that t_w has to be lower than t_s, therefore there are no values in the bottom right area of the plot. The figure shows that for the smallest values of both t_s and t_w precision is at its lowest, with a precision of about 0.35 for $t_s = 0.5$ and $t_w = 0.25$, while the highest precision of 0.94 is reached for $t_s = 4.5$ and $t_w = 2$ for all three distance functions.

Overall we see, that the influence of t_s on the precision is higher. This is to be expected, since the value of t_s and therefore τ_s determines the number of outliers that is detected. The higher t_s gets the fewer outliers are detected and vice versa. In contrast to this, t_w and subsequently τ_w influence the size of detected outliers. Once an inner outlier is detected, it is enlarged until a point whose anomaly score is below τ_w is found. Thus, while a value of t_w that is too low, a few points outside an outlier might be added, it does not add an outlier that is a false positive in its whole.

It is also obvious from the figure, that the behaviour of all three distance functions is relatively similar with no systematic differences in the values.

In Figure 6.4 the effects of t_s and t_w on recall are depicted in a similar fashion as their effects on precision in Figure 6.3. In terms of effects this figure shows to be the opposite of the previous figure for the above average distance and the histogram intersection distance: Lower values of both t_s and t_w unsurprisingly yield better results

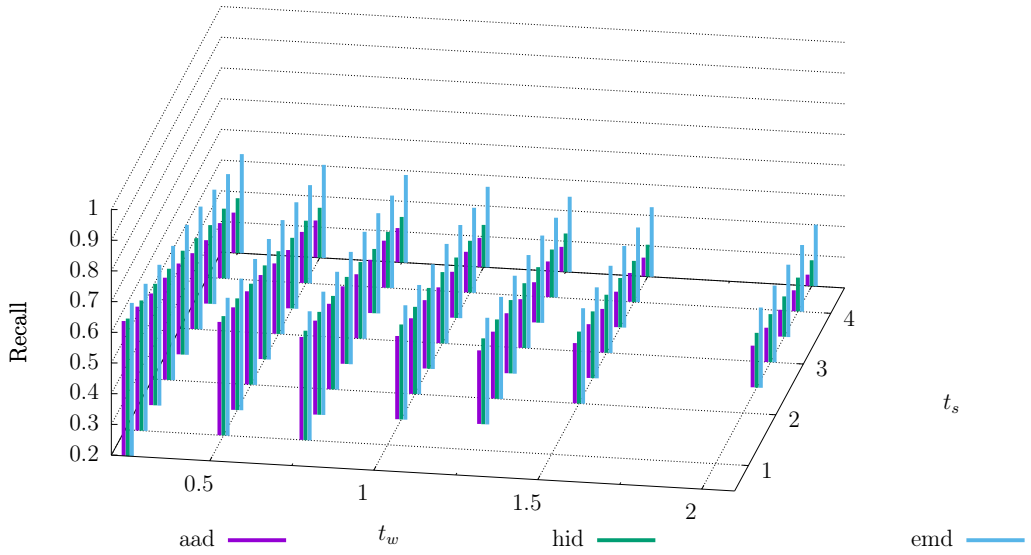


Figure 6.4: Influence of t_w and t_s and τ_{out} on recall without clustering and $l = 1$ on Test Set 1.

in respect to recall. The best recall value is achieved at $t_s = 0.5$, $t_w = 0.25$, being 0.6391 for aad and 0.6480 for hid. The worst value achieved with the above average distance is a recall of 0.2364 and 0.2839 with the histogram intersection distance, each for $t_s = 4.5$ and $t_w = 2.0$.

Overall we see, that the recall values are lower than precision values by about 0.1 for the lower values and 0.2 for the better values. Also, the influence of t_s and t_w on recall is a bit more even than on precision: While t_s again has a greater influence on recall, since it enables finding an outlier per se if it is lowered, higher values for t_w still lower recall significantly. This is due to the fact, that even if we find a sequence of strong anomalies inside an outlier, recall is still low if we miss a large part of that outlier due to a high value of t_w. Recall is better for the histogram intersection distance than the aad by values between 0.0089 for $t_s = 0.5$ and $t_w = 0.25$ and 0.0529 for $t_s = 3$ and $t_w = 1.25$.

For the earth mover's distance, general behaviour is similar to the other two distances in that the best recall of 0.6996 is reached for $t_s = 0.5$ and $t_w = 0.25$ and the worst value is reached at $t_s = 4.5$ and $t_w = 2.0$ with a recall of 0.4012. However, it is observable that the emd is much more robust in respect to recall to changes to the thresholds than the other distance functions. The best values of the earth mover's distance differs only by 0.0605 to the aad and by 0.0516 to the hid, while the difference in the worst values is 0.1648 to the above average distance and 0.1173 to the hid. This is more than double the difference.

Finally, Figure 6.5 shows the F_1 score over both thresholds. For the above average distance it is clear from this plot that the best values regarding F_1 score are achieved for moderately high values of t_s, that yield a relatively high precision, and low values of t_w in order to get outliers that are large enough to detect a high portion of the

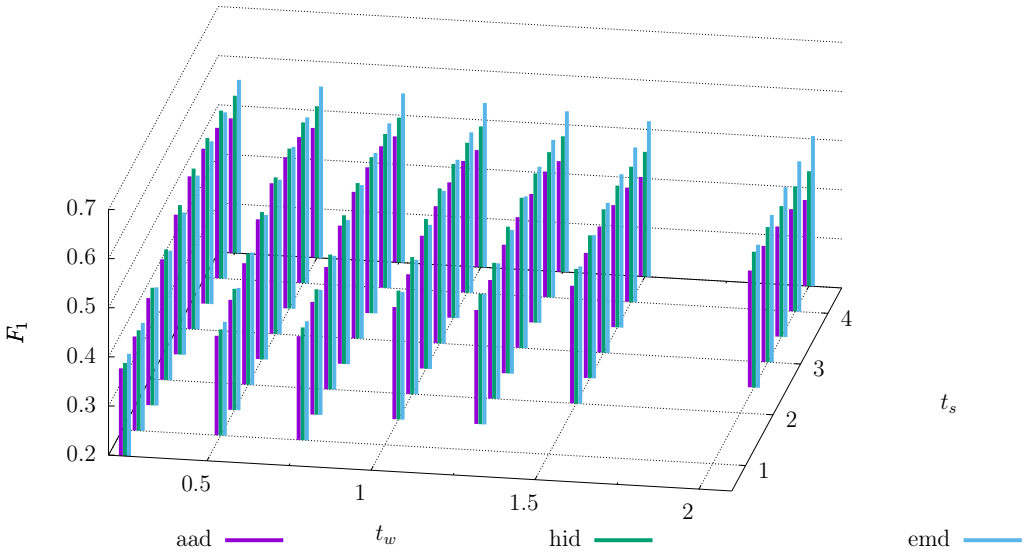


Figure 6.5: Influence of t_w and t_s on F_1 score without clustering and $l = 1$ on Test Set 1.

outliers that have been detected. The best values, i.e. the values that are no more than 0.01 worse than the best score, are achieved for combinations of t_s and t_w of 3.5 and 0.25, 3.0 and 0.25, 3.5 and 0.5, 3.0 and 0.5, and 4.0 and 0.25, achieving F_1 score of 0.5154, 0.5108, 0.5076, 0.5063, and 0.5063, respectively. The lowest values, i.e. the values being no more than 0.01 better than the worst F_1 score, are achieved for the most extreme value combinations of $t_s = 4.5$, $t_w = 2.0$ and a F_1 score of 0.3749 and $t_s = 0.5$ and $t_w = 0.25$ yielding a F_1 score of 0.3782.

The values of the histogram intersection distance show a similar behaviour to the aad but on a slightly higher level due to the higher recall. The best values are reached for $t_s = 4.0$ with an F_1 score of 0.5418 and $t_s = 3.5$ with an F_1 score of 0.5376. Both values are achieved with $t_w = 0.25$. The worst F_1 score of 0.3896 is reached for $t_s = 0.5$ and $t_w = 0.25$.

As expected the earth mover's distance shows the highest F_1 scores, with the difference stemming from the higher recall values. However, because of the greater robustness of the emd's recall in regard to t_s in this case the best values are all achieved for a value of $t_s = 4.5$ with F_1 scores of 0.5532, 0.5492, and 0.5447 achieved for values of t_w of 0.25, 0.5, and 0.75 respectively. The worst value again is reached at $t_s = 0.5$ and $t_w = 0.25$ with a F_1 score of 0.4089.

In order to examine the influence of the minimal length of subsequent strong anomalies in an outlier l on the results, we first look at the best results in respect to F_1 score for values of l in $\{1, 2, 3, 4, 5\}$ shown in Table 6.3 for each of the distance functions. Here, for every value of l the best achieved F_1 score is shown, together with the thresholds t_s and t_w that were used to generate it. The table also shows precision and recall. We can see from this table, that in regard to F_1 score there are no huge differences between the different values for l for each of the distance functions.

Distance	l	t_s	t_w	Precision	Recall	F_1
aad	1	3.5	0.25	0.7031	0.4067	0.5154
	2	3.0	0.25	0.6450	0.4334	**0.5184**
	3	3.0	0.25	0.6882	0.4047	0.5097
	4	2.5	0.25	0.6238	0.4253	0.5058
	5	3.0	0.25	0.7748	0.3668	0.4979
hid	1	4.0	0.25	0.7395	0.4275	**0.5418**
	2	3.5	0.25	0.6893	0.4418	0.5385
	3	3.0	0.25	0.6392	0.4583	0.5338
	4	3.0	0.25	0.6970	0.4324	0.5337
	5	2.5	0.25	0.6422	0.4463	0.5266
emd	1	4.5	0.25	0.5870	0.5231	0.5532
	2	4.5	0.25	0.6142	0.5100	**0.5573**
	3	4.0	0.25	0.6052	0.5051	0.5506
	4	4.0	0.25	0.6598	0.4762	0.5532
	5	4.0	0.25	0.6964	0.4516	0.5479

Table 6.3: Best results regarding F_1 score for each l on Test Set 1.

For the above average distance, the worst value achieved for $l = 5$ is only about 0.02 worse than the best value which occurred for $l = 2$. It is also interesting to note, that in terms of t_w, in all cases a value of 0.25 achieved the best results not only for the aab but for all distance functions. For t_s values range between 2.5 to 3.5. It is noteworthy here, that in case of $l = 2$ and $l = 5$ $t_s = 3$, such that the same combination of thresholds resulted in both the best and worst F_1 score. A value of $t_w = 3$ is also chosen for $l = 3$ and unsurprisingly precision, recall, and F_1 score all fall between the values for $l = 2$ and $l = 3$.

Overall the histogram intersection distance shows behaviour similar to the above average distance. The differences in F_1 scores are marginal, at about 0.015 between the best value for $l = 1$ and the worst for $l = 5$. For t_s the values get smaller the larger l becomes, indicating that both values affect the score more or less the same. In contrast to this, for $l = 3$ and $l = 4$ the same value of t_s of 3 yields the best F_1 score and we see that the differences in F_1 score in theses cases are marginal at 0.0001. However, if we look at precision and recall for these to parameter sets, we see that there are larger differences regarding these scores, such that the decrease in recall is made up by the significant rise in precision.

The earth mover's distance is affected the least by changes of l. This can be seen from the facts that the difference between the best F_1 score at $l = 2$ and the worst F_1 score at $l = 4$ is less than 0.01, and that the value for t_s just changes once from 4.5 at $l = 2$ to 4 at $l = 3$. Surprisingly, for the same thresholds the emd's F_1 score seems to rise rather than fall if l gets larger, at least up to a certain point. This can be seen due to the fact that both for $l = 2$ and $l = 4$ the best values for each t_s are reached, while for the other two distance functions, the best respective value is always reached for the lowest l is occurs with. We suspect that this behaviour has to do with the earth mover's distance's greater robustness to recall.

Regarding precision the table indicates that t_s has a larger impact on both values

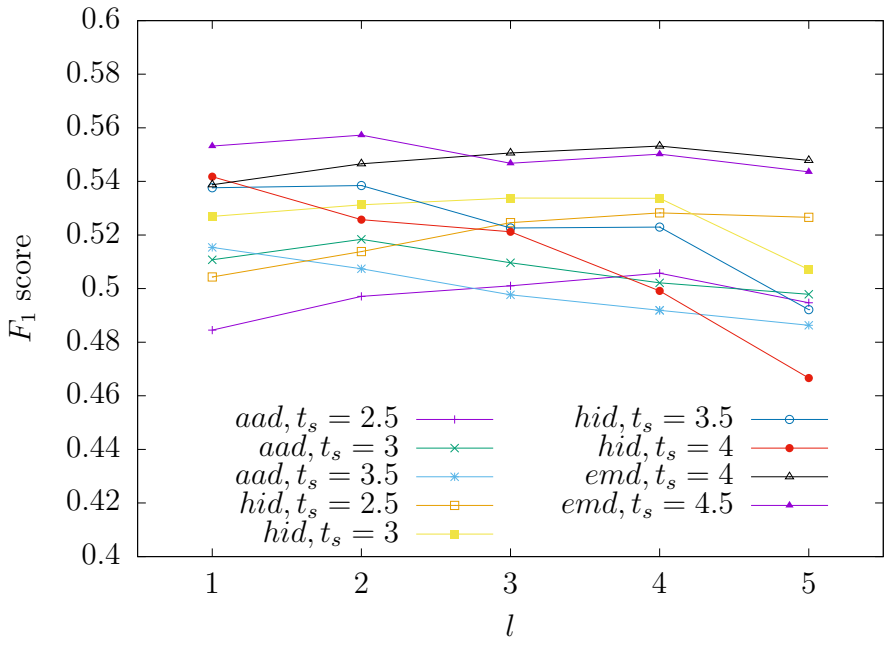


Figure 6.6: Influence of l on F_1 scores on Test Set 1.

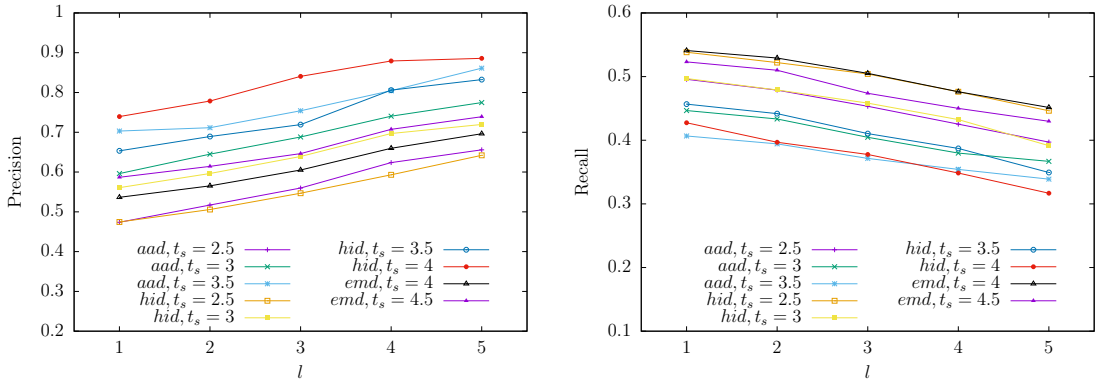
and thus F_1 score than l. This observation is underlined by the fact, that with the aad for $l = 2$, $l = 3$, and $l = 5$, where identical values of t_s were used, precision rises and recall is lowered while for $l = 1$, precision is notably higher than for $l = 3$ and even more so for $l = 2$. With the histogram intersection distance we see a similar behaviour for $l = 1$ to $l = 2$, $l = 2$ to $l = 3$, and $l = 4$ to $l = 5$. For the earth mover's distance this is harder to see, since the value of t_s does change only once at $l = 2$ to $l = 3$. In this case, however, the precision does get lower like with the other distance functions.

On recall, the effects are similar if somewhat weaker: For the aad at $l = 2$ recall is higher than for $l = 3$ and $l = 5$, while due to changes in t_s the other two entries do not seem to follow this trend. The same applies to the histogram intersection distance, where recall is lower for $l = 4$ than for $l = 3$ with the same thresholds, while in the other cases recall is gets higher with lowered t_s. For the emd the effects of t_s on recall are lower, as we have already described above. Therefore the recall for $l = 3$ is slightly lower than at $l = 2$.

In order to further investigate the effect of the minimal inner length parameter l on the F_1 score we have plotted the F_1 scores over l for all combinations of distance function and thresholds that feature in Table 6.3 in Figure 6.6. We can see from this that in general the differences in F_1 score are relatively small. However, in most cases the curves peak at $l = 4$ or earlier and then start to decline from there on. The only case, where this behaviour is not obvious is the histogram intersection distance with an threshold t_s of 2.5. However, even here we note a slight decrease from $l = 4$ to $l = 5$.

Another interesting observation is that in general the hid seems to be the most sensitive distance function in regard to a rising l. Apart from the case of $t_s = 2.5$ that we already discussed above, the other three instances of the histogram intersection distance show a steep decline in contrast to the emd and aad.

Figure 6.7 shows the effects of l on precision (Figure 6.7a) and recall (Figure 6.7b).



(a) (b)

Figure 6.7: Influence of l on precision and recall on Test Set 1.

Again, we see that the histogram intersection distance is more prone to changes of l than the other two distances. In respect to precision we see, that the gain in precision from $l = 3$ to $l = 4$ is the strongest while after that the histogram intersection distance flattens out slightly more than the other to distances. In terms of recall, the decline with rising l is more or less linear for all parameter sets. Again, the hid has a steeper slope than the graphs of the other two distance functions. This is especially apparent for the hid with $t_s = 4$ and the aad with $t_s = 3.5$, that have very similar recall in the l range from 2 to 4. However, the hid's recall for $l = 1$ is higher and the recall for $l = 5$ lower than the corresponding values of the aad. The combination of both of these characteristics then leads to the behaviour seen in Figure 6.6.

Other interesting approaches to evaluate the different combinations of parameters of the outlier detection is the fraction of hit reference outliers compared to all reference outliers, as well as the fraction of hitting detected outliers. Hereby, by hit reference outliers we mean the number of reference outliers that overlap with one of the detected outliers relative the total number of reference outliers. The idea of this measurement is essentially the outlier based recall where a true positive is a detected outlier that intersects with a reference outlier. By hitting outliers we mean the number of detected outliers that intersect with a reference outlier relative to the total number of detected outliers. This can be interpreted as an outlier based precision.

These measurements are meaningful due to the way the adaptive outlier removal in AdaMS works. In affected grid cells the segmentation parameters are adjusted. However, those adjustments do not affect the silhouette inside of that grid cells only. If, for example, due to false segmentation inside a single grid cell g a larger portion of ground pixels gets classified as sky pixels, an adjustment of the brightness factor γ_g can lead to a correct segmentation of the pixels in g and hence other pixels that were classified as sky before are never checked in respect to belonging to the sky and thus stay ground pixels.

Figure 6.8 gives an overview of the influence of the threshold parameters t_s and t_w on the hit outliers. In general these results resemble those of the recall presented in Figure 6.4 in that the earth mover's distance achieves much higher values for larger t_ss than the other two distance functions. Again, the above average distance and the histogram intersection distance perform similarly albeit the hid is 0.0438 better

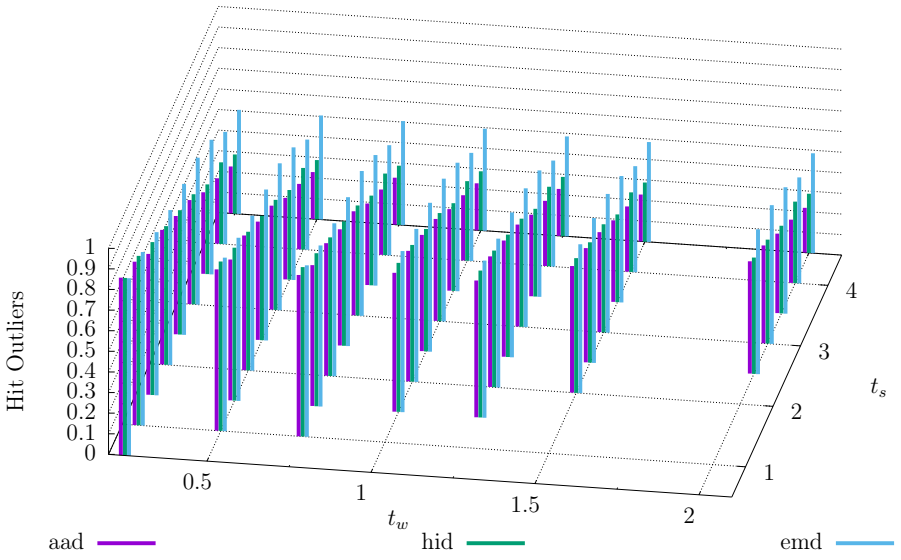


Figure 6.8: Influence of t_w and t_s on the fraction of hit reference outliers without clustering and $l = 1$ on Test Set 1.

on average. Overall, the results values are about 0.2 higher than the recall values. This, however is expected behaviour since the probability of a whole detected outlier intersecting with a reference outlier is higher than the probability of a single detected point inside a detected outlier lying inside a reference outlier as well.

Also, in general, the influence of t_w on the fraction of hit outliers is negligible in all instances. This shows, that for all distance functions if a detected outlier O intersect with a reference outlier R, then O's inner outlier $I(O)$ of strong anomalies does intersect with R as well. A further conclusion to be drawn from this is that cases where one detected outlier hits two or more reference outliers are relatively seldom since it is improbable that the inner outlier portion of detected outliers grows that large.

The fraction of hitting outliers are depicted in Figure 6.9. Again, as with the hit outliers, it is noticeable that the influence of t_w on the number of hitting outliers is insignificant as there are nearly no changes following the rows of the figure from left to right. As before, the histogram intersection distance and the above average distance perform very similarly. However, in this case the aad performs slightly better on average by 0.0073. This effect is higher for values of t_s of 3 and above where the average advantage of the aad is 0.0225.

Interestingly, in contrast to the precision results shown in Figure 6.3 where the earth mover's distance performed on the same level as hid and aad, in this case the emd's performance is clearly the worst of all three distance functions. On average it is 0.1368 worse than the above average distance, while for values of $t_s \geq 3$, where in general the performance of all three distances is the best, the difference rises to an average of 22.09%.

In order to further investigate this phenomenon Table 6.4 shows the absolute number of detected outliers for each distance function and each value of t_s. The other

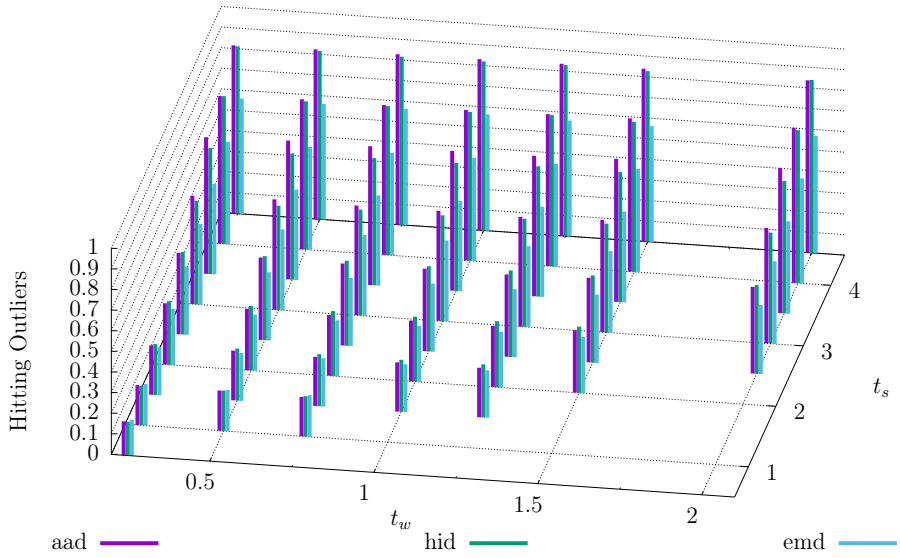


Figure 6.9: Influence of t_w and t_s on the fraction of hitting detected outliers without clustering and $l = 1$ on Test Set 1.

t_s	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5
aad	2222	1550	1033	727	471	287	178	141	92
hid	2019	1488	1004	697	479	324	222	166	117
emd	1917	1485	1144	882	681	528	419	326	266

Table 6.4: Number of detected outliers over t_s for $t_w = 0.25$ and $l = 1$ on Test Set 1.

parameters used to get the numbers in the table are $l = 1$ and $t_w = 0.25$. The threshold t_w was chosen as the lowest possible value since it is the only value that works in combination with all values of t_s. Higher values of t_w marginally increase the number of outliers since it prevents close-by outliers from growing into a single larger outlier.

We can see from the table that with rising t_s the number of detected outliers decreases significantly. For lower values of t_s, i.e. $t_s \leq 1.5$, under all distance functions the relative difference of detected outliers is relatively small. However, in the remaining range of t_s we see that when using the earth mover's distance the number of detected outliers is considerably higher with nearly three times as many detected outliers than the above average distance for $t_s = 4.5$. Meanwhile the difference between the aad and the histogram intersection distance stays very small, as expected when considering the similar performance in the hit and hitting outlier analyses.

In conclusion we see from this table that the high number of hit reference outliers of the emd for larger values of t_s is bought by a higher total number of detected outliers compared to the other distance functions. This results in the low fraction of hitting outliers for the earth mover's distance that we observed in Figure 6.9.

In order to evaluate the overall performance of the variants of the outlier detection

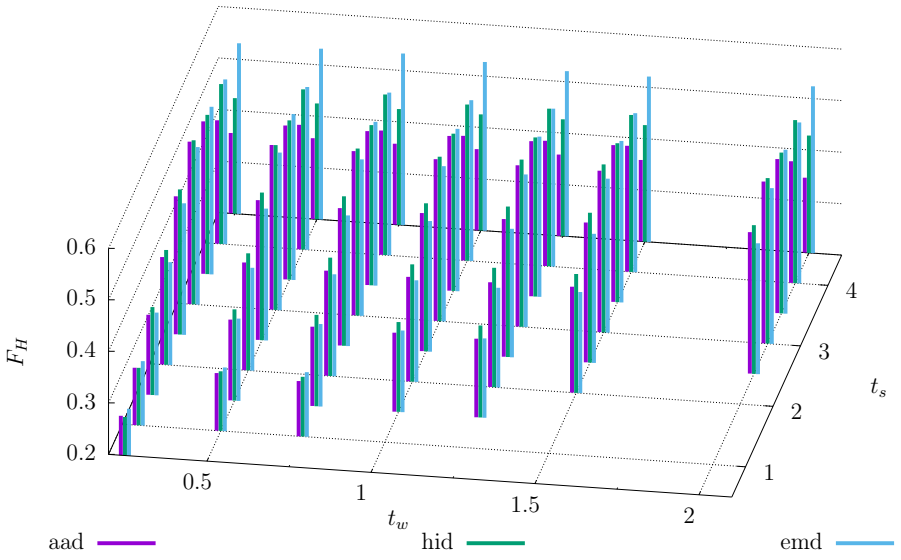


Figure 6.10: Influence of t_w and t_s on the harmonic mean of hit reference outliers and hitting detected outliers without clustering and $l = 1$ on Test Set 1.

algorithm, we also investigated the harmonic mean of the hit outliers and the hitting outliers. Let HR be the fraction of hit reference outliers and HD be the fraction of hitting detected outliers, then the harmonic mean of those two metrics is given by

$$F_H = 2 \cdot \frac{HR \cdot HD}{HR + HD}.$$

If we interpret HR as related to recall and HD as a kind of precision, F_H is the equivalent to the F_1 score for HR and HD.

Figure 6.10 plots this score over the different values for t_s and t_w. As with Figures 6.8 and 6.9 we see that the influence of t_w on the results is once again marginal. For values of t_s up to 3.5 we see that all three distance functions yield very similar results. This comes as no surprise since aad and hid perform similar for the HR and HD while the emd yields better HR scores and worse HD scores in comparison. However, for values of $t_s \leq 3.5$ these effects seem to cancel each other out in respect to F_H.

For higher values of t_s we see a significant drop in F_H score for the above average distance which is caused by the lowest values in HR. The histogram intersection distance drops, too, but not as severe as the aad. Again, the cause of this behaviour are the low HR values for high t_s, however we see that the slightly better values in HR result in F_H scores on the level of the earth mover's distance for $t_s = 4$ and values halfway between the other two distances for $t_s = 4.5$.

While the figure shows that the earth mover's distance is more robust in respect to changes in t_s than the other distances, Table 6.5 reveals that the absolute maximum and minimum values for all three distance functions are relatively close with less than 2% difference between the best and worst values for maximum and minimum values, each.

Dist	Maximum			Minimum		
	t_s	t_w	F_H	t_s	t_w	F_H
aad	3.0	0.5	0.5197	0.5	0.25	0.2766
hid	3.0	1.5	0.5262	0.5	0.25	0.2748
emd	4.5	0.75	0.5330	0.5	0.25	0.2913

Table 6.5: Maximum and minimum F_H scores per distance function for $l = 1$ on Test Set 1.

Distance	l	t_s	t_w	HR	HD	F_H
aad	1	3.0	0.5	0.5050	0.5354	0.5197
	2	3.0	1.5	0.4752	0.6128	**0.5353**
	3	2.5	0.5	0.5248	0.5177	0.5212
	4	2.5	0.5	0.4356	0.5913	0.5017
	5	2.0	1.5	0.4356	0.5246	0.4760
hid	1	3.0	1.5	0.5248	0.5277	**0.5262**
	2	3.0	2.0	0.4653	0.5982	0.5235
	3	3.0	2.0	0.4257	0.6439	0.5126
	4	2.5	2.0	0.4554	0.5975	0.5169
	5	2.0	0.75	0.4752	0.5246	0.4987
emd	1	4.5	0.75	0.5050	0.5645	**0.5330**
	2	3.5	0.25	0.5842	0.4847	0.5298
	3	4.0	0.75	0.4653	0.6089	0.5275
	4	3.0	0.25	0.5149	0.5322	0.5234
	5	3.0	0.25	0.4752	0.5931	0.5277

Table 6.6: Best results regarding F_H score for each l on Test Set 1.

Table 6.6 shows the best F_H scores for every distance function and l values ranging from 1 to 5. Additionally, the thresholds t_s and t_w leading to the value and HR and HD resulting from those parameter combinations are given in a similar fashion as in Table 6.3 for F_1 score. Here we see, that overall, l does have a larger impact on F_H score than on F_1 score, especially when the above average distance is used as distance function that yields both the best overall value with $l = 2$ and the worst overall value in case of $l = 5$. It is also noteworthy that in contrast to the F_1 measure, in this case different values for t_w feature with values from the complete range featuring in combination with the different distance functions. This behaviour stems from the fact, that results regarding HR and HD in general are more similar than for precision and recall and thus small changes caused by variations of t_w have greater influence.

It is also notable that the best values for all three distance functions are closer to each other than in case of the F_1 score. Here the difference between the best value of the best distance and the best value of the worst distance is less than one percent.

In order to verify whether the results obtained above are transferable on a data set which contains obstacles in the following we look at Test Set 2. Figure 6.11 shows the influence of the anomaly thresholds on the F_1 score for $l = 1$ on all three distance

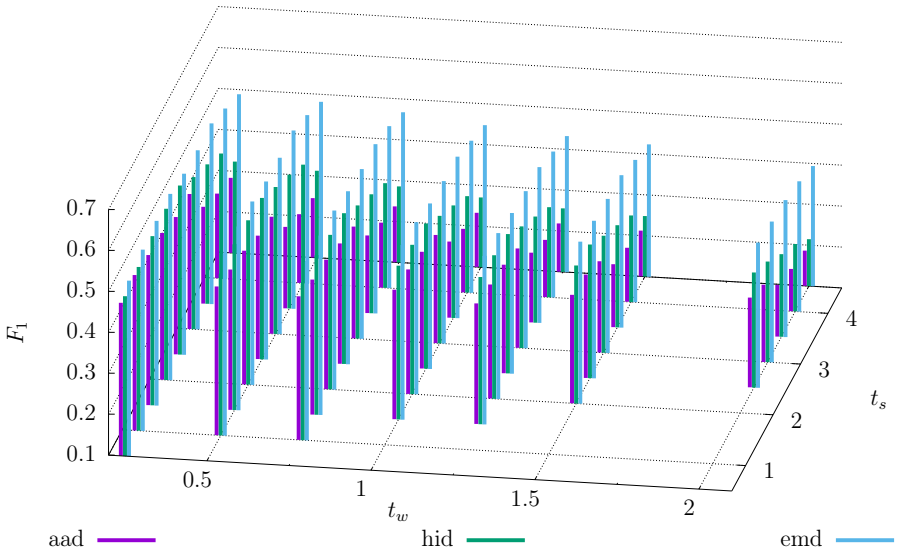
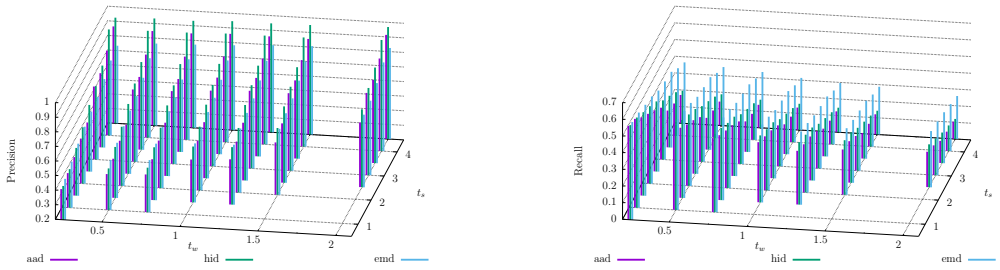


Figure 6.11: Influence of t_w and t_s on F_1 scores without clustering and $l = 1$ on Test Set 2.

functions. Interestingly, the behaviour is quite different on Test Set 2 than it is on Test Set 1. In general, lower values of t_w here seem to result in better F_1 scores over all distance functions. Also, on this data set, lower values of t_s yield better F_1 scores.

As a consequence, the best F_1 scores for the aad, hid, and emd are reached for combinations of t_s and t_w of 1.0 and 0.25, 2.0 and 0.25, and 2.0 and 0.25, respectively. These threshold combinations yield F_1 scores of 0.5058 for the aad, 0.5378 for the hid, and 0.5776 for the emd. For all three distance functions, the worst values are reached for $t_s = 4.5$ and $t_w = 2.0$. Overall the scores are a bit higher than for Test Set 1.

In order to explain this difference in behaviour we have to take a look at Figures 6.12a and 6.12b. Here we see, that the general behaviour of precision and recall under



(a) (b)

Figure 6.12: Influence of t_w and t_s on precision and recall without clustering and $l = 1$ on Test Set 2.

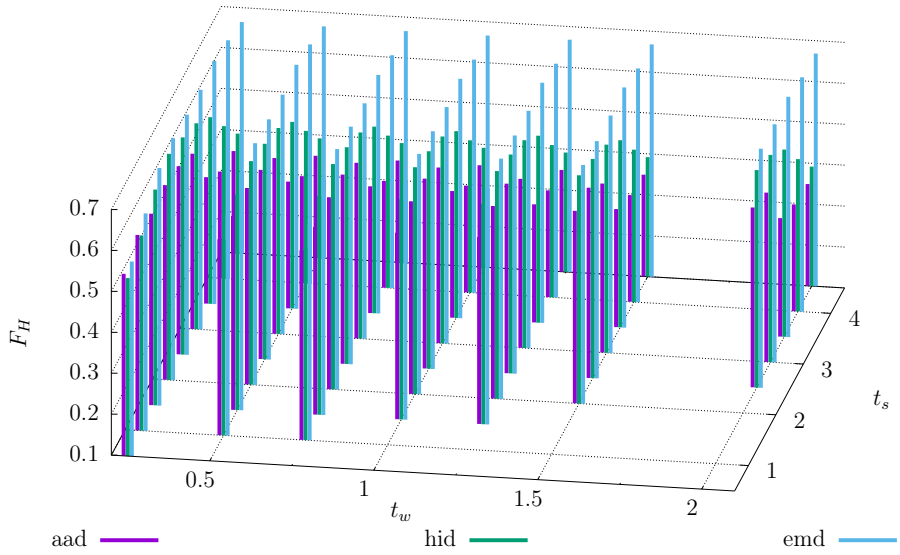


Figure 6.13: Influence of t_w and t_s on F_H scores without clustering and $l = 1$ on Test Set 2.

Dist	Maximum			Minimum		
	t_s	t_w	F_H	t_s	t_w	F_H
aad	2.0	0.5	0.5801	4.5	0.25	0.3487
hid	2.0	1.5	0.6576	4.5	0.25	0.3919
emd	2.0	1.5	0.6576	4.5	0.25	0.3919

Table 6.7: Maximum and minimum F_H scores per distance function for $l = 1$ on Test Set 2.

changes of the thresholds is very similar to that von Test Set 1 and depicted in Figures 6.3 and 6.4. The lowest precision values are reached if both thresholds are low and are mostly influenced by t_s while recall shows an inverted behaviour. It is noticeable however, that in contrast to Test Set 1, the earth mover's distance's precision is significantly lower in comparison to the other two distances for higher values of t_s, while it is better for lower values. Also, the histogram intersection distance in this case outperforms not only the emd but also the above average distance.

In respect to recall, the above average distance is outperformed both by the hid and the emd. Again, as with Test Set 1, the earth mover's distances recall is the most robust in respect to changes to t_s and t_w. Overall, recall values are a bit worse for Test Set 2 than for Test Set 1.

In combination, the overall higher values for precision and lower values for recall lead to the significantly different values for F_1 score that have been described above and especially the more robust values of the earth mover's distance.

Figure 6.13 shows the F_H score for Test Set 2. As with Test Set 1 these are on a

Distance	l	t_s	t_w	HR	HD	F_H
aad	1	2.0	0.5	0.6230	0.5427	0.5801
	2	1.5	0.5	0.7213	0.5123	**0.5991**
	3	1.5	0.5	0.6393	0.5393	0.5851
	4	1.0	0.5	0.6885	0.5024	0.5809
	5	0.5	0.25	0.8197	0.4475	0.5789
hid	1	2.0	1.5	0.6885	0.6294	**0.6576**
	2	1.5	1.25	0.7705	0.5714	0.6562
	3	1.5	0.25	0.7049	0.5843	0.6390
	4	1.0	0.25	0.7213	0.5421	0.6190
	5	1.0	0.25	0.6557	0.5921	0.6223
emd	1	3.5	0.5	0.6557	0.7414	0.6959
	2	3.0	0.5	0.6721	0.7154	**0.6931**
	3	1.5	0.5	0.8033	0.6096	0.6932
	4	2.5	0.5	0.6230	0.7182	0.6672
	5	1.0	0.5	0.7377	0.6168	0.6718

Table 6.8: Best results regarding F_H score for each l on Test Set 2.

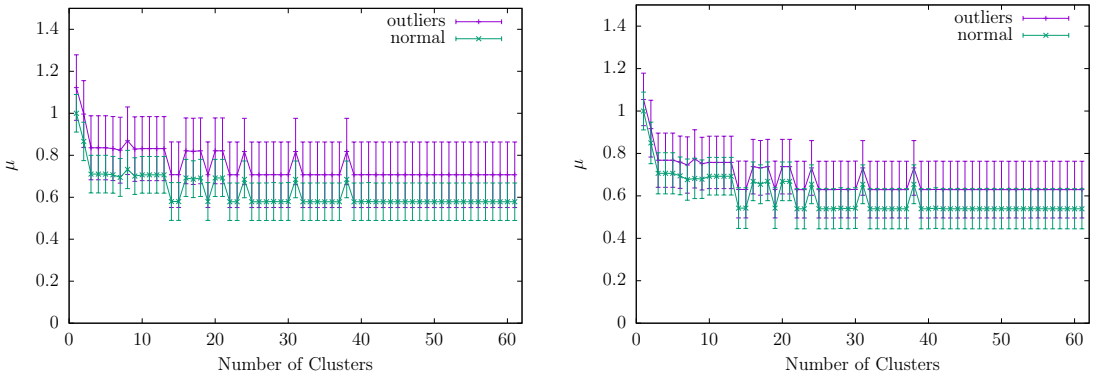
higher level than the F_1 scores but the absolute level on Test Set 2 is again higher as on Test Set 1. The best and worst values are given in Table 6.7. Again we see that on Test Set 2 lower values for t_s lead to better results than on Test Set 1. While on Test Set 1 the worst results were achieved by $t_s = 0.5$ in this case over distance function $t_s = 4.5$ lead to the worst results.

Table 6.8 shows the best threshold combination in respect to F_H for every distance function and $l \in [1, 5]$. Apart from the differences in thresholds discussed above the overall behaviour per distance function is similar to the behaviour on Test Set 1. The above average distance again reaches its best value for $l = 2$ while hid as well as emd get their best values for $l = 1$. The overall results, however, once again show that on Test Set 2 the distance function has a much higher impact than on Test Set 1, with 5% difference between aad and hid and another 5% difference between hid and emd.

6.3 Testing the Multi-Reference Enhancement

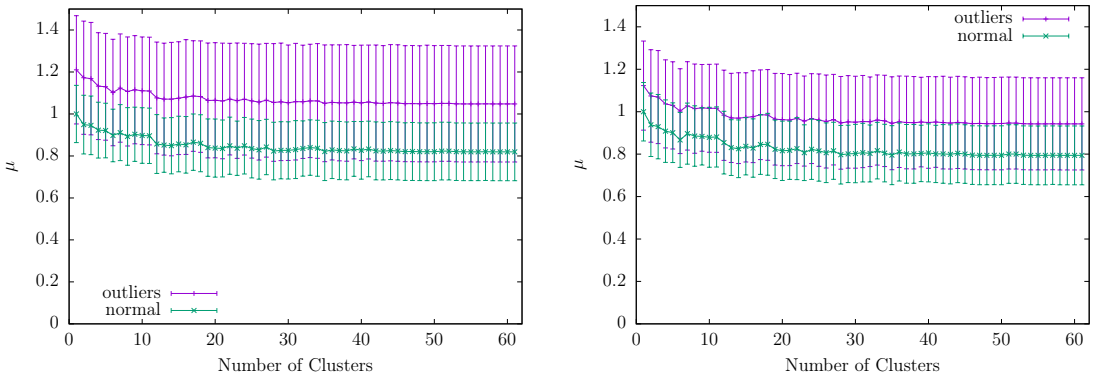
After testing the base algorithm in the previous section, we are going to evaluate the multi-reference enhancement described in Chapter 4.4 on the same data sets as we did with the base algorithm. However, in this section we focus on the effects that are caused by changing the number of clusters during the training phase and therefore the number reference histograms.

The effects of the number of clusters on the training data set have already been described in Section 6.1. There, we have seen, that in general with a rising number of clusters the mean anomaly scores of points of the training silhouettes decreases. In contrast to this, the normalised standard deviation rises slightly for the above average distance and histogram intersection distance, while staying more or less level for the earth mover's distance.



(a) Test Set 1 (b) Test Set 2

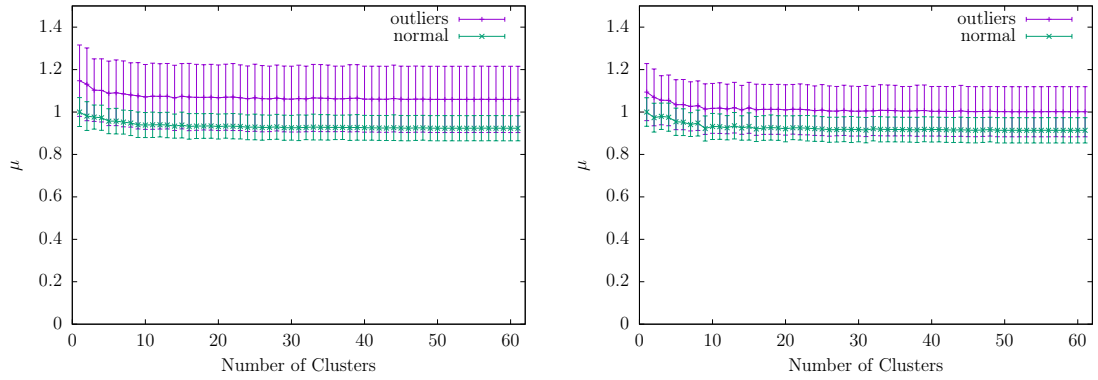
Figure 6.14: Normalised mean and standard deviation over the number of clusters k for aad.



(a) Test Set 1 (b) Test Set 2

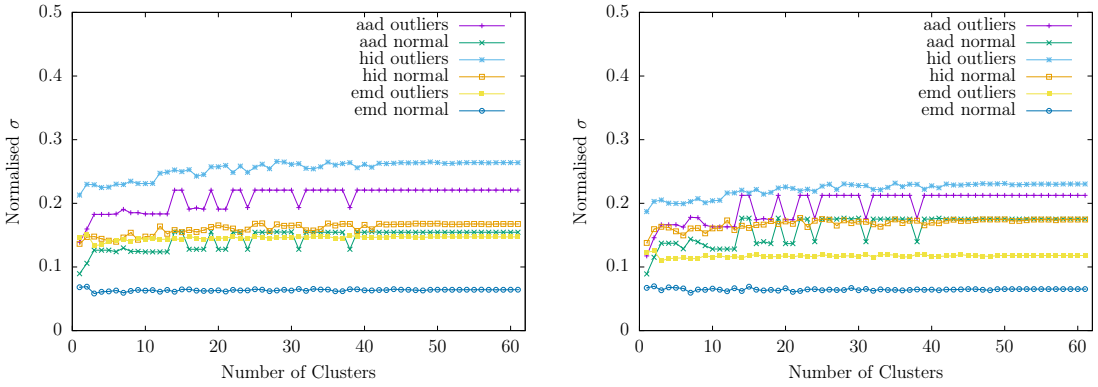
Figure 6.15: Normalised mean and standard deviation over the number of clusters k for hid.

The Figures 6.14, 6.15, and 6.16 shows comparable graphs to Figure 6.1, namely the normalised mean μ over the number of clusters. The whiskers show $\mu \pm \sigma$ for the standard deviation σ. For every distance function, as with Figure 6.1, we normalised all values by dividing through the normal μ for $k = 1$. By normal data we mean data points that are not part of the reference outliers. Additionally the purple curves show the distribution properties over the anomaly scores of points that belong to reference outliers. For the normal data, we see that the distribution mean and standard deviation behave very similar to the ones on the training data, albeit the anomaly scores are slightly higher. It is also observable, that for Test Set 1 in all cases the mean anomaly score for the data points that belong to outliers is farther than one normal data standard deviation away from the mean of the normal data. However, if we look at the outlier points' standard deviation we see that the normal data points' mean is inside one outlier data standard deviation from the outlier data mean. If we take Test Set 2 into account, we see, that for the aad and hid the mean anomaly score for outliers is about one normal data standard deviation away from the normal data mean. In contrast



(a) Test Set 1 (b) Test Set 2

Figure 6.16: Normalised mean and standard deviation over the number of clusters k for emd.



(a) Test Set 1. (b) Test Set 2.

Figure 6.17: Normalised standard deviation σ/μ for test data over the number of clusters.

to this, for the earth mover's distance the outlier mean is farther than one normal data standard deviation away from the normal data mean in both cases although the difference is much higher in case of Test Set 1.

The behaviour of both the histogram intersection distance and especially the earth mover's distance in respect to the number of clusters is relatively smooth. Both distances average values converge towards their final values quickly and reach them at about $k = 20$ for the histogram intersection distance and $k = 10$ for the earth mover's distance. The above average distance shows the same jumps as it has with the reference data. We suspect that the number of histograms that can be computed from the training silhouettes and that make an actual difference in the distance computation is limited. This is due to the way the above average distance is computed, taking into account an binary distinction for every bin only.

Overall it seems like the absolute distance between the mean of the normal data anomaly scores and the outlier data anomaly scores is more or less constant. In theory we expect this to manifest in better results for values where the normal data mean

is the lowest. However, as Figure 6.17 shows in the same way as Figure 6.2 for the training data, the relative standard deviation tends to be larger for both normal and outlier data if the mean gets lower. This could negate the effect described above.

For the earth mover's distance, Figure 6.17 shows that the influence of the number of clusters on the normalised standard deviation is negligible, especially for Test Set 2, where the value is more or less constant over all possible values of k. Also, in general we see, that the normalised standard deviation for Test Set 2 shows more similar values than for Test Set 1. In comparison, the normal data standard deviation is higher on Test Set 2 while the outlier standard deviation is lower. This general distinction in behaviour between the two Test Sets is reflected by above average distance and the histogram intersection distance as well.

The histogram intersection distance shows a slight rise in both normalised standard deviations on both data sets with a rising number of clusters. On both Test Sets we see a stabilisation of the standard deviations for a number of clusters of around 40. For lower numbers of clusters we see both an increase in normal data standard deviation as well as relatively strong jitters between single steps on the x axis. On Test Set 1, it is noticeable that the normalised standard deviation for outlier data rises faster than the normalised standard deviation for the normal data, namely by an overall value of 0.05 between $k = 1$ and $k = 61$ while the change for the normal data is in the region of 0.02. On Test Set 2, however, this behaviour is not replicated. In this case, both types of data behave more or less the same, albeit on a higher level for the outlier data.

Finally, the above average distance again shows the distinctive jumps between different numbers of clusters that have already shown in Figures 6.1, 6.2, and 6.14. Despite the general difference in normalised standard deviation between outlier data and normal data, the changes are more or less uniform in the sense that if there is a rise in normalised standard deviation of the normal for a certain number of clusters, we see a similar rise for the normalised standard deviation for the outlier data. As a consequence of this, the curves for normal data and outlier data seem to be more or less parallel to each other. This, again, leads to the assumption that the above average distance is rather sensitive to the number of clusters in comparison to the other distance functions due to a relatively small amount of effectively different histograms.

In terms of overall difference between outlier and normal data, the above average distance offers the weakest distinction, while the earth mover's distance has the strongest distinction. The histogram intersection distance is in the middle here. At least for $k = 1$ we have already seen in the previous section, that this behaviour is reflected in the overall quality of the results for both F_1 score and F_H score.

For the evaluation of the actual performance of the different distance functions and cluster numbers, we evaluate the maximum F_1 score for each distance function and each number of clusters. For Test Set 1, we see in Figure 6.18 that all distance function show relatively high oscillations for low number of clusters up to about twenty clusters. After that, at least the histogram intersection distance and the earth mover's distance stabilise. Apart from three unusual results for $k = 23$, $k = 31$, and $k = 38$, the above average distance reaches a stable level, too. For these unusual data points clusters are found that have centroid histograms which fit the data in Test Set 1 especially well in the same way as some of the clusterings with smaller numbers of clusters. We also see, that both the earth mover's distance as well as the above average distance reach their absolutely best F_1 score at a number of clusters of nine

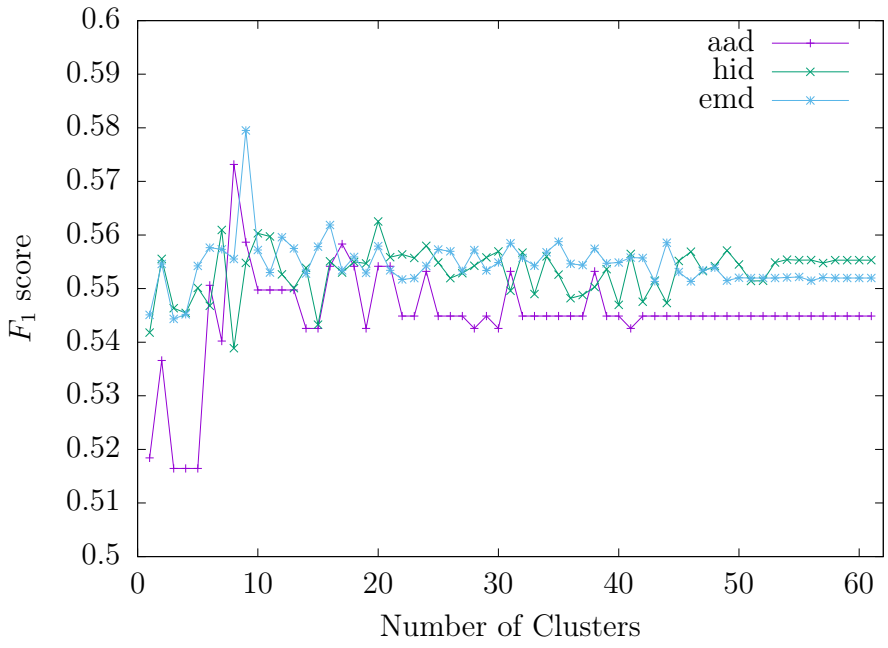


Figure 6.18: Maximum F_1 score over the number of clusters on Test Set 1.

and eight, respectively. The hid reaches the absolute best F_1 score for $k = 20$, though the difference between that best value and the second best value is much lower than for the other two distance functions. The hid's best value is also the worst best value of all three distance functions. Apart from these best values the histogram intersection distance and the earth mover's distance are mostly on similar levels while the above average distance performs about 0.01 worse. Due to the high number of observations we believe that this difference, albeit only small in its results, is significant.

While all three distance functions do profit from the multi-reference approach, it is noteworthy that the above average distance gains the most. For $k = 1$ its best value is more than 0.02 worse than that of the histogram intersection distance, however it then momentarily surpasses both of the other distances at $k = 8$ before staying in the same region as the other distances until about $k = 20$. As with the training data, we again note that for many numbers of clusters, the aad returns similar results. The reason for this again is the limited number of histograms that make an actual difference in regard to distance computation under the above average distance. The histogram intersection distance and the emd do not loose as much F_1 score for higher numbers of clusters, however, it shows that for all three distances the best results are reached for a relatively low number of clusters, as discussed above.

Interestingly, Figure 6.19 shows a rather different behaviour for Test Set 2 than we have seen for Test Set 1. Most notably is the fact that in this case the histogram intersection distance performs on a lower level than on Test Set 1 while the other two distance functions are better than before. This leads to the hid being the worst option on Test Set 2. It is also noteworthy that all distance functions reach their best values for either 22 clusters in case of the earth mover's distance and the aad, which is much higher than for Test Set 1, and for 23 clusters for the histogram intersection distance. Also the above average distance is stable on the best value, in this case reaching the

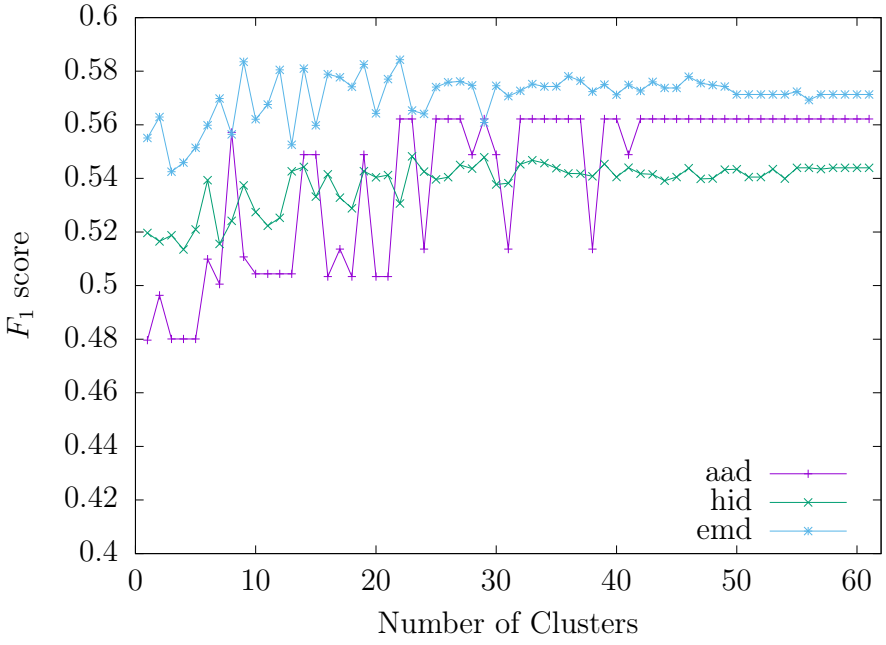


Figure 6.19: Maximum F_1 score over the number of clusters on Test Set 2.

same value for all numbers of clusters between 22 and 61 expect for 24, 28, 30, 31, 38, and 41.

As we have discussed before, in contrast to Test Set 1, Test Set 2 contains both segmentation errors and obstacles. Therefore, we reckon that due to the different kinds of outliers a higher number of clusters provides better matches on this set. It also seems, that for the earth mover's distance and the above average distance, obstacles are easier to detect than segmentation errors, hence the overall better scores apart from the absolute best value on Test Set 1. The histogram intersection distance seems to be better suited to segmentation errors as it is the only distance function that yields worse values for Test Set 2 than for Test Set 1.

In terms of clustering we see that overall for all distance functions and both Test Sets that the multi-reference extension has a positive influence on the overall outlier detection results regarding F_1 score. Also it is noteworthy, that in all cases the best results have been achieved by using between eight and 23 clusters. For higher numbers of clusters the results either stay more or less stable or even recline slightly such that we can conclude that a certain aggregation in terms of reference data has a positive impact on the actual outlier detection.

In order to get a clearer picture of the effects of the clustering of the reference histograms on the F_1 score, some further analysis is necessary. Above, for every distance function and every number of clusters the best combination for l, t_s, and t_w has been plotted. While this does take the fact that different number of clusters can lead to different anomaly score distributions into account it is hard to separate the impact of the clusters from this.

In order to overcome this limitation, we investigate the parameter configurations that yield the best single result for each distance on Test Set 1, aad1, hid1, and emd1, as well as the parameters configurations that gave the best results on Test Set 2, namely

dist	Test Set 1			Test Set 2		
	l	t_s	t_w	l	t_s	t_w
aad	1	4.0	0.25	1	2.0	0.25
hid	1	4.0	0.25	3	1.5	0.25
emd	4	4.5	0.25	2	2.5	0.25

Table 6.9: Choice of parameters for best results.

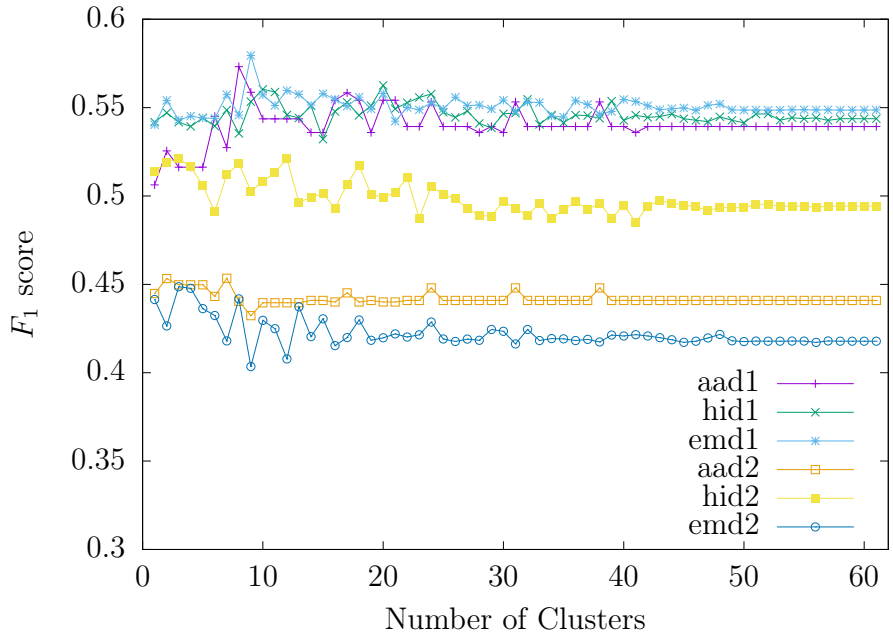


Figure 6.20: Best F_1 scores over the number of clusters on Test Set 1.

aad2, hid2, and emd2. The parameters for each instance are shown in Table 6.9. It is obvious from these values, that Test Set 1 works well with an inner threshold close to the tested maximum while Test Set 2 rewards lower values. The weak anomaly threshold t_w is chosen as 0.25 in all cases, which maximises recall.

Figure 6.20 shows the results of these six parameter sets on Test Set 1. Even on single parameter sets aad1 shows the most influence from the number of clusters, overall. It starts slightly above the 0.5 mark for one centroid reaches its high point at $k = 8$ and then falls to a value of about 0.53, where it stabilises. The earth mover's distance and the histogram intersection distance in the form of emd1 and hid1 are much less influenced by the number of clusters. The earth mover's distance in this case has a single spike at $k = 8$, while it hovers around a F_1 score of 0.55 for most of the other values. In general, it stabilises on that value after the spike for a greater number of clusters while the values before the spike are slightly lower. The highest values are reached around the the spike, e.g. in a range of $k \in [6, 20]$. For the histogram intersection distance the parameter set established on Test Set 1 starts very close to emd1 and emd2, except for the absolute height of the peaks, and stabilises somewhere between emd1 and aad1 without showing a huge impact from the number of clusters.

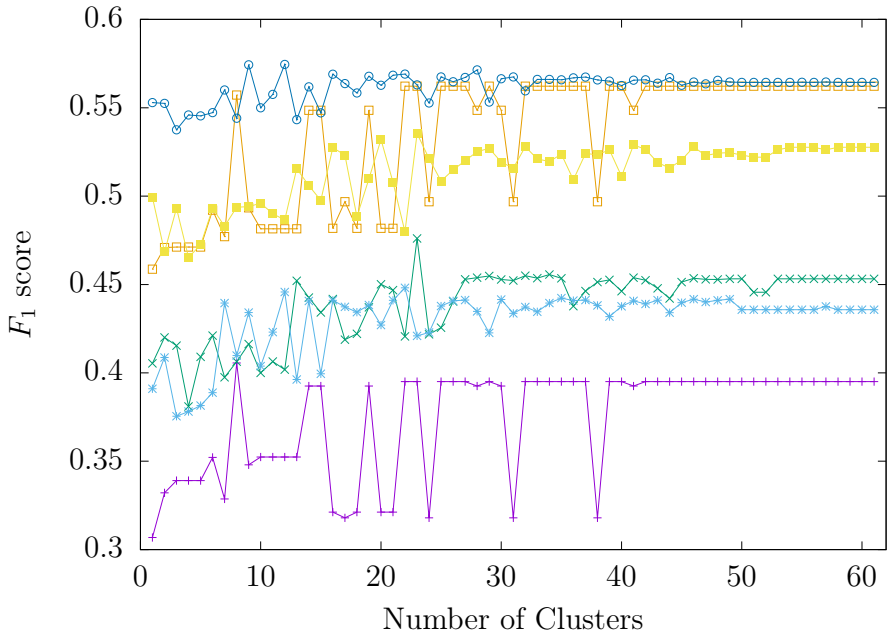


Figure 6.21: Best F_1 score over the number of clusters on Test Set 2. For key see Figure 6.20.

Interestingly, for the variants which are optimised on Training Set 2 the number of clusters in general seems to have a negative impact. For all three distance functions, the best values are reached for $k = 2$ in case of aad2 or $k = 3$ for the histogram intersection and earth mover's distance. In case of the above average distance, results stabilise for ten or more clusters at an F_1 score of about 0.44 while the best values are slightly above 0.45. The earth mover's distance starts with similar, if slightly worse values for less than ten clusters, though with higher oscillations, and then slowly settles for values of around 0.42 F_1 score.

Finally, the histogram intersection distance shows the best values for lower numbers of clusters with strong oscillations as in case of the earth mover's distance. However, in contrast to that, it stabilises later, at about $k = 24$ and from then on slowly stabilises to a value of slightly under 0.5. The best values are in the region of 0.52 to 0.53. In general, the hid2 is situated in the middle between the other sets that have been optimised on Test Set 2 and the ones that have been optimised on Test Set 1.

Figure 6.21 shows the same evaluation as Figure 6.20, only this time for Test Set 2. We see, that again the earth mover distance with the highest value on Test Set 2, emd2, performs the best overall. The highest value is reached for 0 clusters with strong oscillations between $k = 7$ and $k = 16$. After that the values begin to stabilise to values of about 0.56 F_1. The above average distance stabilises on very similar values starting at $k = 22$ though with some larger downward spikes. Before that, starting from eight clusters there is a phase where aad2's results jump around heavily. For lower number of clusters the results are more similar to each other, albeit on a very low level, starting with a F_1 score of 0.46 with one cluster. This shows, that once again, the above average distance is the most sensitive to the number of clusters used.

The hid2 parameter set also starts in a region between 0.46 and 0.5 for low numbers

dist	l	t_s	t_w	k	F_1
aad	2	2.5	0.25	22	0.5306
hid	1	3.5	0.25	13	0.5424
emd	4	3.0	0.25	9	0.5641
aad1	1	4.0	0.25	8	0.5119
hid1	1	4.0	0.25	23	0.5293
emd1	4	4.5	0.25	9	0.5296
aad2	1	2.0	0.25	8	0.5262
hid2	3	1.5	0.25	27	0.5103
emd2	2	2.5	0.25	19	0.5422

Table 6.10: Best combined parameter sets for F_1 score on both test sets.

of clusters and show heavy weaving until 25 clusters, where the best value is reached for $k = 23$. Then values begin to stabilise at about 0.53. This makes hid2 the worst of the parameter sets that have been optimised on Test Set 2. Overall, however, it also is the only parameter set that yields multiple points with an F_1 score higher than 0.5 on both datasets.

Among the datasets optimised on training set 1, hid1 shows the best values. It has the highest peak for 24 clusters and also stabilises at slightly over 0.45. Apart from that there are again strong oscillations in the first half. Overall emd1 behaves very similar, albeit on a lower level and without the high best value shown by hid1. The first parameter set for the above average distance performs the worst. Its behaviour more or less mirrors that of aad2, only for values that are about 1.6 worse in F_1 score.

Overall we see that parameter sets that perform well on one of the datasets are usually worse on the other data set, thus indicating some overfitting. Another indicator for overfitting is the fact, that hid2 does not lead to exceptional results on either data set, however, its results are the most comparable over both test sets.

Table 6.10 shows the best parameter combinations for each distance function over both test sets, where both test sets are weighted the same. In the lower part of the table, the combined results of the best parameter combinations on the single test sets are shown. We can see that again, in all cases t_w is set to 0.25, e.g. the lowest value, while values for l and t_s are more in the middle of the range. The results themselves are in the usual order, where aad performs the worst, hid is about 0.01 better and finally the earth mover's distance again is 2 percent better. For the separate parameter combinations, emd2 with 19 clusters performs the best. If we compare the chosen values to those of the overall best emd settings, we see, that they are relatively close together with l lowered from 4 to 2 and t_s from 3.0 to 2.5. Only in number of clusters a larger difference arises, with emd working best on 9 clusters and emd2 on 19. From this it follows that emd2 has a lower precision and higher recall than the absolute best settings.

Overall we see that in respect to F_1 score the multi-reference extension improves AdaMS OD's results. In all cases, moderate numbers of clusters in the region between 8 and 27 clusters provide the best results. This does support the assumption formulated in Section 4.4, that the right amount of aggregation is important. If very little aggregation is used, i.e. a very high number of clusters is used, mostly precision suffers while

dist	l	t_s	t_w	k	F_H
aad	1	3.0	0.25	22	0.5367
hid	1	3.5	1.25	16	0.5646
emd	1	4.5	0.25	25	0.5924
aad1	2	3.5	1.5	17	0.5085
hid1	1	3.5	1.25	16	0.5646
emd1	1	4.5	0.25	16	0.5918
aad2	2	1.5	0.5	24	0.4126
hid2	2	2.0	0.25	9	0.4669
emd2	2	4.0	0.25	25	0.5901

Table 6.11: Best combined parameter sets for F_H score on both test sets.

recall on average improves. If a strong aggregation in the form of only one reference silhouette or very few clusters is used, recall suffers but precision is improved.

Since Section 6.2 shows a strong distinction between the common F_1 score and the F_H score introduced in that section, in the following we will discuss the influence of the multi-reference approach on that measure as well. Similarly to Table 6.10, Table 6.11 shows the parameter sets for each distance function that retrieved the best results on both data sets, aad, hid, emd, as well as parameter combinations that lead to the best results on the single training sets, marked with 1 and 2, respectively. All F_H scores given in the table are computed for both test sets. Overall we see a similar effect on the choice of parameters as with the F_1 score. In general, the values of t_s resulting in the best outlier detection are between those for Test Set 1 and Test Set 2, with Test Set 1 preferring higher values than Test Set 2. Interestingly, F_H supports higher values for t_w in some cases, which were 0.25 in all cases for F_1 score. As already mentioned in the evaluation of the base algorithm, this is a consequence of the fact, that in order to declare an outlier as hit – and therefore execute a resegmentation of that part of the silhouette in context of the AdaMS framework – does not depend on the size of the overlap. This leads to higher values for t_w in some where an additional size for outliers does not help finding further outliers that would not have been hit otherwise.

In terms of F_H score, Test Set 1 seems to be more relevant to the overall results than Test Set 2. We can see this from the fact, that for the histogram intersection distance the chosen parameters for these two cases are identical, while for the earth mover's distance only the number of clusters deviates. Only the above average distance shows higher values for all parameters apart from the number of clusters for Test Set 1. Apart from emd2 where very similar values to the other cases where chosen and a very similar result overall to the other emd parameter sets was achieved, we see that the parameter sets optimised on Test Set 2 are performing poorly on both datasets combined. The results for the earth mover's distance's parameter sets are all very similar, ranging from 0.5901 to 0.5924 and are the best results, being 0.03 better than the histogram intersection distance with the overall best parameter set that is equal to the best parameter set on Test Set 1. The overall best parameter set for the above average distance is another 0.03 worse than that at 0.5367. This is then followed by the aad2 parameter set. Finally, hid2 and especially aad2 with an F_H score of 0.4669 and 0.4126, respectively, represent the worst presented choices.

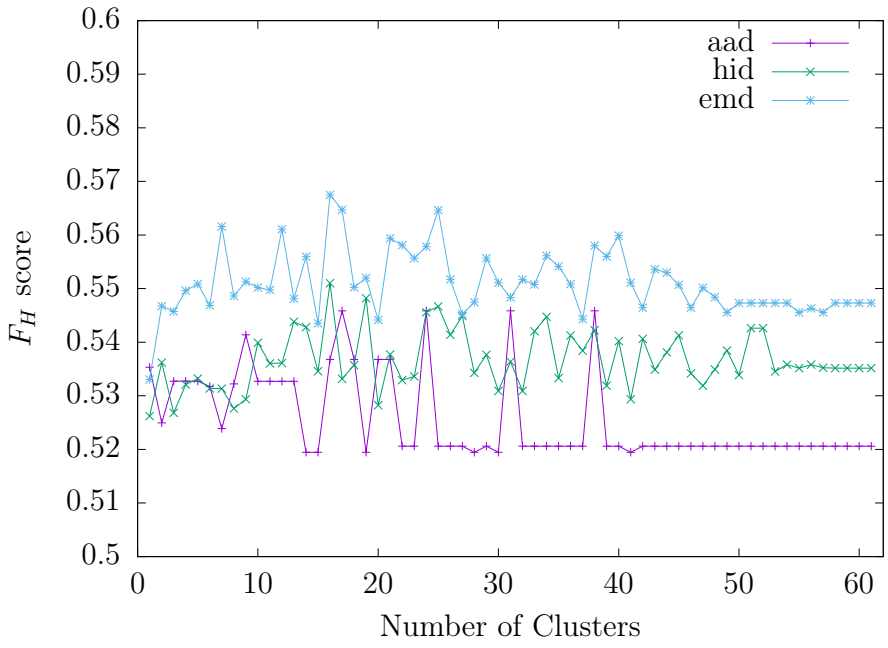


Figure 6.22: Maximum F_H score over the number of clusters on Test Set 1.

Apart from hid2, where a value of $k = 9$ was chosen, all used numbers of clusters lie in the range of 16 to 25. By this we see that the number of clusters for the optimal parameter sets is closer together than when trying to improve F_1 score and also that the average optimal number of clusters is slightly larger. Overall the results presented show once again, that clustering of reference data has a positive impact on the results of the algorithm as in all cases numbers of clusters in the middle range yield better results than the results without clustering presented in the previous section.

In order to further investigate the influence of the number of clusters on the outlier detection, Figure 6.22 shows the maximum F_H score for each number of clusters and distance function. We see that for the earth mover's distance the number of clusters has a greater influence on the F_H score than on the F_1 score, even if there are strong oscillations for huge parts of the graph. In fact the values only start to stabilise at about 45 clusters and then on a relatively low level compared to the numerous high points especially in the lower half, i.e. up to 30 clusters. The best value, as already has been shown in Table 6.11 is reached for 16 clusters. The lowest value by some margin is reached for 1 reference silhouette, that is for the base algorithm. Apart from this occurrence the lowest values overall are reached in some spikes and at the stabilised end with a high number of clusters.

The histogram intersection distance behaves similar to the earth mover's distance though on a lower level of performance. Again, the worst result is achieved for one cluster, even if in this case by a smaller margin, and the best value is reached for 16 clusters in an area with strong deviations. In case of the hid the stabilisation of the values is reached even later than in case of the earth mover's distance beginning at 53 clusters. In contrast to the emd and also the above average distance, the hid stabilises at values that are only slightly below the mean of the spectrum.

For the above average distance, the last values, starting at 39 clusters also mark

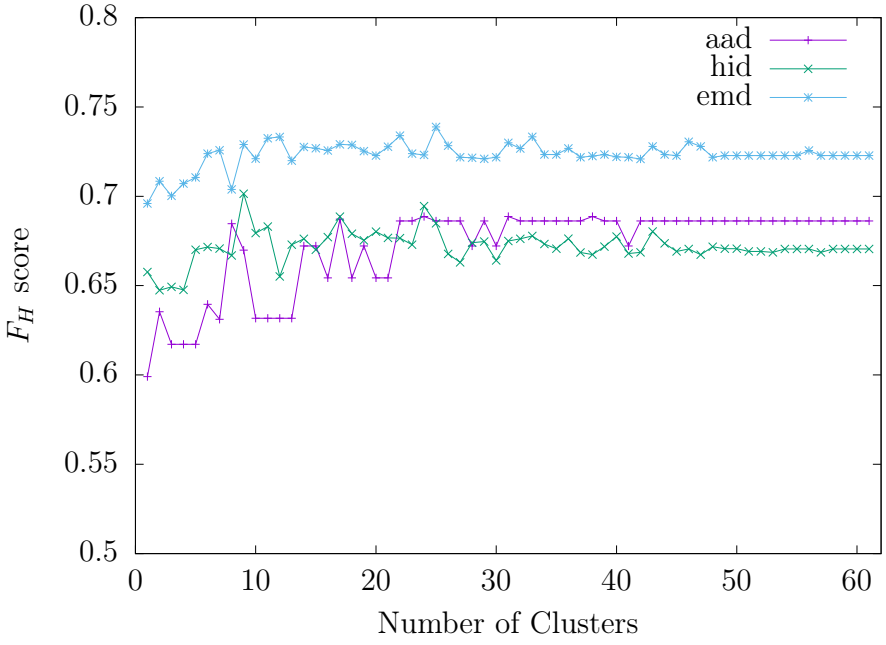


Figure 6.23: Maximum F_H score over the number of clusters on Test Set 2.

the lowest values, together with similar values earlier on. Overall, we see the usual behaviour of the above average distance, i.e. huge steps between different values but overall a low number of different reached scores. There are four high spikes between 17 and 38 clusters, that reach relatively good values and a slightly lower one for 10 clusters. Apart from these the scores are about the quality of the hid for up to ten clusters and are lower from then on. Interestingly, the value for $k = 1$ is the best of all three distance functions, as discussed in the previous section. In that sense it could be counted as one of the high points of the aad.

Overall it is noteworthy that for all three distance functions the best values on Test Set 1 are worse than the F_H scores over both datasets as shown in Table 6.11 for the same parameter combinations. This is a contrast to the results regarding F_1 score. If we take the number of outliers of certain lengths, as given in Table 6.2, into account, we see that for Test Set 1 65 of 101 used outliers have a length of less than 20 points. These outliers are represented more strongly by the F_H score than the F_1 score, which is more of a point-wise measure.

Figure 6.23 shows the same evaluation for Test Set 2 as Figure 6.22 does for Test Set 1. First of all we see much higher results for this data set, thus supporting our assumption that larger outliers are easier to detect. Another apparent observation is the fact that all three curves are much smoother than in case of Test Set 1, which indicates that a wider variety of parameters perform decently on this set. The best results are once more reached by the earth mover's distance with the best value reached at $k = 25$. The worst value the earth mover's distance reaches is once again with one cluster and values stabilise beginning at $k = 34$ at an F_H score of about 0.725. Between $k = 1$ and $k = 11$ the scores rise on average, and begin to plateau after that.

The histogram intersection distance starts about 0.04 lower than the earth mover's distance and then falls further until $k = 5$. The highest peak is then reached for 9

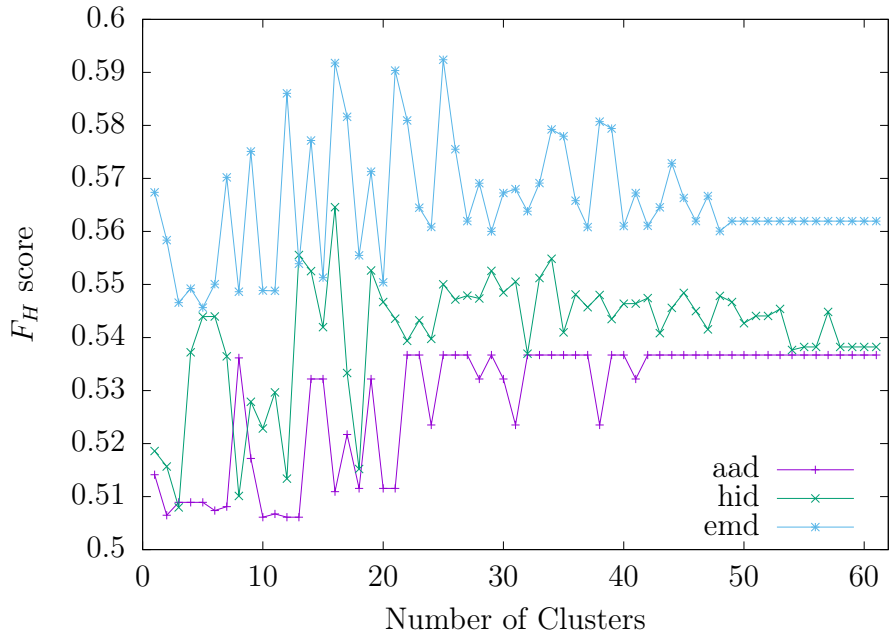
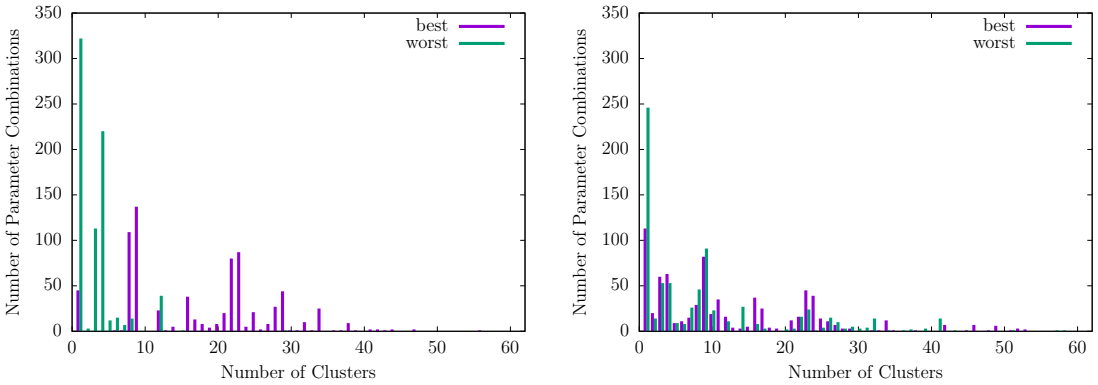


Figure 6.24: Best F_H scores over the number of clusters on both test sets.

clusters with a value of 0.7 which is followed by a period of oscillations until 31 clusters are reached. From there on the F_H score stabilises at around 0.67. The above average distance in this case starts with the lowest value for one cluster at an F_H score of 0.6 and then quickly rises to its first maximum of about 0.68 at eight clusters. This is then followed by a huge drop in results and then some steps upwards until a plateau is reached for 22 clusters. Apart from three smaller drops the values stabilise here, with three values that are minimal better at $k = 24, 31$, and 38. The last of the three drops happens at $k = 41$. This behaviour leads to the fact that the stabilised value of the aad is reached the earliest of all three distance functions and it also is slightly better than that of the histogram intersection distance.

In Figure 6.24 the best parameter sets from Table 6.11 are plotted over the number of clusters. In contrast to the figures before we here only change the number of clusters per series instead of choosing the best parameter combination for each distance function and number of clusters. For all three distance functions huge oscillations are noticeable especially up to the thirty clusters mark. In case of both the earth mover's distance and the histogram intersection distance also the best overall values are reached in this part. For the above average distance, as before with the F_1 score, the best values are reached for multiple values, the first of which is $k = 8$. The same value is then again reached multiple times beginning from $k = 22$ and finally also stabilises there. It is noteworthy at this point that these oscillations while close in terms numbers of clusters to the oscillations we noted with the aad in Figure 6.18 are for different numbers of clusters. The other two distance function show lower oscillations for values larger than 30 clusters and stabilise at rather low values compared to the best values, especially for the earth mover's distance. For both the histogram intersection distance as well as the above average distance, the final values are significantly better than those for only one cluster. However, in case of the emd this is not the case. Here, the results for

one cluster are about half a percent better than those of the stabilised values, though beginning at $k = 7$ clusters the first value is bettered.



(a) F_1 score (b) F_H score

Figure 6.25: Best and worst values per number of clusters.

Tables 6.10 and 6.11 already showed that neither for F_1 nor for F_H score the best parameter combination uses only one clustering. For every number of clusters, Figure 6.25 shows the number of parameter combinations in regard to l, t_s, and t_w in which that number of clusters performed best or worst. It is obvious that especially for one cluster the number of worst results is extremely high. In respect to F_1 measure as Figure 6.25a, the majority of worst results is located for a number of clusters between 1 and 5. The number of best clusters is more evenly spread. As is noticeable from Figure 6.25b the spreading in respect to F_H score is more even as that for F_1 score though the main characteristics are the same. Again $k = 1$ has the highest number of worst results and the majority of worst results spreads to to about 10 clusters. However, in this case, $k = 1$ also has the highest number of best results.

In terms of quality of feature combinations however it holds that the best feature combination using one cluster in respect to F_1 is only the 774th best and the 410th best regarding F_H score. This further supports our claim that a clustering of the reference histograms in order to get a better trade-off between aggregation and detailing improves the outlier detection quality.

6.4 Evaluation with Outlier-type Distinction

In this section we will discuss the results of the second proposed enhancement for the outlier detection algorithm. The basic idea of this approach is to use a different set of features for miscellaneous kinds of outliers. In the following, we will first evaluate the results of the contrast features, separately. Then, we show the performance of the combined algorithm.

The assumption that forms the basis for this enhancement is the idea, that segmentation errors and obstacles have different properties regarding the positional as well as contrast-wise features. More specifically, we suspect that obstacle have high contrast values as obstacles are part of the foreground but unusual overall forms. Otherwise,

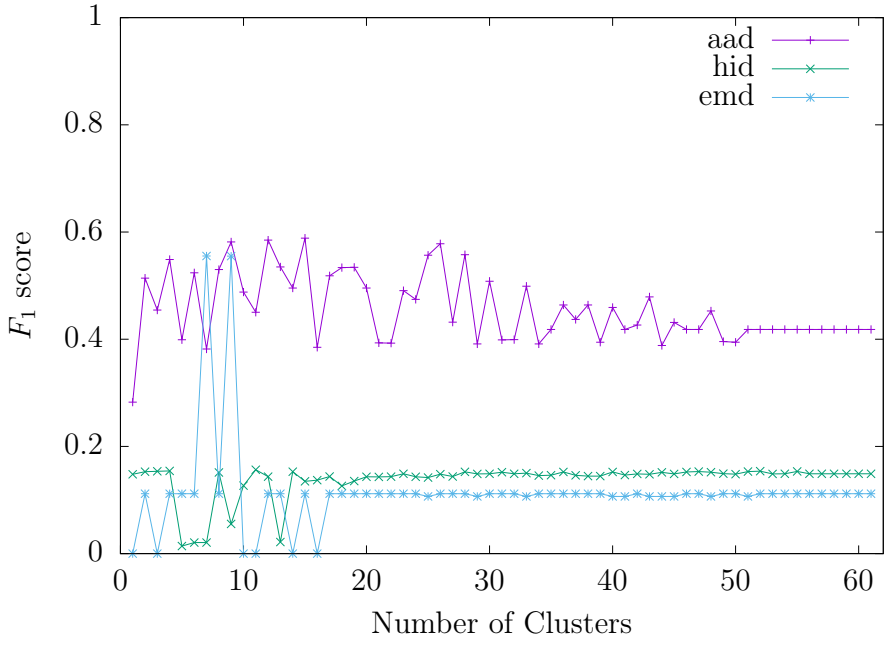


Figure 6.26: Maximum F_1 scores with contrast features on Test Set 1.

segmentation errors can have usual forms as well as unusual ones, but have lower contrast values at least in parts as their borders are either in the sky or in the foreground. If these assumptions hold, the results for the contrast features alone should be strong on Test Set 1, as this consists of segmentation errors, only, and worse on Test Set 2, as this set includes both segmentations errors and obstacles.

However, Figure 6.26 shows overall bad results for Test Set 1, especially for the histogram intersection and earth mover's distances, that in most cases have results below an F_1 score of 0.2. The above average distance performs significantly better, mostly reaching F_1 scores between 0.4 and 0.6. If we compare these values to the F_1 scores of the positional features on Test Set 1 as shown in Figure 6.18, we see that the best values of the contrast features are slightly better than those of the positional ones, although the positional features are much more robust to parameter choice. The best value for the contrast features is reached with the above average distance at 15 clusters with a F_1 score of 0.5886, while for the positional parameters a maximal F_1 score of 0.5795 for 9 clusters is reached under the earth mover's distance.

One of the reasons for this behaviour lies in the fact that the images in Test Set 1 were scaled down from higher resolutions to a resolution of 1024×768 pixels using ImageMagick in its standard settings which uses linear colour interpolation. Thus, by resizing the images the contrast at the borders between foreground and background gets reduced. This makes the detection of segmentation errors in this dataset particularly difficult. By only comparing the frequent bins in the histograms, the above average distance implements an effect similar to a sharpening filter, which effectively increases contrast. This explains the better scores in comparison to the other two distance functions. The same argumentation also gives an explanation to the fact that the histogram intersection distance works better than the earth mover's distance: While the hid works on all bins of the histograms, it only compares bins that have the same

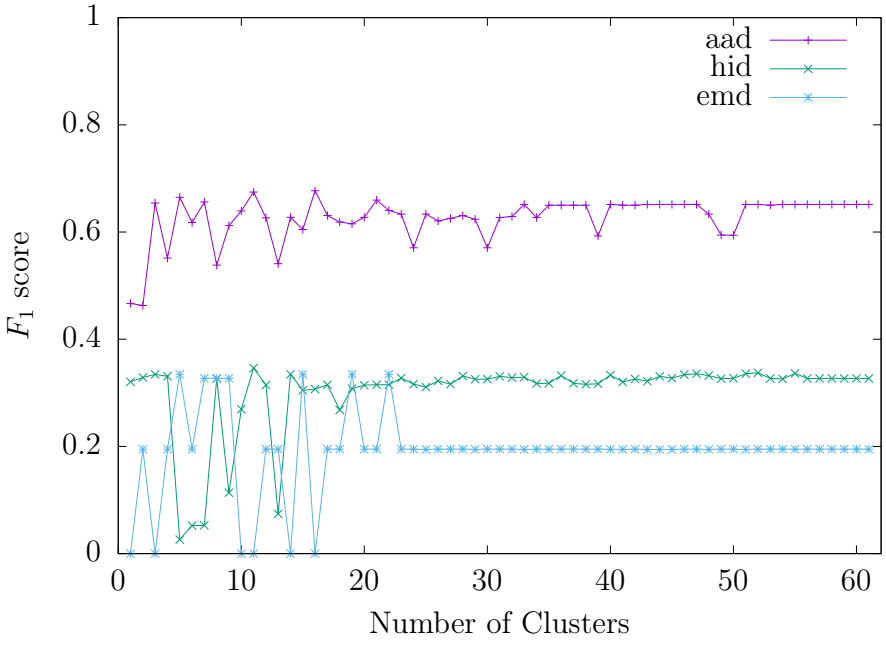


Figure 6.27: Maximum F_1 scores with contrast features on Test Set 2.

borders. The earth mover's distance additionally applies a softening to this by taking bins in a closer proximity into account.

The results for Test Set 2, presented in Figure 6.27, are similar to those for Test Set 2, albeit on an overall higher standard. In fact, this change is higher than the increase in F_1 scores we saw in the positional features on Test Set 2 in comparison to Test Set 1. Apart from this, the above average distance outperforms the hid and emd again, this time with values that surpass the best values on the positional parameters. There are also less jumps in F_1 score for the different numbers of clusters for the above average distance in contrast to Test Set 1. The largest jump for the above average distance happens from two clusters to three clusters, with an F_1 increase of 0.2.

While still trailing the aad by a huge margin, the histogram intersection distance performs about twice as good om Test Set 2 than on Test Set 1. Apart from some jumps between five and 14 clusters it performs very stable on this dataset. The earth mover's distance, too, performs about twice as good on this dataset than on Test Set. The best results reach a F_1 score of about 0.35, which is in the region of the best scores of the hid. However, with a stabilized F_1 score of about 0.2 for 23 and more clusters, it still produces very weak results.

These results support our thesis that relative to each other, the different distance functions implement a varying degree of softening respectively sharpening. While this also happens when applying the distance functions to positional features, the effects are contrary: On the positional features, a softer comparison yields better results than a sharper comparison, while on contrast features, a sharper comparison improves the results over a softer approach.

Figure 6.28 shows the F_H score for the contrast features on Test Set 1. Overall, the F_H scores behave similar to the F_1 scores. Again, the above average distance shows the best results, with maximum scores surpassing 0.6. The best result is reached for 10

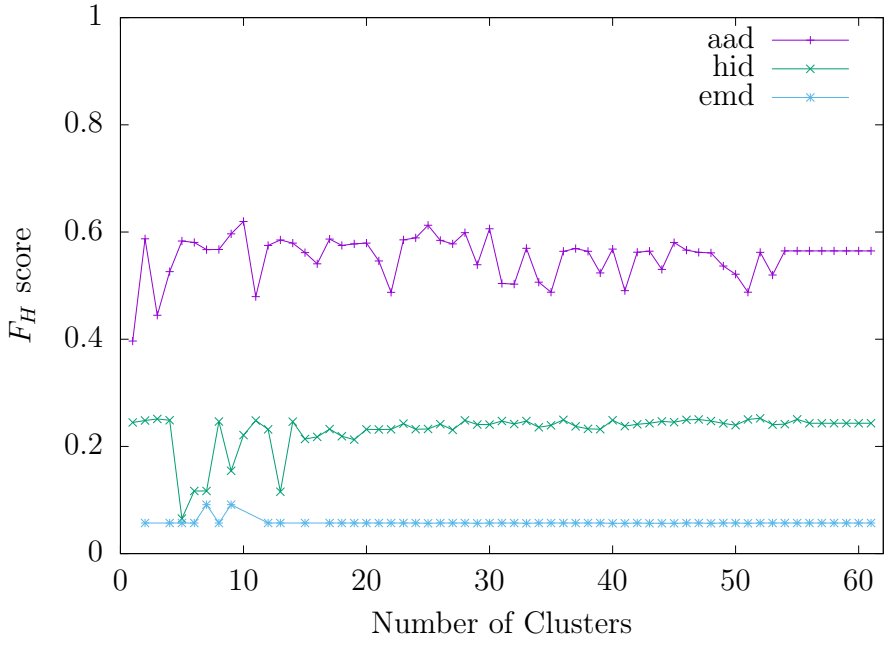


Figure 6.28: Maximum F_H scores with contrast features on Test Set 1.

clusters and surpasses all results of the positional features as given in Section 6.3. The histogram intersection distance reaches F_H scores of about 0.25, that are stable except for some oscillations between five and 14 clusters. Finally, the earth mover's distance, again, proves to be the worst choice in case of the contrast features. Its result never surpass a F_H score of 0.1.

As the F_H score for both the aad and the hid are higher than the corresponding F_1 scores, we conclude that precision in these cases is higher than recall. For the outlier detection on mountain silhouettes, this behaviour is desired, as we described argued in previous chapters. Since the adaptive algorithm re-evaluates a larger section of a silhouette than only the detected outlier, a complete detection of the outlier is not necessary. While higher recall values could be produced by lowering t_w, this could cause a higher false positive rate. This, in turn, would instantiate unnecessary corrections of silhouettes and could in fact induce errors to parts of silhouettes that were fault-free in the first place.

For the earth mover's distance, however, the F_H score is lower than the F_1 score. This shows, that the outliers that are detected by the emd are detected in a large portion, but only few outliers are detected overall. This result shows, that the general characteristic of the earth mover's distance, that shows a higher ratio of recall to precision than the other two distance functions, is invariant of the underlying features.

The F_H scores for Test Set 2 are depicted in Figure 6.29. Here, the above average distance reaches very high and stable F_H scores, that surpass 0.8 for a multitude of cluster numbers for 30 clusters and less. After that, the values stabilise at about 0.75. The F_H scores of the earth mover's distance are improved, too. They are now slightly more than 0.2 in most cases with some spikes that lead to scores in the region of 0.3. Still, the performance of the earth mover's distance is very weak on the contrast features.

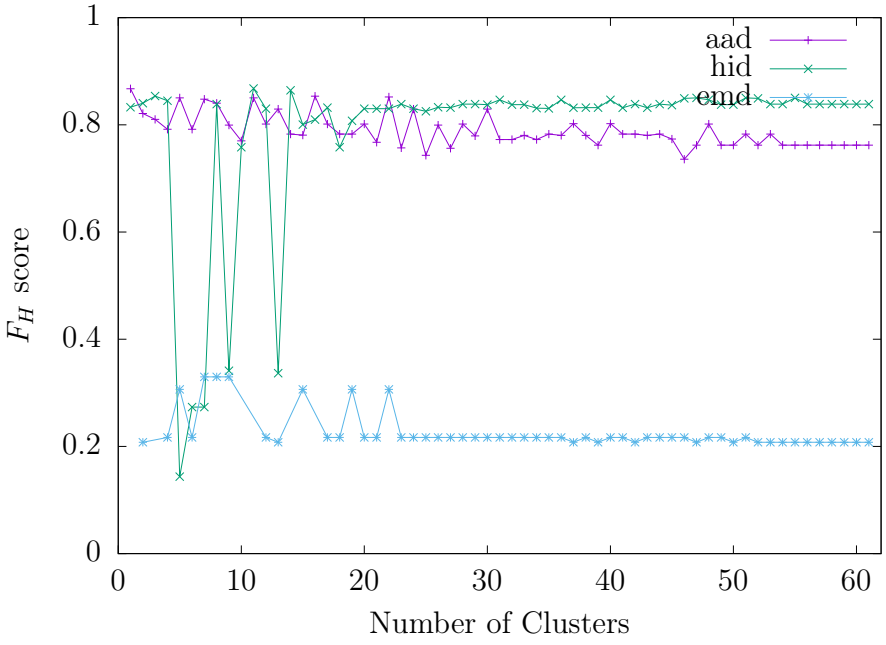
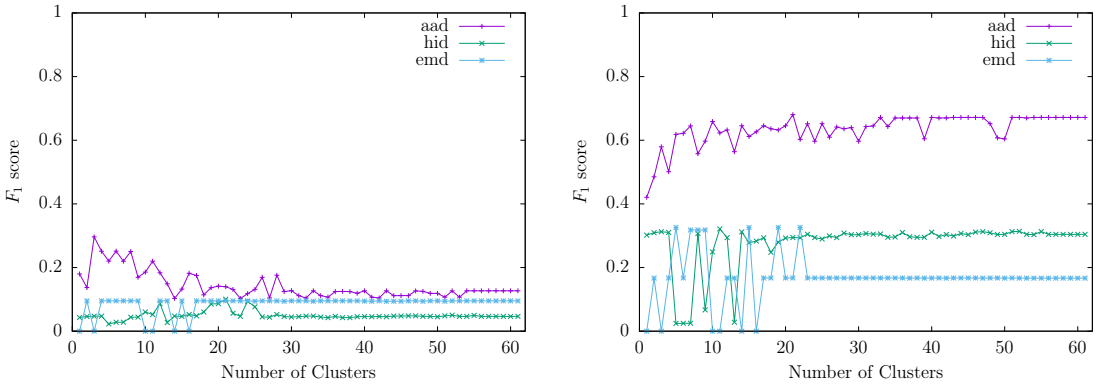


Figure 6.29: Maximum F_H scores with contrast features on Test Set 2.

The performance of the histogram intersection distance is somewhat surprising, as it reaches the highest overall scores, with a maximum of 0.8679 at 11 clusters, and also the highest stabilised score. Between five and 18 clusters, there are some heavy oscillations, with a lowest values of 0.1437 at five clusters and some other values that go below 0.4 as well. The value at five clusters interestingly is the worst value achieved by all three distance functions. In contrast to the erratic behaviour with less than 20 clusters, F_H scores then stabilize quickly at about 0.85. This stabilized value, as well as the above average distance's stabilised value, is higher than all values reached by positional features on both datasets.

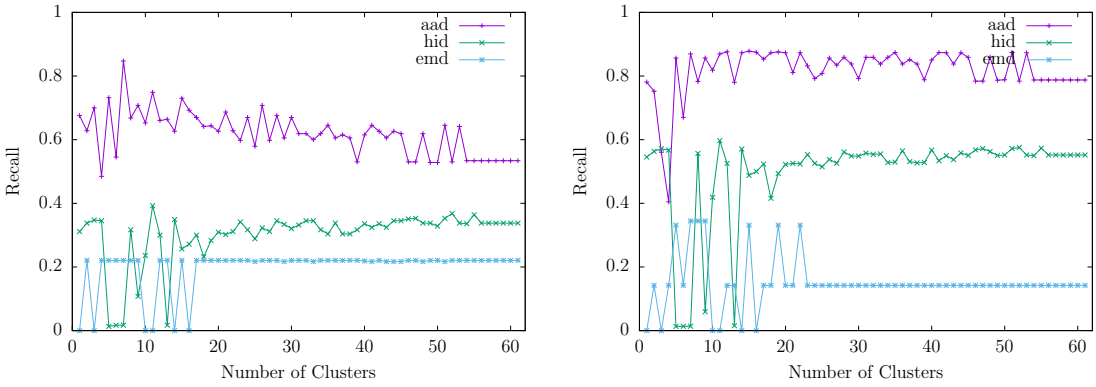
At first view, the good results of contrast features on Test Set 2 are surprising, as this set contains obstacles as well as segmentation errors. Before the experiments, we expected the contrast features to be especially proficient on segmentation errors, due to the general changes in contrast that occur in these situations. However, as the results in Figures 6.28 and 6.29 show, these features are able to detect obstacles as well. An explanation for this is due the fact, that the contrast features contain a certain amount of positional information as well, namely in the contrast in x direction and in the contrast in y direction. The quality of the results on Test Set 2 lets us hypothesise that the contrast directions of obstacles deviate from the contrast directions of normal mountain silhouettes far enough to be distinguishable by AdaMS OD.

To test our assumption, in the following we separate the outliers in Test Set 2 by their class, i.e. we evaluate obstacles and segmentation errors on their own. First, we look at the F_1 scores for the obstacles, that are depicted in Figure 6.30a. The results we see here, are extremely low in all cases, although the above average distance shows significantly better values for the range of three to eight clusters than the other distance functions. The main reason for these results lies in the fact, that the precision values are extremely low in this case. This is because of the 61 outliers in Test Set 2,



(a) Obstacles (b) Segmentation Errors

Figure 6.30: Maximum F_1 scores with contrast features of Test Set 2 separated by outlier class.



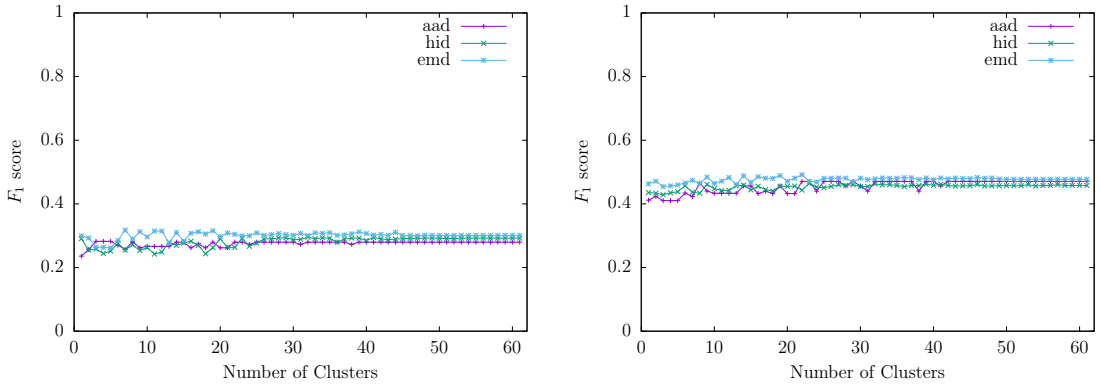
(a) Obstacles (b) Segmentation Errors

Figure 6.31: Maximum recall with contrast features of Test Set 2 separated by outlier class.

only 11 are obstacles. As segmentation errors in this case are not considered but there are outliers detected where those occur, precision is predictably low. The F_1 scores for the segmentation errors are shown in Figure 6.30b. As these are the majority of the outliers, the resulting curve is very similar to that of the F_1 scores for all outliers.

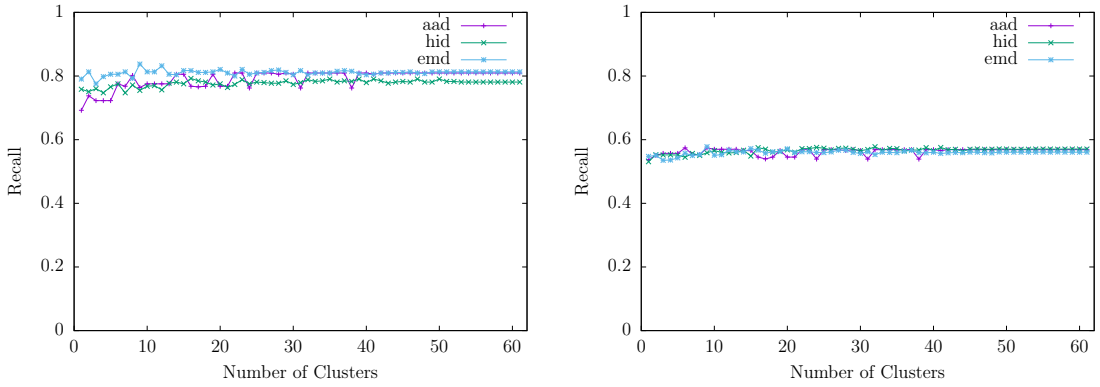
In order to get more meaningful results for these separately, we also look at the recall values for both types of outliers. These are shown in Figure 6.31. It is noticeable from these figures that the results for both types of outliers are much more similar for the recall metric. While the earth mover's distance results in overall worse recall values for segmentation errors than for obstacles, we see, that both the histogram intersection distance and the above average distance yield better results for segmentation errors than for obstacles, which is expected for the contrast features as discussed earlier.

The above average distance shows the best values in all four tests, resulting in the best results as presented previously. Especially recall on the segmentation error set shown in Figure 6.31b is impressive, where scores greater than 0.8 are reached for most clusters. Recall on the obstacle set is noticeably worse, where most values are



(a) Obstacles (b) Segmentation Errors

Figure 6.32: Maximum F_1 scores with positional features of Test Set 2 separated by outlier class.



(a) Obstacles (b) Segmentation Errors

Figure 6.33: Maximum recall with positional features of Test Set 2 separated by outlier class.

in the range between 0.7 and 0.5. The histogram intersection distance shows similar behaviour, but on a weaker level. It has better recall on the segmentation set as well, however precision, especially on the obstacles is very low. This can be seen as the hid's recall on the obstacles is higher than that of the earth mover's distance, while the F_1 scores are worse, as Figure 6.30a shows.

In contrast to the other two histogram distance functions, the earth mover's distance has a lower recall on segmentation errors than on obstacles. Again, it seems likely that this has to do with the softening properties of the emd that have more influence on the segmentation errors than on the obstacles. However, as recall and subsequently F_1 scores are very weak in both cases, coincidently better detected obstacles could be the reason as well.

Figures 6.32 and 6.33 show evaluations for F_1 scores and recall for the positional features used in Section 6.3, separated by outlier type. As expected from the results shown in Figure 6.19 on this data set, all three distance functions perform more or less the same. Also, there are no large differences in between the different number of

metric	dist	l	t_s	t_w	k	score
seg. F_1	aad	5	4.0	0.25	10	0.6804
seg. F_H	aad	4	3.0	1.00	7	0.8132
obs. F_1	emd	2	3.5	0.75	7	0.3176
obs. F_H	hid	1	3.0	2.00	1	0.2879

Table 6.12: Best parameter combinations for different target metrics.

Setting	Strategy	F_1	F_H
F_1	Merge	0.7231	0.6584
F_1	Merge to Seg.	0.7231	0.6584
F_1	Split & Merge	0.8304	0.7251
F_H	Merge	0.6350	0.7840
F_H	Merge to Seg.	0.6350	0.7840
F_H	Split & Merge	0.6482	0.8116

Table 6.13: Combined results on Test Set 2.

clusters. In all evaluations the earth mover's distance performs slightly better than the other two distance functions. More interesting at this point, we notice in Figures 6.32a and 6.32b that the F_1 scores for segmentation errors are better for segmentation errors than for obstacles. However, the difference between both outlier types is much smaller for the positional features than for the contrast features shown in Figure 6.30.

As Figure 6.33 shows, the reason for this lies again in the fact that the number of obstacles in the test set is much lower than the number of segmentation errors, which results in bad precision for the obstacle detection. In contrast to this, recall for obstacles is in the region of 0.8 for the emd for all clusters and marginally worse for the other distance functions. For segmentation errors, recall is at about 0.55 for all three distance functions over all clusters. Overall, these results show that our initial intuition was right: Contrast features are indeed better suited to detect segmentation errors, at least with the above average distance, while positional features show better recall on obstacles.

In order to evaluate the combined outlier detection, we use the parameters that yield the best F_1 scores and F_H scores on the corresponding outlier classes. This means, for the contrast features we choose the two sets of parameters that have the best scores on the segmentation error set, while for the positional features, we look for the parameters that give the best results on the obstacles. In detail, the results are shown in Table 6.12.

For the combined results, we combine the settings from seg. F_1 and obs. F_1 from Table 6.12 to detect segmentation errors and obstacles with a high F_1 score, as well as the seg. F_H and obs. F_H settings. In the following, we refer to the first combination as F_1 settings and to the second as F_H settings.

The results of all combinations of parameter settings and merge strategies are shown in Table 6.13. Unsurprisingly, the combinations with the F_1 settings result in a higher F_1 score than the combinations with F_H settings, that achieve higher F_H scores than the

Setting	Strategy	Obstacles	Segmentation Errors	Total
F_1	Merge	0.6364	0.0000	0.1148
F_1	Merge to Seg.	0.4545	0.3000	0.3279
F_1	Split & Merge	0.6364	0.2200	0.2951
F_H	Merge	0.3636	0.7200	0.6557
F_H	Merge to Seg.	0.3636	0.8200	0.7377
F_H	Split & Merge	0.6364	0.8000	0.7705

Table 6.14: Correctly classified outliers by class over all outliers.

F_1 combinations. It is noteworthy though, that the split and merge strategy results in higher scores than the two merge strategies. The reason for this are two effects. First, with this strategy the number of outliers becomes larger than with the other two strategies. Therefore, a higher chance of hitting outliers, i.e. parts of the silhouette that are detected as an outlier and intersect with an actual one, rises. Second, some of the outliers produced by this variant are shorter than ten points. In this case, as described in the beginning of this chapter, these outliers are ignored.

In general, we see that the combined results are better than the best single setting results, thus we can conclude, that using different outlier detection strategies for different kinds of outliers is benevolent. Next, we look into the classification quality that comes with the outlier-type distinction model.

Table 6.14 shows the ratios of correctly classified outliers for each type of outlier and in total. In order to achieve the results in the table, an outlier is counted as correctly classified if the algorithm outputs an intersecting outlier with the right class. This means, that outliers that have not been detected at all are not counted as correctly classified. It is also noteworthy that the number of outliers of each type is different. As mentioned above the test set contains 11 obstacles and 50 segmentation errors.

Overall the results show that the parameter settings are more important to the quality of the classification than the merging strategy. For the parameters that ensure the best F_1 scores we see very low numbers of correctly classified outliers, especially in regard to segmentation errors. As the obstacles in the test set are larger than the segmentation errors, this is reflected by the parameter settings and thus the detected outliers. The result of this are detected obstacles, that are larger than the segmentation errors on average. In cases of overlaps, this results to many of the detected outliers labelled as obstacles. Another point that leads to the low classification results is the fact, that for settings that optimise the F_1 scores, many of the shorter segmentation errors are not detected at all. In case of the F_1 merge to segmentation error strategy noted in the second line of the table this becomes especially clear: Despite the fact that an outlier is classified as segmentation error as long as a single segmentation error is detected as part of the overlapping set, less than one third of the segmentation errors are correctly classified.

When changing the settings to the parameters that result in good F_H scores, we see drastic improvements in the overall number of correctly classified outliers. While the number of obstacles that are detected, suffers for the merge and merge to segmentation error strategies, it stays on the same level for the split and merge strategy, compared to the F_1 optimised settings. At the same time, the classification of segmentation errors

is greatly improved with the F_H parameter settings. As discussed above, this is mostly due to the case that more segmentation errors have been found at all. In case of the split & merge strategy for the F_H parameters, 77 percent of all outliers have been correctly classified. For these settings, 51 of the 61 outliers in the test set have been detected by the algorithm. Taking this into account, the ratio of correctly classified detected outliers for this setting is 0.9216.

6.5 OutlierNet

This section discusses the results of the outlier detection with convolutional neural networks as introduced in Chapter 5. Both architectures have been trained with the same 61 training images that have been used to train the Adams Outlier Detection algorithms. Additional, wrong segmentations have been generated as described in Section 5.1. In all cases we trained the networks for 2000 epochs and compare the results with the training status after 1000 epochs, at which point training accuracy was already very stable.

All results have been computed on Test Set 1 as introduced in Section 6.2. It consists of 46 images that have been segmented by the AdaMS segmentation algorithm that has been introduced in Chapter 3 and includes 101 outliers that are considered in the evaluation. In order to evaluate the performance of the neural network driven outlier detection, three evaluation metrics are taken into account, namely the receiver operator characteristic, the precision and recall curve, and the F_1 score for different thresholds. All three of these metrics rely on the fact that the output of the classifier is a score from the interval $\text{pred}(v_i) \in [0, 1]$ for each vertex v_i in the silhouette. We have to chose a threshold $t \in [0, 1]$ such that a vertex v_i is regarded as part of an outlier if and only if $\text{pred}(v_i) \geq t$. By selecting different values of t, obviously different parts of the silhouette are regarded as outliers and thus the mentioned metrics are influenced.

We first analyse the receiver operator characteristic (ROC) curve as presented in Figure 6.34. The ROC curve shows the false positive rate that has to be accepted to achieve a certain true positive rate. The false positive rate is influenced by the choice of the threshold t, i.e. the lower t is chosen the more points that are not parts of outliers are regarded as outliers vice versa. For $t = 0$ every point is considered as an outlier which results in a false positive rate of one as well as a true positive rate of one, while for $t = 1$ no point is regarded as an outlier and such both rates are zero. A higher run of the curve represents a better performance as we have to accept less false positives to be able to detect a certain ratio of true positives. A weighted random process is expected to result in a curve close to the diagonal in the plot.

Figure 6.34 shows that all four variants are well below the diagonal, which means that the produced results are better than a random process. It is also noteworthy that the curves of the two network architectures behave different up to a false positive rate of about 0.6. The second network architecture achieves steeper slopes in the first part of the curve for both training states and stay noticeably higher than the corresponding curves of the first network architecture up to a false positive rate of about 0.5. Only then, the first – smaller – network architecture is able to catch up with the larger network.

A similar effect can be attributed to the number of training epochs, where we see an earlier slump of the curves in both cases for the training state after 1000 epochs

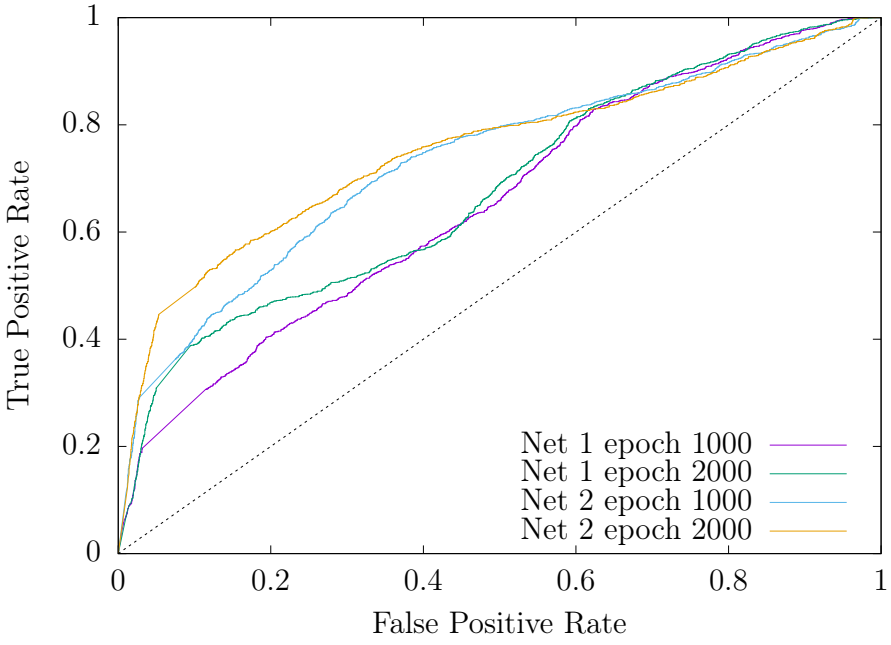


Figure 6.34: Receiver Operator Characteristics of OutlierNet.

compared to that after 2000 epochs. In fact, for a false positive rate in the interval of about 0.05 to 0.1, the curves of the first network architecture with 2000 epochs and of the second network with 1000 epochs are nearly identical.

Next, we look at the precision and recall curve. Here, the y axis shows the achieved recall value while the x axis shows the corresponding precision. Again, the higher the run of the curve is, the better is the performance of the classifier. As in most case high precision is detrimental to high recall and contrariwise, the curves usually decline in precision with increasing recall. For a recall of zero, no points are detected as outliers and thus by definition precision is one.

All in all, we see somewhat similar results to the ROC curve. Again, the larger network results in better scores up to a recall of 0.8, at which point all precision scores are bad. Training time also makes a difference again, with 2000 epochs staying on a higher level for larger recall values. In case of Net 1 with 1000 training epochs, precision drops around a reached recall value of 0.2. With 2000 training epochs, the drop happens slightly after a recall of 0.3 has been reached. The same applies for the Net 2 architecture with 1000 training epochs, albeit on a higher precision level, while the second network architecture even reaches a recall of more than 0.5 while maintaining a precision of over 0.6.

Similar effects of training epochs and network architecture also show in regards to the F_1 score with varying threshold parameter t. This is depicted in Figure 6.36. Again, the second network architecture with 2000 training epochs performs best, as it reaches a maximum F_1 score of about 0.52. None of the other combinations is able to surpass a F_1 score of 0.45. Interestingly, though, all four combinations achieve their best scores for values of t of about 0.9. This means, that the detection works best if the network is very sure of the vertices being outliers.

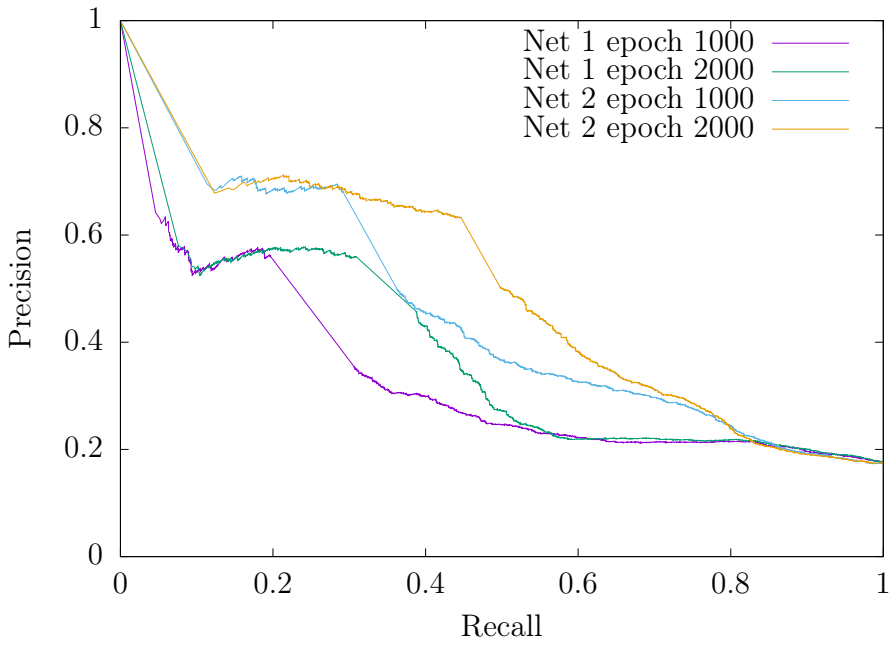


Figure 6.35: Precision and recall curve of OutlierNet.

6.6 Summary

In this chapter all outlier detection techniques, that have been presented in this thesis have been evaluated. In Section 6.1 we described the training sets and showed some differences between the distance functions used with the AdaMS outlier detection.

The actual evaluation of the different variants of the outlier detection algorithms that have been introduced in Chapter 4, is in the next three sections. Section 6.2 shows the results for the base algorithm. After a detailed exploration of the consequences to changes of the parameters of the outler detection algorithm, this section shows that the base algorithm works best with the earth mover's distance. In this case, F_1 scores of 0.56 and 0.58 were reached on Test Set 1 and 2, respectively. The earth mover's distance also yielded the best F_H scores on average, namely 0.53 on Test Set 1 and 0.69 on Test Set 2. On Test Set 1 the above average distance performed slightly better, reaching an F_H score of 0.54.

In Section 6.3 the mutli-reference enhancement was benchmarked. Here, it showed that using multiple reference silhouettes indeed enhances the quality of the outlier detection. Overall, again the earth mover's distance performed the best with F_1 scores of 0.58 on both test sets and F_H scores of 0.57 and 0.74. This section also showed that in most cases the best results were reached for about ten clustered reference silhouettes.

Section 6.4 showed the results of the outlier-type distinction enhancement. Interestingly, even the contrast features alone bettered the results of the positional features on both test sets, with the above average distance the best performer with F_1 scores of 0.59 and 0.68, as well as F_H scores of 0.62 and 0.87. The combined strategies where only tested on Test Set 2, as only this set included both obstacles and segmentation errors. In this setup, the results are even more impressive, as a F_1 score of 0.83 is reached. The F_H score though suffers as it reduces to 0.81 in order to provide a better

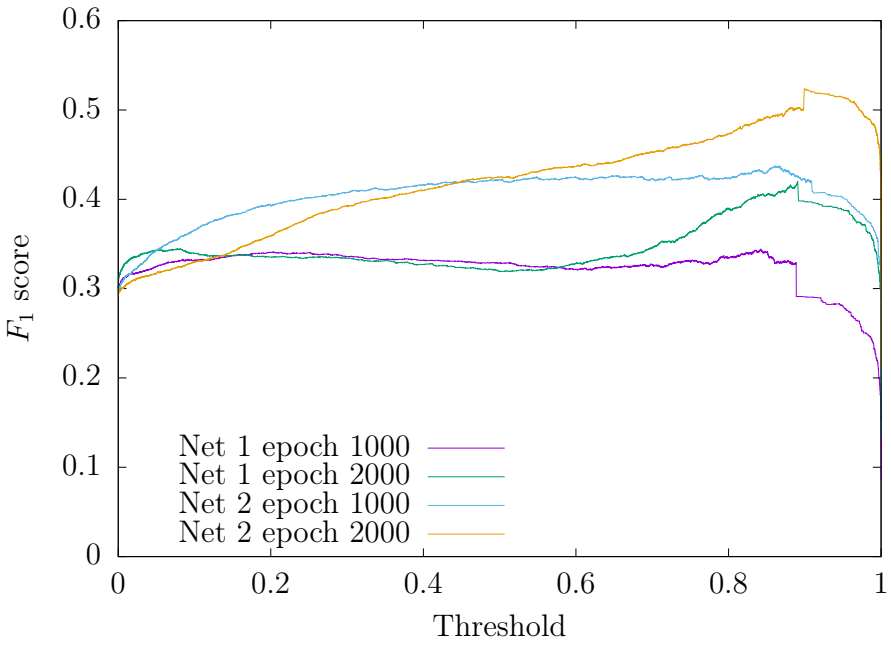


Figure 6.36: F_1 score over thresholds.

outlier classification. In this, a maximum ration 0.77 correctly identified and classified outliers is reached.

The previous section showed the results of the OutlierNet as introduced in Chapter 5. The evaluation showed that the simple approach presented here cannot compete with the specialised algorithms from the previous section, as the best reached F_1 score is only 0.52. However, we have seen that larger network architectures and a higher number of training epochs leads to better results, so there is still room for improvements.

7

Conclusion

In this chapter we summarise the work on outlier detection on sequences presented in this thesis. Afterwards, we give a brief outlook on how to pursue on this topic by future work.

7.1 Summary

As discussed in the introduction the main motivation to this thesis is the detection of obstacles and segmentation errors that occur when detecting the silhouettes of mountains in photos. Similar problems may arise in other segmentation applications as well, for example in more generic skyline detection applications, object recognition approaches that require the detection of precise edges, or in the analysis of medical imaging methods such as CT scans or magnetic resonance images. Depending on the task at hand and the amount of data that has to be computed, outlier detection on the segments' edges can be used to either automatically improve the segmentation by adaptive approaches like AdaMS or by notifying unusual findings to human experts that then can concentrate on a smaller fraction of the data.

During the thesis we first introduce the adaptive segmentation part of AdaMS in Chapter 3. The basic segmentation algorithm uses a region growing approach to compute an initial segmentation. In order to accommodate the adaptive improvements the picture gets partitioned into grid cells. If an outlier is detected the segmentation parameters of adjacent grid cells are adjusted and this part of the image gets segmented again to correct segmentation errors. For obstacles, a new segmentation is not valuable since the image does not hold information about parts concealed by the obstacle. In those cases, the silhouette can be approximated by straight lines or splines.

Chapter 4 presents the outlier detection algorithms developed during the work on this thesis. We start by defining outliers as subsequences of the polygonal chains that form the silhouettes. Each outlier must contain at least a subsequence of l strong anomalies, that is, at least l adjacent points that are clearly unusual in their properties. We also introduce a number of histogram distances to be able to compute the anomaly

scores. Notably, we introduce the above average distance, which is a new distance measure and prove that it is a pseudometric. Based on these definitions, the first and most basic version of the AdaMS outlier detection algorithm is presented which only uses one reference histogram and positional features.

The remainder of Chapter 4 also raises two possibilities to improve on the basic version mentioned above. The multi-reference enhancement introduces multiple reference histograms instead of using a single one. In order to accommodate this change, we use the minimal reference distance defined as

$$\mathrm{dist}_{mr}(H, \mathcal{H}_{ref}) = \min\{\mathrm{aad}(H, H_{ref}) | H_{ref} \in \mathcal{H}_{ref}\}.$$

By using this approach, we get a more granular presentation of normal mountain silhouettes. This is supposed to help the outlier detection as a great variety of forms in mountains exists.

Secondly, the outlier-type distinction extension uses different sets of features for the detection of different kinds of outliers, namely positional features for the detection of obstacles and contrast based features for the detection of segmentation errors. Essentially, the idea of this approach is similar to other ensemble methods, however, the combination of the results is a bit more difficult than in many other cases. The reason for this lies in the fact that the results in our case are not class labels as in classification ensemble methods, but outliers, that can overlap. Therefore, we propose three different merging algorithms that are able to solve this problem.

An approach to outlier detection with the help of artificial neural networks is proposed in Chapter 5. In the past, convolutional neural networks have yielded very good results in classification problems without the need for feature extraction. We argue that outlier detection tasks can be reduced to classification problems if training data for the outlier class can be produced. Therefore, we first concentrate on the creation of training data by describing how training samples for outlier-free are generated. Then we describe the creation of training data for the outliers, which is generated by deviating from the correct silhouettes of the training images.

Afterwards we give an overview of convolutional neural networks and how their main layer types work, before we describe the two architectures we utilise for the outlier detection networks. Both are heavily influenced by the VGG structure, however, due to the small size of the input image patches, we opt for a much swallower design. Finally, we also describe how the output of the network should be interpreted. For every image patch the classifier returns the probability of that patch being an outlier, however, outliers in silhouettes can consist of more than one patch. Because of this, we compute outlier probabilities for each single silhouette point and then find sequences with outlier probabilities that are higher than a certain threshold.

Finally, Chapter 6 evaluates the outlier detection algorithms presented in this thesis. The chapter starts with an exploration of the training data set, which consists of 61 error free images and corresponding silhouettes. We note that the usage of different distance functions leads to highly different mean distances. Also it is noteworthy, that the above average distance's mean is most sensitive to a higher number of clusters. However, we see that a number of clusters higher than 15 does not lead to any reduction of mean or changes in the standard deviation for any of the distance functions. Next, we conducted an in-depth analysis of the parameters of the base algorithm, starting of with the two thresholds τ_s and τ_w. Those thresholds determine whether a point is

regarded as a strong anomaly or a weak anomaly, respectively. Overall we note that for the base algorithm, relatively high values for the inner threshold τ_s and low values for the outer threshold τ_w yield the best results overall in respect to F_1 scores, while slightly higher values for τ_w result in better F_H scores. In terms of the minimal length of the inner outlier, values for l of 1 or 2 lead to the best results.

Next, we concentrate on the evaluation of the multi-reference enhancement, that introduces an additional parameter for the number of reference silhouette clusters. On the test sets, the distances' means and standard deviations for the different distance functions behave similar to those of the training data. Overall, this high level analysis hints that the best separation of normal and outlier data is achieved by the earth mover's distance. This is then confirmed by the total evaluation of both trainings sets, where the emd yields the best results for both the F_1 and the F_H scores.

This is followed by the evaluation of the outlier-type distinction. There are two types of outliers that exist in the silhouettes: Segmentation errors, that are usually characterised by weaker contrast on the silhouette, and obstacles, which have unusual forms. By introducing a separate set of features that concentrates on the strength of contrast and its direction we aim to better identify segmentation errors. We opt to implement this set of features by an additional outlier detection step that in order to keep the number of bins in the reference histograms low enough for meaningful results. This gives us the additional benefit of getting an implicit outlier-type classification. By this, we get a ratio of 77% of correctly classified outliers, including those that have not been detected at all, and a combined outlier detection with an F_H score of 0.81 in the best case.

Lastly, the neural networks that have been introduced in Chapter 5 are evaluated. Overall the results are on the same level as those of the base algorithm. With the larger networks and 2000 training epochs, an F_1 score of 0.52 is reached. However, we have shown, that larger network architectures and longer training times result in more precise outlier detection.

7.2 Future Work

There are several areas that allow for future work. Regarding the AdaMS outlier detection, one drawback of the current solution with multi-reference and outlier-type distinction extensions lies in the fact that many parameters have to be chosen. While Chapter 6 showed that the ideal combination of distance functions is the earth mover's distance for the positional features and the above average distance for the contrast features, we have also seen that the best settings for the other parameters change between both test sets. Thus, in order to improve the ease of use of the algorithm an additional module that chooses good settings for those parameters can be developed. A promising way to implement this could be to look at the properties of the silhouettes in the images in terms of positional and contrast attributes. Based on these and stored settings for the parameters, a classification algorithm such as k nearest neighbours could be used to choose a good set of parameters.

Another way to automate the parameter settings, that could be explored in addition to the one mentioned previously, is to search for dependencies between parameters. Obviously there are dependencies between the type of distance function that is used and the other parameters, especially the thresholds. Given the related nature of the

threshold parameters τ_s and τ_w it does not seem unlikely that some kind of functional dependency for good settings between those two exists as well.

The usage of artificial neural networks for outlier detection also contains additional research opportunities. As we already discovered, networks with more layers yield better results, so some future research can be dedicated to exploring a good size for those. Sizing up the networks too much, however, not only results in much longer training times, but also more possibilities of overfitting the training data. The creation of the training data for the outlier class could be refined, too. Currently, this is a somewhat random process since we only ensure that the silhouettes used for this deviate from the true silhouettes. However, we know that the detection of the spot where a detected silhouette deviates from the correct silhouette is the key for identifying and correcting outliers. Therefore, an approach that creates branches of outlier off the true silhouettes and uses those for training the network could also lead to better results.

Bibliography

[ABKS99] Ankerst, Mihael ; Breunig, Markus M. ; Kriegel, Hans-Peter ; Sander, Jörg: OPTICS: Ordering Points to Identify the Clustering Structure. In: *ACM Sigmod record* Bd. 28, 1999

[Aiz64] Aizerman, Mark A.: Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning. In: *Automation and Remote Control* 25 (1964)

[Alt92] Altman, Naomi: An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. In: *The American Statistician* 46 (1992), Nr. 3

[Ans60] Anscombe, Frank J.: Rejection of Outliers. In: *Technometrics* 2 (1960), Nr. 2

[AY01] Aggarwal, Charu C. ; Yu, Philip S.: Outlier Detection for High Dimensional Data. In: *CM Sigmod Record* Bd. 30, 2001

[BA11] Buu, Huynh Tran Q. ; Anh, Duong T.: Time Series Discord Discovery Based on iSAX Symbolic Representation. In: *Third International Conference on Knowledge and Systems Engineering*, 2011

[BČES11] Baboud, Lionel ; Čadík, Martin ; Eisemann, Elmar ; Seidel, Hans-Peter: Automatic Photo-to-terrain Alignment for the Annotation of Mountain Pictures. In: *Proc. of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, 2011

[BDVJ03] Bengio, Yoshua ; Ducharme, Réjean ; Vincent, Pascal ; Jauvin, Christian: A Neural Probabilistic Language Model. In: *Journal of Machine Learning Research* 3 (2003)

[BGV92] Boser, Bernhard E. ; Guyon, Isabelle M. ; Vapnik, Vladimir N.: A Training Algorithm for Optimal Margin Classifiers. In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 1992 (COLT '92)

[BKNS00] Breunig, Markus M. ; Kriegel, Hans-Peter ; Ng, Raymond T. ; Sander, Jörg: LOF: Identifying Density-based Local Outliers. In: *ACM Sigmod Record* Bd. 29, 2000

[BL+07] Bengio, Yoshua ; LeCun, Yann u. a.: Scaling Learning Algorithms towards AI. In: *Large-scale Kernel Machines* 34 (2007), Nr. 5

[BLPL07] Bengio, Yoshua ; Lamblin, Pascal ; Popovici, Dan ; Larochelle, Hugo: Greedy Layer-wise Training of Deep Networks. In: *Advances in neural information processing systems*, 2007

[BS15] Braun, Daniel ; Singhof, Michael: Automated Silhouette Extraction for Mountain Recognition. In: *GI GvDB Workshop 2015*, 2015

[BSC16] Braun, Daniel ; Singhof, Michael ; Conrad, Stefan: AdaMS: Adaptive Mountain Silhouette Extraction from Images. In: *Proc. of the MLDM 2016*, 2016

[BSCB00] Bertalmio, Marcelo ; Sapiro, Guillermo ; Caselles, Vincent ; Ballester, Coloma: Image Inpainting. In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000

[BSKP12] Baatz, Georges ; Saurer, Olivier ; Köser, Kevin ; Pollefeys, Marc: Large Scale Visual Geo-Localization of Images in Mountainous Terrain. In: *Computer Vision - ECCV 2012.* 2012

[BTG06] Bay, Herbert ; Tuytelaars, Tinne ; Gool, Luc V.: Surf: Speeded Up Robust Features. In: *European Conference on Computer Vision*, 2006

[Can86] Canny, John: A Computational Approach to Edge Detection. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* PAMI-8 (1986), Nov, Nr. 6

[CMS12] Ciregan, Dan ; Meier, Ueli ; Schmidhuber, Jürgen: Multi-column Deep Neural Networks for Image Classification. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012

[CS01] Chan, Tony F. ; Shen, Jianhong: Nontexture Inpainting by Curvature-driven Diffusions. In: *Journal of Visual Communication and Image Representation* 12 (2001), Nr. 4

[CV95] Cortes, Corinna ; Vapnik, Vladimir: Support-vector Networks. In: *Machine Learning* 20 (1995), Nr. 3

[DLR77] Dempster, Arthur P. ; Laird, Nan M. ; Rubin, Donald B.: Maximum Likelihood from Incomplete Data via the EM Algorithm. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1977)

[EKSX96] Ester, Martin ; Kriegel, Hans-Peter ; Sander, Jörg ; Xu, Xiaowei: A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* Bd. 96, 1996

[Fer61] Ferguson, Thomas S.: On the Rejection of Outliers. In: *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability* Bd. 1, 1961

[FFT14] Fedorov, Roman ; Fraternali, Piero ; Tagliasacchi, Marco: Mountain Peak Identification in Visual Content Based on Coarse Digital Elevation Models. In: *Proc. of the 3rd ACM International Workshop on Multimedia Analysis for Ecological Data*, 2014

[GBQG09] Gammeter, Stephan ; Bossard, Lukas ; Quack, Till ; Gool, Luc V.: I Know What You Did Last Summer: Object-level Auto-annotation of Holiday Snaps. In: *IEEE 12th International Conference on Computer Vision*, 2009

[GD04] Grauman, Kristen ; Darrell, Trevor: Fast Contour Matching Using Approximate Earth Mover's Distance. In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on* Bd. 1, 2004

[GDDM14] Girshick, Ross ; Donahue, Jeff ; Darrell, Trevor ; Malik, Jitendra: Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014

[Gir15] Girshick, Ross: Fast R-CNN. In: *Proceedings of the IEEE International Conference on Computer Vision*, 2015

[GKVL06] Gardner, Andrew B. ; Krieger, Abba M. ; Vachtsevanos, George ; Litt, Brian: One-class Novelty Detection for Seizure Analysis from Intracranial EEG. In: *Journal of Machine Learning Research* 7 (2006), Nr. Jun

[Gru50] Grubbs, Frank E.: Sample Criteria for Testing Outlying Observations. In: *The Annals of Mathematical Statistics* (1950)

[Güm58] Gümbel, Emil J.: *Statistics of Extremes*. 1958

[HA91] Haddad, Richard A. ; Akansu, Ali N.: A Class of Fast Gaussian Binomial Filters for Speech and Image Processing. In: *IEEE Transactions on Signal Processing* 39 (1991), Nr. 3

[Haw80] Hawkins, Douglas M.: *Identification of Outliers*. 1980

[HOT06] Hinton, Geoffrey E. ; Osindero, Simon ; Teh, Yee-Whye: A Fast Learning Algorithm for Deep Belief Nets. In: *Neural computation* 18 (2006), Nr. 7

[JH99] Jaakkola, Tommi ; Haussler, Davis: Exploiting Generative Models in Discriminative Classifiers. In: *Advances in Neural Information Processing Systems*, 1999

[Jon72] Jones, Karen S.: A Statistical Interpretation of Term Specificity and its Application in Retrieval. In: *Journal of Documentation* 28 (1972), Nr. 1

[KA12] Khanh, Nguyen Dang K. ; Anh, Duong T.: Time Series Discord Discovery Using WAT Algorithm and iSAX Representation. In: *Proceedings of the Third Symposium on Information and Communication Technology*, 2012

[KB14] Kingma, Diederik P. ; Ba, Jimmy: Adam: A Method for Stochastic Optimization. In: *CoRR* abs/1412.6980 (2014)

[Kla17] Klassen, Gerhard: *Classification of Edges in Images*, Master's Thesis, 2017

[KLF04] Keogh, Eamonn ; Lin, Jessica ; Fu, Ada: HOT SAX: Finding the Most Unusual Time Series Subsequence: Algorithms and Applications. In: *Proc. of the 5th IEEE Int'l Conf. on Data Mining*, 2004

[KLF05] Keogh, Eamonn ; Lin, Jessica ; Fu, Ada: HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In: *Fifth IEEE International Conference on Data Mining (ICDM'05)*, 2005

[KSH12] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks. In: *Advances in Neural Information Processing Systems 25*. 2012

[KSZ08] Kriegel, Hans-Peter ; Schubert, Matthias ; Zimek, Arthur: Angle-based Outlier Detection in High-dimensional Data. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008

[KYM07] Kawahara, Yoshinobu ; Yairi, Takehisa ; Machida, Kazuo: Change-point Detection in Time-series Data Based on Subspace Identification. In: *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, 2007

[LBD+90] LeCun, Yann ; Boser, Bernhard E. ; Denker, John S. ; Henderson, Donnie ; Howard, Richard E. ; Hubbard, Wayne E. ; Jackel, Lawrence D.: Handwritten Digit Recognition with a Back-propagation Network. In: *Advances in Neural Information Processing Systems*, 1990

[LBOM98] LeCun, Yann ; Bottou, Léon ; Orr, Genevieve B. ; Müller, Klaus-Robert: Efficient Backprop. In: *Neural Networks: Tricks of the Trade*. 1998

[LHL08] Lee, Jae-Gil ; Han, Jiawei ; Li, Xiaolei: Trajectory Outlier Detection: A Partition-and-detect Framework. In: *IEEE 24th International Conference on Data Engineering*, 2008

[LKLC03] Lin, Jessica ; Keogh, Eamonn ; Lonardi, Stefano ; Chiu, Bill: A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In: *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2003

[LLLH05] Lie, Wen-Nung ; Lin, Tom C-I ; Lin, Ting-Chih ; Hung, Keng-Shen: A Robust Dynamic Programming Algorithm to Extract Skyline in Images for Navigation. In: *Pattern recognition letters* 26 (2005), Nr. 2

[LM14] Le, Quoc ; Mikolov, Tomas: Distributed Representations of Sentences and Documents. In: *Proceedings of the 31st International Conference on Machine Learning*, 2014

[Low04] Lowe, David G.: Distinctive Image Features from Scale-invariant Keypoints. In: *International Journal of Computer Vision* 60 (2004), Nr. 2

[LSD15] Long, Jonathan ; Shelhamer, Evan ; Darrell, Trevor: Fully Convolutional Networks for Semantic Segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015

[Mac67] MacQueen, James: Some Methods for Classification and Analysis of Multivariate Observations. In: *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967

[MAKK+11] Masud, Mohammad M. ; Al-Khateeb, Tahseen M. ; Khan, Latifur ; Aggarwal, Charu ; Gao, Jing ; Han, Jiawei ; Thuraisingham, Bhavani: Detecting Recurring and Novel Classes in Concept-drifting Data Streams. In: *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, 2011

[MP03a] Ma, Junshui ; Perkins, Simon: Online Novelty Detection on Temporal Sequences. In: *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003

[MP03b] Ma, Junshui ; Perkins, Simon: Time-series Novelty Detection Using One-class Support Vector Machines. In: *Neural Networks, 2003. Proceedings of the International Joint Conference on* Bd. 3, 2003

[MSC+13] Mikolov, Tomas ; Sutskever, Ilya ; Chen, Kai ; Corrado, Greg S. ; Dean, Jeff: Distributed Representations of Words and Phrases and their Compositionality. In: *Advances in Neural Information Processing Systems*, 2013

[NH10] Nair, Vinod ; Hinton, Geoffrey E.: Rectified Linear Units Improve Restricted Boltzmann Machines. In: *Proceedings of the 27th International Conference on Machine Learning*, 2010

[NMMI97] Naval Jr, Prospero C. ; Mukunoki, Masayuki ; Minoh, Michihiko ; Ikeda, Katsuo: Estimating Camera Position and Orientation from Geographical Map and Mountain Image. In: *38th Research Meeting of the Pattern Sensing Group, Society of Instrument and Control Engineers*, 1997

[NS06] Nister, David ; Stewenius, Henrik: Scalable Recognition with a Vocabulary Tree. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* Bd. 2, 2006

[OFL07] Ozuysal, Mustafa ; Fua, Pascal ; Lepetit, Vincent: Fast Keypoint Recognition in Ten Lines of Code. In: *IEEE Conference on Computer Vision and Pattern Recognition, 2007*, 2007

[Orl97] Orlin, James B.: A Polynomial Time Primal Network Simplex Algorithm for Minimum Cost Flows. In: *Mathematical Programming* 78 (1997), Nr. 2

[PBV+14] Porzi, Lorenzo ; Buló, Samuel R. ; Valigi, Paolo ; Lanz, Oswald ; Ricci, Elisa: Learning Contours for Automatic Annotations of Mountains Pictures on a Smartphone. In: *Proceedings of the International Conference on Distributed Smart Cameras*, 2014

[PCC+07] Poultney, Christopher ; Chopra, Sumit ; Cun, Yann L. u. a.: Efficient Learning of Sparse Representations with an Energy-based Model. In: *Advances in Neural Information Processing Systems*, 2007

[Pic85] Picard, Dominique: Testing and Estimating Change-points in Time Series. In: *Advances in applied probability* 17 (1985), Nr. 4

[PKGF03] Papadimitriou, Spiros ; Kitagawa, Hiroyuki ; Gibbons, Phillip B. ; Faloutsos, Christos: Loci: Fast Outlier Detection Using the Local Correlation Integral. In: *Proceedings of the 19th International Conference on Data Engineering*, 2003

[PLD10] Pham, Ninh D. ; Le, Quang L. ; Dang, Tran K.: HOT aSAX: A Novel Adaptive Symbolic Representation for Time Series Discords Discovery. In: *Asian Conference on Intelligent Information and Database Systems*, 2010

[PLSP10] Perronnin, Florent ; Liu, Yan ; Sánchez, Jorge ; Poirier, Hervé: Large-scale Image Retrieval with Compressed Fisher Vectors. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010

[Pre70] Prewitt, Judith M.: Object Enhancement and Extraction. In: *Picture Processing and Psychopictorics* 10 (1970), Nr. 1

[PSF05] Papadimitriou, Spiros ; Sun, Jimeng ; Faloutsos, Christos: Streaming Pattern Discovery in Multiple Time-series. In: *Proceedings of the 31st International Conference on Very Large Data Bases*, 2005

[PW08] Pele, Ofir ; Werman, Michael: A Linear Time Histogram Metric for Improved Sift Matching. In: *Computer Vision–ECCV 2008*, 2008

[PW09] Pele, Ofir ; Werman, Michael: Fast and Robust Earth Mover's Distances. In: *2009 IEEE 12th International Conference on Computer Vision*, 2009

[Qui86] Quinlan, J. R.: Induction of Decision Trees. In: *Machine learning* 1 (1986), Nr. 1

[RC15] Rischka, Magdalena ; Conrad, Stefan: Image Landmark Recognition with Hierarchical K-Means Tree. In: *BTW 2015*, 2015

[RDS+15] Russakovsky, Olga ; Deng, Jia ; Su, Hao ; Krause, Jonathan ; Satheesh, Sanjeev ; Ma, Sean ; Huang, Zhiheng ; Karpathy, Andrej ; Khosla, Aditya ; Bernstein, Michael ; Berg, Alexander C. ; Fei-Fei, Li: ImageNet Large Scale Visual Recognition Challenge. In: *International Journal of Computer Vision (IJCV)* 115 (2015), Nr. 3

[RHGS15] Ren, Shaoqing ; He, Kaiming ; Girshick, Ross ; Sun, Jian: Faster R-CNN: Towards Real-time Object Detection with Region Proposal Networks. In: *Advances in Neural Information Processing Systems*, 2015

[Rob63] Roberts, Lawrence G.: *Machine Perception of Three-dimensional Solids*, Massachusetts Institute of Technology, Diss., 1963

[Ros61] Rosenblatt, Frank: Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms. 1961. – Forschungsbericht

[RTG98] Rubner, Yossi ; Tomasi, Carlo ; Guibas, Leonidas J.: A Metric for Distributions with Applications to Image Databases. In: *Computer Vision, 1998. Sixth International Conference on* IEEE, 1998

[RTG00] Rubner, Yossi ; Tomasi, Carlo ; Guibas, Leonidas J.: The Earth Mover's Distance as a Metric for Image Retrieval. In: *International Journal of Computer Vision* 40 (2000), Nr. 2

[SB91] Swain, Michael J. ; Ballard, Dana H.: Color Indexing. In: *International Journal of Computer Vision* 7 (1991), Nr. 1

[SBC16] Singhof, Michael ; Braun, Daniel ; Conrad, Stefan: Finding Trees in Mountains – Outlier Detection on Polygonal Chains. In: *Proc. of the Conference LWDA 2016*, 2016

[Sch00] Scharr, Hanno: *Optimale Operatoren in der digitalen Bildverarbeitung*, Diss., 2000

[SJ08] Shirdhonkar, Sameer ; Jacobs, David W.: Approximate Earth Mover's Distance in Linear Time. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008

[SK08] Shieh, Jin ; Keogh, Eamonn: iSAX: Indexing and Mining Terabyte Sized Time Series. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* ACM, 2008

[SKBC17] Singhof, Michael ; Klassen, Gerhard ; Braun, Daniel ; Conrad, Stefan: Detection and Implicit Classification of Outliers via Different Feature Sets in Polygonal Chains. In: *BTW*, 2017

[Sob90] Sobel, Irvin: An Isotropic 3× 3 Image Gradient Operator. In: *Machine Vision for Three-dimensional Scenes* (1990)

[SZ14] Simonyan, Karen ; Zisserman, Andrew: Very Deep Convolutional Networks for Large-scale Image Recognition. In: *CoRR* (2014)

[Tar97] Tarjan, Robert E.: Dynamic Trees as Search Trees via Euler Tours, Applied to the Network Simplex Algorithm. In: *Mathematical Programming* 78 (1997), Nr. 2

[Vas69] Vaserstein, Leonid N.: Markov Processes over Denumerable Products of Spaces, Describing Large Systems of Automata. In: *Problemy Peredachi Informatsii* 5 (1969), Nr. 3

[VC64] Vapnik, Vladimir ; Chervonenkis, Alexey: A Note on One Class of Perceptrons. In: *Automation and Remote Control* 25 (1964), Nr. 1

[VSNI+14] Van der Walt, Stefan ; Schönberger, Johannes L. ; Nunez-Iglesias, Juan ; Boulogne, François ; Warner, Joshua D. ; Yager, Neil ; Gouillart, Emmanuelle ; Yu, Tony: scikit-image: Image Processing in Python. In: *PeerJ* 2 (2014)

[Wu83] Wu, CF J.: On the Convergence Properties of the EM Algorithm. In: *The Annals of Statistics* (1983)

List of Figures

List of Tables

A

List of Publications

This chapter lists the publications of Michael Singhof. The first section gives an overview of publications that contributed to the dissertation as well as the contributions of the author. The second section shows additional publications that did not contribute directly to this work.

Publications Used in the Dissertation

1. M. Singhof, G. Klassen, D. Braun, S. Conrad:
   Detection and Implicit Classification of Outliers via Different Feature Sets in Polygonal Chains
   In: *Datenbanksysteme für Business, Technologie und Web (BTW 2017)*, 2017
   Contributions: Michael Singhof contributed the idea for the outlier detection with two different sets of features. The implementation of this algorithm was carried out jointly with Gerhard Klassen. The annotation of the outliers for the test set and the evaluation was carried out jointly with Daniel Braun. The manuscript was prepared jointly with Gerhard Klassen.

2. M. Singhof, D. Braun, S. Conrad:
   Finding Trees in Mountains – Outlier Detection on Polygonal Chains
   In: *Proceedings of the Conference Lernen, Wissen, Daten, Analysen (LWDA 2016)*, 2016
   Contributions: The multi-reference-enhancement for the outlier detection was invented and implemented by Michael Singhof, as well as the proof showing that the above average distance is a pseudometric. The annotation of the outliers for the test set and the evaluation was carried out jointly with Daniel Braun. The manuscript was also prepared together with Daniel Braun.

3. D. Braun, M. Singhof, S. Conrad:
   AdaMS: Adaptive Mountain Silhouette Extraction from Images

In: *Machine Learning and Data Mining in Pattern Recognition (MLDM 2016)*, 2016
Contributions: The idea and implementation of the outlier detection and classification are by Michael Singhof. The preparation of the manuscript and the evaluation where carried out jointly with Daniel Braun.

4. D. Braun, M. Singhof, S. Conrad:
An Adaptive Grid Segmentation Algorithm for Mountain Silhouette Extraction from Images
In: *Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB*, 2015
Contributions: The idea of the outlier detection and classification parts are by Michael Singhof. The manuscript was prepared with Daniel Braun.

5. D. Braun, M. Singhof:
Automated Silhouette Extraction for Mountain Recognition In: *Proceedings of the 27th GI-Workshop on Foundations of Databases*, 2015
Contributions: The idea and implementation of the silhouette cleaning are by Michael Singhof. The manuscript was written in conjunction with Daniel Braun.

Other Publications

1. D. Braun, M. Singhof, M. Tatusch, S. Conrad:
Convolutional Neural Networks for Multidrug-resistant and Drug-sensitive Tuberculosis Dinstinction
In: *CLEF2017 Working Notes*, 2017

2. G. Klassen, M. Singhof:
Shape Based Outlier Detection in SLIC Superpixels
In: *Proceedings of the 29th GI-Workshop on Foundations of Databases*, 2017

3. M. Singhof:
Analysis of DDoS Detection Systems
In: *Proceedings of the 25th GI-Workshop on Foundations of Databases*, 2013

B

Data Sets

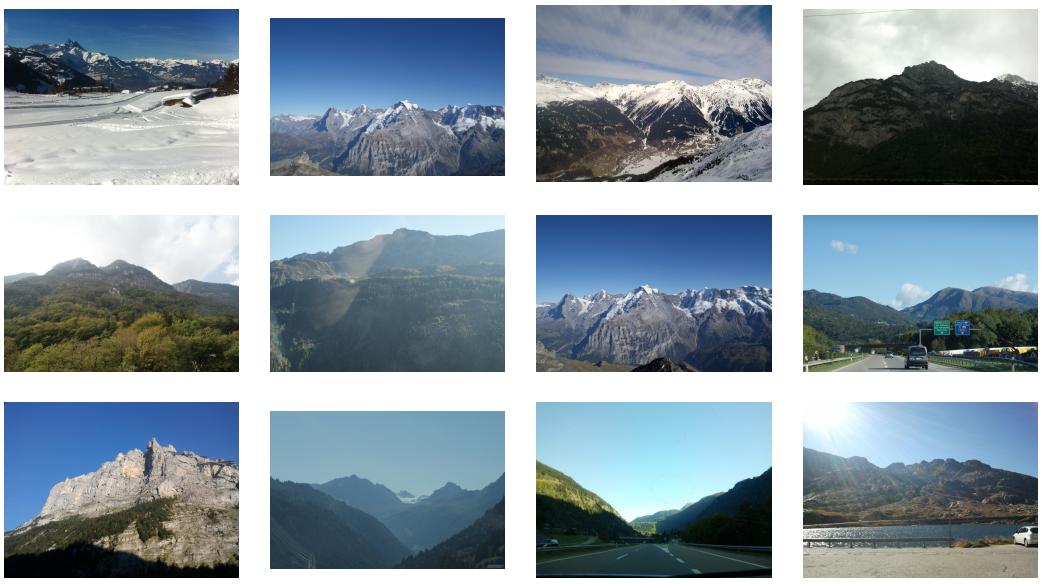
The Switzerland data set consists of 203 images and is provided with [BSKP12]. The whole set of images is available for download at

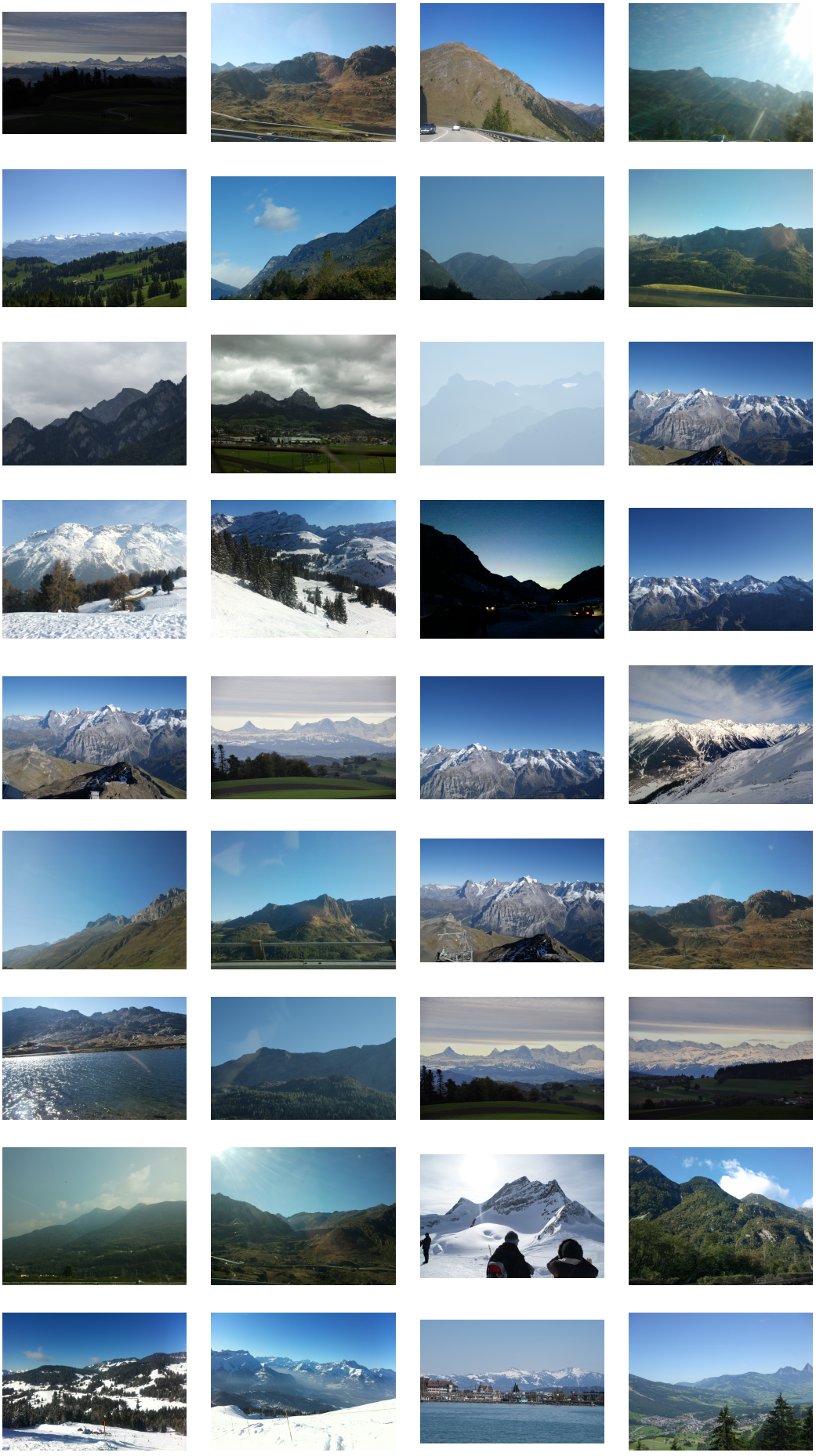
`http://cvg.ethz.ch/research/mountain-localization/`

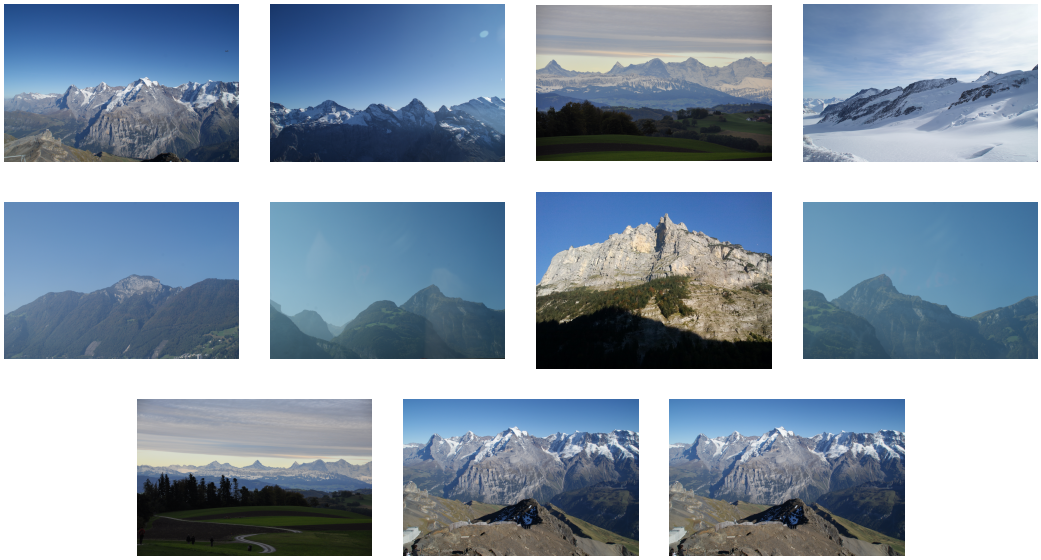
While 196 pictures are licensed under the creative commons license, seven photos are not covered by this license and are therefore not shown in this appendix.

B.1 Training Data Set

The Training Data Set consists of a portion of 61 of the images of the Switzerland Dataset, for which the segmentation algorithm was able to compute fault-free silhouette out of the box. Of those 61 photos, two are not covered by the creative commons license and are therefore excluded here.





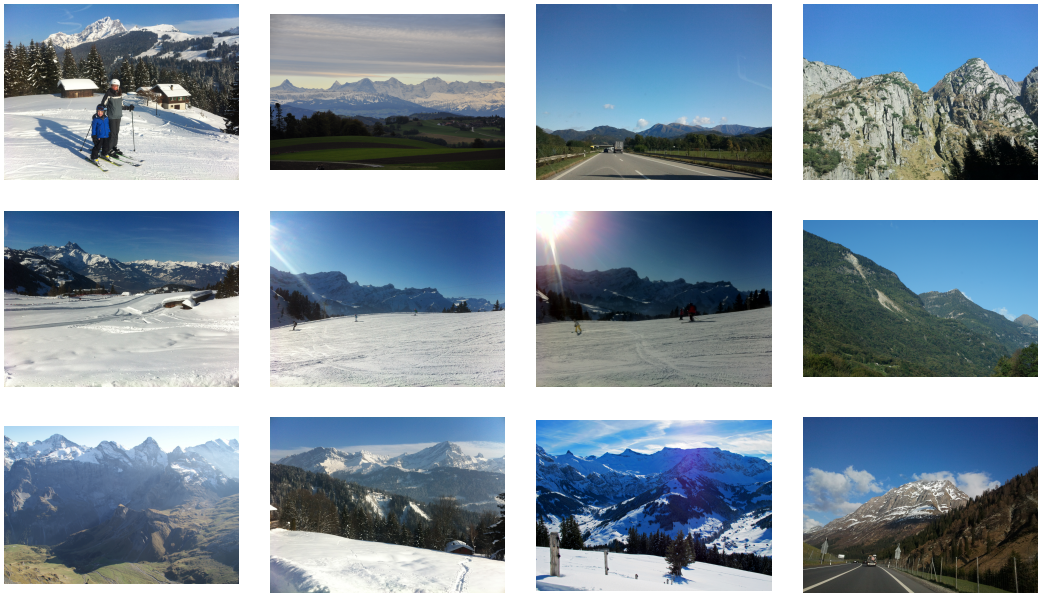


B.2 Switzerland Data Set / Test Set 2

The Switzerland data set as described in the evaluation in Chapter 6, consists of 14 of the photos from the whole data set published with [BSKP12]. On these photos we manually annotated 70 outliers on the silhouettes computed by the segmentation algorithm of which the 61 that are longer than four silhouette points have been used in the evaluation:

Length	Number of outliers
1 – 4	9
5 – 9	41
10 – 19	24
20 – 99	31
≥ 100	6
Total used	61

The following image are used:




B.3 Our Data Set / Test Set 1

This data set consists of 46 images taken by Michael Singhof. The images have been manually segmented by Daniel Braun. If the computed silhouette differs more than two pixels from the manually detected one, this part of the silhouette is defined as an outlier. By this approach, we get the following outliers:

Length	Number of outliers
1 – 4	1213
5 – 9	135
10 – 19	65
20 – 99	29
≥ 100	7
Total used	101

Again only those outliers that are at least five silhouette points long have been used in the evaluation. The following photos are used in this data set: