Kernel methods for vessel trajectories
de Vries, G.K.D.

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: http://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
KERNEL METHODS FOR VESSEL TRAJECTORIES

KERNEL METHODS FOR VESSEL TRAJECTORIES

GERBEN KLAAS DIRK DE VRIES

GERBEN KLAAS DIRK DE VRIES
KERNEL METHODS FOR VESSEL TRAJECTORIES

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. D.C. van den Boom
ten overstaan van een door het college voor promoties ingestelde
commissie, in het openbaar te verdedigen in de Agnietenkapel
op woensdag 4 april 2012, te 12:00 uur

door

Gerben Klaas Dirk de Vries

geboren te Amsterdam
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

This work has been carried out as a part of the Poseidon project in collaboration with Thales Nederland, under the responsibilities of the Embedded Systems Institute (ESI). This project is partially supported by the Dutch Ministry of Economic Affairs under the BSIK program.

Cover design by Menno Endt of The Crooked Line, based on a photograph by Theun Hendrikx. Printed by Ipskamp Drukkers BV.

Copyright © 2012 by Gerben Klaas Dirk de Vries
# CONTENTS

**PREFACE**  ix  

1 **INTRODUCTION**  1  
1.1 Approach  1  
1.2 Research Questions  2  
1.3 Poseidon Project  4  
1.4 Outline  4  

2 **PRELIMINARIES**  7  
2.1 Machine Learning  7  
2.1.1 Kernel Methods  9  
2.1.2 Evaluation Metrics  14  
2.2 Spatio-Temporal Data-Mining  15  
2.2.1 Moving Object Trajectories  15  
2.2.2 Trajectory Similarity Measures  16  
2.2.3 Data-Mining Tasks for Trajectories  17  
2.2.4 Temporal Data-Mining and Kernel Methods  18  
2.3 Maritime Safety and Security Systems  19  
2.3.1 Vessel Trajectories from AIS  20  
2.3.2 Clustering  21  
2.3.3 Classification  21  
2.3.4 Outlier Detection  23  

3 **TRAJECTORY COMPRESSION**  25  
3.1 Introduction  25  
3.2 Trajectory Compression  26  
3.2.1 Trajectory Definitions  26  
3.2.2 Piecewise Linear Segmentation  27  
3.2.3 A Problematic Example  30  
3.2.4 Two Stage Piecewise Linear Segmentation  31  
3.2.5 Error bounds and Complexity  32  
3.3 Evaluation  33  
3.3.1 Stop Retention  33  
3.3.2 Experimental Set-Up  34  
3.3.3 Results  35  
3.3.4 Discussion  38  
3.4 Deriving Complex Behavior  39  
3.4.1 Semantic Web  39  
3.4.2 Simple Event Model  40  
3.4.3 Segments  40  
3.4.4 Additional Semantics  41  
3.5 Conclusions & Future Work  43
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td><strong>Alignment Based Similarities</strong></td>
<td>45</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>45</td>
</tr>
<tr>
<td>4.2</td>
<td>Trajectory Alignment Kernels</td>
<td>46</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Alignments</td>
<td>47</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Alignment Kernels</td>
<td>49</td>
</tr>
<tr>
<td>4.3</td>
<td>Experiments</td>
<td>51</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Experimental Setup</td>
<td>52</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Clustering</td>
<td>54</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Classification</td>
<td>58</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Outlier Detection</td>
<td>62</td>
</tr>
<tr>
<td>4.4</td>
<td>Conclusions &amp; Future Work</td>
<td>63</td>
</tr>
<tr>
<td>5</td>
<td><strong>Integral Based Similarities</strong></td>
<td>67</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>67</td>
</tr>
<tr>
<td>5.2</td>
<td>Integral Based Trajectory Similarity</td>
<td>68</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Full Trajectory Similarity Kernels</td>
<td>70</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Subtrajectory Similarity Kernels</td>
<td>76</td>
</tr>
<tr>
<td>5.3</td>
<td>Experiments</td>
<td>78</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Experimental Set-Up</td>
<td>78</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Clustering</td>
<td>79</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Classification</td>
<td>80</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Outlier Detection</td>
<td>82</td>
</tr>
<tr>
<td>5.4</td>
<td>Conclusions &amp; Future Work</td>
<td>84</td>
</tr>
<tr>
<td>6</td>
<td><strong>Adding Domain Knowledge</strong></td>
<td>87</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>87</td>
</tr>
<tr>
<td>6.2</td>
<td>Geographical Domain Knowledge</td>
<td>88</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Domain Ontologies</td>
<td>88</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Enriching Trajectories</td>
<td>91</td>
</tr>
<tr>
<td>6.3</td>
<td>Enriched Trajectory Kernels</td>
<td>92</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Substitution Functions for Sets of Geo-Labels</td>
<td>93</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Kernels</td>
<td>94</td>
</tr>
<tr>
<td>6.4</td>
<td>Experiments</td>
<td>96</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Experimental Set-Up</td>
<td>97</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Clustering</td>
<td>97</td>
</tr>
<tr>
<td>6.4.3</td>
<td>Classification</td>
<td>100</td>
</tr>
<tr>
<td>6.4.4</td>
<td>Outlier Detection</td>
<td>105</td>
</tr>
<tr>
<td>6.5</td>
<td>Conclusions &amp; Future Work</td>
<td>108</td>
</tr>
<tr>
<td>7</td>
<td><strong>Conclusion</strong></td>
<td>109</td>
</tr>
<tr>
<td>7.1</td>
<td>Answers to Research Questions</td>
<td>109</td>
</tr>
<tr>
<td>7.2</td>
<td>Discussion</td>
<td>111</td>
</tr>
<tr>
<td>7.3</td>
<td>Future Work</td>
<td>112</td>
</tr>
<tr>
<td>A</td>
<td><strong>Stop Retention Full Results</strong></td>
<td>115</td>
</tr>
<tr>
<td>B</td>
<td><strong>Edit Distance Soft-Max Kernel Proof</strong></td>
<td>121</td>
</tr>
<tr>
<td>CONTENTS</td>
<td>V</td>
<td></td>
</tr>
</tbody>
</table>

<p>| SUMMARY       | 125 |
| SAMENVATTING  | 129 |
| BIBLIOGRAPHY  | 133 |
| SIKS DISSERTATION SERIES | 143 |</p>
<table>
<thead>
<tr>
<th>ACRONYMS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSD</td>
<td>Positive Semi-Definite</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>RDF</td>
<td>Resource Description Framework</td>
</tr>
<tr>
<td>STDM</td>
<td>Spatio-Temporal Data-Mining</td>
</tr>
<tr>
<td>AIS</td>
<td>Automatic Identification System</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>MSS</td>
<td>Maritime Safety and Security</td>
</tr>
<tr>
<td>MMSI</td>
<td>Maritime Mobile Service Identity</td>
</tr>
<tr>
<td>DTW</td>
<td>Dynamic Time Warping</td>
</tr>
<tr>
<td>ED</td>
<td>Edit Distance</td>
</tr>
<tr>
<td>LCSS</td>
<td>Longest Common SubSequence</td>
</tr>
<tr>
<td>PLS</td>
<td>Piecewise Linear Segmentation</td>
</tr>
</tbody>
</table>
**SYMBOLS**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{X}$</td>
<td>An input space.</td>
</tr>
<tr>
<td>$\mathcal{Y}$</td>
<td>An output space.</td>
</tr>
<tr>
<td>$\mathbb{N}$</td>
<td>The set of natural numbers.</td>
</tr>
<tr>
<td>$\mathbb{R}$</td>
<td>The set of real numbers.</td>
</tr>
<tr>
<td>$h$</td>
<td>A hypothesis function $h: \mathcal{X} \to \mathcal{Y}$.</td>
</tr>
<tr>
<td>$x$</td>
<td>A vector.</td>
</tr>
<tr>
<td>$T, S$</td>
<td>Trajectories as sequences of elements, i.e. $T = \langle x_1, y_1, t_1 \rangle, \langle x_2, y_2, t_2 \rangle, \ldots, \langle x_n, y_n, t_n \rangle$.</td>
</tr>
<tr>
<td>$</td>
<td>T</td>
</tr>
<tr>
<td>$\mathcal{T}$</td>
<td>A set of trajectories as sequences of elements, i.e. $\mathcal{T}$.</td>
</tr>
<tr>
<td>$T: \mathbb{R} \to \mathbb{R} \times \mathbb{R}$</td>
<td>Trajectories as continuous functions, i.e. $T$.</td>
</tr>
<tr>
<td>$k(., .)$</td>
<td>A kernel function.</td>
</tr>
<tr>
<td>$K$</td>
<td>A kernel matrix.</td>
</tr>
<tr>
<td>$E(., ., .)$</td>
<td>Error function used in piecewise linear segmentation.</td>
</tr>
<tr>
<td>$\pi$</td>
<td>Alignment between two trajectories $S$ and $T$.</td>
</tr>
<tr>
<td>$</td>
<td>\pi</td>
</tr>
<tr>
<td>$s(S, T, \pi)$</td>
<td>Score of alignment $\pi$ between two trajectories $S$ and $T$.</td>
</tr>
<tr>
<td>sub$(., .)$</td>
<td>A substitution function.</td>
</tr>
<tr>
<td>$g$</td>
<td>Gap penalty.</td>
</tr>
<tr>
<td>$\text{sim}(S, T)$</td>
<td>Alignment similarity function between two trajectories $S, T$.</td>
</tr>
<tr>
<td>$D(S, T)$</td>
<td>A distance function between continuous trajectories $S, T$.</td>
</tr>
<tr>
<td>$d(., .)$</td>
<td>A distance function between points in a trajectory, often the Euclidean distance.</td>
</tr>
</tbody>
</table>
Symbols

β

A scaling parameter, for instance used in the Gaussian kernel.

Parameters labels

We use the convention that parameters are indicated with a superscript and labels, indicating which variant of a function is used, with a subscript.
Let me start these words of gratitude by saying that I thoroughly enjoyed my four years as a PhD-student. A first year student is often warned by his PhD-elders for the stress and feelings of helplessness that he will encounter along the way. Fortunately, I have never really suffered much from those. This is in no small part due to the help, support and friendship of the people that I am about to thank in the paragraphs below.

First of all, I am very much indebted to my great supervisors, Maarten van Someren and Pieter Adriaans. The relative ease with which I feel that this thesis was put together can be very much attributed to your stress-free attitude towards doing quality research. Together you created an environment in which I felt very comfortable doing my job as a PhD-student.

Pieter, due to unfortunate circumstances, our cooperation was not as close as I had hoped. However, I have always enjoyed, and learned from, your theoretical and overarching scientific interests and insights. I am glad that I continue to work with you on a new project and like to go sailing one day.

Maarten, I enjoyed having you as a daily supervisor very much, always being enthusiastic about my ideas and those of others. You gave me a lot of freedom in deciding my research direction, but also steered me when necessary. My work has greatly benefited from your attention to detail and clarity. I hope and think that we will work together in the future.

I am very thankful to my committee members, Cees de Laat, Frans Groen, David Aha, Eric Postma and Marc van Kreveld, for agreeing to be on my committee and critically reading my thesis.

I enjoyed being part of the Poseidon team while doing my research. I especially thank my co-authors from the project, Willem van Hage, Véronique Malaisé, Jeroen Janssens, Niels Willems and Guus Schreiber. The project was managed very well by the people from the Embedded Systems Institute: Frans Reckers, Michael Borth, Piërre van de Laar and Jan Tretmans. Furthermore, I got to work with some nice people within Thales: Maurice Glandrup, Jacek Skowronek, Hans Hiemstra, Bob Eskes and Huub de Waard.

Within the University of Amsterdam I had a lot of fine and wonderful colleagues. We had very fun and fruitful discussions within the AIM subgroup, thanks to: Victor, Sophia, Niels, Vera and Wico. Having moved three times, I have had many different cool offices mates: Mattijs, Floris, Jochem, Miriam, Henriëtte, Wouter, Patrick, Sander, Jan, Katja, Jafar, Naser, Mahdi, Mahdieh, Hodjat and Sijali. And then
there are the other nice people at HCS and FCN: Simon, Bob, Frank, Vanessa, Bert, Wouter, Hamideh, Jacobijn, Frank and Saskia.

For quite a while I was part of the institute’s borrelcommissie. Tim, Edgar, Julian, Wouter, Gosia and Virginie, too bad we could not continue the borrels in the new building. During my last year or so, I played a lot of relaxing football, during and after office hours, with the ISLA futsal group and as a member of the team. It is time to become champions, dudes.

Dan komen we bij mijn lieve vrienden en vriendinnen, die allen voor de nodige afleiding hebben gezorgd. Thijs, Angela, Coen, Theun, Annelot, Daan, Joska, Menno, Martyn, Remy, Leslie, Joost, Thatcher, Annelies, bedankt voor jaren en jaren goede vriendschap. En natuurlijk, Jerke, Yvonne, Marloes, Romi, Jana, Franc, en de rest van Sjambacajow, in de afgelopen jaren hebben we veel opgetreden en heel veel andere leuke dingen gedaan, bedankt! Verder, wil ik alle andere mensen met wie ik de afgelopen tijd een biertje of koffie heb gedronken, of iets anders gezelligs heb gedaan, bedanken.

Mijn zorgzame en lieve ouders, Frits en Annemarie, en mijn eigenwijze broertje, Roelof, hebben altijd veel belangstelling gehad voor mijn onderzoek en me gesteund tijdens mijn hele opleiding. Bedankt voor alles.

En tenslotte, mijn moppie Femke. Ze zeggen dat je tijdens een promotie eigenlijk geen tijd hebt om verliefd te worden, zeker in het laatste jaar. Dat heeft ons gelukkig niet tegengehouden! Nu gaan we leuke dingen doen.
INTRODUCTION

In the past decade, tracking of people and objects in geographical space has become ubiquitous. Smart phones have GPS sensors, cars are equipped with navigation systems and vessels are carrying special transponders to transmit information called AIS. All of this tracking data can easily be stored, generating a type of data called the moving object trajectory.

In most cases, these moving object trajectories are tracked and stored as is, i.e. they have no attached labels indicating, for instance, what the person or object is doing. Adding semantics to the tracks can help end-users or operators to interpret the data. For instance, tracking passengers in an airport and grouping these into different passenger types, such as tourist and business man, can greatly help in optimizing walking route lay-outs.

Another example is in maritime safety and security, where vessels can be tracked with GPS or radar. Trajectories made by fishing vessels are totally different from the paths generated by tankers or cargo ships, which follow regular routes. Identifying whether a trajectory belongs to one or the other class can help in determining whether a vessel exhibits unwanted behavior and thus should be further investigated.

Consider all the vessel movements that occur around a major port, like the busy harbors of Singapore and Rotterdam. Grouping these movements into clusters of similar behavior can help to get an overview of the general movement patterns. This overview can assist the operator to better spot irregular movements. Performing tasks like those mentioned above on vessel trajectories is the subject of study in this thesis.

One option to perform these tasks of assigning and discovering labels and classes for vessel trajectories is to design algorithms for them by hand. This is time consuming and difficult, and made more complicated by the inherent sensor noise in the data. However, we can also use automatic techniques from the fields of machine learning and data-mining that work with available historical movement data directly. This is the approach that we take in this thesis.

1.1 APPROACH

The tasks of assigning semantic labels to vessel trajectories are handled using techniques from machine learning and data-mining. We consider clustering of trajectories to discover groups of similar move-
ment behavior. Predicting the type of vessel belonging to a trajectory is a classification task. In outlier detection we try to find a strange vessel trajectory among a set of normal trajectories. These three tasks assign different types of semantic labels to trajectories.

The main element in solving these three tasks is computing a similarity measure between vessel trajectories. We study alignment measures from the time-series field and similarities, based on taking the integral of the distance between trajectories over time, from the field of computational geometry. Moving object trajectories essentially combine these two fields. On the one hand, they are multidimensional time-series. On the other hand, their shape is more important than in traditional time-series, and shapes are the subject of computational geometry.

Vessels produce temporally long trajectories that are very regular, therefore we investigate ways to compress the trajectory data. Compression is a useful preprocessing step, that allows higher level reasoning by creating larger units of movement events. Also, it speeds up further computations of alignment measures, although this potentially has a negative impact on task performance.

The machine learning algorithms for clustering, classification and outlier detection that we use in conjunction with trajectory similarity measures are kernel methods. The similarity measures are therefore defined as kernels. Kernel methods provide a unified framework in which the kernel is the central component for performing different tasks. For classification and outlier detection we use Support Vector Machines, which are well behaved and understood algorithms. Kernels are positive semi-definite functions, but trajectory similarities are not. Therefore we apply, and sometimes define new, corresponding positive semi-definite kernels for each trajectory similarity.

Vessel trajectories exist in a geographical space for which there are a lot of semantic concepts, such as harbor types, shipping lanes and anchoring areas. Adding this information to the similarity measures can potentially improve the performance on the clustering, classification and outlier detection tasks. Alignment measures and kernel methods allow for good integration of this domain knowledge.

1.2 RESEARCH QUESTIONS

The goal of this thesis is to provide automatic techniques to label vessel trajectories with semantic labels, such as the type of vessel or whether it shows strange behavior. The vessel trajectory data have a number of properties that make it interesting to apply automatic methods. First, there is often a large amount of data per trajectory. And, second, these trajectories are temporal, and hence sequential, in nature and commonly of different length.

In general, we investigate the following research question.
1.2 Research Questions

How do we, effectively and efficiently, assign semantic labels to vessel trajectory data?

We answer this question by applying machine learning techniques to vessel trajectories. By using classification, types can be assigned. Clustering is used for grouping vessels into similar behavior classes. Outlier detection allows for discovering abnormal behavior among regular trajectories. The algorithms that we use are based on kernel methods. While answering the above question by applying machine learning, we pose a number of more specific research questions.

The first question addresses the problem that each trajectory is based on a lot of samples, and these samples are often redundant.

1. How do we reduce the large volume and redundancy of vessel trajectories, and make them suitable for higher level reasoning?

This question is answered in Chapter 3 by looking at a trajectory compression technique. We adapt this technique in such a way that we generate representations of trajectories that can be used in higher level reasoning, for which we give a number of examples. This compression is also the first step in applying machine learning algorithms to vessel trajectories.

Secondly, we consider the issue of how to perform typical machine learning tasks with trajectories. The sequential and non-fixed length nature of trajectories suggests a similarity measure, as opposed to feature based, approach to do the tasks of clustering, classification and outlier detection. More specifically we answer the question:

2a. How do we compare vessel trajectories to be able to apply machine learning techniques?

In Chapters 4 and 5 we investigate this question by defining a common set of similarity measures from the fields of time-series analysis and computational geometry and use these in clustering, classification and outlier detection. We test these measures in a kernel method framework and research both their traditional and kernelized versions.

To further investigate the use of trajectory compression we also look at the following question.

2b. What is the influence of applying trajectory compression on comparing vessel trajectories for machine learning?

This question is answered in Chapter 4 by applying a subset of the similarity measures that is sensitive to compression to uncompressed and compressed trajectory data for a range of compression settings.

Most types of moving object trajectories occur in a space with semantics. In contrast to, for instance, handwritten digits or eye movements, our trajectories occur in a geographical world filled with places and regions. Utilizing this type of information can enhance machine learning and leads to the following question.
3. How do we incorporate geographical domain knowledge to improve machine learning for vessel trajectories?

We answer this question in Chapter 6 by defining similarity measures that can handle both raw trajectory information and conceptual domain knowledge simultaneously. We show how this measure can improve classification performance over using just raw trajectories or just geographical domain knowledge and give clustering results that illustrate the discovery of concepts that combine raw trajectories and domain knowledge.

1.3 POSEIDON PROJECT

The research presented in this thesis has been carried out in the context of the Poseidon project. The aim of this project was to develop a Maritime Safety and Security (MSS) system. To create such a system, research had to be done on integration and testing of the components of such a system, and also on the data that such an MMS system provides and needs to analyze. For the research in this thesis we worked on interpreting and analyzing the vessel trajectory data created by the MSS system and linking this data to higher level concepts and labels. We did this by creating solutions for the three tasks of clustering, classification and outlier detection, introduced above.

A summary of the research and achievements of the entire Poseidon project, including our own, is to be published in the forthcoming book by van de Laar et al. (to appear).

1.4 OUTLINE

The rest of this thesis is structured as follows.

In Chapter 2 we give preliminary definitions and concepts that are used throughout the rest of the thesis. We give a brief introduction to machine learning and kernel methods. Then we give an overview of the field of spatio-temporal data-mining, in which the research in this thesis is situated. We end with introducing the datasets that we used in this thesis and the Poseidon project.

We extend an existing trajectory compression technique in Chapter 3 so that it works better with vessel trajectory data, which we show with experimental results. We also illustrate how this compression can be used as the first step in higher level reasoning.

Alignment similarity measures are the subject of Chapter 4. We study two common measures, and their kernel versions. The presented experiments investigate which measures perform best in the three tasks of clustering, classification and outlier detection. Furthermore, we specifically consider what effect the compression method

1 http://www.esi.nl/poseidon
from the previous chapter has on task performance using alignment measures.

In Chapter 5 we study a family of integral over time based similarity measures. For these measures we give corresponding positive semi-definite kernel versions, some of which did not exist previously in the literature. In the experimental section we provide results on which measures perform best and the difference with alignment techniques.

Chapter 6 investigates the use of geographical domain knowledge in the machine learning tasks. This knowledge is incorporated in the alignment similarity measures.

Finally, Chapter 7 revisits the research questions posed in the Introduction and answers them.
This chapter introduces concepts and definitions that are used throughout this thesis. We begin with a short introduction to machine learning and more specifically to kernel methods. Then we give an overview of the research and necessary concepts in spatio-temporal data-mining. We end with a description of the tasks of a maritime safety and security system and the specific datasets of vessel trajectories used in our experiments.

2.1 MACHINE LEARNING

In machine learning several different learning tasks with respect to an input space of objects $X$ are recognized. The most iconic task in machine learning is a form of supervised learning, called classification. In classification the input space $X$ is labeled from a set of labels $Y$. A training example is then an instance $(x_i, y_i) \in X \times Y$. The goal for a learning algorithm is to find a hypothesis function, or classifier, $h: X \rightarrow Y$, which predicts a label from $Y$ for any instance of $X$. The function $h$ is an approximation of some unknown target function $t$ that describes the data optimally, thus ideally we would like to have $\forall x_i \in X: h(x_i) = t(x_i)$. However, for any realistic problem, the training set is a (small) subset of $X$ and contains noise, and therefore $h$ will most likely never be equal to $t$. Multiple strategies, i.e. multiple learning algorithms, exist to identify $h$, which all have different advantages and disadvantages; some handle noise better, some can deal well with skewed training sets, and others require very little training time.

The prototypical unsupervised machine learning task is clustering. In clustering we have a dataset from an unlabeled input space $X$, and we construct a partitioning of this dataset into clusters $\{c_1, \ldots, c_k\}$. To construct such a partitioning, a notion of similarity between objects in $X$ is required. Clustering algorithms usually have parameters that control the number of clusters that are found. For centroid based algorithms, like k-means, the parameter $k$ directly sets the number of clusters created. Density based algorithms, such as DBSCAN (Ester et al. (1996)), consider as clusters groups of objects that are close together, or densely connected. These algorithms have parameters to regulate this density, in terms of the closeness together and the number of objects required in a cluster.

Generally the task of outlier detection is considered a form of unsupervised learning, since usually the dataset that belongs to input
space $\mathcal{X}$ is unlabeled. The goal is to learn a function $h: \mathcal{X} \rightarrow \{-1, +1\}$, where the value $-1$ indicates an outlier or abnormal observation, and $+1$ an inlier or normal observation. The main reason that outlier detection is considered an unsupervised task is that there is great variation in types of outliers, and there is often no single class of outlier. Moreover, the available training data (almost) only contains normal observations, since outliers are supposed to occur infrequently. Thus, an outlier detection algorithm constructs a model $h$ of the training data, such that for (nearly) all $x_i \in \mathcal{X}$, $h(x_i) = +1$. If a new observation $x'$ fits the model well enough to be considered an inlier, then $h(x') = +1$, else $h(x') = -1$, and $x'$ is an outlier.

The structure of the objects in $\mathcal{X}$ and the models that a machine learning algorithm creates can vary widely. The most common type of object in $\mathcal{X}$ is the vector of a fixed number of features $m$: $x = \langle x_1, \ldots, x_m \rangle$. These features can be nominal or numerical. Traditional machine learning algorithms like rule learners, decision trees and Naive Bayes work on these representations (Mitchell (1997)). However, more complex types of data, like moving object trajectories, can be difficult to fit into the fixed length feature vector type of representation. In this case, methods based on similarity provide a solution. A similarity measure for complex data is often more straightforward and natural to define than a feature vector representation. Clustering and outlier detection are naturally suited to use similarity measures. For classification the $k$-nearest neighbor algorithm can be used. Kernel methods, which we describe below, are also based on similarity measures.

A possible alternative to a similarity based approach is to use a probabilistic (Bayesian) framework. Trajectories can be modeled as (non)-linear dynamic systems, using Kalman filters, hidden Markov models, or more advanced dynamic Bayesian network techniques. However, the modeling choices needed to represent trajectories as probabilistic variables are non-trivial, given that there is no fixed set of times and locations. Classification can be done by training models for each class. For a new instance we select as its class the one that belongs to the model that has the highest probability of generating that instance. Outlier detection can be done by training models for the normal data and labeling new instances as outliers when they have a low probability of being generated by one of the models. For clustering a form of similarity between probabilistic models for trajectories is needed.

We feel that a similarity/kernel based framework fits moving object trajectory data well. It is a direct approach to solving machine learning tasks in this domain. However, we recognize that other solutions can also be viable for trajectories.
2.1.1 Kernel Methods

Kernel methods are machine learning techniques that use the notion of similarity between objects (Schölkopf and Smola (2001); Shawe-Taylor and Cristianini (2004)). The framework of kernel methods provides a unified way to handle different machine learning tasks, like classification and clustering. For each task well-known algorithms exist to solve it, given that we can define a kernel function, introduced below, between the objects in our dataset. Kernels are most often associated with the Support Vector Machine (SVM) classification algorithm (Cortes and Vapnik (1995); Schölkopf et al. (2000)). However, kernels can also be used for clustering (Dhillon et al. (2007)), and for outlier detection (Schölkopf et al. (2001)).

The key insight of kernel methods is that a number of machine learning techniques, like the SVM and the perceptron, can be redefined to use a dot-product between vectors when creating a model of the data, and that this dot-product can be replaced by a mathematically more general notion called a kernel. This is what is referred to as the kernel trick. Kernels can be defined for objects with other structures than feature vectors, like trees, graphs or trajectories, in contrast to dot-products, which are computed between same length feature vectors. Many types of kernels create an implicit mapping of objects into a higher dimensional feature space. This mapping is implicit, because the kernel function directly computes the dot-product in this higher dimensional feature space, without computing the mapping. Computing this is often much cheaper than doing the mapping explicitly, if it all, and computing the dot-product in the high dimensional space.

In order to ensure that kernel functions represent a dot-product in a certain feature space, the functions need to be Positive Semi-Definite (PSD) as defined in Definition 2.1.1. A simple example of such a function is the canonical dot-product $x \cdot x'$ or the identity function.

**Definition 2.1.1.** Let $\mathcal{X}$ be a set of objects. Then a function $k: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a positive semi-definite kernel iff $k$ is symmetric, i.e. $\forall x, x' \in \mathcal{X} : k(x, x') = k(x', x)$ and,

$$\sum_{i,j=1}^{n} a_i a_j k(x_i, x_j) \geq 0 ,$$

for any $n \in \mathbb{N}$, $a_1, \ldots, a_n \in \mathbb{R}$ and $x_1, \ldots, x_n \in \mathcal{X}$.

The assumption of positive semi-definiteness is essential to ensure convexity in the quadratic programming approach used in Support Vector Machines.\(^1\)

\(^1\)The machine learning literature often refers to the properties of positive definiteness and positive semi-definiteness interchangeably, since the difference is not of great importance for machine learning. We will always use the term Positive Semi-Definite (PSD).
In practice, good machine learning results have also been achieved with kernels that are indefinite, i.e. still symmetric but not positive semi-definite. Some research has been done on their behavior, see Haasdonk (2005). In this thesis we will use both indefinite and PSD kernels.

Using simple kernels, more complex kernels can be constructed using different kernel closure properties. For a family of PSD kernels $k_1, \ldots, k_n$ the following holds (Berg et al. (1984)):

- the sum $\sum_{i=1}^n c_ik_i$ is a kernel, given $c_1, \ldots, c_n \geq 0$,
- the product $k_1^{c_1} \cdots k_n^{c_n}$ is a kernel, given $c_1, \ldots, c_n \in \mathbb{N}$,
- the limit $k := \lim_{n \to \infty} k_n$ is a kernel, if the limit exists.

These closure properties can be used to construct more complex kernels from simpler kernels.

A PSD kernel that we will encounter in the rest of the thesis is the Gaussian kernel, for two vectors $x, x'$ and standard deviation $\sigma$:

$$k_{\sigma}(x, x') = \exp\left(-\frac{\|x - x'^2\|}{2\sigma^2}\right). \quad (2.1)$$

Often, the $\frac{1}{2\sigma^2}$ part is replaced by just the scaling parameter $\beta > 0$.

We note that we refer to both a kernel function and the kernel matrix based on that function as kernel. However, we will use the small $k$ for functions and the capital $K$ for matrices.

**Mapping kernels**

To construct kernels on more complex structures than same length vectors, different solutions have been proposed. In this thesis we make use of mapping kernels (Shin and Kuboyama (2008)). Mapping kernels are generalizations of convolution kernels as defined by Haussler (1999).

**Definition 2.1.2.** A mapping kernel on a set of objects $\mathcal{X}$ requires a map $M(x, y) \subseteq \mathcal{X}' \times \mathcal{X}'$ between two objects $x, y \in \mathcal{X}$, where each $x \in \mathcal{X}$ is associated with a finite subset $\mathcal{X}'_x$ of a common space $\mathcal{X}'$. We define the mapping kernel:

$$k(x, y) = \sum_{(x'_i, y'_i) \in M(x, y)} k'(x'_i, y'_i).$$

This kernel is positive semi-definite iff the kernel $k'$ is positive semi-definite on $\mathcal{X}'$ and the map $M$ is transitive: $\forall x, y, z \in \mathcal{X} ((x', y') \in M(x, y) \land (y', z') \in M(y, z)) \rightarrow (x', z') \in M(x, z))$.

The approach to constructing a mapping kernel is to find a common space of less complex structured objects, called $\mathcal{X}'$, for the objects in $\mathcal{X}$, such that a positive semi-definite kernel can be defined on $\mathcal{X}'$. For instance, an example of complex objects are strings, and less
structured objects are the characters that make up the strings. A PSD kernel is easily defined on characters, e.g. the identity function. Each more complex object $x \in X$ is associated with a finite subset of $X'$, indicated as $X'_x$. In the strings example, a string would be associated with the characters that make up the string. Next, we need a transitive map $M$ between two $x, y \in X$. This map $M$ is a subset of $X_x' \times X_y'$. With the map $M$ we can create a positive semi-definite kernel on $X$, using the positive semi-definite kernel on the common space $X'$, as in Definition 2.1.2. A possible transitive map for the string example would be to map each character, with position, in one string to each character in the other string.

2.1.1.1 Support Vector Machines

The Support Vector Machine is a method that seeks to find a binary classifier for points in a $d$-dimensional space, by finding their optimal linear separation. For points $x_i \in X$ belonging to two classes, $y = +1$ or $y = -1$, it tries to find values for $w$ and $b$ such that the function

$$f(x_i) = w \cdot x_i + b,$$  \hspace{1cm} (2.2)

optimally separates the two classes. This means that the hyperplane defined by $w \cdot x + b = 0$ has the same distance to the closest point with class $+1$ as to the closest point with class $-1$. Figure 1 illustrates this for the 2-dimensional separable case.

The general version of this problem, i.e. for datasets that are not fully linearly separable, can be solved by minimizing

$$\frac{1}{2}||w||^2 + C \sum_i \xi_i,$$  \hspace{1cm} (2.3)
with respect to the following constraints:

\[ x_i \cdot w + b \geq +1 - \xi_i \quad \text{for } y_i = +1 , \]
\[ x_i \cdot w + b \leq -1 + \xi_i \quad \text{for } y_i = -1 , \]
\[ \xi_i \geq 0 \quad \forall i . \]

Here, \( \xi_i \) are so-called slack variables to deal with the non-separability, and \( C \) is a user determined cost parameter to set the influence of classification errors.

In practice, Lagrange multipliers \( \alpha_i \) are added to solve this optimization problem and a dual form

\[ \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j , \]

is introduced, which needs to be maximized, subject to

\[ 0 \leq \alpha_i \leq C , \]
\[ \sum_i \alpha_i y_i = 0 . \]

In this dual form we see why SVMs are considered a kernel method. The dot-product \( x_i \cdot x_j \) can be replaced by a kernel \( k(x_i, x_j) \). All \( x_i \) for which \( \alpha_i > 0 \) are called support vectors. These are the vectors that “support” the separating hyperplane. Multiple options exist for extending SVMs to multi-class classifiers. For more details and an excellent tutorial see the work by Burges (1998).

**One-Class SVM** For outlier detection there exists a type of Support Vector Machine called the one-class SVM (Schölkopf et al. (2001)). A one-class SVM tries to create a model of unlabeled data. The technique tries to separate most of the data points from the origin with maximum margin, using a hyperplane. This idea results in equations very similar to regular SVMs. One-class SVMs try to minimize:

\[ \frac{1}{2} \|w\|^2 + \frac{1}{\nu n} \sum_i \xi_i - \rho , \]

subject to

\[ x_i \cdot w \geq \rho - \xi_i \quad \forall i , \]
\[ \xi_i \geq 0 \quad \forall i . \]

Here, \( \rho \) is the distance of the hyperplane to the origin, \( n \) is the number of data points and \( \nu \) is a user configurable parameter that controls the trade-off between the number of support vectors and the amount of data captured by the model. To solve this optimization problem a dual form similar to regular SVM exists, which can also be expressed in terms of dot-products and hence can make use of a kernel function.
Due to the popularity of Support Vector Machines there exist a lot of good implementations that can be used off-the-shelf. We use the library LibSVM (Chang and Lin (2001)), which is freely available and can be easily called directly from MatLab. LibSVM implements a multi-class version of the often used C-SVM, which we described above, and also has an implementation of one-class SVM.

2.1.1.2 Kernel k-means

The popular and well-known clustering algorithm k-means can be turned into a kernel based algorithm, dubbed kernel k-means, in a relatively straightforward manner (Shawe-Taylor and Cristianini (2004); Dhillon et al. (2007)). This adds flexibility to the original algorithm in terms of the possible input data. Other kernel based clustering algorithms exist as well, such as support vector clustering (Ben-Hur et al. (2001)), which does not require the specification of the number of clusters k, but has parameters for controlling the neighborhood.

For a set of points $x_i \in \mathcal{X}$, regular k-means tries to find a clustering $C$ into k partitions:

$$C = \{c_1, \ldots, c_k\} ,$$

such that for all $x_i \in \mathcal{X}$, $x_i$ is in exactly one $c_i \subseteq \mathcal{X}$, and that minimizes the objective function:

$$\sum_{j=1}^{k} \sum_{x_i \in c_j} \|x_i - m_j\|^2 ,$$

where

$$m_j = \frac{\sum_{x_i \in c_j} x_i}{|c_j|} .$$

In kernel k-means, the squared distance $\|x_i - m_j\|^2$ is rewritten using dot-products, which can be replaced by kernels $k(x_i, x_j)$ between arbitrary objects $x_i, x_j$,

$$k(x_i, x_i) - \frac{2}{|c_j|} \sum_{x_l \in c_j} k(x_i, x_l) + \frac{1}{|c_j|^2} \sum_{x_l, x_m \in c_j} k(x_l, x_m) .$$

In Dhillon et al. (2007), weights $w_i$ are added for each $x_i$ to create weighted kernel k-means, leading to the objective function:

$$\sum_{j=1}^{k} \sum_{x_i \in c_j} w_i \left( k(x_i, x_i) - \frac{2}{|c_j|} \sum_{x_l \in c_j} w_l k(x_i, x_l) \right) + \frac{\sum_{x_l, x_m \in c_j} w_l w_m k(x_l, x_m)}{\left( \sum_{x_l \in c_j} w_l \right)^2} .$$
Using different settings for the weights $w_i$, variants of popular graph cut based clustering algorithms can be solved without doing expensive eigen value computations. For instance, the often used normalized cut (Shi and Malik (2000)) can be computed using weighted kernel k-means. Let $K$ be the kernel matrix computed by kernel function $k$. If we let $D$ be the diagonal degree matrix with $D_{ii} = \sum_j K(i,j)$, then $w_i = D_{ii}$ and $K' = D^{-1}K D^{-1}$, where $K'$ is the kernel matrix used in weighted kernel k-means. The advantage of the normalized cut over regular kernel k-means is that normalized cut can handle clusters of different sizes better.

To compute weighted kernel k-means the most straightforward strategy is the same as for regular k-means. This means that we start out with a random initialization of $k$ clusters. At each iteration we try to minimize the distance between the cluster centers and objects in the clusters by first assigning objects to the closest cluster and then recomputing the cluster centers, until we arrive at a stable clustering. Because different random initializations can lead to different stable partitions $C$, the kernel k-means clustering is run a number of times. The partitioning with the lowest intra cluster spread is kept as the final clustering.

### 2.1.2 Evaluation Metrics

In machine learning different metrics are used to evaluate the performance of different algorithms. The basic score to measure the performance of classification algorithms is accuracy. Let $X_{\text{test}} \subset X \times Y$ be a test set and let $h$ be a trained classifier, then accuracy is

$$\text{accuracy}(h, X_{\text{test}}) = \frac{\sum_{(x_i, y_i) \in X_{\text{test}}} [h(x_i) = y_i]}{|X_{\text{test}}|},$$

i.e. simply the fraction (usually given as a percentage) of correctly classified instances in the test set.

In evaluating a binary classifier, such as an outlier detector, with class labels $Y = \{+1, -1\}$ the measures of precision and recall are often used. Let $\text{pos}_h = \{x_i \mid (x_i, y_i) \in X_{\text{test}}, h(x_i) = +1\}$ and $\text{pos}_{\text{test}} = \{x_i \mid (x_i, y_i) \in X_{\text{test}}, y_i = +1\}$. Precision is

$$\text{precision}(\text{pos}_h, \text{pos}_{\text{test}}) = \frac{|	ext{pos}_h \cap \text{pos}_{\text{test}}|}{|\text{pos}_h|}.$$  \hfill (2.19)

This is the fraction of instances that are classified as positive that also belong to the positive class. Recall is

$$\text{recall}(\text{pos}_h, \text{pos}_{\text{test}}) = \frac{|	ext{pos}_h \cap \text{pos}_{\text{test}}|}{|\text{pos}_{\text{test}}|},$$

\hfill (2.20)
which is the fraction of positive instances that are classified as positive. Precision and recall are often combined into the $F_1$-measure,

$$F_1(pos_h, pos_{test}) = \frac{2|pos_h \cap pos_{test}|}{|pos_h| + |pos_{test}|},$$

which is the harmonic mean between precision and recall.

In tasks where a classifier produces a ranked list of results, for instance in outlier detection, we are often only interested in the top-$n$ most results. Let $X_{test}^n$ be the $n$ highest ranked positive results. Then precision@n is computed as the precision for the the set $X_{test}^n$, which gives the fraction of true positives among the top-$n$. Note that the definitions above can easily be applied to the negative class, which we are interested in in outlier detection, when we switch $+1$ and $-1$.

To evaluate the quality of a clustering compared to a gold standard we use a scoring function that takes the best $F_1$-score for each cluster $g_i$ in the gold standard $G$ and average over these scores (Liao (2005)). Thus the clustering score function $clus\_score$ of a clustering $C$ of size $k$ with respect to a gold standard $G$ of size $k$ is

$$clus\_score(C, G) = \frac{1}{k} \sum_{g_i \in G} \max_{c_j \in C} F_1(c_j, g_i).$$

### 2.2 Spatio-Temporal Data-Mining

The field of machine learning/data mining that is concerned with moving object trajectories and other types of spatio-temporal data is called Spatio-Temporal Data-Mining (STDM). STDM is a relatively young area of research for which there have been a number of workshops. For instance the series of Workshops on Spatial and Spatio-Temporal Data-Mining (SSTDM), such as the one in 2010 as part of ICDM 2010 (Fan et al. (2010)). Work in the STDM area builds upon work in Spatial Data-Mining and Temporal or Time-Series Data-Mining. One particular problem that results from the infancy of the field is the lack of standard datasets for testing new methods. A good overview of the research that has been done until recently is given by Nanni et al. (2008).

#### 2.2.1 Moving Object Trajectories

Research in STDM has a strong focus on solving typical machine learning/data-mining tasks, such as clustering, pattern mining and classification, for spatio-temporal data. The most common type of spatio-temporal data is that of the moving object trajectory. Moving object trajectories are continuous movements in a geographical space. This continuous movement is sampled using Global Positioning System (GPS) sensors or similar techniques. In the following we will
mostly consider a trajectory from the sampled perspective, i.e. as a sequence of temporally labeled positions. A difference is made between unconstrained trajectories, e.g. planes and animals, and constrained trajectories, e.g. cars that follow a road network.

We define a moving object trajectory formally in Definition 2.2.1.

**Definition 2.2.1.** A moving object trajectory $T$ in 2-dimensional space is represented by a sequence of vectors: $T = \langle x_1, y_1, t_1 \rangle, \langle x_2, y_2, t_2 \rangle, \ldots, \langle x_n, y_n, t_n \rangle$, where $x_i$ and $y_i$ represent the position of the object at time $t_i$ and $t_{i+1} > t_i$. The length of a trajectory, i.e. the number of vectors, is denoted as: $|T|$. Furthermore, let $T(i) = \langle x_i, y_i, t_i \rangle$ and $T(i, j) = \langle x_i, y_i, t_i \rangle, \ldots, \langle x_j, y_j, t_j \rangle$.

The sample rate of trajectories is not always fixed, thus the difference between consecutive values $t_i, t_{i+1}$ is not the same. In some tasks, there are more dimensions to trajectories that can be derived from the $x, y, t$ information that we need to consider, such as speed and direction. These dimensions can just be added to the $\langle x, y, t \rangle$ vector as extra variables. In the following we refer to a vector $\langle x_i, y_i, t_i \rangle$ as a trajectory element or point. In the analysis of trajectories the concepts of stop and move are essential (Spaccapietra et al. (2008)). What should be considered stops and moves is application dependent. A stop is the non-empty time interval for which the traveling object does not move from the application’s perspective, the rest of the trajectory is considered a move.

The approaches to data-mining for moving object trajectories can be partitioned into two broad types (Nanni et al. (2008)). On the one hand we have the approach that focuses on using standard data-mining/machine learning techniques and applying these to trajectory data. This requires defining features or similarity measures, depending on the technique, for trajectories. In this approach, all knowledge about trajectory data is put into the features/similarity measures. On the other hand there is the type of data-mining for trajectories that has the aim to specifically design methods for this type of data, using the specific properties of trajectory data in the machine learning algorithms.

### 2.2.2 Trajectory Similarity Measures

Due to the nature of trajectories being different in temporal length, distance traveled and the number of data points, using a feature based approach is not so obvious. Often similarity measure based techniques are used for trajectories, for which there are a number of possibilities. Note that we use the term ‘similarity’ generically, for any measure that expresses some form of ‘sameness’ between trajectories. Thus, we use it for ‘distances’, where a larger value means less equal, ‘similarities’, where a larger value means more equal, and ‘dissimilarities’, where larger values mean less equal.
One approach is to consider a trajectory as a sequence of points. For two trajectories we search for an optimal alignment between their points, according to some scoring scheme. How this alignment is determined and the scoring scheme is defined leads to different similarity measures. For instance, there exists: Dynamic Time Warping (DTW) \cite{Vlachos2004}, and various forms of Edit Distance (ED), such as edit distance with real penalties \cite{ChenNg2004}, edit distance on real sequences \cite{ChenOzsu2005} and Longest Common SubSequence (LCSS) \cite{Vlachos2002,Vlachos2005}. The DTW similarity measure has its origin in the time-series literature, whereas edit distance measures were originally defined for strings of characters. We will treat these methods more extensively in Chapter 4.

Instead of looking at a trajectory as a sequence, the similarity measures considered by Nanni \cite{Nanni2002,NanniPedreschi2006,vanKreveldLuo2007,Frentzos2007,Buchin2009} interpret a trajectory as a continuous function and take as a similarity measure the integral over $t$ for the distance function $d$ that gives the distance between two trajectories for each time point $t$. The measure given by Buchin et al. \cite{Buchin2009} generalizes previous versions of this measure with both a variable time-shift and a variable length of matching. Pelekis et al. \cite{Pelekis2007} describe a different method that calculates the surface between trajectories, instead of calculating the integral over time, projections to the $xy$- and $t$-plane are used.

The key difference between alignment like approaches and the integral based approach is the treatment of time. In alignments, time is often treated implicitly, i.e. only by the ordering of the sequence elements, whereas it is treated explicitly in the integral approach.

### 2.2.3 Data-Mining Tasks for Trajectories

With similarity measures such as those described above, typical tasks like clustering and classification can be performed.

For clustering, Vlachos et al. \cite{Vlachos2002} take an approach using longest common subsequence combined with a density based algorithm. An integral over time similarity measure is used by Nanni \cite{Nanni2002} with $k$-means clustering, and with density based clustering by Nanni and Pedreschi \cite{NanniPedreschi2006}. Trajectories are first converted into a grid based representation and then clustered using fuzzy $c$-means by Pelekis et al. \cite{Pelekis2009}, which is an approach not directly based on similarity measures. The same holds for the framework by Lee et al. \cite{Lee2007}, which uses segmentation of trajectories based on the minimum description length principle first, and then clusters the segments as subtrajectories using density based clustering. The somewhat older work by Gaffney and Smyth \cite{GaffneySmyth1999} employs a very different mixture of regression models method.
Very little literature exists on classification of moving object trajectories. Vlachos et al. (2005) use a one-nearest-neighbor classifier with the LCSS similarity measure. The main use for this classifier is to test the power of the measure; the actual results for the task are of less importance. Using the partition-and-group framework developed by Lee et al. (2007) a classification method for subtrajectories is defined by Lee et al. (2008b). A voting algorithm for segments of trajectories is introduced by Panagiotakis et al. (2009) to classify trajectories into their representative paths.

Also, there is not much work on outlier detection for trajectories. With the partition-and-group framework from Lee et al. (2007, 2008) an outlier detection method for subtrajectories is introduced by Lee et al. (2008a). Bu et al. (2009) define an outlier detection algorithm for trajectory streams which uses a Euclidean distance between trajectories, where the main focus is to provide an efficient datastructure to perform outlier detection online.

The prototypical data-mining task of frequent pattern mining has received a lot of attention in the spatio-temporal research. For instance, Giannotti et al. (2007) introduce a method to mine so-called T-patterns from trajectories, which is based on an existing pattern mining technique. T-patterns are concise descriptions of frequent behaviors in terms of both the regions visited during movements and the duration of the movements. Instead of searching for any type of frequent spatio-temporal pattern there is also research on finding specific patterns. For instance, this includes the search for spatio-temporal group patterns, such as, flocking, convergence and divergence (Andersson et al. (2008)). An obvious task in the context of moving objects is that of prediction, i.e. discovering where a moving object will go next. One option is to employ clustering or sequential pattern mining models for this task. For more on these types of research, see the chapter by Nanni et al. (2008).

Closely related to the field of spatio-temporal data-mining is the field of visual analytics for trajectory data. Techniques for visualization as described by, for instance, Andrienko et al. (2007, 2009), make extensive use of data-mining methods for moving objects, especially clustering.

2.2.4 Temporal Data-Mining and Kernel Methods

Moving object trajectories are a special type of time-series. Temporal data-mining has been around longer and there exists a large body of literature on it. The typical data-mining tasks mentioned above have all been studied for time-series. Some of the similarity measures mentioned earlier have their origin in time-series research. A large part of the time-series studied in temporal data-mining are rather different from moving object trajectories, e.g. financial time-series, electro car-
diagrams, etc. Typically, the objects studied are 1-dimensional time-series. For a recent overview see the paper by Fu (2011), or the less recent work by Antunes and Oliveira (2001); Roddick and Spiliopoulou (2002). Liao (2005) gives an overview of clustering methods for time-series data.

**KERNELS FOR TIME-SERIES**

To the best of our knowledge, kernel methods have not been applied to tasks on moving object trajectories. However, there exists a body of work on kernels for other types of time-series. Joder et al. (2008); Gudmundsson et al. (2008); Chaovalitwongse and Pardalos (2008) use kernels based on dynamic time warping (DTW) in time-series classification. The regular form of DTW is used by Gudmundsson et al. (2008); Chaovalitwongse and Pardalos (2008), whereas Joder et al. (2008) use DTW-kernels based on definitions given by Cuturi et al. (2007), which are guaranteed to be positive semi-definite. Kernels based on the longest common subsequence are used for classification by Gruber et al. (2006).

Kernels that are not based on alignments are also used, e.g. Rueping (2001) introduces some elementary time-series kernels. Sivaramakrishnan et al. (2007) define a kernel based on linear combinations of piecewise polynomial functions. Other types include the autoregressive kernel (Cuturi and Doucet (2011)) and the cross-correlation kernel (Wachman et al. (2009)). An edit distance based kernel is used by Chandrakala and Sekhar (2008) for density based clustering of time-series.

### 2.3 MARITIME SAFETY AND SECURITY SYSTEMS

The goal of the Poseidon project, the project in which the research in this thesis was conducted, was to create a Maritime Safety and Security (MSS) System. Such a system has the task to acquire, store and analyze data and information that are collected from various sources and enable users to analyze and operate on this information. The sources of information available in the project were vessel movement data obtained via the Automatic Identification System (AIS, introduced below), and historical information about vessels and geographical domain knowledge extracted from the Internet. Furthermore, an MSS system will likely include radar information, image and video data, which were unavailable in the project.

An important part of an MSS system is the automatic analysis of vessel movements stored by the system. The analysis of these trajectories can be further enhanced by integrating information from other sources such as geographical domain knowledge about harbors, anchoring areas, etc. Different types of analysis are possible that lead to different types of information. To gain insight into different vessel behaviors, groups of similar movements need to be created, which
is a clustering task. These unlabeled groups can be assigned meaningful labels by hand. Predicting known labels/properties, such as the vessel type, is another form of analysis, which is a classification task. Identifying strange trajectories, i.e. showing irregular behavior, among a large group of normal trajectories is an outlier detection task. More analysis tasks are possible. However, these three tasks represent three distinct and often occurring possibilities. For each of these tasks we have created experimental datasets, which we introduce below.

There exist some very specific research on detecting anomalies in vessel trajectories tracked with the Automatic Identification System. This work takes a different approach than this thesis, it tries to create probabilistic models on the basis of individual AIS position messages, i.e. they do not consider trajectories. Ristic et al. (2008) present a method using kernel density estimation, whereas Laxhammar (2008) introduces a Gaussian mixture model approach. Laxhammar et al. (2009) compare both methods and the kernel density estimation technique is found to be slightly better.

2.3.1 Vessel Trajectories from AIS

The vessel trajectory data that we use for the experiments in this thesis are collected using the Automatic Identification System (AIS). An AIS system is mandatory for vessels over 300 metric tons. The system broadcasts a number of different types of messages. The most often sent messages are the dynamic messages, with updates on position information, every couple of seconds, and the static messages, with static properties of the vessel, every couple of minutes. The dynamic message contains fields like a unique vessel identifier, the MMSI number, the position of the vessel in latitude, longitude format, the vessel’s speed and a timestamp. In the static message information about the vessel’s size, callsign and type is transmitted. Position of the vessel is most often determined using the vessel’s onboard GPS. These vessel trajectories are a type of unconstrained trajectories, though vessels moving in a harbor are sometimes considered to be constrained and even at open sea, more and more obstacles appear each year.

To compute distances between two latitude, longitude positions $(\phi_1, \lambda_1), (\phi_2, \lambda_2)$ one can use the haversine formula:

$$2 \arcsin \left( \sqrt{\sin^2 \left( \frac{\phi_1 - \phi_2}{2} \right) + \cos(\lambda_1) \cos(\lambda_2) \sin^2 \left( \frac{\lambda_1 - \lambda_2}{2} \right)} \right).$$

(2.23)

However, compared to taking a straightforward Euclidean norm between two $\langle x, y \rangle$ vectors, this is a slow computation. Thus to speed up

http://en.wikipedia.org/wiki/Automatic_identification_system
most computations in the rest of this thesis, we project latitude, longitude coordinates \((\phi, \lambda)\) to a 2-dimensional \(x, y\)-plane using a simple sinusoidal projection:

\[
x = (\lambda - \lambda_0) \cos(\phi) , \quad (2.24)
\]
\[
y = \phi - \phi_0 . \quad (2.25)
\]

where \((\phi_0, \lambda_0)\) is the center of our dataset. Using this sinusoidal projection we get moving object trajectories as in Definition 2.2.1. More complicated projections are also possible, but this projection is sufficient for the size of the areas that our datasets are concerned with. For our datasets, the sinusoidal projection leads to differences of less than 0.5% in distance computations at the edge of the area. For analysis on datasets that cover larger areas, or areas closer to the poles, this simple projection might lead to problems and more sophisticated projections can be necessary.

Using the two-stage compression algorithm described in Chapter 3 we cut the trajectories at the stops for each of the datasets that we describe below. Thus the trajectories that we consider are moves in terms of the stop-move model by Spaccapietra et al. (2008). Moreover, we do not allow appearing or disappearing trajectories. This means that we only keep trajectories that start from a stop, or enter our area of observation and that either stop somewhere, or exit our area of observation. For all trajectories \(t_1\) is set to 0, thus we consider only the relative time of trajectories.

### 2.3.2 Clustering

For our clustering experiments we created a dataset of a 25km radius consisting of 714 vessel trajectories around the Texel island in the Netherlands. These trajectories are partitioned into 8 different clusters, creating a gold standard \(G = g_1, \ldots, g_8\). The clusters are very different in size, ranging from 8 to 348 trajectories. The average length of a trajectory is a sequence of 300 \((x, y, t)\) vectors.

We illustrate the clustering in Figure 2. The general direction of movement for each of the 8 clusters is indicated with an arrow. These clusters are unlabeled, but they can be given descriptive labels, such as “leaving port to the north” for 3 and “from Den Helder to Texel” for 8. Together, these 8 clusters give a summary, in terms of the different types of movements/behavior, of what goes on around Texel island.

### 2.3.3 Classification

To do classification experiments we use vessel trajectories in a 50km radius around the Port of Rotterdam. This set consists of 2219 trajectories. The AIS system provides a number of different vessel types.
Figure 2: An overview of the clustering vessel trajectory dataset with the 8 different clusters of the gold standard indicated with an arrow giving the general direction. A solid dot indicates the start of a trajectory and an asterisk indicates the end.

The type distribution in our set is very skewed. For our classification experiments we use only the four most common types: cargo ship, tanker, tug and law-enforcement vessel. The average sequence length for this set is 689 elements.

The trajectories for the four types are illustrated in Figure 3. From the figure we see that cargo ships and tankers have similar trajectories, and law enforcement vessels and tugs as well. Thus, separating cargo ships from tankers and tugs from law enforcement will be the hardest classification task. More useful classification tasks than predicting the type of vessel can be thought of, since the vessel type is already available via AIS. However, predicting the vessel type is still useful to check whether the vessel type field is filled in correctly, and hence determines the trustworthiness of the vessel. Moreover, predicting the vessel type is a good example classification task, precisely because the labels are readily available via AIS, whereas other labels are harder to come by.
2.3.4 Outlier Detection

For outlier detection we selected only the cargo ships and tankers from the classification dataset described above. In this set 39 trajectories were labeled as outliers because they show strange behavior compared to the 747 trajectories that were labeled as regular. For this set, the average sequence length is 918 vectors.

The outliers are plotted against the regular trajectories in Figure 4. From the figure we see that trajectories can be outliers for different reasons: vessels can sail to an irregular place, have extra turns, or stop unexpected, etc.
Figure 3: An overview of the classification vessel trajectory dataset. A solid dot indicates the start of a trajectory and an asterisk indicates the end.

Figure 4: An overview of the outlier detection vessel trajectory dataset. A solid dot indicates the start of a trajectory and an asterisk indicates the end.
This chapter extends work by Willems et al. (2010); van Hage et al. (2011, 2009). The ferry example was first discussed by de Vries et al. (2008).

The volume of moving object trajectory data is huge and often quite regular, thus it makes sense to apply a trajectory compression technique to make the data more manageable. In doing analysis of moving objects, the concepts of stop and move are essential movement primitives. We show in this chapter that standard trajectory compression techniques, based on line-simplification, are not ideally suited to retain the stop and move information. Therefore we propose a simple extension to this method that looks at speed along a trajectory first. We evaluate standard trajectory compression and our technique on the task of stop retention using a dataset of vessel trajectories. The conclusion from this evaluation is that our method generally has better stop retention at the same compression rate. Furthermore, we show how we can use this method to create the semantic building blocks for higher level reasoning about vessel behavior.

3.1 INTRODUCTION

Moving object trajectories are often sampled every couple of seconds, which results in a large amount of data. However, in a number of cases, such as cars following a road, or ships in a sea-lane, the data is highly regular. In such cases it makes sense to make the data more manageable without losing much information by using a trajectory compression technique. For instance, large scale data reduction makes computation of similarity between trajectories much faster, as we see in Chapter 4, and it also allows for higher level reasoning, of which we give examples in this chapter.

As recognized by Spaccapietra et al. (2008), the concepts of stop and move are essential semantic movement primitives for the analysis of moving objects. Thus, when using a trajectory compression method we should not compress movement data in such a way that information about these concepts is lost. In other words, trajectory compression should retain stops and moves. We dub this problem stop retention.

For our vessel trajectory data we consider a standard trajectory compression technique based on line-simplification (Gudmundsson et al. (2009)), which we call Piecewise Linear Segmentation (PLS). We will show the performance of this method at stop retention for a number of different error measures and conclude that this stan-
standard method is not ideal. Therefore, we propose a modification of PLS-based trajectory compression that explicitly looks at the speed, i.e. the partial derivative to time of the trajectory, of a moving object first. This modification allows for a better retention of the stop and move concepts at the same compression rate. Stop retention also has the advantage that it allows us to define the concept of segments on compressed trajectory data. These segments are used as atomic movement events that allow for reasoning to derive more complex behavior patterns, which we illustrate with an example.

The goal of our method is not the task of stop detection on its own, for which there are methods such as SMoT (Alvares et al. (2007)), CB-SMoT (Palma et al. (2008)) and DB-SMoT (Rocha et al. (2010)). Our method aims at retaining stops when applying trajectory compression.

The rest of this chapter is structured as follows. In Section 3.2 we introduce the standard piecewise linear segmentation compression technique and introduce our own extension that looks at speed first. We evaluate the standard technique and our extension with respect to stop retention in Section 3.3. Section 3.4 introduces the segment representation that we define upon compressed trajectories and gives examples of reasoning towards detecting complex vessel behavior with these segments. We end with some conclusions in Section 3.5.

3.2 Trajectory Compression

In this section we first review some trajectory related definitions. Then we describe piecewise linear segmentation with a number of different error measures. We give a problematic example for this standard algorithm and propose a solution in terms of an extension. Finally, we say something about query error-bounds and computational complexity.

3.2.1 Trajectory Definitions

For a definition of a trajectory we refer to Definition 2.2.1 in Chapter 2. In this chapter we also explicitly consider the speed of a moving object \( v \). Therefore let \( T_v = (x_1, y_1, v_1, t_1), \ldots, (x_n, y_n, v_n, t_n) \) be a trajectory for which we also have the speed \( v_i \) for the moving object at time \( t_i \). This speed can either be reconstructed from \( T \), or derived from a special sensor.

Spaccapietra et al. (2008) argue that in analyzing the behavior of moving objects the concepts of stop and move are essential. What are stops and moves is application dependent. The part of a trajectory that constitutes a stop is the non-empty time interval for which the traveling object does not move from the application's perspective. A
move is a part of a trajectory that is not a stop. We see in Section 3.3.1 what are considered stops and moves in our application.

3.2.2 Piecewise Linear Segmentation

In the field of moving object databases different techniques have been studied to compress trajectory data (Meratnia and de By (2004); Cao et al. (2006); Potamias et al. (2006); Frentzos and Theodoridis (2007); Gudmundsson et al. (2009); Ni and Ravishankar (2007)). The most common one is a line-simplification method, adapted to trajectory data, known as the Douglas-Peucker algorithm (Douglas and Peucker (1973)) in cartography, and Ramer’s algorithm (Ramer (1972)) in image processing. In the world of data-mining from time-series it yet again goes by other names (Keogh et al. (2001)), we shall refer to it, rather generically, as Piecewise Linear Segmentation (PLS). This method is intuitive, easy to implement, relatively fast (compared to optimal methods (Cao et al. (2006))) and gives good results.

The PLS-algorithm compresses a trajectory T into linear segments by recursively keeping the points that have maximum error higher than a fixed threshold \( \epsilon \). The pseudo code for this algorithm is given in Algorithm 3.2.1. It works in the following way. The first and last points, \( T(1) = (x_1, y_1, v_1, t_1) \) and \( T(n) = (x_n, y_n, v_n, t_n) \), of a trajectory \( T \) of length \( n \) are selected, and for all intermediate points we compute the error, using the function \( \Phi \), with respect to these points. If the maximum of these errors is greater than a certain threshold, then we apply the procedure again, with the corresponding point \( T(i) \) both as start and end-point, i.e. we recursively apply the procedure to the trajectory from \( T(1) \) to \( T(i) \) and \( T(i) \) to \( T(n) \). When there is no error greater than the given threshold, then we just keep the points \( (x_1, y_1, v_1, t_1) \) and \( (x_n, y_n, v_n, t_n) \). Thus, the goal of the algorithm is to reduce the number of points in a trajectory while keeping the maximum deviation, or error, from the original trajectory within the threshold \( \epsilon \).

For trajectories, \( \Phi \) can be a number of different error functions \( E \) (Algorithm 3.2.1, line 5), which we will introduce below. We define all these error functions in terms of the line-segment model, because the line model can lead to some counter-intuitive results (Gudmundsson et al. (2009)). The line-segment model considers the line-segment between two points when computing the error according to some \( E \), whereas the line model considers the line through two points, which is infinite in length. Thus, a point can be very far away from a line-segment between two points, but very close to, or even lie on, the infinite line through those points.

A number of authors have proposed error functions \( E \) to use in the above algorithm. Most recently, Gudmundsson et al. (2009) proposed...
Algorithm 3.2.1 pls_{\mu}(T, \epsilon)

1. We use end to indicate the index of the last element of a trajectory.

2. \(d_{\text{max}} = 0\)
3. \(i_{\text{max}} = 0\)
4. \(\text{for } i = 2 \text{ to } \text{end} - 1 \text{ do}\)
5. \(d = \Phi(T(i), T(1), T(\text{end}))\)
6. \(\text{if } d > d_{\text{max}} \text{ then}\)
7. \(i_{\text{max}} = i\)
8. \(d_{\text{max}} = d\)
9. \(\text{end}\)
10. \(\text{end}\)
11. \(\text{if } d_{\text{max}} \geq \epsilon \text{ then}\)
12. \(A = \text{pls}(T(1, i_{\text{max}}), \epsilon)\)
13. \(B = \text{pls}(T(i_{\text{max}}, \text{end}), \epsilon)\)
14. \(T_{C} = A, B(2, \text{end})\)
15. \(\text{else}\)
16. \(T_{C} = T(1), T(\text{end})\)
17. \(\text{end}\)
18. \(\text{return } T_{C}\)

an error function \(E_{\mu}\) that generalizes most of the earlier defined error functions, so we consider this one first.

Definition 3.2.1.

\[E_{\mu}(\langle x_i, y_i, t_i \rangle, \langle x_1, y_1, t_1 \rangle, \langle x_n, y_n, t_n \rangle) = \| \langle x_i, y_i, \mu t_i \rangle - \langle x_i', y_i', \mu t_i' \rangle \|,\]

where \(\langle x_i', y_i', t_i' \rangle\) is the closest point to \(\langle x_i, y_i, t_i \rangle\) on the line-segment \(\langle x_1, y_1, t_1 \rangle, \langle x_n, y_n, t_n \rangle\).

The parameter \(\mu\) determines the ratio between the space and time dimensions. Different settings of \(\mu\) lead to different previously defined error functions, which we illustrate in Figure 5.

In terms of \(E_{\mu}\), the original error measure \(E_2\) of the Douglas-Peucker algorithm is defined as:

Definition 3.2.2.

\[E_2 = E_{\mu=0}.\]

With this function, a trajectory is treated as a line in 2-dimensional space, ignoring the time dimension. \(E_2\) is illustrated in Figure 5.

Time can also be treated as just another dimension, i.e. a trajectory is a line in a 3-dimensional space. This leads to the standard error function \(E_3\), also shown in Figure 5.

Definition 3.2.3.

\[E_3 = E_{\mu=1}.\]
Figure 5: Examples of different error functions for the point $\langle 4, 2, 1 \rangle$ and the line segment $\langle 1, 1, 0 \rangle, \langle 5, 5, 4 \rangle$. Based on a similar figure by Cao et al. (2006).

Also very common is $E_u$ (Cao et al. (2006); Meratnia and de By (2004)), where the time dimension is treated differently, because we take the difference between the point and its linear interpolation based on time on the line-segment, see Figure 5.

**Definition 3.2.4.**

$$E_u = E_{µ=∞}.$$ 

Cao et al. (2006) also define the error measure $E_t$, illustrated in Figure 5, which is more or less the dual of $E_u$ and not definable in terms of $E_µ$. Instead of determining the spatial difference at the same time, we determine the temporal difference at the same place, or closest place. Compression using just this error measure is incomplete, because errors in the spatial dimension are ignored. Contrast this with $E_2$, which ignores the temporal dimension completely.

**Definition 3.2.5.**

$$E_t(⟨x_i, y_i, t_i⟩, ⟨x_1, y_1, t_1⟩, ⟨x_n, y_n, t_n⟩) = \sqrt{(t_i - t'_i)^2},$$

where $t'_i$ is the time of the point $⟨x'_i, y'_i⟩$ on the 2-dimensional projection of $⟨x_1, y_1, t_1⟩, ⟨x_n, y_n, t_n⟩$ on the xy-plane that is closest to the 2-dimensional projection of $⟨x_i, y_i, t_i⟩$ on the xy-plane.
Therefore, Cao et al. (2006) define an error measure that combines $E_u$ and $E_t$. This is done by simply taking the maximum of the two error functions. However, it is not entirely clear how the difference in scale between the spatial ($E_u$) and temporal ($E_t$) error should be treated. In our implementation we divide the error by the error threshold $\epsilon$ for $E_u$ and $E_t$ separately, and take the maximum of those two values.

**Definition 3.2.6.**

$$E_u \land E_t = \max(E_u, E_t)$$

We will use a subscript to indicate which error function is used with PLS, for example $\text{pls}_{E_u}$.

### 3.2.3 A Problematic Example

Consider the example of the application of PLS with $E_u$ in Figure 6 (a) and (b). This figure illustrates the compression of a trajectory consisting of 5 points, $p_1, \ldots, p_5$. Figure 6 (a) shows the initial state of the compression, where the trajectory is approximated by the line-segment between $p_1$ and $p_5$. In this situation $p_4$ has the greatest distance to this segment. This distance is over the threshold $\epsilon$, thus we retain point $p_4$, and we get Figure 6 (b). The stop in the trajectory between $p_2$ and $p_3$ is not retained in the compression, because the distances to the simplification $p_1, p_4$ are not large enough and hence not over $\epsilon$. The result is that in the compressed trajectory it seems as if the object has moved (slowly) between $p_1$ and $p_4$, instead of two faster moves with a stop in between. Should the stop between $p_2$ and $p_3$ be of a longer duration then eventually also the points $p_2$ and $p_3$ should be retained. However, if the distance between $p_1$ and $p_2$, and $p_3$ and $p_4$ is below $\epsilon$ then this will never happen, no matter how long
the stop takes. Using $E_2$, $E_3$ or $E_\mu$ would not change things, since they are even less strict than $E_u$ (Cao et al. (2006)).

The stop in the example would be retained if we used $E_t$ or $E_u \wedge E_t$. However, we will see in the evaluation section that this comes at the cost of quite a bit lower compression rates.

### 3.2.4 Two Stage Piecewise Linear Segmentation

To deal with the problem presented above we extend the standard algorithm. The intuition behind this extension is that the stopping and moving behavior of moving objects is more apparent in the derivative, i.e. the speed time-series, of the trajectory than in the trajectory itself. Therefore we propose a simple extension to the earlier defined trajectory compression, which we give in Algorithm 3.2.2. The idea is that first piecewise linear segmentation is applied to the speed time-series of the trajectory, using essentially a one-dimensional variant of the $E_u$ error measure. We call this error measure $E_v$ and define it below for trajectories $T_v$, i.e. those that include speed.

**Definition 3.2.7.**

$$E_v((x_i, y_i, v_i, t_i), (x_1, y_1, v_1, t_1), (x_n, y_n, v_n, t_n)) = \sqrt{(v_i - v'_i)^2},$$

where $(x'_i, y'_i, v'_i, t_i)$ is the point on the line-segment $(x_1, y_1, v_1, t_1)$, $(x_n, y_n, v_n, t_n)$ with time $t_i$.

Essentially we take the difference between the actual speed $v_i$ and the linearly interpolated speed $v'_i$.

To the resulting subtrajectories, which are created in this speed-only segmentation step, we apply regular PLS with an error measure of our choosing: $E_2$, $E_3$, $E_u$ or $E_\mu$. Should we skip the second step, or alternatively just set $\epsilon_p = \infty$, then we have trajectory compression purely based on speed. This is incomplete in the same way as compression with just $E_t$ is. As we did for PLS we indicate with a subscript which error measure is used in the second step, e.g. $2\text{stage-pls}_{E_2}$.

Figure 6 (c) is the speed time-series corresponding to the trajectory from (a) and (b). The graph illustrates that when using compression on the speed time-series, the point with the greatest distance to the line $p_1$, $p_5$ is now $p_2$ and not $p_4$. In other words, if we look at the speed, the stop is now the most important point to keep.

Since speed is derived from the trajectory, the compression in terms of speed that we do here is not entirely accurate. Leaving out a point from a trajectory (because of compression) will change the total displacement of a moving object. This means that, to be more precise, we should compare the speed at a certain point and time to the estimation of the speed at the same time, but with the point left out of the trajectory. This is different from how it is done above, because
we compare the speed at a certain time to a linear interpolation of the speed at that time. However, this linear interpolation is based on start and end speeds that are derived with the point that we are comparing to still part of the trajectory.\footnote{We have also experimented with reestimating the speed, but found this not to make any significant differences in our current experiments. However, it does increase computation time significantly.}

For higher level reasoning that we will illustrate in Section \ref{sec:reasoning}, we prefer to combine \texttt{2stage-pls} with $E_2$, because this allows us to attach the most interesting semantics to the trajectory. When doing compression with $E_3$, $E_\mu$ and $E_\mu$ using regular PLS or using \texttt{2stage-pls} we have to assume constant speed between points. However, with \texttt{2stage-pls}_{E_2} we can assume constant acceleration between points. This means that apart from separating stops and moves we can also classify moves in a trajectory as being accelerating or decelerating. Furthermore, there is a clear separation between the compression in the temporal dimension (using speed) and the spatial dimension (using $E_2$). Thus, we know whether points are kept because they are (temporal) speed change events or because they are (spatial) course change events. However, if we use $E_2$, then the interpolation of a position for a timestamp between two points that are kept is slightly more complex, since we do not assume constant speed, but constant acceleration.

\subsection{Error bounds and Complexity}

An important point in research into trajectory compression techniques are the error bounds that different error measures give on a number of standard moving object database queries. Regarding these bounds, Gudmundsson et al. (2009) generalize the work of Cao et al. (2006). They show the error bounds for their $E_\mu$ measure for the \texttt{Where-At} and \texttt{When-At} queries and how these are dependent on $\mu$. It is easy to see that if we use \texttt{2stage-pls}_{E_\mu}, we have the same error bounds,
because for every subtrajectory created by the speed compression, these error bounds hold since these subtrajectories are compressed using $E_\mu$.

The error bounds for $E_\mu$ vanish when $\mu = 0$ and consequently the bounds for $2\text{stage-pls}_{E_2}$ are not good. The error for the first speed step is bounded. However, an error in speed leads to a quadratic error in position. Position is not further bounded, since $E_2 = E_{\mu=0}$.

The worst case running time of standard piecewise linear segmentation is $O(n^2)$. The $2\text{stage-pls}$ algorithm is in the same order of complexity, because it is essentially two applications of PLS. The first (speed) step is just a regular full recursion of PLS and we can consider the second step as if we are already at a certain level of the recursion of PLS.

3.3 Evaluation

Typically the performance of a trajectory compression algorithm is judged by the reconstruction error introduced when recreating the original trajectory from the compressed trajectory. One way to do this is to prove query error bounds (Cao et al. (2006); Gudmundsson et al. (2009)). An empirical option is to compute the distance between the compressed trajectory and the original trajectory for each point in time (Meratnia and de By (2004)). As we mentioned in the previous section, our method has the same error bounds as regular PLS with $E_\mu$, so we do not consider reconstruction error in our evaluation.

The goal of our two-stage method is to be better at retaining stop information than the traditional error measures while still providing good compression rates. Or to put it differently, our method should provide better stop retention at the same compression rate. This is what our evaluation below focuses on. To show this, we use a dataset from our set of AIS vessel trajectories, described in more detail in Chapter 2. AIS contains speed information about the vessel. However, this is sometimes based on a different sensor, and furthermore is not always available in other applications. To show the more general applicability of the above method we have also reconstructed speed, based on the position information alone. To reduce the influence of noise in the AIS data, caused by for instance GPS, we apply some standard moving average smoothing\(^2\) to the trajectory and also to the recomputed speed.

3.3.1 Stop Retention

As was mentioned earlier, the definition of a stop is application dependent. In this application we consider a ship to be stopped whenever its movement speed is (close to) zero for some (short) amount

\(^2\)Simple moving average, such as the standard MatLab smooth function.
of time. This means that a ship can stop in port but also at sea or
in an anchoring area. With this in mind, we selected trajectories from
our dataset that appeared to have (a lot of) stops in them. We plotted
these trajectories on a map and hand-labeled the stops in the trajec-
tory, indicating them by a start and end time. This stop dataset, for
instance, contains a ferry between two ports and a cargo ship docking
in a harbor.

Next, we need to define when a compressed trajectory has suc-
cessfully retained a stop, which we do as follows. If a compressed
trajectory $T_C$ contains a segment $T_C(i, i+1)$, such that

$$\text{dist}(\langle x_i, y_i \rangle, \langle x_{i+1}, y_{i+1} \rangle) /
(t_{i+1} - t_i) \leq \theta,$$

then that segment is a stop. In other words, a vessel is stopped if
the average speed in a segment is below a certain threshold. If the
temporal interval $t_i, t_{i+1}$ overlaps with the temporal interval of a
hand-labeled stop (i.e. they have at least one time-point in common),
then we say that the compression has retained that stop. The stop-
threshold $\theta$ should at least be as high as the maximum speed among
the hand-labeled stops. Note that a stop can only be retained once,
e.g. if there are two segments below the threshold that overlap with
one labeled stop, then this is counted as one. This also holds the other
way around, if there are two hand-labeled stops that overlap with one
segment, then we also count this as one. Furthermore, we define the
stop retention ratio as the total number of retained stops divided by
the total number of hand-labeled stops. Apart from stops that are not
retained, the opposite can also occur. The average speed in a segment
can be below the stop threshold, but it is not labeled a stop. We do
not consider this option in the stop retention ratio definition, since it
did not occur in our experiments.

What a good stop threshold is in our application is not immedi-
ately obvious, therefore we consider stop retention at three different
stop thresholds. Because the data is noisy and hand-labeling is far
from perfect, the number of hand-labeled stops that fall below these
thresholds differs slightly. At stop threshold $\theta = 0.1$ knots we have
291 stops, at $\theta = 0.05$ knots we have 288 and at $\theta = 0.025$ knots there
are 281 stops. Thus, for each threshold the total number of stops that
should maximally be retained differs.

3.3.2 Experimental Set-Up

We are interested in increasing stop retention while maintaining com-
pression rate. Thus, for the two algorithms, the different error mea-
ures and a range of parameter settings, we compute the stop re-
tention for the hand-labeled stop dataset. Clearly, this set is biased
towards trajectories with stops. To get a more general compression
result for our data we compute the compression rate for the different settings on a regular unbiased dataset containing 600 randomly selected trajectories.

As baseline methods we will look at the regular pls algorithm with the error measures: $E_2$, $E_3$, $E_u$, $E_{\mu=\frac{1}{2}}$, $E_{\mu=2}$, $E_t$ and $E_u \wedge E_t$, essentially what is defined in Section 3.2.2. We recall that the settings $E_{\mu=0,1,\infty}$ are equal to $E_2$, $E_3$ and $E_u$, respectively. $E_{\mu=\frac{1}{2}}$ is in between $E_2$ and $E_3$, and $E_{\mu=2}$ is in between $E_3$ and $E_u$.

Before any of these measures can be applied, the time dimension needs to be scaled to the spatial dimensions. The intuition between the measure $E_{3}/E_{\mu=1}$ is that differences in the temporal and spatial dimensions are equal. Thus, we have done the scaling of the temporal dimension such that the average temporal difference between samples, in the original scale of the temporal dimension, equals the average difference in position, under the $E_3 / E_{\mu=1}$ setting.

All of the above baselines are compared to our two-stage method 2stage-pls. We combine 2stage-pls with the error measure $E_2 / E_{\mu=0}$, which we like for semantic reasons, and also with the $E_{\mu}$ setting that scores the best among our baselines. Furthermore, we consider the situation where we take the speed information directly from AIS and the scenario where we recompute the speed.

Finally, we consider other intuitive variants that use two error measures, like 2stage-pls. The first variant is using the $E_t$ error measure as the first step in 2stage-pls, because $E_t$, like $E_v$, is a candidate for being a good stop detector. We test this variant with $E_2$ and the same $E_{\mu}$ setting as above. We also create two variations on pls with $E_u \wedge E_t$: $E_2 \wedge E_v$ and $E_{\mu} \wedge E_v$. As the last alternative we change 2stage-pls so that we reverse the two compression steps, i.e. we do compression with $E_2$, respectively $E_{\mu}$, first, and do the compression with $E_v$ in the second step.

For all experiments, the $\epsilon$ settings that we range over are the same. For the error measures $E_2$, $E_3$, $E_u$ and $E_{\mu}$, $\epsilon$ ranges from 0.01 to 0.08 kilometers with increments of 0.01. For $E_t$, $\epsilon$ ranges from 90 to 720 seconds, with increments of 90. Finally, for $E_v$, $\epsilon_v$ ranges from 0.5 to 4 knots, with increments of 0.5.

### 3.3.3 Results

Because of the large number of different settings that we tested, we only present the most important results of the experiments in the following section. For a complete overview of all the results we refer the reader to Appendix A.

From the baselines we only show the results for the $E_{\mu=2}$ setting, the best performing setting in terms of stop retention versus compres-

---

3 Typically the time is in seconds, and the spatial dimension in arc lengths or kilometers.
Figure 7: Results of $\text{zstage-pls}_{E_2}$ under different experimental settings. Results for $\text{pl}_{E_{\mu-2}}$ are included for reference.
Figure 8: Results of 2stage-pls$_{E_{\mu=2}}$ under different experimental settings. Results for pls$_{E_{\mu=2}}$ are included for reference.
sion rate. Furthermore we give the results for $\textit{2stage-pls}$ with $E_2$ and with $E_{\mu=2}$. The other variants using two error measures are outperformed by these two $\textit{2stage-pls}$ variants and we therefore do not give the results, but instead refer to Appendix A.

Figures 7 and 8 shows results for $\textit{2stage-pls}$ with $E_2$, and $\textit{2stage-pls}$ with $E_{\mu=2}$ respectively, for a number of settings. Each figure also contains the result for regular $\textit{pls}$ with $E_{\mu=2}$ for reference. The graphs plot stop retention on the vertical axis versus compression rate on the horizontal axis. Each graph shows results for one setting of the stop threshold $\theta$ (vertically) and recomputed speed (left) or AIS speed (right). Ideally, we have good stop retention with high compression rate, which we find in the top right corner of the graphs. Each line in the graph is created by running the algorithm and error measure combination for the range of $\epsilon$ that we defined above. However, $\textit{2stage-pls}$ has two parameters, thus there are 8 lines, representing the different settings of $\epsilon_v$. For each figure it holds that the highest line in the graph is for $\epsilon_v = 0.5$, the lowest line for $\epsilon_v = 4$ and the rest of the settings are in between.

3.3.4 Discussion

We first consider the performance of $\textit{2stage-pls}_{E_{\mu=2}}$. From Figure 8 we see that this variant of $\textit{2stage-pls}$ outperforms regular $\textit{pls}_{E_{\mu=2}}$ under almost all parameter settings. Thus, what we do with two stage PLS, adding an initial speed compression step to the best performing baseline, is a good idea. If we do this, we can retain more stops at the same compression rate.

The results in Figure 7 for our semantically motivated method $\textit{2stage-pls}_{E_2}$ show that we do not outperform our best baseline for all parameter settings. However, because $\textit{2stage-pls}$ has two parameters we can tune our method as to how well we want to detect stops and how much spatial error we allow.

What is also interesting is that the performance of both $\textit{2stage-pls}_{E_{\mu=2}}$ and $\textit{2stage-pls}_{E_2}$ does not degrade much when we lower the stop threshold. On the other hand our baseline $E_{\mu=2}$ does perform quite a bit worse with a stricter stop threshold. This suggest that the added speed compression step of $\textit{2stage-pls}$ adds robustness when it comes to stop retention.

Furthermore, stop retention is better using the speed coming from the AIS sensor. One reason for this seems to be that recomputing the speed from the trajectory is not trivial business, since the GPS used by AIS creates quite some noise, which is amplified in the speed recomputation. This is something to consider for applications where the speed of the moving object is not directly available.

Though we have introduced a rather general definition of a stop there are also other possibilities. Another definition of a stop might
make a difference in performance. For instance, one could define a stop as a point, instead of a segment of a trajectory, for which the speed has gone below the stop threshold. Such a definition might be more favorable to our approach. However, we have not investigated this.

Finally, we note that using piecewise linear segmentation leads to a large data reduction with compression rates over 90% for the realistic $\epsilon$ settings that we used.

### 3.4 Deriving Complex Behavior

In this section we show how we create atomic movement events that we call *segments* using the compression method described above. These segments are represented in the Simple Event Model (SEM) (van Hage et al. (2011)). Additional concepts are added from external sources and via reasoning, and are also represented in SEM. Using these extra concepts we can derive more abstract, and complex vessel behavior, which we will illustrate with an example. In this section we first briefly describe the semantic web and then the simple event model; for more details on this we refer to the papers (Willems et al. (2010); van Hage et al. (2011)). Secondly we discuss how the simple event model is combined with the segments.

#### 3.4.1 Semantic Web

In 2001 the concept of the Semantic Web was introduced (Berners-Lee et al. (2001)). The aim of the semantic web is to create a network of interlinked data that is readable for machines, as opposed to the old world wide web, which is understandable for humans. This development has brought new standards such as the Resource Description Framework (RDF) and the Web Ontology Language (OWL).

RDF is the foundation for knowledge representation, i.e. ontologies, on the semantic web and is based on the idea of making statements about resources in a subject-predicate-object form. Such expressions are dubbed *triples*. The RDF specification defines a number of classes, for the subjects and objects, and properties, for the predicates. Moreover, users can, and should add their own classes and objects. For example, suppose that we have an ontology about fruits, called fruit. Then the RDF triple `fruit:Pear rdfs:subClassOf fruit:Fruit` expresses the fact the the class of Pear is a sub-class of the class of Fruit. And the triple `fruit:elstar rdf:type fruit:Apple` expresses the fact that an elstar is an instance of the class Apple. The colon notation is used to indicate from which ontology a class or property is used, i.e. the class Pear comes from the example fruit ontology that we created ourselves, but the property `type` is defined by the RDF standard. These notations are shorthands for full-fledged Universal Resource Identifiers (URI) that
uniquely identify the specific ontology used and where to find it, thereby forming the backbone of the semantic web. The triple format of RDF is essentially a labeled directed multi-graph and pieces of RDF are often represented as graphs in figures for better readability.

A lot more on RDF and the semantic web can be found online.4

3.4.2 Simple Event Model

The simple event model (SEM) (van Hage et al. (2011)) was designed to represent events, in the broad sense of the word, derived from various sources, such as, the web, sensor data, etc. SEM has four core classes: sem:Event (what happens), sem:Place (where did it happen), sem:Actor (who or what participated), and sem:Time (when did it happen). Each core class has an associated sem:Type that is used to indicate the type of a core individual. For instance, the core class sem:Place has the sem:PlaceType class associated to it.

SEM has three kinds of properties: sem:eventProperties, sem:type properties, and a few miscellaneous properties like the sem:hasTimeStamp’s subproperties. The sem:eventProperties relate sem:Events to other individuals. They also contain the sem:hasSubEvent property to aggregate events. A sem:type relates individuals of the core classes to individuals of sem:Type. There are a number of sem:hasTimeStamp properties, among them: sem:hasBeginTimeStamp and sem:hasEndTimeStamp for time intervals.

3.4.3 Segments

We use 2stage-pls to create segments that are represented using SEM. Let T_{v,c} be a trajectory as defined earlier, which next to speed v also includes course c. We compress this trajectory into

\[ T_C = 2\text{stage-pls}_{E_2}(T_{v,c}, \epsilon_v, \epsilon_p) , \]

where \( \epsilon_v = 2.5 \) knots and \( \epsilon_p = 50 \) m are chosen such that we have high compression, but also good stop retention, according to Figure 7. For all \( i \) such that \( T_C(i) = (x_i, y_i, v_i, c_i, t_i) \) and \( T_C(i + 1) = (x_{i+1}, y_{i+1}, v_{i+1}, c_{i+1}, t_{i+1}) \) we create a segment. A segment is an instance of sem:Event. It furthermore has the additional sem:eventType etype_stopped or etype_moving, indicating whether the segment is a stop or a move, determined as described in Section 3.3.1 with a stop threshold \( \theta = 0.1 \). Event information about the speed along the segment, i.e. whether it is constant, increasing or decreasing can also similarly be added as an event type. The coordinates \( x_i, y_i \) and \( x_{i+1}, y_{i+1} \), converted back to latitude and longitude, are represented as instances of sem:Place which are connected to the segment event by the prop-

4 http://www.w3.org/standards/semanticweb/
properties `seg:hasBeginPlace` and `seg:hasEndPlace`, subproperties of `sem:hasPlace`. The segment also has the properties `seg:hasBeginTimestamp`, `seg:hasEndTimestamp`, `sem:hasBeginSpeedOverGround`, `seg:hasBeginCourseOverGround`, `seg:hasEndSpeedOverGround`, and `sem:hasEndCourseOverGround`, which take the values $t_i$ and $t_{i+1}$, $v_i$ and $v_{i+1}$, and $c_i$ and $c_{i+1}$, respectively. In Figure 9 the RDF representation of a segment is illustrated by the white boxes.

Under the current compression settings we have a compression of around 98%, thus a data reduction by a factor of 50. This data reduction is important to make the reasoning that we will see in the rest of the section feasible.

### 3.4.4 Additional Semantics

In order to derive more complex behavior patterns extra semantics for the segments are needed, such as the type of place a vessel goes to, and the type of the vessel itself.

**Places** By using spatial proximity, the latitude and longitude coordinates of the `sem:Places` are connected to concepts in the GeoNames ontology. The GeoNames ontology associates instances of places with geo-coordinates to GeoNames feature codes like `geo:H.HBR` (harbor), and `geo:H.ANCH` (anchorage). If such a GeoNames concept is close, we add a `sem:Place` instance for this concept and relate it to a segment using a `seg:hasBeginPlace` or `seg:hasEndPlace` property. This allows us, for instance, to add the additional `sem:EventType type_stopped_in_harbor` for segments that are of type `type_stopped` and for which the related place has the type `geo:H.HBR`. The addition of this type of information to the segments is illustrated in Figure 9 by the red (C) boxes.

**Actors** Each segment belongs to a single vessel. This vessel is identified as the `sem:Actor` of the segment, related by the `sem:hasActor` property. The vessel has an `ais:mmsi` property to the value of its Maritime Mobile Service Identity (MMSI) number. Using this MMSI number additional information about the vessel is pulled from different web sources and new properties, like `ais:length` and `ais:callsign`, of the vessel are introduced. Also type information is added, which is aligned to WordNet. For instance, the type “passenger vessel” would be translated to `atype_passenger_vessel`, which is aligned to `wordnet:synset-passenger_ship-noun-1`. Extra actor information is shown in Figure 9 by the yellow (D) boxes.

**Deriving Complex Events** With the extra semantic information in place we can derive more complex behavior than the simple

---

5 http://www.geonames.org
6 http://www.wordnet.org
Figure 9: An RDF-graph example of part of a ferry trip. Green (A) and purple (B) boxes are SEM core classes. The white boxes are concepts from the basic segments, the red (C) boxes illustrate added place semantics, the yellow (D) boxes are added actor semantics, and the blue (E) boxes are concepts inferred using rules.
etypere_stopped and etype_moving events. This is done using rules. For instance, to derive the complex behavior “trip” we use a rule that is based on the assumption that if we do not know that an explicit stop between consecutive moving events exists, then it does not exist. This closed world assumption allows us to deal with missing ship observations, which occur frequently. We conclude with this rule, that if we do not know about a stop at a harbor between two stops at harbors $a$ and $b$, then there was a trip between harbors $a$ and $b$. This trip is added as a new event, which has \texttt{sem:hasSubEvent} property relations to the segments that compose the trip. The harbors of departure and arrival become \texttt{sem:hasBeginPlace} and \texttt{sem:hasEndPlace} properties of the trip. Subsequently we construct a rule that defines a “ferry trip” as a trip back and forth between two different harbors, allowing us to recognize ferries. All the information inferred using rules is illustrated in Figure 9 by the blue (E) boxes.

An important advantage of having different levels of abstraction of events is that for the higher level events it does not matter where the subevents were derived from. For instance, one subevent \texttt{stopped_at_harbor} of the trip event could be derived as described above, whereas another could be derived from another source, like the ferry schedule on the web. This would still allow the derivation of the ferry trip event.

3.5 Conclusions & Future Work

We presented a simple extension to standard trajectory compression based on Piecewise Linear Segmentation (PLS). The idea of our compression method is to add an initial compression step based on the speed of the moving object. Afterwards we apply compression with a spatial error measure of our choosing. The motivation for introducing this method is that standard error measures and PLS are not ideally suited for stop retention.

For most parameter settings, this two-stage method allows us to retain more stops at the same (high) compression rate than regular piecewise linear segmentation using any of the error measures. Furthermore, the two-stage method allows for better tuning of the retained stops versus compression rate trade-off. The method is also more robust in terms of stop retention, using stricter stop thresholds decreases stop retention only slightly whereas this is not the case with regular PLS.

Furthermore, we saw examples of how we can use compression to create atomic movement events in terms of the Simple Event Model (SEM). The representation in terms of SEM allows us to incorporate more domain knowledge and do higher level reasoning about vessel behavior, leading to rules recognizing, for example, ferry movements.
In future work we would like to compare to the generic segmentation framework presented by Buchin et al. (2010), which we unfortunately only became aware of after research on this chapter was complete.
This chapter is an extension of the work by de Vries and van Someren (2010). Preliminary research on compression and prediction was done by de Vries and van Someren (2009).

In this chapter we apply two alignment measures: dynamic time warping and edit distance, in three typical machine learning tasks for vessel trajectories: clustering, classification, and outlier detection. The alignment measures are defined as kernels and are used in kernel based algorithms. We investigate the performance of these alignment kernels in combination with the trajectory compression method of Chapter 3. The experiments indicate that compression has a positive effect on the performance in the three tasks. The regular kernels, based on taking the best alignment, perform best in the clustering and classification tasks, whereas the soft-max kernels, based on summing over all alignments, show the best performance in the outlier detection task.

4.1 Introduction

To solve the machine learning tasks of clustering, classification and outlier detection for vessel trajectories, a measure that defines the similarity between these moving object trajectories is required. Vessel trajectories have the property that they are usually different in temporal length, distance traveled and the number of data points. Alignment methods, such as dynamic time warping and edit distance, are designed to handle these kinds of variations. Thus, such methods can make a suitable similarity measure for moving object trajectories.

Computing alignment measures can be expensive, if the number of elements in a trajectory is high. In Chapter 3 we saw that vessel trajectories can be compressed very well. This compression speeds up the computation of alignments by several orders of magnitude, because there is a large reduction in data points. However, the quality of the alignments does not necessarily remain the same. Compression loses information and may have a bad effect on the alignment.

We approach the tasks of clustering, classification and outlier detection in the framework of kernel methods. In this chapter we define different alignment kernels for vessel trajectories based on two well-known alignment measures. We consider three kernel versions of these measures. Not all of these kernels are positive semi-definite. We use the kernels in clustering, classification and outlier detection experiments. The goal of these experiments is to investigate the perf-
formance of the different kernels on the three different tasks. We are especially interested in the influence of applying trajectory compression. The experimental results show that a number of the alignment kernels achieve better performance when the trajectories are compressed first.

The rest of this chapter is organized as follows. Section 4.2 contains the technical definitions of the alignments and the kernels that are based upon them. Our experiments with clustering, classification and outlier detection of vessel trajectories using alignment kernels are described in Section 4.3. We end with some conclusions and directions for future work.

4.2 TRAJECTORY ALIGNMENT KERNELS

In this section we look at alignment similarities between moving object trajectories and how these are used as kernels. We consider the Dynamic Time Warping (DTW) and Edit distance with Real Penalties (ERP) alignment measures. Good results have also been achieved by Longest Common SubSequence (LCSS) (Vlachos et al. (2005)) and Edit Distance on Real sequences (EDR) (Chen and Özsu (2005)). However, due to the way these measures are defined, they are more difficult to turn into positive semi-definite kernels, and we have not included them in this chapter. Table 1 gives an overview of all the kernels defined in this chapter. As mentioned in Chapter 2, we remark that we use the term ‘similarity’ generically, to indicate, for example, ‘distances’, ‘similarities’ and ‘dissimilarities’.

<table>
<thead>
<tr>
<th>Shorthand</th>
<th>Parameters</th>
<th>Description</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{\text{max,DTW}}$</td>
<td>none</td>
<td>Direct kernel of regular DTW</td>
<td>4.2.2 &amp; 4.2.10</td>
</tr>
<tr>
<td>$K_{\text{maxnorm,DTW}}$</td>
<td>none</td>
<td>Direct kernel of normalized DTW</td>
<td>4.2.2 &amp; 4.2.10</td>
</tr>
<tr>
<td>$K_{\text{softmax,DTW}}$</td>
<td>$\beta$</td>
<td>Soft-max kernel of DTW</td>
<td>4.2.2 &amp; 4.2.11</td>
</tr>
<tr>
<td>$K_{\text{max,ED}}$</td>
<td>$g$</td>
<td>Direct kernel of regular edit distance</td>
<td>4.2.3 &amp; 4.2.10</td>
</tr>
<tr>
<td>$K_{\text{maxnorm,ED}}$</td>
<td>$g$</td>
<td>Direct kernel of normalized edit distance</td>
<td>4.2.3 &amp; 4.2.10</td>
</tr>
<tr>
<td>$K_{\text{softmax,ED}}$</td>
<td>$\beta, g$</td>
<td>Soft-max kernel of edit distance</td>
<td>4.2.3 &amp; 4.2.11</td>
</tr>
</tbody>
</table>
4.2.1 Alignments

We define two types of alignments between two trajectories. Based on these alignments we define kernels that express similarity between trajectories in Section 4.2.2. The goal of an alignment is to bring the elements of two sequences, e.g. trajectories, in a correspondence, typically with the aim to maximize (or minimize) a scoring function defined on this correspondence.

Assume in the following that there is a function \( \text{sub}(S(i), T(j)) \), which we define further on in this section, that gives a score to substituting the trajectory element \( S(i) \) with \( T(j) \). This score represents the similarity between these two elements. To enhance readability, we will not use superscripts to indicate the specific type of alignment when this cannot lead to confusion.

We define different alignments for two trajectories below, following the notation and definitions from Vert et al. (2004).

**Definition 4.2.1.** An alignment \( \pi \), possibly with gaps, of \( p \) positions, where \( p \geq 0 \), \( p \leq |S| \) and \( p \leq |T| \), between two trajectories \( S \) and \( T \) is a pair of \( p \)-tuples:

\[
\pi = ((\pi_1(1), \ldots, \pi_1(p)), (\pi_2(1), \ldots, \pi_2(p))) \in \mathbb{N}^{2p} .
\]

The idea behind this definition is that the alignment encodes \( p \) elements in each trajectory that are aligned to each other. That is, the \( \pi_1(i) \)th element in \( S \) is aligned to the \( \pi_2(i) \)th element in \( T \). Multiple elements of \( S \) can be aligned to one element in \( T \) and vice versa. Furthermore, not all elements have to be aligned to an element in the other trajectory, in this case an element is a gap.

**Dynamic Time Warping.** A very popular alignment method, initially designed for time-series, is Dynamic Time Warping (DTW). In Definition 4.2.2 we follow Cuturi et al. (2007).

**Definition 4.2.2.** A DTW alignment \( \pi_{\text{DTW}} \) is an alignment according to Definition 4.2.1, but has the additional constraints that there are unitary increments and no simultaneous repetitions, thus \( \forall (1 \leq i \leq p - 1) \),

\[
\pi_1(i + 1) \leq \pi_1(i) + 1, \quad \pi_2(i + 1) \leq \pi_2(i) + 1
\]

and

\[
(\pi_1(i + 1) - \pi_1(i)) + (\pi_2(i + 1) - \pi_2(i)) \geq 1 .
\]

Furthermore,

\[
1 = \pi_1(1) \leq \pi_1(2) \leq \ldots \leq \pi_1(p) = |S| ,
\]

\[
1 = \pi_2(1) \leq \pi_2(2) \leq \ldots \leq \pi_2(p) = |T| .
\]
This means that all elements in both trajectories are aligned, which might require repeating elements from a trajectory, but in the alignment we cannot simultaneously repeat an element in both trajectories. Furthermore, the start and end of trajectories are aligned by default.

**Edit Distance.** Edit distances are a popular method for comparing the similarity of strings. This similarity is computed in terms of the number of substitutions, deletions and insertions that are required to transform one string into another string. We define (Definition 4.2.3 and 4.2.4) a version for continuous elements similar to how Chen and Ng (2004) define it for time-series. However we chose to consider fixed gap penalties, i.e. deletion and insertion costs, because this seems more natural in the trajectory case. Chen and Ng (2004) define the gap penalties as essentially the value of the gapped element minus a fixed constant. For trajectories this would turn out to mean that something on the edge of the map, having high x, y coordinates, would have a higher penalty than in the middle, which has 0, 0 coordinates, which we think is counter-intuitive.

**Definition 4.2.3.** An edit distance alignment $\pi_{ED}$ is an alignment according to Definition 4.2.1, but also has the constraints:

\[
1 \leq \pi_1(1) < \pi_1(2) < \ldots < \pi_1(p) \leq |S| , \\
1 \leq \pi_2(1) < \pi_2(2) < \ldots < \pi_2(p) \leq |T| .
\]

This means that not all elements have to be aligned and there is no repetition of elements.

We define a score function for the two types of alignments below in Definition 4.2.4.

**Definition 4.2.4.** The score for an alignment $\pi$ of two trajectories $S$ and $T$, using a substitution function $\text{sub}$ and gap penalty $g$, is equal to:

\[
s_{\text{sub},g}(S,T,\pi) = g \cdot ((|S| - |\pi|) + (|T| - |\pi|)) + \sum_{i=1}^{|
\pi|} \text{sub}(S(\pi_1(i)), T(\pi_2(i))) .
\]

This score sums the substitution scores computed by sub for all aligned elements, and adds a constant gap penalty for all elements that are not aligned. Considering the two types of alignments that are defined above, we note that the most important difference is in how they treat gaps. In the DTW case there are no gaps. What would be a gap in the edit distance case is treated in DTW by repeating a trajectory element, and thus gets a score according to the substitution function sub. In the edit distance case gaps have a fixed penalty $g$. Thus, the value of $g$ has no influence for $s_{\text{sub},g}$ if $\pi$ is a DTW alignment.
4.2.2 Alignment Kernels

Below we give different ways to create a kernel to express similarity between trajectories, using the alignments defined above. Two types of kernels are based on taking the score of the alignment that maximizes the score function. This differs from the third kernel which takes the soft-max of the scores of all alignments.

First, we define two very similar substitution functions. The first one is the negative of the $L^2$ norm, i.e. the regular Euclidean distance, in Definition 4.2.5. The second one is the negative of the squared $L^2$ norm, see Definition 4.2.6.

Definition 4.2.5.

$$\text{sub}_1(\langle x_i, y_i, t_i \rangle, \langle x_j, y_j, t_j \rangle) = -\|\langle x_i - x_j, y_i - y_j, t_i - t_j \rangle\|.$$ 

Definition 4.2.6.

$$\text{sub}_2(\langle x_i, y_i, t_i \rangle, \langle x_j, y_j, t_j \rangle) = -\|\langle x_i - x_j, y_i - y_j, t_i - t_j \rangle\|^2.$$ 

Other functions are possible here, but we have not experimented with this. Note that $x$ and $y$ are of the same dimension, but $t$ is not directly comparable to $x$ and $y$. Hence, in the experiments we apply a weight to $t$ to make it comparable. In practice, this weight will depend on the domain and goal of one’s application, cf. $E_3$ in Chapter 3.

Above, in Section 4.2.1, we have defined a number of alignments between two trajectories and their score functions. However, this does not give a similarity between two trajectories yet, because there are a lot of possible alignments, and corresponding scores, for one type of alignment.

One option for similarity between two trajectories is to take the score of the alignment that maximizes the score function $s$. We will call this similarity $\text{sim}_{\text{max}}$ and define it below in Definition 4.2.7. This similarity corresponds to the typical way that DTW and edit distance are defined.

Definition 4.2.7. Given two trajectories $S$ and $T$, let $\Pi_\phi(S, T)$ be the set of all possible alignments between these trajectories under a certain alignment measure $\phi$ and $g \leq 0$, then

$$\text{sim}_{\text{max}, \phi}^g(S, T) = \max_{\pi \in \Pi_\phi(S, T)} s_{\text{sub}_1, g}(S, T, \pi).$$

Depending on the application, this score is normalized by dividing by the sum of the lengths of $S$ and $T$, resulting in the average score per element; see Definition 4.2.8.

Definition 4.2.8. Given two trajectories $S$ and $T$, let $\phi$ be an alignment measure and $g \leq 0$, then

$$\text{sim}_{\text{max norm}, \phi}^g(S, T) = \frac{\text{sim}_{\text{max}, \phi}^g(S, T)}{|S| + |T|}.$$
However, it has been argued (Cuturi et al. (2007); Vert et al. (2004)) that taking the soft-max\(^1\) of the scores of all possible alignments leads to better kernels, because all scores are taken into account. We will call this alignment score \(\text{sim}^{\text{softmax}}\). It is given in Definition 4.2.9 below. The soft-max similarity has a scaling parameter \(\beta\) and uses substitution function \(\text{sub}_2\), which is required for positive semi-definiteness.

**Definition 4.2.9.** Given two trajectories \(S\) and \(T\), let \(\Pi_\phi(S,T)\) be the set of all possible alignments between these trajectories under a certain alignment measure \(\phi\), \(g \leq 0\), and \(\beta > 0\) then
\[
\text{sim}^{g,\beta}_{\text{softmax},\phi}(S,T) = \sum_{\pi \in \Pi_\phi(S,T)} \exp(\beta s_{\text{sub}_2}(S,T,\pi)) .
\]

These similarities are not kernels yet. We define a kernel based on \(\text{sim}_{\text{max}}\) or \(\text{sim}_{\text{maxnorm}}\) in Definition 4.2.10. What we define below is a kernel matrix. Note that in the following two definitions we leave out the alignment measure, and the parameters \(g\) and \(\beta\) for readability.

**Definition 4.2.10.** For a set of trajectories \(\mathcal{T}\), we compute the kernel matrix \(K_\Psi\), where \(\Psi = \{\text{max}|\text{maxnorm}\}\) as:
\[
K_\Psi(i,j) = \text{sim}_\Psi(S,T) ,
\]
where \(i\) and \(j\) are indexes for \(S,T \in \mathcal{T}\). Furthermore, we normalize and make a kernel out of \(K_\Psi\) by:
\[
K_\Psi = 1 - \frac{K_\Psi}{\min(K_\Psi)} .
\]

The kernel based on \(\text{sim}_{\text{softmax}}\) is given in Definition 4.2.11.

**Definition 4.2.11.** For a set of trajectories \(\mathcal{T}\), we compute the normalized kernel matrix \(K_{\text{softmax}}\) as:
\[
K_{\text{softmax}}(i,j) = \frac{\text{sim}_{\text{softmax}}(S,T)}{\sqrt{\text{sim}_{\text{softmax}}(S,S)\text{sim}_{\text{softmax}}(T,T)}} ,
\]
where \(i\) and \(j\) are indexes for \(S,T \in \mathcal{T}\).

### 4.2.1 Positive Semi-Definiteness

The \(K_{\text{max}}\) and \(K_{\text{maxnorm}}\) kernels are not proper kernels in the sense that they are not Positive Semi-Definite (PSD). However it has been observed that non-PSD kernels can still work well in practice (for DTW for instance by Gudmundsson et al. (2008)). We notice this in our experiments in Section 4.3 as well. The soft-max version of the DTW alignment kernel was proven to be PSD by Cuturi et al. (2007), given a proper substitution function like the negative of the squared Euclidean distance, i.e. \(\text{sub}_2\).

\(^1\)Given a set of positive scalars \(Z = z_1, \ldots, z_n\), the soft-max of \(Z\) is \(\log \sum e^{z_i}\).
To the best of our knowledge there exists no proof in the current literature for the positive semi-definiteness of the soft-max version of the edit distance that we have defined above. However, this kernel is very close to the more complex local alignment kernel that is defined by Vert et al. (2004). The proof that $\text{sim}_{\text{softmax,ED}}$ is positive semi-definite can be found in Appendix B.

4.2.2.2 Concluding Remarks

Cuturi et al. (2007) and Vert et al. (2004) notice that the soft-max type of kernel often suffers from the diagonal dominance issue and they remedy this by taking the logarithm of this kernel. We have experimented with taking the logarithm since we also ran into diagonal dominance problems, but this did not improve results.

The classic alignment similarities, $\text{sim}_{\text{max}}$ and $\text{sim}_{\text{maxnorm}}$, are efficiently computable via a dynamic programming approach. By replacing the max-sum algebra with a sum-product algebra we can do the same for the $\text{sim}_{\text{softmax}}$ versions. This is shown for DTW by Cuturi et al. (2007). We can see that this works when we, for instance, work out $\text{sim}_{\text{softmax,DTW}}$ as in Equation 4.1. We leave out the subscript DTW and the parameter $g$, since it does not actually influence DTW.

$$
\text{sim}^\beta_{\text{softmax}}(S, T) = \sum_{\pi \in \Pi(S, T)} \exp\left(\sum_{i=1}^{\left|\pi\right|} \text{sub}_2(S(\pi_1(i)), T(\pi_2(i)))\right) \\
\sum_{\pi \in \Pi(S, T)} \prod_{i=1}^{\left|\pi\right|} \exp\left(\beta \text{sub}_2(S(\pi_1(i)), T(\pi_2(i)))\right)
$$

(4.1)

To sum up, we have now defined a number of kernel functions. Three for the DTW measure and three variants for the edit distance measure; see Table 1.

4.3 Experiments

The goal of the experiments below is to investigate the performance of the different alignment kernels that we defined above on three typical machine learning tasks, clustering (Section 2.3.2), classification (Section 2.3.3) and outlier detection (Section 2.3.4). We are interested in which measure performs best overall and more specifically what the influence is of applying trajectory compression on task performance.

The possibility of compression having a negative impact on alignment similarities is illustrated in Figure 10. The figure shows two trajectories of 7 points, which align nicely together. However, it is possible that compression removes different points from the two paths, as illustrated by the open dots, which means that the remaining solid
dots align very badly. This leads to a bad alignment score, even though the trajectories are very similar.

4.3.1 Experimental Setup

To investigate the influence of trajectory compression on the performance of the three tasks we used 7 compression settings for Piecewise Linear Segmentation (PLS) with Algorithm 3.2.1 and error measure $E_{\mu=2}$, i.e. $\text{pls}_{E_{\mu=2}}$. Note that we used the two-stage variant (Algorithm 3.2.2) to separate stops and moves in the datasets that we use. However, since the datasets only consist of moves, we use regular PLS in the following experiments. We set the $\mu$ parameter to the best stop retention versus compression rate value from Chapter 3 and did not vary it for these experiments, since we found it had little effect on the overall outcome. The first of the compression settings is $\epsilon = 0$, thus, we apply no compression. In the other settings we applied $\text{pls}_{E_{\mu=2}}$ to each trajectory with $\epsilon = 10, 50, 100, 1000, 10000, 100000$ m, respectively. The high epsilon settings are not realistic application settings, since they reduce trajectories to only a few points, but these settings introduce a form of baseline performance.

Because it is unclear how to weigh the time dimension against the space dimensions in the substitution functions, we have defined a number of settings for this. In the first setting we treat the trajectories as if they are sampled with a fixed sample rate. This means that some positions have to be linearly interpolated between two positions from the original data. In this setting a trajectory is a sequence of $(x, y)$ points and time is implicitly represented by the fixed time interval between these points. We introduced this condition, because dynamic time warping is commonly applied to fixed sample rate time-series. Furthermore, by decreasing the sample rate we get a baseline form of compression to compare to $\text{pls}_{E_{\mu=2}}$. 

---

Figure 10: Example of a bad alignment due to compression. Two (partial) trajectories of 7 points. The open dots are points removed by compression and the solid dots are points retained. The retained points align badly, whereas the original points would align well. For simplicity of the illustration, the trajectories are drawn as straight lines. In this case, most likely none of the points would be retained, except for the first and last.
Table 2: Average compression rate (cr) and corresponding sample rate (sr) for different \( \epsilon \) settings.

<table>
<thead>
<tr>
<th>( \epsilon ) (m)</th>
<th>0</th>
<th>10</th>
<th>50</th>
<th>100</th>
<th>1000</th>
<th>10000</th>
<th>100000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clustering</td>
<td>cr (%)</td>
<td>0</td>
<td>92</td>
<td>97</td>
<td>98</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>sr (Hz)</td>
<td>0.11</td>
<td>0.0082</td>
<td>0.0030</td>
<td>0.0023</td>
<td>0.00059</td>
<td>0.0001</td>
<td>0.0001</td>
</tr>
<tr>
<td>Classification</td>
<td>cr (%)</td>
<td>0</td>
<td>85</td>
<td>94</td>
<td>96</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td>sr (Hz)</td>
<td>0.11</td>
<td>0.0053</td>
<td>0.0018</td>
<td>0.0012</td>
<td>0.00045</td>
<td>0.0001</td>
<td>0.00001</td>
</tr>
<tr>
<td>Outlier Detection</td>
<td>cr (%)</td>
<td>0</td>
<td>80</td>
<td>92</td>
<td>95</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>sr (Hz)</td>
<td>0.11</td>
<td>0.0038</td>
<td>0.0014</td>
<td>0.00091</td>
<td>0.00038</td>
<td>0.0001</td>
<td>0.00001</td>
</tr>
</tbody>
</table>

In the other settings we have 4 weights \( w \) for the time dimension, so that a trajectory essentially becomes: \( T = (x_1, y_1, w \cdot t_1), \ldots, (x_n, y_n, w \cdot t_n) \). As weights we took \( w = 0, \frac{1}{2}, 1, 2 \). With \( w = 0 \) we ignore the time dimension and with the other weights we increasingly weigh the time dimension more heavily. The \( w = 1 \) setting means that the average difference between two points in the space dimension is roughly equal to the average difference in the time dimension, cf. \( E_{\mu} \) in Chapter 3.²

In the fixed sample rate setting we set the sample rate to be roughly equal to the compression rate achieved under the different compression settings; see Table 2. We set the sample rate for the no compression case to the average temporal distance between two consecutive trajectory samples in the uncompressed dataset; again see Table 2. Note that these settings differ per dataset.

For each of the different combinations of settings, i.e. for each combination of time setting and compression setting, we compute a number of kernels. For an overview of the kernels, see Table 1. For DTW we compute \( K_{\text{max,DTW}}, K_{\text{maxnorm,DTW}} \), and \( K_{\text{softmax,DTW}}^\beta \), with \( \beta = 1, 4, 16, 64, 256, 1024 \). For the edit distance measure we compute \( K_{\text{max,ED}}^g, K_{\text{maxnorm,ED}}^g \) with \( g = -0.01, -0.025, -0.05, -0.075, -0.1 \). And we compute \( K_{\text{softmax,ED}}^g, K_{\text{softmax,ED}}^g \) with \( g = -0.01^2, -0.025^2, -0.05^2, -0.075^2, -0.1^2 \) and again \( \beta = 1, 4, 16, 64, 256, 1024 \).

**RUNNING TIME** Table 3 gives some results on running times. These computation times are for running our MatLab Mex implementation of the DTW alignment kernels in the clustering experiment³ on an AMD Phenom II X6 1055 (3.2 Ghz) cpu system with 8 GB of ram.

² To enhance readability we do not give the actual values for the weights, because they depend on the units that the dimensions are in.
³ The results are comparable for other kernels.
Table 3: Running time for computing one DTW kernel for the clustering experiment.

<table>
<thead>
<tr>
<th>Kernel type</th>
<th>$\epsilon = 0$</th>
<th>$\epsilon = 0$</th>
<th>$\epsilon = 0$</th>
<th>$\epsilon = 0$</th>
<th>$\epsilon = 0$</th>
<th>$\epsilon = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{\text{max},\text{DTW}}$</td>
<td>172s</td>
<td>1.48</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>$K_{\text{maxnorm},\text{DTW}}$</td>
<td>172s</td>
<td>1.48</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>$K_{\text{softmax},\text{DTW}}$</td>
<td>239s</td>
<td>1.78</td>
<td>1.08</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
</tr>
</tbody>
</table>

4.3.2 Clustering

The goal of the clustering task is to investigate which trajectory kernel best reconstructs the gold standard clustering of Section 2.3.2. The kernels defined above are used as input for the weighted kernel k-means algorithm of Section 2.1.1.2. For each kernel, clustering is done 100 times with random initializations. This process is repeated 10 times. We set $k = 8$, i.e. the same number of clusters as in the gold standard. The function clus_score as defined Section 2.1.2 is used to evaluate the quality of a clustering.

4.3.2.1 Results

In the result tables the mean clus_score of a kernel is computed over the scores of 10 clusterings, thus for each kernel $N = 10$. Per kernel we statistically compare the scores for the high compression/low sample rates settings to the scores for the no compression/high sample rate setting using a two-tailed Student t-test with $p < 0.05$.

The tables present the results per kernel type. First we will give the scores for the three types of DTW kernels and then the scores for the three types of edit distance kernels. The goal of these tables is to show the performance of the kernel type under different compression (the columns) and time weight (the rows) settings. For the kernels that have 1 or 2 parameters, we do not give the results for all the settings, but instead provide the results for the setting that provided the best score and give the mean, and minimum and maximum scores computed over all the parameter settings. The top row of the tables indicate the different compression settings that were used, in case of the fixed sample rate setting these should read the different sample rates. However, we omitted these for readability, Table 2 shows which $\epsilon$ setting corresponds to which sample rate.

Scores marked with $^+$ indicate a significant difference between that clustering score, according to the above defined test, and the clustering score for the no compression ($\epsilon = 0$) case, with the scores marked with $^+$ being higher. Scores marked with $^-$ indicate a significant difference between that score and the score for the no compression ($\epsilon = 0$) case, with the score marked with $^-$ being lower. For completeness we give the mean score of 10 random clusterings, which is 0.14. Scores in bold face indicate the best score for the kernel type. Note
that the scores for the setting $\epsilon = 100000$ are missing, since for the clustering dataset, this produces the same compressed trajectories as $\epsilon = 10000$, reducing all trajectories to 2 points.

Table 4 gives the results for the different types of dynamic time warping kernels. The best overall DTW score (0.86) is achieved by the $K_{\text{maxnorm,DTW}}$ kernel in the fixed sample rate setting of 0.0023 Hz. This score does not differ significantly from the best score (0.80) for the $K_{\text{max,DTW}}$ kernel, but does differ significantly from the best score (0.79) for the $K_{\text{softmax,DTW}}$ kernel.

Furthermore, we see that for some of the $K_{\text{max,DTW}}$ kernel settings performance under compression is significantly better than performance with no compression. For the $K_{\text{softmax,DTW}}$ kernel this holds for all the settings. This is different in the case of the $K_{\text{maxnorm,DTW}}$ kernels, which show significantly less performance than the no compression setting in some cases. However, we must remark here that the soft-max kernels generated in the no compression setting suffer from problems with numerical machine precision and are very diagonally dominant. As mentioned, we experimented with taking the logarithm of these kernels, which unfortunately did not solve this problem.

The kernels for the $\epsilon = 10000$ setting are nearly identical, since this compression setting reduces every trajectory to just its start and end point. This makes the $K_{\text{max,DTW}}$ and $K_{\text{maxnorm,DTW}}$ kernels the same. Therefore the scores for this setting are also very similar.

The results for the different types of edit distance kernels are given in Table 5. The best overall edit distance score (0.89) is achieved by the $K_{\text{max,ED}}$ kernel in the $w = 0$ setting. This score does not differ significantly from the best score (0.86) for the $K_{\text{maxnorm,DTW}}$ kernel or the best score (0.84) for the $K_{\text{maxnorm,ED}}$ kernel. It does differ significantly from the best score (0.82) for the $K_{\text{softmax,ED}}$ kernel.

Like with the $K_{\text{max,DTW}}$ kernel, we see that in the edit distance max and max normalized kernel cases the performance is better for some compression settings than the no compression setting. The same problems with numerical machine precision occur with the edit distance soft-max kernels as well. Furthermore, we note that for the edit distance kernels the same holds as for the dynamic time warping kernels: under $\epsilon = 10000$ the kernels are almost identical.

4.3.2.2 Discussion

The main result of the above experiments is that for most kernel types, for at least a number of settings, the performance on the compressed trajectories is just as good as, or better than, the performance on the uncompressed data, even though we see from Tables 2 and 3 that the dataset is reduced by 92% or more and kernel computation time is at least 100 times faster.
<table>
<thead>
<tr>
<th>β_{best} (\text{max})</th>
<th>79.0</th>
<th>78.0</th>
<th>77.0</th>
<th>76.0</th>
<th>75.0</th>
<th>74.0</th>
<th>73.0</th>
<th>72.0</th>
<th>71.0</th>
<th>70.0</th>
<th>69.0</th>
<th>68.0</th>
<th>67.0</th>
<th>66.0</th>
<th>65.0</th>
<th>64.0</th>
<th>63.0</th>
<th>62.0</th>
<th>61.0</th>
<th>60.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>β_{best} (\text{mean})</td>
<td>70.0</td>
<td>69.0</td>
<td>68.0</td>
<td>67.0</td>
<td>66.0</td>
<td>65.0</td>
<td>64.0</td>
<td>63.0</td>
<td>62.0</td>
<td>61.0</td>
<td>60.0</td>
<td>59.0</td>
<td>58.0</td>
<td>57.0</td>
<td>56.0</td>
<td>55.0</td>
<td>54.0</td>
<td>53.0</td>
<td>52.0</td>
<td>51.0</td>
</tr>
<tr>
<td>β_{best} (\text{min})</td>
<td>52.0</td>
<td>51.0</td>
<td>50.0</td>
<td>49.0</td>
<td>48.0</td>
<td>47.0</td>
<td>46.0</td>
<td>45.0</td>
<td>44.0</td>
<td>43.0</td>
<td>42.0</td>
<td>41.0</td>
<td>40.0</td>
<td>39.0</td>
<td>38.0</td>
<td>37.0</td>
<td>36.0</td>
<td>35.0</td>
<td>34.0</td>
<td>33.0</td>
</tr>
</tbody>
</table>

Table 1: Mean value of clus-score for different DTW kernels.
Table 5: Mean value of clus._score for different ED kernels.

|                        | best (g = -0.25) | best (g = -0.05) | best (g = 0) | best (g = 0.05) | best (g = 0.25) | best (g = 0.5) | best (g = 1) | best (g = 1.5) | best (g = 2) | best (g = 2.5) | best (g = 3) | best (g = 3.5) | best (g = 4) | best (g = 4.5) | best (g = 5) | best (g = 5.5) | best (g = 6) | best (g = 6.5) | best (g = 7) | best (g = 7.5) | best (g = 8) | best (g = 8.5) | best (g = 9) | best (g = 9.5) | best (g = 10) |
|------------------------|------------------|------------------|--------------|-----------------|-----------------|----------------|--------------|----------------|--------------|----------------|--------------|----------------|--------------|----------------|--------------|----------------|--------------|----------------|--------------|----------------|--------------|----------------|--------------|----------------|
| **K_{clus,ED}** for    |                  |                  |              |                 |                 |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |
| w = 0                  | 0.80             | 0.75             | 0.77         | 0.77            | 0.77            | 0.77           |       |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |
| w = 1/2                | 0.77             | 0.72             | 0.70         | 0.70            | 0.70            | 0.70           |       |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |
| w = 1                  | 0.71             | 0.69             | 0.67         | 0.67            | 0.67            | 0.67           |       |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |
| w = 2                  | 0.67             | 0.65             | 0.62         | 0.62            | 0.62            | 0.62           |       |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |
| **K_{inout,ED}** for    |                  |                  |              |                 |                 |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |
| w = 0                  | 0.77             | 0.73             | 0.70         | 0.69            | 0.68            | 0.66           |       |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |
| w = 1                  | 0.72             | 0.69             | 0.67         | 0.66            | 0.65            | 0.64           |       |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |
| w = 2                  | 0.71             | 0.68             | 0.65         | 0.64            | 0.63            | 0.62           |       |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |                 |              |

(Continued on next page)
Furthermore, the regular max and max normalized versions of the DTW and edit distance kernels perform better than their soft-max counterparts. However we must make the proviso that, for the uncompressed case, computation of the soft-max kernels runs into problems with numerical machine precision.

The best score was achieved by the $K_{\text{max,ED}}$ kernel. However, this was not significantly better than the best score for the $K_{\text{max,DTW}}$ kernel or the $K_{\text{maxnorm,ED}}$ kernel. Moreover, these best scores were all reached under different time weight settings, suggesting that different treatments of time are beneficial for different alignment types.

Scores almost as good as the best score can be achieved by reducing the trajectories to just two points, i.e. the start and end, and using these two to compute similarity. Inspecting the dataset suggests that, to determine the cluster of a trajectory indeed this information is almost sufficient. However, one does not expect this to hold in general.

### 4.3.3 Classification

In the classification task we use the trajectory kernels in combination with a Support Vector Machine (SVM) to predict the type of vessel related to a trajectory. For this we use the classification dataset as described in Section 2.3.3. This dataset consists of trajectories of the four most common vessel types. We randomly select 100 trajectories for each vessel type to create a dataset of 400 trajectories. This unskewed selection makes sure that we get the biggest difference in results.

For the classification algorithm we use the C-SVC implementation of the Support Vector Machine algorithm in LibSVM, we refer to Section 2.1.1.1 for more details. To evaluate the performance of the different kernels we use a 10-fold cross validation set-up. For each fold the classification accuracy is computed as the performance measure. Within each fold the C parameter is optimized by selecting it from the range $10^{-4}$, $10^{-3}$, $10^{-2}$, 0.1, 1, 10, $10^2$, $10^3$, $10^4$ using, again, 10-fold cross validation.

#### 4.3.3.1 Results

In the results below we present the mean classification accuracy over 10 folds, hence $N = 10$ for each kernel. To statistically compare the results for two kernels we use a two-tailed paired t-test with $p < 0.05$, since the 10 folds are the same for all kernels. The presentation of the results below has the same structure as used for the clustering experiment. With 4 equally sized classes, the mean random classification accuracy is 25%.

Table 6 presents the results for the different types of dynamic time warping kernels. The best performance (74.5%) is achieved by the max and max norm kernels. This score differs significantly from the best (68.75%) soft-max kernel performance.
For all kernel types we see that compression has a positive effect on classification, since the best scores are achieved under high compression settings. We also must remark again that the soft-max kernels suffer from numerical machine precision problems in the uncompressed case.

We note that in the $\epsilon = 100000$ compression setting the kernels are nearly identical, since this setting reduces trajectories to start and end points only. Since experiment folds are the same for every kernel we see that the same settings result in identical scores.

In Table 7 we give the results for the edit distance kernels. The best score (76.25%) is achieved by the $K_{\text{maxnorm,ED}}$ kernel. This score does not differ significantly from the best score (76%) for the max kernel nor the best score (75.25%) for the soft-max kernel. There is also no significant difference with the best DTW score (74.5%).

Like the DTW case, compression has a positive effect on performance, since for all kernels the best results are achieved under high compression rates. Also here, the soft-max kernels have problems with numerical machine precision. And again, kernels for the $\epsilon = 100000$ setting are nearly identical.

### 4.3.3.2 Discussion

From the above results we see that compression has a positive result on the performance in the classification task. However, the performance in case of very high compression ($\epsilon = 100000$), which amounts to reduction to just start and end points, is not significantly worse than the overall best performance. This means that the current classification task can be solved just as well by considering only the start and end points of the trajectories, making the proposed alignment measures unnecessary in this task. On the one hand this suggests that our classification task is too simple. However, there is still room for almost 24% performance gain. When we inspect the confusion matrix for the highest performance, we see that the most misclassifications are made when separating tankers from cargo ships. These two types have almost identical trajectories, and hence are nearly impossible to distinguish on the basis of their trajectories alone.

Similar to the clustering experiment, we see that the soft-max kernels are outperformed by their regular (max and max norm) counterparts. When it comes to which treatment of time to use, and thus whether piecewise linear segmentation is more useful than reducing the sample rate, the classification task does not provide a lot of insight, since the performance is similar under the fixed sample rate, $w = 0$ and $w > 0$ settings.

We note that 400 trajectories is a relatively small amount of data for the classification experiments. However, because of the results of the experiment, which shows that our classification task does not provide great distinction between measures, we chose not to repeat the
<table>
<thead>
<tr>
<th>Alignment based similarities</th>
<th>Mean accuracy (%) for different DTW kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>Table 6: Mean accuracy (%) for different DTW kernels.</td>
</tr>
<tr>
<td>$c$</td>
<td>$e = 0 \text{m}$</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------</td>
</tr>
<tr>
<td>$K_{\text{mean,ED}}$ for best ($g = -0.01$)</td>
<td>66.5</td>
</tr>
<tr>
<td></td>
<td>$\text{mean}_{\text{min}}^{\text{max}}$</td>
</tr>
<tr>
<td>$w = 0$</td>
<td>best ($g = -0.01$)</td>
</tr>
<tr>
<td></td>
<td>$\text{mean}_{\text{min}}^{\text{max}}$</td>
</tr>
<tr>
<td>$w = \frac{1}{2}$</td>
<td>best ($g = -0.01$)</td>
</tr>
<tr>
<td></td>
<td>$\text{mean}_{\text{min}}^{\text{max}}$</td>
</tr>
<tr>
<td>$w = 1$</td>
<td>best ($g = -0.025$)</td>
</tr>
<tr>
<td></td>
<td>$\text{mean}_{\text{min}}^{\text{max}}$</td>
</tr>
<tr>
<td>$w = 2$</td>
<td>best ($g = -0.01$)</td>
</tr>
<tr>
<td></td>
<td>$\text{mean}_{\text{min}}^{\text{max}}$</td>
</tr>
<tr>
<td>$K_{\text{mean,ED}}$ for best ($g = -0.01$)</td>
<td>68.75</td>
</tr>
<tr>
<td></td>
<td>$\text{mean}_{\text{min}}^{\text{max}}$</td>
</tr>
<tr>
<td>$w = 0$</td>
<td>best ($g = -0.01$)</td>
</tr>
<tr>
<td></td>
<td>$\text{mean}_{\text{min}}^{\text{max}}$</td>
</tr>
<tr>
<td>$w = \frac{1}{2}$</td>
<td>best ($g = -0.01$)</td>
</tr>
<tr>
<td></td>
<td>$\text{mean}_{\text{min}}^{\text{max}}$</td>
</tr>
<tr>
<td>$w = 1$</td>
<td>best ($g = -0.025$)</td>
</tr>
<tr>
<td></td>
<td>$\text{mean}_{\text{min}}^{\text{max}}$</td>
</tr>
<tr>
<td>$w = 2$</td>
<td>best ($g = -0.01$)</td>
</tr>
<tr>
<td></td>
<td>$\text{mean}_{\text{min}}^{\text{max}}$</td>
</tr>
<tr>
<td>$K_{\text{mean,ED}}$ for best ($g = -0.01, \beta = 1024$)</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>$\text{mean}_{\text{min}}^{\text{max}}$</td>
</tr>
<tr>
<td>$w = 0$</td>
<td>best ($g = -0.01, \beta = 1024$)</td>
</tr>
<tr>
<td></td>
<td>$\text{mean}_{\text{min}}^{\text{max}}$</td>
</tr>
<tr>
<td>$w = \frac{1}{2}$</td>
<td>best ($g = -0.01, \beta = 1024$)</td>
</tr>
<tr>
<td></td>
<td>$\text{mean}_{\text{min}}^{\text{max}}$</td>
</tr>
<tr>
<td>$w = 1$</td>
<td>best ($g = -0.05, \beta = 256$)</td>
</tr>
<tr>
<td></td>
<td>$\text{mean}_{\text{min}}^{\text{max}}$</td>
</tr>
<tr>
<td>$w = 2$</td>
<td>best ($g = -0.01, \beta = 1024$)</td>
</tr>
<tr>
<td></td>
<td>$\text{mean}_{\text{min}}^{\text{max}}$</td>
</tr>
</tbody>
</table>
experiment with more trajectories, which would require a substantial amount of extra computation time.

4.3.4 Outlier Detection

For outlier detection we use trajectory kernels with one-class SVMs to find outlying trajectories among a set of vessel trajectories of cargo ships and tankers. This dataset is described in Section 2.3.4. It consists of 786 vessel trajectories of which 39 are labeled as outliers.

For outlier detection we use the one class version of Support Vector Machines in LibSVM, we refer to Section 2.1.1.1 for more details. To evaluate the performance of the different kernels we randomly split the 747 normal trajectories in the dataset into a training set containing $\frac{2}{3}$ of the dataset and a test set containing the other $\frac{1}{3}$. To the test set we add the 39 outlier trajectories. Per kernel we do this split 10 times. For each split we train a one class SVM model on the training set, using 10-fold cross validation to determine the optimal $\nu$ from the range $0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$. In this experimental set-up we assume that in an application setting we have cleaned the training set from outliers such that they have no influence on the model. Therefore we also do not include outliers in our training sets.

To test the performance of the different kernels in outlier detection we use the decision values generated by the one class SVM to rank the test trajectories. We select the top 39 ranked outlying trajectories and compute as performance score the fraction of labeled outliers among these 39, i.e. the precision@39, see Section 2.1.2.

4.3.4.1 Results

Below we present the mean precision@39 scores of a kernel computed over 10 splits, thus for each kernel $N = 10$. Per kernel we statistically compare the scores for the high compression/low sample rates settings to the scores for the no compression/high sample rate setting using a two-tailed Student t-test with $p < 0.05$. The following tables, presenting the results, have the same structure as the tables for the clustering and classification experiments. The mean precision@39 score for 10 random rankings of the test set is 0.16.

Table 8 presents the results for the different dynamic time warping kernels. The best performance is clearly achieved by the $K_{\text{softmax,DTW}}$ kernels, with the best score (0.84) differing significantly from the best scores of the $K_{\text{max,DTW}}$ and $K_{\text{maxnorm,DTW}}$ kernels.

For all three types of kernels we see that most compression settings significantly outperform the no compression setting. Again, as in the clustering and classification experiments, the soft-max kernels suffer from problems with numerical machine precision. As in the classification experiment, the kernels for the $\epsilon = 100000$ setting are nearly
identical, since this setting reduces every trajectory to just start and end points, hence the very similar scores.

In Table 9 we present the results for the different edit distance kernels. Again we see that the best performance is achieved by the softmax type kernels, with the best score (0.77) being significantly better than the best scores for the max and max norm kernels. However, this score is significantly worse than the best score (0.84) for the soft-max DTW kernels.

Like with the DTW kernels, we see that most compression settings significantly outperform the no compression setting. Again we must remark that the soft-max kernels suffer from problems with numerical machine precision. Also, the kernels for the $\epsilon = 100000$ setting are nearly identical, resulting in similar performance.

4.3.4.2 Discussion

As with the clustering and classification experiments, data reduction has a largely positive effect on outlier detection task performance. However, in the outlier detection task the soft-max kernels shine and outperform the max and max norm kernels, which are the best in the clustering and classification tasks, by a large margin.

Most outlying trajectories are outliers because part of the trajectory is strange. This strange part does not align well with normal trajectories. The soft-max kernels take into account all possible alignments, and in all these possible alignments this strange part is likely difficult to align well. Thus outlying trajectories will end up being very dissimilar from regular trajectories. This effect is the strongest in the case of the dynamic time warping soft-max kernel. For DTW badly aligning points can lead to high penalties, because there is no maximum gap penalty $g$.

Another consequence of the fact that outliers are only strange in a part of the trajectory is that considering only the start and end points of trajectories does not work as well. A large part of the outlying trajectories have very normal start and end points.

The outlier detection task favors the piecewise linear segmentation form of compression over reducing the sample rate, for both the max and softmax type of kernels. This makes sense, because the strange part of an outlier is likely to be retained with piecewise linear segmentation, the strangeness is likely outside the error threshold. With reduction of sample rate the strange part can just be missed, because the low sample rate misses those points.

4.4 Conclusions & Future Work

The results from the three experiments above show that, in all the tasks, applying compression to trajectories can have a positive influence on task performance. This is a good result, since this com-
<table>
<thead>
<tr>
<th>m</th>
<th>precision@8</th>
<th>recall@8</th>
<th>fsr</th>
<th>fsr</th>
<th>soft-max, DTW</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>1</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>2</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>3</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>4</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>5</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>6</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>7</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>8</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>9</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>10</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>11</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>12</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>13</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>14</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>15</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>16</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
<tr>
<td>17</td>
<td>0.000000</td>
<td>0.0000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

Table 5: Precision@8 for different DWM kernels.
Table 9: Precision@39 for different ED kernels.

<table>
<thead>
<tr>
<th>K_{norm, ED}</th>
<th>w = 0</th>
<th>w = 1/2</th>
<th>w = 1</th>
<th>w = 2</th>
<th>K_{norm, ED}</th>
<th>w = 0</th>
<th>w = 1/2</th>
<th>w = 1</th>
<th>w = 2</th>
<th>K_{norm, ED}</th>
<th>w = 0</th>
<th>w = 1/2</th>
<th>w = 1</th>
<th>w = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c = 0\text{m}$</td>
<td>0.38</td>
<td>0.36</td>
<td>0.34</td>
<td>0.37</td>
<td>$c = 10\text{m}$</td>
<td>0.34</td>
<td>0.34</td>
<td>0.49</td>
<td>0.38</td>
<td>$c = 50\text{m}$</td>
<td>0.34</td>
<td>0.34</td>
<td>0.49</td>
<td>0.38</td>
</tr>
<tr>
<td>$c = 100\text{m}$</td>
<td>0.34</td>
<td>0.40</td>
<td>0.37</td>
<td>0.44</td>
<td>$c = 1000\text{m}$</td>
<td>0.34</td>
<td>0.40</td>
<td>0.44</td>
<td>0.38</td>
<td>$c = 5000\text{m}$</td>
<td>0.34</td>
<td>0.40</td>
<td>0.44</td>
<td>0.38</td>
</tr>
<tr>
<td>$c = 10000\text{m}$</td>
<td>0.34</td>
<td>0.40</td>
<td>0.44</td>
<td>0.38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$K_{norm, ED}$</th>
<th>$w = 0$</th>
<th>$w = 1/2$</th>
<th>$w = 1$</th>
<th>$w = 2$</th>
<th>$K_{norm, ED}$</th>
<th>$w = 0$</th>
<th>$w = 1/2$</th>
<th>$w = 1$</th>
<th>$w = 2$</th>
<th>$K_{norm, ED}$</th>
<th>$w = 0$</th>
<th>$w = 1/2$</th>
<th>$w = 1$</th>
<th>$w = 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c = 0\text{m}$</td>
<td>0.38</td>
<td>0.36</td>
<td>0.34</td>
<td>0.37</td>
<td>$c = 10\text{m}$</td>
<td>0.34</td>
<td>0.34</td>
<td>0.49</td>
<td>0.38</td>
<td>$c = 50\text{m}$</td>
<td>0.34</td>
<td>0.34</td>
<td>0.49</td>
<td>0.38</td>
</tr>
<tr>
<td>$c = 100\text{m}$</td>
<td>0.34</td>
<td>0.40</td>
<td>0.37</td>
<td>0.44</td>
<td>$c = 1000\text{m}$</td>
<td>0.34</td>
<td>0.40</td>
<td>0.44</td>
<td>0.38</td>
<td>$c = 5000\text{m}$</td>
<td>0.34</td>
<td>0.40</td>
<td>0.44</td>
<td>0.38</td>
</tr>
<tr>
<td>$c = 10000\text{m}$</td>
<td>0.34</td>
<td>0.40</td>
<td>0.44</td>
<td>0.38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$K_{norm, ED}$</th>
<th>$w = 0$</th>
<th>$w = 1/2$</th>
<th>$w = 1$</th>
<th>$w = 2$</th>
<th>$K_{norm, ED}$</th>
<th>$w = 0$</th>
<th>$w = 1/2$</th>
<th>$w = 1$</th>
<th>$w = 2$</th>
<th>$K_{norm, ED}$</th>
<th>$w = 0$</th>
<th>$w = 1/2$</th>
<th>$w = 1$</th>
<th>$w = 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c = 0\text{m}$</td>
<td>0.38</td>
<td>0.36</td>
<td>0.34</td>
<td>0.37</td>
<td>$c = 10\text{m}$</td>
<td>0.34</td>
<td>0.34</td>
<td>0.49</td>
<td>0.38</td>
<td>$c = 50\text{m}$</td>
<td>0.34</td>
<td>0.34</td>
<td>0.49</td>
<td>0.38</td>
</tr>
<tr>
<td>$c = 100\text{m}$</td>
<td>0.34</td>
<td>0.40</td>
<td>0.37</td>
<td>0.44</td>
<td>$c = 1000\text{m}$</td>
<td>0.34</td>
<td>0.40</td>
<td>0.44</td>
<td>0.38</td>
<td>$c = 5000\text{m}$</td>
<td>0.34</td>
<td>0.40</td>
<td>0.44</td>
<td>0.38</td>
</tr>
<tr>
<td>$c = 10000\text{m}$</td>
<td>0.34</td>
<td>0.40</td>
<td>0.44</td>
<td>0.38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
pression furthermore reduces computation time for alignment measures substantially, by a factor 100 or more. The outlier detection task clearly favors the method of piecewise linear segmentation compression over just reducing the sample rate. This trajectory compression algorithm works by recursively retaining the most salient elements of a trajectory, reducing noise. As we mentioned in the introduction, this compression can have a potentially negative influence on alignments. However, in our dataset, similar trajectories have similar salient elements, vessels stop and turn in the same places due to geographical and legislative constraints. This means that similar, compressed trajectories can be aligned well.

In the clustering and classification tasks it is a form of regular edit distance, i.e. the max or normalized max version, that achieves the highest scores, albeit that these scores are not statistically significantly different from some other settings. Also the regular versions of DTW outperform their soft-max counterpart. So for clustering and classification the regular alignments work better. This is the opposite in case of outlier detection, where the soft-max kernels substantially outperform the regular versions.

In most application scenarios these alignment measures will be applied in an online context, and in future work it will be especially interesting to compare their performance and the possibly needed adaptations in such a setting.

We mentioned that outliers usually only have a part of the trajectory that is strange. Therefore, it would make sense to consider alignment measures that consider subsequences, such as Smith-Waterman (Vert et al. (2004)), for outlier detection.

The performance of the alignment kernels in all three tasks suggests that the influence of the temporal component is not very large, often the best scores are achieved in the setting where time is ignored. In future work it is interesting to look at tasks and datasets where the role of the temporal component is larger. In case of vessel trajectories the outlier detection task seems to provide the best opportunities for this.
In this chapter we study similarity measures for vessel trajectories that are based on taking the integral over time of the distance between two trajectories. For these similarities we define positive semi-definite kernel variants in an approach similar to Chapter 4. We test these kernels in the tasks of clustering, classification and outlier detection. These experiments indicate that these kernel variants work, but they are outperformed by the alignment methods from the previous chapter. As in the previous chapter the soft-max kernel variants perform well in outlier detection.

5.1 Introduction

The alignment measures used in clustering, classification and outlier detection in Chapter 4 are one type of similarity that can be used for these tasks. Another solution, coming from the field of computational geometry is to express the similarity between trajectories as an integral of the distance between two trajectories over time. To deal with the temporal length and distance traveled variations in trajectories, this similarity has the option of allowing time shifts and subtrajectories.

As in the chapter about alignments, we study this similarity in the kernel framework. Therefore we define, for each combination of allowing time shift and/or subtrajectories, positive semi-definite kernel variants of this similarity measure. To the best of our knowledge these positive semi-definite kernels have not been studied in the field of spatio-temporal data-mining. However, some of the kernels that we define are equal to kernels already considered in the time-series literature. As in Chapter 4 we also consider direct variants of the defined similarity measure, which are not positive semi-definite.

We use the defined kernels in the same experiments as Chapter 4. The goal of these experiments is to discover which of the variants of the similarity measure performs best and how the performance compares to the alignment kernels. It turns out that the alignment kernels outperform the integral based kernels.

In the rest of this chapter we discuss the following. Section 5.2 defines a general version of trajectory similarity based on the integral over time. We then give more specific versions of this similarity and corresponding positive semi-definite kernels. The experiments with clustering, classification and outlier detection, using these ker-
5.2 Integral Based Trajectory Similarity

Buchin et al. (2009) give algorithms to compute various cases of Definition 5.2.1, a form of integral over time based trajectory similarity. A more specific version of this similarity appeared in earlier literature, such as the work by Nanni and Pedreschi (2006). There exist other definitions for shape matching that do not explicitly consider time that can also be applied to trajectories, but we do not consider those here.

In Definition 5.2.1, trajectories are continuous functions $T: \mathbb{R} \to \mathbb{R} \times \mathbb{R}$, from time to position, i.e. $T(t) = (x, y)$. We use bold face capital letters ($T$) to distinguish trajectories as continuous functions from trajectories as sequences ($T$) as in Definition 2.2.1. Furthermore, for two trajectories $S, T$, let $d$ be the Euclidean distance function:

$$d(S(t), T(t)) = \| (S(t) - T(t)) \| . \quad (5.1)$$

**Definition 5.2.1.** Let $S$ and $T$ be two trajectories. And let $\tau \leq \min(t_S, t_T - t_{\text{shift}})$, where $t_S$ is the maximum $t$ for which $S(t)$ exists and $t_T$ is the maximum $t$ for which $T(t)$ exists. Furthermore, $\tau \geq t_{\min} > 0$. Then, we define trajectory similarity $D_{\text{sub,shift}}(S, T)$ between $S$ and $T$ as

$$D_{\text{sub,shift}}(S, T) = \min_{\tau, t_{\text{shift}}} \frac{\int_0^\tau d(S(t), T(t + t_{\text{shift}})) \, dt}{\tau} .$$

Figure 11 illustrates this similarity function. The variable $\tau$ represents the temporal length over which the integral of $d$ over time is computed. With $t_{\text{shift}}$, the trajectory $T$ can be temporally shifted with respect to the trajectory $S$. The similarity value is the value of the integral over time divided by $\tau$ for the $\tau$ and $t_{\text{shift}}$ that minimize this integral divided by $\tau$. Furthermore $\tau$ is constrained such that it is larger than some minimal subtrajectory length $t_{\min}$.

In this rest of this section we will first consider the more specific cases of this similarity. We consider the version that does not allow for
subtrajectories, i.e. where \( \tau \) is fixed, both with and without time shift \( t_{\text{shift}} \) in Section 5.2.1. Then we treat the versions with subtrajectories, i.e. variable \( \tau \), in Section 5.2.2. For these four variants we introduce corresponding positive semi-definite kernels. Most of these kernels are similar to the soft-max kernels of Chapter 4. We also consider direct kernels of the four variants. A list of all the kernels introduced in this chapter is given in Table 10.

As mentioned in Chapter 2, we use the term ‘similarity’ generically, for similarities and distances alike. Hence we refer to the distance defined in Definition 5.2.1 as a similarity. Furthermore, we recall that we refer to both a kernel function and the kernel matrix based on that function as kernel. However, we use the small \( k \) for functions and the capital \( K \) for matrices. We use the concept of a mapping kernel (Shin and Kuboyama (2008)), given in Definition 2.1.2, to ensure PSD-ness of some of the kernels that we define. We could also do this directly via the ‘standard’ closure properties (Section 2.1.1.1). However, we think that the mapping kernel leads to more elegant definitions.

### Table 10: List of kernels defined in this chapter.

<table>
<thead>
<tr>
<th>Shorthand</th>
<th>Parameters</th>
<th>Description</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( K_{D_{\text{noshift}}} )</td>
<td>none</td>
<td>Direct kernel version of ( D_{\text{noshift}} )</td>
<td>5.2.2 &amp; 5.2.12</td>
</tr>
<tr>
<td>( K_{D_{\text{shift}}} )</td>
<td>( \tau_{\min} )</td>
<td>Direct kernel version of ( D_{\text{shift}} )</td>
<td>5.2.4 &amp; 5.2.12</td>
</tr>
<tr>
<td>( K_{D_{\text{sub_noshift}}} )</td>
<td>( \tau_{\min} )</td>
<td>Direct kernel version of ( D_{\text{sub_noshift}} )</td>
<td>5.2.8 &amp; 5.2.12</td>
</tr>
<tr>
<td>( K_{D_{\text{sub_shift}}} )</td>
<td>( \tau_{\min} )</td>
<td>Direct kernel version of ( D_{\text{sub_shift}} )</td>
<td>5.2.1 &amp; 5.2.12</td>
</tr>
<tr>
<td>( K_{\text{noshift}} )</td>
<td>( \beta )</td>
<td>PSD kernel version of ( D_{\text{noshift}} )</td>
<td>5.2.3 &amp; 5.2.11</td>
</tr>
<tr>
<td>( K_{\text{sumshift}} )</td>
<td>( \beta )</td>
<td>PSD kernel version of ( D_{\text{shift}} )</td>
<td>5.2.5 &amp; 5.2.11</td>
</tr>
<tr>
<td>( K_{\text{sub_fixed}} )</td>
<td>( \beta, n )</td>
<td>PSD kernel version of ( D_{\text{shift}} )</td>
<td>5.2.7 &amp; 5.2.11</td>
</tr>
<tr>
<td>( K_{\text{sub_noshift}} )</td>
<td>( \beta, \tau_{\min} )</td>
<td>PSD kernel version of ( D_{\text{sub_noshift}} )</td>
<td>5.2.9 &amp; 5.2.11</td>
</tr>
<tr>
<td>( K_{\text{sub_shift}} )</td>
<td>( \beta, \tau_{\min} )</td>
<td>PSD kernel version of ( D_{\text{sub_shift}} )</td>
<td>5.2.10 &amp; 5.2.11</td>
</tr>
</tbody>
</table>
### 5.2.1 Full Trajectory Similarity Kernels

In this section we look at variants of Definition 5.2.1 that do not allow subtrajectories. We define positive semi-definite kernels corresponding to these similarities.

#### 5.2.1.1 No Time Shift

First, we consider the case of Definition 5.2.1 where \( \tau = \min(t_S, t_T) \), i.e. we consider full trajectory similarity. We also do not allow for time shift, which reduces Definition 5.2.1 to Definition 5.2.2 below.

**Definition 5.2.2.** Let \( S \) and \( T \) be two trajectories. And let \( \tau = \min(t_S, t_T) \), where \( t_S \) is the maximum \( t \) for which \( S(t) \) exists and \( t_T \) is the maximum \( t \) for which \( T(t) \) exists. Then, we define trajectory similarity \( D_{\text{noshift}}(S, T) \) between \( S \) and \( T \) as

\[
D_{\text{noshift}}(S, T) = \frac{\int_0^\tau d(S(t), T(t)) \, dt}{\tau}. \tag{5.2}
\]

Thus, the integral over time is computed over the entire temporal length for which the two trajectories both exist. This similarity is the same as what time-series literature refers to as the Euclidean distance for time-series (Faloutsos et al. (1994)).

Trajectories are always sampled and therefore piecewise linear, which leads to the fact that \( d \) is a piecewise hyperbolic function. Each piece \( d^* \) of \( d \) is of the form:

\[
d^*(t) = \sqrt{At^2 + Bt + C}, \tag{5.3}
\]

for some constants \( A, B, C \), which leads to a symbolically solvable integral for each piece. Note that the similarity function is a metric, if the time \( \tau \) is the same for every two trajectories in our set.

Before we turn the similarity measure in Definition 5.2.2 into a positive semi-definite kernel we recall the kernel closure properties from Section 2.1.1.1.

Furthermore, we know that the Gaussian kernel, with scaling parameter \( \beta > 0 \),

\[
\exp(-\beta d(S(t), T(t))^2) \tag{5.4}
\]

is positive semi-definite.

In the vein of the similarity in Definition 5.2.2 we define the following kernel \( k \), for two trajectories \( S, T \in \mathcal{U} \), where \( \mathcal{U} \) contains trajectories of the same length \( \tau \). Let the interval \([0, \tau]\) be divided into subintervals of equal width \( \Delta t \), and from each interval choose a point \( t_i \), then

\[
k^\beta(S, T) = \int_0^\tau \exp(-\beta d(S(t), T(t))^2) \, dt = \lim_{n \to \infty} \sum_{i=1}^n \exp(-\beta d(S(t_i), T(t_i))^2) \Delta t. \tag{5.5}
\]
We can see that this is a proper kernel by the closure under sum and limit properties, since \( \tau \) is finite, and \( \exp(-\beta d(S(t), T(t))^2) \) is a kernel.

We can extend this kernel to trajectories of variable length. Suppose \( V \) is a set of trajectories with variable temporal length. Let \( t_{\text{ends}} = 0, t_1, \ldots, t_m \) be the ordered list of the last time points for each \( T \in V \), starting with \( 0 \). Let \( T(t_i, t_j) \) be the part of the trajectory \( T \) from \( t_i \) to \( t_j \). Now consider the following kernel \( k' \), with \( S, T \in V \).

\[
k'^\beta(S, T) = \sum_{n=1}^{m} k^\beta(S(t_{\text{ends}}(n), t_{\text{ends}}(n+1)), T(t_{\text{ends}}(n), t_{\text{ends}}(n+1)))
\]

(5.6)

where \( k(S(t_{\text{ends}}(n), t_{\text{ends}}(n+1)), T(t_{\text{ends}}(n), t_{\text{ends}}(n+1))) = 0 \) if either \( S \) or \( T \) does not exist for the interval \( t_{\text{ends}}(n), t_{\text{ends}}(n+1) \). Note that by zero extension \( k \) remains a valid kernel. Hence, by closure under sum, \( k' \) is a valid kernel.

In practice the integral defined in the above kernel is computationally intensive to compute compared to approximating the integral by sampling \( d(S(t), T(t)) \) at a fixed sample rate, especially when considering time shift, as we will do further on in the chapter.

From now on we will only consider the fixed sample rate case, instead of the somewhat more complex case of continuous trajectory functions. We do this for two reasons. Firstly, the following kernel definitions are clearer and easier to read for the fixed sample rate case. Secondly and more importantly, in practice we use an approximation by fixed sample rate to compute both the similarity functions \( D(S, T) \) and the kernels that we define.

In the rest of this chapter a fixed sample rate trajectory is a sequence of points \( (\mathcal{T} = \{(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_n, y_n, t_n)\}) \), as in Definition 2.2.1, indicated with an italic font face \( (\mathcal{T}) \). We recall the shorthand \( T(i) = (x_i, y_i, t_i) \). Since the sample rate is fixed, \( d(S(i), T(i)) \) reduces to \( d((x_{S,i}, y_{S,i}), (x_{T,i}, y_{T,i})) \), with \( S(i) = (x_{S,i}, y_{S,i}, t_{S,i}) \) and \( T(i) = (x_{T,i}, y_{T,i}, t_{T,i}) \), because \( t_{S,i} = t_{T,i} \). I.e. in practice \( d \) is the same as in the continuous trajectory case.

Below we give the fixed sample rate version definition of kernel \( k' \).

**Definition 5.2.3.** Let \( \mathcal{T} \) be a set of fixed sample rate trajectories and \( \beta > 0 \). Then we define the kernel function \( k_{\text{oshift}} \) between two trajectories \( S, T \in \mathcal{T} \) as:

\[
k_{\text{oshift}}^\beta(S, T) = \lim_{n \to \infty} \sum_{i=1}^{n} \exp(-\beta d(S(i), T(i))^2)
\]

where \( \exp(-\beta d(S(i), T(i))^2) \) is defined to be 0 if one of the trajectories is undefined for that \( i \).
This is finite, because trajectories have finite length. Like before, this is a valid kernel because of the closure properties under sum and limit.

Note that the kernel defined above allows us to compare a set of trajectories of different length. On the other hand, the original $D_{\text{no_shift}}(S, T)$ is not a metric for a set of trajectories with different temporal length, and using the simple approach of directly putting $D$ in the Gaussian kernel ($\exp^{-D(S, T)^2}$) does not yield a proper positive semi-definite kernel for trajectories of different length.

As mentioned above, we use the concept of a mapping kernel (Shin and Kuboyama (2008)), given in Definition 2.1.2, further on in this chapter to show that the defined kernels are PSD. We illustrate the use of the mapping kernel by showing that the above kernel is also a mapping kernel.

We define $\mathcal{J}'$ as the set of all $(x, y)$ points in the trajectories $\mathcal{J}$, and $\mathcal{J}'_T$ as the subset of $\mathcal{J}'$ that contains only the points of the trajectory $T$. Furthermore, we define the map $M_{\text{noshift}}(S, T)$ below:

$$M_{\text{noshift}}(S, T) = \{(S(i), T(i)) \mid 0 \leq i \leq \min(|S|, |T|)\}.$$  

Each point in $S$ is mapped to the point in $T$ with the same index, if it exists. It is easy to see that this map is transitive. Consider any three trajectories $S, T, U \in \mathcal{J}$ and the three maps $M_{\text{noshift}}(S, T), M_{\text{noshift}}(T, U)$ and $M_{\text{noshift}}(S, U)$. If, for any $i$, $(S(i), T(i)) \in M_{\text{noshift}}(S, T)$ and $(T(i), U(i)) \in M_{\text{noshift}}(T, U)$, then $0 \leq i \leq |S|$ and $i \leq |U|$. Thus $(S(i), U(i)) \in M_{\text{noshift}}(S, U)$ by definition of the map.

With this map, we can give an alternative version of Definition 5.2.3:

$$k_{\text{noshift}}^\beta(S, T) = \sum_{(S(i), T(i)) \in M_{\text{noshift}}(S, T)} \exp(-\beta d(S(i), T(i))^2).$$  

5.2.1.2 Time Shift

Now, we consider the variant of Definition 5.2.1 that includes time shift, but not subtrajectories. This version is given below in Definition 5.2.4.

**Definition 5.2.4.** Let $S$ and $T$ be two trajectories. And let $\tau = \min(t_S, t_T - t_{\text{shift}})$, where $t_S$ is the maximum $t$ for which $S(t)$ exists and $t_T$ is the maximum $t$ for which $T(t)$ exists. Furthermore, $\tau \geq t_{\min} > 0$. Then, we define trajectory similarity $D_{\text{shift}}(S, T)$ between $S$ and $T$ as

$$D_{\text{shift}}(S, T) = \min_{t_{\text{shift}}} \int_0^\tau d(S(t), T(t + t_{\text{shift}})) \, dt.$$  

Essentially this measure computes the integral over time from $D_{\text{noshift}}$ for the value of $t_{\text{shift}}$, i.e. the time shift, that minimizes this function. In other words, this similarity tries to optimally align $S$ and $T$ temporally, such that the integral is smallest.
It makes sense to constrain the maximum allowed time shift \( t_{\text{shift}} \) to be larger than some \( t_{\text{min}} \). For example, a shift of \( t_{\text{S}} \) or \( t_{\text{T}} \) only compares the first and last point of two trajectories. This would make two trajectories with the same path, but in opposite direction, completely equal to each other.

To introduce time shift into the kernel of Definition 5.2.3, we treat time shift as a shift in a number of trajectory points \( (i_{\text{shift}}) \). Instead of comparing the points \( S(i) \) and \( T(i) \), we compare \( S(i) \) with \( T(i + i_{\text{shift}}) \). Thus, naively incorporating time shift, in a similar fashion as in Definition 5.2.4, into Definition 5.2.3 leads to the following kernel:

\[
k^\beta(S, T) = \min_{i_{\text{shift}}} \left( \lim_{n \to \infty} \sum_{i=1}^{n} \exp\left(-\beta d(S(i), T(i + i_{\text{shift}}))^2\right) \right).
\] (5.10)

The combination of the optimal value of \( i_{\text{shift}} \) for a pair of trajectories \( S \) and \( T \) and the optimal \( i_{\text{shift}} \) for a pair \( T, U \), does not necessarily give the optimal \( i_{\text{shift}} \) for the pair of trajectories \( S \) and \( U \). This results in the fact that a map \( M(S, T) \) similar to the one above is not transitive for this kernel, because of \( i_{\text{shift}} \). The above kernel is not positive semi-definite. We can create a positive semi-definite kernel that takes into account time shift by defining a map \( M(S, T) \) that is transitive and that represents this time shift in some fashion.

One obvious solution is to sum over all possible \( i_{\text{shift}} \) between two trajectories \( S \) and \( T \), like the soft-max kernels in Chapter 4. However, this does not result in a PSD kernel. Somewhat more complex maps are required to solve this problem. In the rest of this section we will define two different kernels and corresponding maps that give PSD solutions which take into account time shift.

Suppose that all trajectories in \( T \) are sampled with sample rate \( \phi \). Let

\[
\text{shift}(T, i_{\text{shift}}) = \langle x_1, y_1, t_1 - \frac{i_{\text{shift}}}{\phi}, \ldots, x_{|T|}, y_{|T|}, t_{|T|} - \frac{i_{\text{shift}}}{\phi} \rangle,
\]

with \( T(i) = \langle x_i, y_i, t_i \rangle \).

Then, \( \text{shift}(T, i_{\text{shift}}) \) defines a time shifted version of \( T \) over \( \frac{i_{\text{shift}}}{\phi} \). We define the set of all time shifts \( \text{shifts}(T) \) for one trajectory \( T \) as:

\[
\text{shifts}(T) = \{\text{shift}(T, i_{\text{shift}}) \mid 0 \leq i_{\text{shift}} \leq |T|\}.
\] (5.12)

In terms of a mapping kernel, if \( T \) is a set of trajectories, then \( T' = \bigcup_{T \in T} \text{shifts}(T) \), and hence \( T'_T = \text{shifts}(T) \). As a map we take

\[
M_{\text{sumshift}}(S, T) = \text{shifts}(S) \times \text{shifts}(T).
\] (5.13)

This map is trivially transitive, since it consists of the cross-product.

To construct a mapping kernel with this map, we need a kernel on \( T' \). This kernel is a slightly altered version of \( k_{\text{noshift}} \). We need to change the kernel, since trajectories in \( T' \) do not necessarily have
t = 0 for the first element, hence we cannot directly compare \( S(i) \) to \( T(i) \), since the t’s might not be equal. We give this altered version based on the mapping kernel variant of \( k_{\text{noshift}} \). We need to redefine the map \( M'_{\text{noshift}}(S, T) \) from Equation 5.7 to \( M'_{\text{noshift}}(S, T) \) such that each element in \( S \) is mapped to the element in \( T \) with the same time \( t \).

\[
M'_{\text{noshift}}(S, T) = \{ (S(i), T(j)) \mid t_i = t_j, S(i) = (x_i, y_i, t_i), T(j) = (x_j, y_j, t_j) \}. \tag{5.14}
\]

Note that this map is also transitive and that \( M_{\text{noshift}}(S, T) \) is a special case of this map. Using this map, \( k_{\text{noshift}} \) is redefined to \( k'_{\text{noshift}} \):

\[
k'_{\text{noshift}}(S, T) = \sum_{(S(i), T(j)) \in M'_{\text{noshift}}(S, T)} \exp(-\beta d(S(i), T(j))^2). \tag{5.15}
\]

Putting these ingredients together leads to the following kernel in Definition 5.2.5.

**Definition 5.2.5.** Let \( \mathcal{T} \) be a set of fixed sample rate trajectories and \( \beta > 0 \). Then we define the kernel function \( k_{\text{sumshift}} \) between two trajectories \( S, T \in \mathcal{T} \) as:

\[
k_{\text{sumshift}}^\beta(S, T) = \sum_{(S', T') \in M_{\text{sumshift}}(S, T)} k'_{\text{noshift}}(S', T').
\]

To ensure transitivity of the map \( M_{\text{sumshift}} \) and hence PSD-ness, this kernel compares the same shifted trajectories multiple times. We can see this by the fact that comparing \( \text{shift}(S, 0) \) with \( \text{shift}(T, 0) \) is the same as comparing \( \text{shift}(S, 1) \) and \( \text{shift}(T, 1) \), etc. This means that the larger shifts, i.e. higher \( i_{\text{shift}} \), get less weight, since they are repeated less times in the mapping. We can use the fact that the same shifted trajectories are compared multiple times to speed up the computation of this kernel, since it turns computing \( |S||T| \) shifts into computing \( |S| + |T| \) shifts.

Another option to incorporate time shift is to compare subtrajectories of fixed length and all possible mappings between those. Note that we do consider subtrajectories here, but of a fixed length. Hence we can consider the kernel that we define as somewhere in between taking full trajectories and subtrajectories. Because we will use it in the following, we define a variant of Definition 5.2.3 that takes the product over all \( \exp(-\beta d(S(i), T(i))^2) \).

**Definition 5.2.6.** Let \( \mathcal{T} \) be a set of fixed sample rate trajectories of length \( n \) and \( \beta > 0 \). Then we define the kernel function \( k_{\text{prod}} \) between two trajectories \( S, T \in \mathcal{T} \) as:

\[
k_{\text{prod}}^\beta(S, T) = \prod_{i=1}^{n} \exp(-\beta d(S(i), T(i))^2).
\]
This is a kernel by the closure under product property. The important difference with Definition 5.2.3 is that all trajectories have the same length $n$. It would also be a kernel for variable length trajectories, if we take $\exp(-\beta d(S(i), T(i))^2)$ to be 0 if one of the trajectories is not defined for $i$. However, this would result in the kernel computing $0$ for two trajectories of unequal length, which is not very useful. The kernel is known as the radial basis function kernel for time-series (Rueping (2001)).

Next we define $\text{subtraj}(T, n)$ which gives all the subtrajectories of $T$ of length $n$ as:

$$\text{subtraj}(T, n) = \{T(i, i + n - 1) | 0 \leq i \leq |T| - n + 1\} \quad (5.16)$$

With this we create the transitive map $M^n_{\text{subtraj}}(S, T)$, between all subtrajectories of length $n$ of $S$ and $T$,

$$M^n_{\text{subtraj}}(S, T) = \text{subtraj}(S, n) \times \text{subtraj}(T, n) \quad (5.17)$$

This allows us to define the subtrajectory kernel in Definition 5.2.7.

**Definition 5.2.7.** Let $\mathcal{T}$ be a set of fixed sample rate trajectories and $\beta > 0$. Then we define the kernel function $k_{\text{fixed}_\text{sub}}$ between two trajectories $S, T \in \mathcal{T}$ as:

$$k_{\text{fixed}_\text{sub}}^n(S, T) = \sum_{(S', T') \in M^n_{\text{subtraj}}(S, T)} k^\beta_{\text{prod}}(S', T') \quad .$$

If we set this $n$ to be relatively high compared to the length of a trajectory, for instance $n = \min_{T \in \mathcal{T}}(|T|)$, then only long subtrajectories are considered, therefore we say that the kernel is somewhere between a subtrajectory and full trajectory kernel. Also, the fact that $n$ is fixed limits the number of time shift that are considered. For instance, situations in which the first point of one trajectory is only compared to the last point of the other trajectory are not possible, unless $n = 1$. The computation time of this kernel is an order of magnitude worse than the previous kernel. We have to consider $|S| + |T|$ possible time shifts, which each have $|U| - n$ subtrajectories, where $|U|$ is the length of the overlap between $S$ and $T$ for a certain time shift i.e. $|U|$ is maximally $\min(|S|, |T|)$. This leads to computing similarity for $(|S| + |T|)(|U| - n)$ subtrajectories.

When we work out this definition to:

$$k_{\text{fixed}_\text{sub}}^n(S, T) = \sum_{(S', T') \in M^n_{\text{subtraj}}(S, T)} k^\beta_{\text{prod}}(S', T')$$

$$= \sum_{(S', T') \in M^n_{\text{subtraj}}(S, T)} \prod_{i=1}^{n} \exp(-\beta d(S'(i), T'(i))^2)$$

$$= \sum_{(S', T') \in M^n_{\text{subtraj}}(S, T)} \exp(-\beta \sum_{i=1}^{n} d(S'(i), T'(i))^2) \quad ,$$

$$\quad (5.18)$$
then we see a close resemblance to the soft-max kernels of Chapter 4. In this kernel we compute the soft-max of all similarities between subtrajectories of length \( n \). We note that this kernel is the same as the subsequence kernel for time-series and sequences (as, e.g. by Rueping (2001)), but we derived it differently. The given definition uses \( k_{\text{prod}} \) as the ‘inner’ kernel, but \( k_{\text{noshift}} \) can be used instead.

**Relation to Alignments** When we consider the similarity measures \( D \) as computed via a fixed sample rate, then \( D_{\text{shift}} \) in Definition 5.2.4 is quite similar to the dynamic time warping (DTW) alignment. Both have computation times that are quadratic. DTW is flexible in catering for slight differences in speed by allowing repetition of elements. In \( D_{\text{shift}} \) we allow for time shifts, which do not introduce an extra penalty, and hence the amount of shifting needs to be constraint. DTW also allows for some time shift, by repeating the first or last elements of a trajectory. The difference between \( D_{\text{shift}} \) and edit distance is similar, but edit distance furthermore has the fixed gap penalty \( g \).

The alignment methods benefit greatly from trajectory compression, because this speeds up their computation. We cannot do this in a similar fashion in the \( D_{\text{shift}} \) case, since we require a high sample rate to approximate the integral. However, we can speed up computation by reducing the number of shifts to consider, by setting the shift as a multiple of the sample rate.

### 5.2.2 Subtrajectory Similarity Kernels

In the previous section we looked at kernels that considered similarity between full trajectories. Albeit that the kernel in Definition 5.2.7 already considered subtrajectories, but of a fixed length. In this section we will look at kernel versions of subtrajectory similarity.

First, we consider the version of \( D \), in Definition 5.2.8, that considers subtrajectories, but no time shift.

**Definition 5.2.8.** Let \( S \) and \( T \) be two trajectories. And let \( \tau \leq \min(t_S, t_T) \), where \( t_S \) is the maximum \( t \) for which \( S(t) \) exists and \( t_T \) is the maximum \( t \) for which \( T(t) \) exists. Furthermore, \( \tau \geq t_{\text{min}} > 0 \). Then, we define trajectory similarity \( D(S, T) \) between \( S \) and \( T \) as

\[
D_{\text{sub,noshift}}(S, T) = \min_{\tau} \frac{\int_0^\tau d(S(t), T(t)) dt}{\tau}.
\]

This variant of \( D \) considers all possible subtrajectories longer than \( t_{\text{min}} \), but does not allow for time shift. This means that we require a different map than \( M_{\text{subtraj}} \) for subtrajectories of length \( n \):

\[
M_{\text{subtraj}}^n(S, T) = \{(S', T') | t_{S, 0} = t_{T, 0}, S'(0) = (x_{S, 0}, y_{S, 0}, t_{S, 0}), T'(0) = (x_{T, 0}, y_{T, 0}, t_{T, 0}), S' \in \text{subtraj}(S, n), T' \in \text{traj}(T, n)\}.
\]

(5.19)
In this map, which is transitive, we map the subtrajectories of $S$ and $T$ that have the same start time. We replace this map in $k_{\text{fixed}_\text{sub}}$ to get:

$$k_{\text{fixed}_\text{sub}}^{n_1,\beta}(S, T) = \sum_{(S', T') \in M_{n_1}^{\text{subtraj}}(S, T)} k_{\text{prod}}^{\beta}(S', T') . \quad (5.20)$$

We use this kernel to define (Definition 5.2.9) a kernel that takes the soft-max of all similarities of subtrajectories larger than $i_{\text{min}}$.

**Definition 5.2.9.** Let $T$ be a set of fixed sample rate trajectories and $\beta > 0$. Then we define the kernel function $k_{\text{sub}_\text{noshift}}$ between two trajectories $S, T \in T$ as:

$$k_{\text{sub}_\text{noshift}}^{i_{\text{min}},\beta}(S, T) = \lim_{n \to \infty} \sum_{i=1}^{n} k_{\text{subtraj}}^{i,\beta}(S, T) .$$

This is a positive semi-definite kernel by the closure under the limit property since the maximum length of a subtrajectory is finite. For computing this kernel we have to consider $|T| - i_{\text{min}}$ subtrajectories, where $T$ is the shortest trajectory.

The final variant of $D$ that we look at is the one already given in Definition 5.2.1. This version considers both subtrajectories of minimum length $t_{\text{min}}$ and time shifts. Given all that we defined above, the kernel variant of this similarity is straightforward, and we can immediately give it in Definition 5.2.10. We use the earlier defined $k_{\text{fixed}_\text{sub}}$.

**Definition 5.2.10.** Let $T$ be a set of fixed sample rate trajectories and $\beta > 0$. Then we define the kernel function $k_{\text{sub}_\text{shift}}$ between two trajectories $S, T \in T$ as:

$$k_{\text{sub}_\text{shift}}^{i_{\text{min}},\beta}(S, T) = \lim_{n \to \infty} \sum_{i=1}^{n} k_{\text{fixed}_\text{sub}}^{i,\beta}(S, T) .$$

Again, this kernel is PSD by the closure under the limit property. This kernel clearly has the worst running time of all the kernels, considering $(|S| + |T|)(|U| - i_{\text{min}})^2$ subtrajectories, where $|U|$ is the length of the overlap between $S$ and $T$ for a certain time shift.

**Normalization and direct kernels of $D$** We normalize the positive semi-definite kernels defined above the same way as the soft-max kernels of Chapter 4, defined in Definition 5.2.11. This normalization step preserves PSD-ness.

**Definition 5.2.11.** For a set of trajectories $T$ we compute the normalized kernel matrix $K$ for a kernel $k$ as:

$$K(i, j) = \frac{k(S, T)}{\sqrt{k(S, S)k(T, T)}} ,$$

where $i$ and $j$ are indexes for $S, T \in T$. 
We also want to use the different versions of $D$ in our experiments with kernel algorithms in the next section. We turn these into kernels in Definition 5.2.12 the same way as we did in Chapter 4 with the regular versions of dynamic time warping and edit distance. Like with the kernels that we defined above, we compute $D$ using a fixed sample rate approximation.

**Definition 5.2.12.** For a set of trajectories $\mathcal{T}$ we compute the kernel matrix $K_D$ for a similarity measure $D$ as:

$$K_D(i,j) = D(S,T),$$

where $i$ and $j$ are indexes for $S, T \in \mathcal{T}$. Furthermore, we normalize and make a kernel out of $K_D$ by:

$$K_D = 1 - \frac{K_D}{\min(K_D)}.$$

We use the subscript $D$ to distinguish the kernel matrices defined above from the kernel matrices defined for the positive semi-definite kernel functions $k$ and matrices $K$ defined earlier in this section.

For a summary of all the kernels that we defined above we refer to Table 10.

### 5.3 Experiments

The goal of the experiments below is to test the defined similarity measures and their corresponding kernel variants in the three typical machine learning tasks of clustering (Section 2.3.2), classification (Section 2.3.3) and outlier detection (Section 2.3.4) for vessel trajectory data. The main goal of the experiments is to determine which measures work best and if the positive semi-definite kernel variants perform better or worse.

#### 5.3.1 Experimental Set-Up

The set-up of the experiments is very similar to the experiments of Chapter 4 that investigate the different alignment kernels. Therefore, we are slightly more brief in stating the algorithm set-up for each experiment. The same datasets and algorithms are used and hence results in this section can be compared directly to the results in the previous chapter. The main difference is that the influence of trajectory compression is not investigated and that the kernels defined in the current chapter have different parameters.

We run experiments for all the kernels in Table 10. For the parameter $n$ in $K_{n,B}^{\text{sub\_fixed}}$ and $i_{\min}$ in $K_{D_{\text{shift}}}^{\text{min}}, K_{D_{\text{sub\_noshift}}}^{\text{min}}, K_{D_{\text{noshift}}}^{\text{min}}$, and $K_{\text{sub\_shift}}^{\text{noshift}}$ and $K_{\text{sub\_shift}}^{\text{noshift}}$ we try $\lambda = \frac{1}{4}, \frac{1}{2}, 1$ in $\lambda \cdot \min_{T \in \mathcal{T}}(|T|)$ as different settings,
i.e. for each experiment we take the length of the shortest trajectory in the dataset and set \( n \), resp. \( i_{\min} \), to a fraction of that length.

To speed up the computation of the kernels that shift over the trajectory to consider different time shifts or subtrajectories, i.e. all kernels but \( K_{D \text{no shift}} \) and \( K_{\text{no shift}} \), we introduce the parameter \( \gamma \). This \( \gamma \) determines the interval, in number of samples, over which we shift, by \( \gamma \cdot \lambda \cdot \min_{T \in \mathcal{T}}(|T|) \). For each setting of \( \lambda \) we consider two settings \( \gamma = \frac{1}{2}, 1 \). Using this \( \gamma \) keeps the positive semi-definiteness of the kernels. The smallest settings of \( \gamma = \frac{1}{2} \) and \( \lambda = \frac{1}{2} \) lead to shifts of approximately 10 samples, which is a 100-fold computation-time speed up for the slowest kernels.

For the three kernels \( K_{\text{sub fixed}}, K_{\text{sub no shift}} \) and \( K_{\text{sub shift}} \) that have \( k_{\text{prod}} \) as inner kernel we also test a variant where this kernel is replaced by \( k_{\text{no shift}} \).

We use \( \beta = 1, 4, 16, 64, 256, 1024 \) for the kernels that have parameter \( \beta \). However for the kernels that make use of \( k_{\text{no shift}} \) as inner kernel, i.e. the three mentioned above and \( K_{\text{sum shift}} \) we noticed that better results are achieved by higher values for \( \beta \), here we take \( \beta = 64, 256, 1024, 4000, 64000, 256000 \).

All the kernels are computed on trajectories that have been resampled at a 0.111Hz sample rate, which is the same as we used for the no compression setting in Chapter 4.

5.3.2 Clustering

In the clustering task we investigate which of the kernels defined in the previous section reconstructs the gold standard of Section 2.3.2 best. We use the kernels as input in weighted kernel k-means; see Section 2.1.1.2. For each kernel we generate 10 clusterings, with \( k = 8 \), the amount in the gold standard. We evaluate the quality of a clustering with the formula from Section 2.1.2.

5.3.2.1 Results

Below we present the mean score of a kernel over 10 clusterings, i.e. \( N = 10 \) for each kernel. Statistical comparisons between two kernels are done using a two-tailed Student t-test, with \( p < 0.05 \).

In the table below we present the results per kernel type. The different columns show the results of the kernels for various settings of \( \lambda \) and \( \gamma \), when applicable. The kernels \( K_{D \text{no shift}} \) and \( K_{\text{no shift}} \) are not influenced by \( \lambda \) and \( \gamma \), thus there is only one score. Also, we note that not all columns are filled for the kernel \( K_{\text{sum shift}} \). This is because this kernel is only influenced by the \( \gamma \) parameter, which means that some combinations of \( \gamma \) and \( \lambda \) lead to the same setting, e.g. \( \gamma = \frac{1}{2}, \lambda = 1 \) and \( \gamma = 1, \lambda = \frac{1}{2} \). For the kernels that have a \( \beta \) parameter we give the results for the setting of \( \beta \) that gives the best score and we provide
the mean, minimum and maximum scores over all the $\beta$ settings. We give the mean score of 10 random clusterings, which is 0.14.

Table 11 presents the results for all the different kernels. The best clustering score (0.80) is achieved by the $K_{D_{\text{shift}}}$ kernel under the $\lambda = 1, \gamma = 1$ setting. This score is significantly better than the best scores for the other kernels, except for the best score for $K_{\text{sumshift}}$ which is the best performing PSD kernel. The best clustering score (0.89) from Chapter 4 is significantly better than the best score from Table 11.

5.3.2.2 Discussion

Like we saw with the alignment kernels in the previous chapter, the best result is achieved by a non positive semi-definite kernel. However, the differences are not as big here as they are in that chapter. When we consider the results for $K_{D_{\text{noshift}}}$ and $K_{D_{\text{shift}}}$ we see that allowing for a little time shift ($\lambda = 1$) is beneficial for the results, but allowing for more shift degrades performance.

For the other kernels, the different shift settings ($\lambda$ and $\gamma$) have no effect, or only marginal (for $K_{D_{\text{subshift}}}$ and $K_{\text{subshift}}$). There is no difference in performance for the three kernels with $K_{\text{noshift}}$ as inner kernel, compared to their regular counterparts.

The kernels that are more expensive to compute, i.e. because they include time shift and/or subtrajectories, show better performance than the computationally cheap kernels $K_{D_{\text{noshift}}}$ and $K_{\text{noshift}}$.

5.3.3 Classification

In the classification task we predict the type of vessel related to a trajectory in the dataset of Section 2.3.3 using a Support Vector Machine (SVM). In this dataset we have trajectories of the four most common vessel types. For each of these types we select 100 trajectories to create a dataset of 400 vessel trajectories, i.e. the same dataset is used as in the previous chapter.

As our SVM algorithm we use the C-SVC implementation of LibSVM; see Section 2.1.1.1 for details. We use a 10-fold cross validation set-up to evaluate the different kernels, where we compute the classification accuracy for each fold. Within each fold the C parameter is optimized using, again, 10-fold cross validation.

5.3.3.1 Results

We present the mean classification accuracy over 10 folds, i.e. $N = 10$ for each kernel. Statistical comparisons between the results of two kernels are done using a two-tailed paired t-test with $p < 0.05$, since the 10 folds are the same for each kernel. The table presenting the results below has the same structure as used for the clustering experiment. The mean random classification accuracy is 25%.
Table 11: Mean value of clus_score for different integral over time kernels.

<table>
<thead>
<tr>
<th>$\lambda = \frac{1}{4}$</th>
<th>$\gamma = \frac{1}{4}$</th>
<th>$\lambda = \frac{1}{2}$</th>
<th>$\gamma = \frac{1}{2}$</th>
<th>$\lambda = 1$</th>
<th>$\gamma = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{D_{noshift}}$</td>
<td>0.62</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$K_{D_{shift}}$</td>
<td>0.59</td>
<td>0.54</td>
<td>0.65</td>
<td>0.75</td>
<td>0.74</td>
</tr>
<tr>
<td>$K_{D_{sub_noshift}}$</td>
<td>0.53</td>
<td>0.54</td>
<td>0.57</td>
<td>0.57</td>
<td>0.59</td>
</tr>
<tr>
<td>$K_{D_{sub_shift}}$</td>
<td>0.58</td>
<td><strong>0.69</strong></td>
<td>0.66</td>
<td>0.69</td>
<td>0.66</td>
</tr>
<tr>
<td>$K_{noshift}$</td>
<td>best ($\beta = 64$)</td>
<td><strong>0.63</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mean$^{max}_{min}$</td>
<td>0.53$^{0.63}_{0.43}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$K_{noshift}$</td>
<td>best ($\beta = 256$)</td>
<td>0.65</td>
<td>0.75</td>
<td>0.72</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mean$^{max}_{min}$</td>
<td>0.62$^{0.69}_{0.56}$</td>
<td>0.63$^{0.75}_{0.54}$</td>
<td>0.62$^{0.72}_{0.52}$</td>
<td></td>
</tr>
<tr>
<td>$K_{sub_noshift}$</td>
<td>best ($\beta = 16$)</td>
<td>0.69</td>
<td>0.72</td>
<td>0.58</td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td>mean$^{max}_{min}$</td>
<td>0.59$^{0.71}_{0.50}$</td>
<td>0.61$^{0.72}_{0.55}$</td>
<td>0.62$^{0.71}_{0.55}$</td>
<td>0.62$^{0.71}_{0.54}$</td>
</tr>
<tr>
<td>$K_{sub_shift}$</td>
<td>best ($\beta = 1$)</td>
<td>0.70</td>
<td>0.71</td>
<td>0.71</td>
<td>0.71</td>
</tr>
<tr>
<td></td>
<td>mean$^{max}_{min}$</td>
<td>0.62$^{0.70}_{0.54}$</td>
<td>0.61$^{0.71}_{0.55}$</td>
<td>0.63$^{0.71}_{0.54}$</td>
<td>0.63$^{0.71}_{0.52}$</td>
</tr>
<tr>
<td>$K_{fixed_noshift}$</td>
<td>best ($\beta = 64$)</td>
<td>0.71</td>
<td>0.71</td>
<td>0.71</td>
<td>0.70</td>
</tr>
<tr>
<td></td>
<td>mean$^{max}_{min}$</td>
<td>0.56$^{0.71}_{0.48}$</td>
<td>0.58$^{0.71}_{0.51}$</td>
<td>0.57$^{0.71}_{0.51}$</td>
<td>0.56$^{0.70}_{0.49}$</td>
</tr>
<tr>
<td>$K_{fixed_shift}$</td>
<td>best ($\beta = 64$)</td>
<td>0.62</td>
<td>0.62</td>
<td>0.62</td>
<td>0.60</td>
</tr>
<tr>
<td></td>
<td>mean$^{max}_{min}$</td>
<td>0.54$^{0.62}_{0.43}$</td>
<td>0.52$^{0.62}_{0.44}$</td>
<td>0.52$^{0.62}_{0.41}$</td>
<td>0.53$^{0.62}_{0.44}$</td>
</tr>
<tr>
<td>$K_{sub_shift}$</td>
<td>best ($\beta = 64$)</td>
<td>0.71</td>
<td>0.71</td>
<td>0.71</td>
<td>0.71</td>
</tr>
<tr>
<td></td>
<td>mean$^{max}_{min}$</td>
<td>0.58$^{0.71}_{0.48}$</td>
<td>0.58$^{0.71}_{0.48}$</td>
<td>0.57$^{0.71}_{0.50}$</td>
<td>0.59$^{0.71}_{0.49}$</td>
</tr>
</tbody>
</table>
We present the classification results for all the different kernels in Table 12. The best accuracy (68.25) is achieved by the $K_{\text{sumshift}}$ kernel. This accuracy is significantly better than the accuracies for the different $K_D$ kernels, but not significantly better than the accuracies for the other PSD kernels. The best accuracy (76.25) achieved for the alignment kernels is significantly better than the best result achieved in this chapter.

5.3.3.2 Discussion

Contrary to the alignment kernels of Chapter 4, the best results are not achieved by the non-PSD kernels, but by the positive semi-definite kernels, by a large margin. For almost all the kernels the influence of the $\lambda$ and $\gamma$ settings is nearly non-existent. The exception being the $K_{D_{\text{sub shift}}}$ kernel, which performs somewhat better with higher $\lambda$. Like we saw in the clustering experiment, there is not much difference between the performance of the three kernels with $k_{\text{noshift}}$ as their inner kernel, and the performance of their corresponding regular versions. Similar to the results for alignment kernels, it does not pay off to compute more expensive kernels, since the $K_{D_{\text{noshift}}}$ kernel does not perform worse than the more expensive $K_D$ kernels, and the same holds for the $K_{noshift}$ kernel.

5.3.4 Outlier Detection

In the outlier detection task we use the kernels with one-class Support Vector Machines to find the outliers among a set of trajectories of cargo ships and tankers; see Section 2.3.4. This set consists of 786 trajectories, 39 of which are outliers.

We use the one-class version of SVM in LibSVM. We randomly split the 747 normal trajectories into a training set of $\frac{2}{3}$ and a test set of the rest. To this test set we add the outlying trajectories. We do this split 10 times per kernel. For each split we train a one-class SVM model. The $\nu$ parameter is optimized during training using 10-fold cross validation. The decision values that this model gives on the test set are used to generate a ranking of the test trajectories. On this ranking, we use the precision@39 measure to evaluate the performance of the different kernels.

5.3.4.1 Results

The mean precision@39 scores for each kernel are computed over 10 splits, thus $N = 10$. Statistical comparisons between two kernels are done using a two-tailed Student t-test, with $p < 0.05$. The following table presents the result in the same way as for the clustering and classification experiments. The mean precision@39 for 10 random rankings of the test set is 0.16.
Table 12: Mean accuracy (%) for different integral over time kernels.

<table>
<thead>
<tr>
<th></th>
<th>$\lambda = \frac{1}{4}$</th>
<th>$\lambda = \frac{1}{2}$</th>
<th>$\lambda = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma = \frac{1}{4}$</td>
<td>44.45</td>
<td>42.5</td>
<td>40.0</td>
</tr>
<tr>
<td>$\gamma = \frac{1}{2}$</td>
<td>39.25</td>
<td>40.25</td>
<td>38.5</td>
</tr>
<tr>
<td>$\gamma = 1$</td>
<td>43.75</td>
<td>43.75</td>
<td>48.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Best ($\lambda = 1024$)</th>
<th>Mean (max, min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{D_{noshift}}$</td>
<td>65.5</td>
<td>58.21, 53.25</td>
</tr>
<tr>
<td>$K_{D_{shift}}$</td>
<td>68.25</td>
<td>64.58, 62.25</td>
</tr>
<tr>
<td>$K_{D_{sub_noshift}}$</td>
<td>67.75</td>
<td>64.83, 62.25</td>
</tr>
<tr>
<td>$K_{D_{sub_shift}}$</td>
<td>67.25</td>
<td>62.25, 62.25</td>
</tr>
<tr>
<td>$K_{K_{noshift}}$</td>
<td>67.75</td>
<td>64.83, 62.25</td>
</tr>
<tr>
<td>$K_{K_{sub_noshift}}$</td>
<td>67.25</td>
<td>62.25, 62.25</td>
</tr>
<tr>
<td>$K_{K_{sub_shift}}$</td>
<td>67.25</td>
<td>62.25, 62.25</td>
</tr>
<tr>
<td>$K_{fixed_sub}$</td>
<td>67.75</td>
<td>64.83, 62.25</td>
</tr>
<tr>
<td>$K_{fixed_shift}$</td>
<td>67.75</td>
<td>64.83, 62.25</td>
</tr>
</tbody>
</table>

53 Experiments
Table 13 presents the results for the outlier detection experiment. The best score (0.53) is achieved by the $K_{\text{sub\_noshift}}$ kernel. This score is significantly better than all the other best scores, except for the best score (0.49) for the $K_{\text{noshift}}$ kernel and the best score (0.51) for the $K_{\text{sub\_noshift}}$ with $k_{\text{noshift}}$ kernel. However, these scores are significantly worse than the best score (0.84) for the alignment kernels.

5.3.4.2 Discussion

Like we saw with the soft-max kernels in Chapter 4, the PSD kernels perform a lot better on the outlier detection task than the non-PSD kernels. We see, as in the previous two experiments, that the influence of the shifting parameters $\lambda$ and $\gamma$ is marginal. Also similar to the previous two experiments, the three kernels with $k_{\text{noshift}}$ as inner kernels do not differ much in performance from their regular counterparts. The not significantly worse performance of the cheap $K_{\text{noshift}}$ kernel does not warrant the computation of more expensive kernels on the outlier detection task.

The rather large performance difference between the kernels defined in this chapter and the alignment kernels from Chapter 4 for the outlier detection task is striking. This might be due to the fact that piecewise linear segmentation ‘magnifies’ the strange part in a trajectory, since this strange part adds extra retained points, compared to a regular trajectory. This effect leads to bad alignments. However, in the fixed sample rate, integral based approach, the strange part is not particularly ‘magnified’.

5.4 Conclusions & Future Work

The main result for this chapter is that kernels based on integrating the distance between two trajectories over time perform worse than the kernels based on alignment measures from Chapter 4. In all the three experiments we obtained significantly worse results than achieved in the previous chapter. Moreover, apart from the most basic kernels ($K_{D\text{noshift}}$ and $K_{\text{noshift}}$) these integral based kernels are more expensive to compute, since they do not benefit from trajectory compression, i.e. they require a large number of samples to approximate the integral.

Like in the previous chapter we see that for the different tasks, different kernels work best. Especially noteworthy are the results of the outlier detection experiment, where the concept of soft-max works reasonably well, as it also did for the soft-max alignment kernels.

For future work it would be interesting to investigate in which kinds of datasets and tasks the performance of the alignment measures breaks down and where the more precise approach of computing the integral gives better results. In the same vein it would be interesting
Table 13: Precision@39 for different integral over time kernels.

<table>
<thead>
<tr>
<th>$\lambda = \frac{1}{2}$</th>
<th>$\lambda = \frac{1}{2}$</th>
<th>$\lambda = \frac{1}{2}$</th>
<th>$\lambda = 1$</th>
<th>$\lambda = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma = \frac{1}{4}$</td>
<td>$\gamma = 1$</td>
<td>$\gamma = \frac{1}{2}$</td>
<td>$\gamma = 1$</td>
<td>$\gamma = \frac{1}{2}$</td>
</tr>
<tr>
<td>$K_{D_{noshift}}$</td>
<td>0.23</td>
<td>0.12</td>
<td>0.12</td>
<td>0.13</td>
</tr>
<tr>
<td>$K_{D_{shift}}$</td>
<td>0.17</td>
<td>0.16</td>
<td>0.18</td>
<td>0.16</td>
</tr>
<tr>
<td>$K_{D_{sub,noshift}}$</td>
<td>0.16</td>
<td>0.16</td>
<td>0.15</td>
<td>0.16</td>
</tr>
</tbody>
</table>

| $K_{noshift}$            | best ($\beta = 256$)   | 0.49                    | 0.46          | 0.46          | 0.46          | 0.46          |
|                         | mean               | $0.36^{0.49}_{0.25}$   | $0.41^{0.46}_{0.26}$ | $0.40^{0.47}_{0.26}$   | $0.41^{0.46}_{0.29}$   | $0.41^{0.47}_{0.25}$   |
| $K_{samehift}$           | best ($\beta = 1024$) | 0.46                    | 0.47          | 0.46          | 0.46          | 0.46          |
|                         | mean               | $0.36^{0.49}_{0.25}$   | $0.41^{0.46}_{0.26}$ | $0.40^{0.47}_{0.26}$   | $0.41^{0.46}_{0.29}$   | $0.41^{0.47}_{0.25}$   |
| $K_{fixed_{sub}}$        | best ($\beta = 64$)  | 0.38                    | 0.38          | 0.39          | 0.38          | 0.46          |
|                         | mean               | $0.31^{0.39}_{0.16}$   | $0.30^{0.38}_{0.17}$ | $0.33^{0.41}_{0.18}$   | $0.32^{0.42}_{0.18}$   | $0.34^{0.46}_{0.16}$   | $0.34^{0.46}_{0.19}$   |
| $K_{sub,noshift}$        | best ($\beta = 16$)  | 0.49                    | 0.51          | 0.53          | 0.47          | 0.48          |
|                         | mean               | $0.41^{0.59}_{0.28}$   | $0.41^{0.51}_{0.29}$ | $0.42^{0.53}_{0.30}$   | $0.43^{0.55}_{0.35}$   | $0.41^{0.48}_{0.31}$   | $0.43^{0.51}_{0.33}$   |
| $K_{sub,shift}$          | best ($\beta = 64$)  | 0.38                    | 0.38          | 0.38          | 0.39          | 0.41          | 0.45          |
|                         | mean               | $0.30^{0.39}_{0.17}$   | $0.30^{0.38}_{0.17}$ | $0.31^{0.39}_{0.18}$   | $0.33^{0.42}_{0.18}$   | $0.34^{0.41}_{0.18}$   | $0.35^{0.45}_{0.22}$   |
| $K_{fixed_{sub}}$        | best ($\beta = 4000$) | 0.37                    | 0.38          | 0.38          | 0.39          | 0.39          | 0.42          |
|                         | mean               | $0.30^{0.38}_{0.17}$   | $0.31^{0.39}_{0.18}$ | $0.31^{0.39}_{0.18}$   | $0.31^{0.39}_{0.17}$   | $0.31^{0.40}_{0.17}$   | $0.31^{0.42}_{0.17}$   |
| $K_{sub,noshift}$        | best ($\beta = 1024$) | 0.49                    | 0.50          | 0.47          | 0.49          | 0.51          | 0.47          |
|                         | mean               | $0.42^{0.49}_{0.34}$   | $0.42^{0.5}_{0.33}$ | $0.42^{0.5}_{0.33}$   | $0.42^{0.5}_{0.33}$   | $0.43^{0.51}_{0.33}$   | $0.42^{0.49}_{0.33}$   |
| $K_{sub,shift}$          | best ($\beta = 4000$) | 0.38                    | 0.38          | 0.38          | 0.38          | 0.40          | 0.43          |
|                         | mean               | $0.31^{0.38}_{0.18}$   | $0.31^{0.38}_{0.18}$ | $0.31^{0.39}_{0.17}$   | $0.31^{0.41}_{0.17}$   | $0.31^{0.41}_{0.17}$   | $0.32^{0.43}_{0.17}$   |
to research what tasks and datasets are advantageous for the positive semi-definite kernels, compared to the non-PSD kernels.
This chapter is an extension of the work described by de Vries et al. (2010).

In this chapter we investigate similarity measures that combine low-level trajectory information with geographical domain knowledge to compare vessel trajectories. These similarity measures are largely based on the alignment techniques of Chapter 4. In a clustering experiment we show how these measures can be used to discover behavior concepts in vessel trajectories that are dependent both on the low-level trajectories and the domain knowledge. We also apply these measures in a classification and outlier detection task. In the classification experiment we show that significantly better classification accuracy can be achieved by combining trajectory information and geographical domain knowledge. The outlier detection experiment shows no significant increase in performance due to the added domain knowledge.

6.1 INTRODUCTION

The vessel trajectories that we are concerned with in this thesis, as well as other types of moving object trajectories, exist in spaces where places and regions have semantics of their own. In the case of vessels there are concepts such as anchoring areas, sea lanes and harbors. If we add this information to the trajectories and incorporate it in the similarity measure, then we can potentially discover more complex and interesting behavior patterns.

A simple example of such a pattern is the trajectories of large cargo ships and tankers that use a shipping lane to sail north. Using the shipping lane information the trajectories of these vessels can be discriminated from other vessels moving in that direction, such as the fishing vessels that do not use a shipping lane to go fishing up north. Also different types of ports, e.g. petrol docks and cargo terminals, can help us discriminate between the trajectories of tankers and those of cargo vessels, which look really similar in terms of movement.

In this chapter we present alignment based similarity measures that combine low-level vessel trajectories with geographical domain knowledge, such as the name and type of the regions that vessels pass through and where they stop. As a basis for this we use the alignment kernels from Chapter 4, because they outperform the measures in Chapter 5 and their integration of domain knowledge is also more intuitive. We use the similarity measures in a clustering, classification and outlier detection task. The clustering experiment shows
adding domain knowledge

a number of interesting vessel behaviors that are discovered based on
the combination of trajectories and geographical domain knowledge.
In the classification task we try to predict the type of the vessel. The
combination of trajectories and domain knowledge gives the best av-
erage classification accuracy. Integration of domain knowledge can
lead to a similar performance in outlier detection as using only trajec-
tory data, but it does not significantly increase performance.

Our work has similarities with the work by Andrienko et al. (2007),
where the authors use a series of clusterings with different similarity
measures to discover interesting movement patterns. In this chapter
we incorporate more domain knowledge and also provide the com-
bination of trajectory and domain knowledge as one measure where
weights can be used to determine the influence of each component.

The rest of this chapter is organized as follows. In Section 6.2 we
will describe the geographical domain knowledge that we used and
how we enriched trajectories with this knowledge. Our similarity
measure for low-level trajectories labeled with geographical information is described in Section 6.3. The clustering, classification and out-
lier detection experiments are presented in Section 6.4. We end with
some conclusions and suggestions for future work.

6.2 GEOGRAPHICAL DOMAIN KNOWLEDGE

In this section we introduce the geographical domain knowledge that
we use and how we enrich vessel trajectories with this knowledge.

6.2.1 Domain Ontologies

Our geographical domain knowledge comes in the form of two sim-
ple ontologies. Both ontologies are stored as RDF; see Section 3.4 for
more details. One ontology contains the definition of different an-
chorages, clearways and other areas at sea, which we call Anchorages-
AndClearways. All of these geographical features were converted to
RDF from shape files from Rijkswaterstaat (RWS), part of the Nether-
lands Ministry of Transport, Public Works and Water Management.
The other ontology has definitions for different types of harbors, such
as liquid bulk and general cargo (containers), which we call Harbors.
All harbors were manually copied from the harbor branches map of
the Port of Rotterdam Authority. The concepts in these ontologies
have a unique identifier, are assigned polygon regions, and have a
1 http://www.portofrotterdam.com/en/Port/port-maps/Pages/branches.aspx
type. The different concepts in these ontologies are illustrated in Fig-
ure 12.

The modeling of the concepts follows the GeoNames ontology, with
the exception of the positioning properties (wgs84:{lat|long}) and the
type property (geo:featureCode). GeoNames specifies feature types with
Figure 12: Visualization of the geographical domain knowledge in KML. All the clearways and approach areas (translucent, A), anchorages (dark red, B), restriction zones (dark blue, C), and separation zones (yellow, D) are shown in the picture on the left. The various harbor types and the deep water lane (light blue, E) for large vessels are shown in the picture on the right, which corresponds to the rectangular area outlined with a white line in the picture on the left.
Figure 13: Example of the RDF representation of the geographical domain knowledge using a combination of the GeoNames ontology and GeoRSS. This figure shows a dry bulk cargo harbor.

the geo:featureClass property for general classes, like geo:P for populated feature types and geo:H for hydrographical feature types. Specific types, like geo:H.HBR for harbors, are specified with the geo:featureCode property. For our experiments we require more specific types than just harbor, e.g. dry bulk harbor. We assigned these specific types as extra types to the features. The specific types are modeled as subclasses of the original geo:featureCodes. To allow RDFS reasoning over the featureCodes and their new subclasses we temporarily asserted that geo:featureCode is a subproperty of rdf:type, which makes each featureCode an rdfs:Class containing all the features of that type as instances. GeoNames uses WGS84 latitude longitude coordinates, while we use polygons of WGS84 coordinates which we express in the GeoRSS Simple vocabulary.

The polygons that define the different regions can be overlapping. For example, an anchorage area can overlap with a harbor approach. Each of the harbor regions is assigned a polygon demarcating the land area of the harbor (the port) and not the part of the water (the dock), because the same dock can be shared by two ports of different types. For instance, there can be container cranes on one side of the basin and oil valves on the other. This is not the case for harbors found in GeoNames, because these are located by points in the middle of the dock. An example of the representation of a harbor with a specific type and polygon shape can be found in Figure 13.

We have created two web services to enrich trajectories with geographical features. One of these services, NearestHarbor, matches a latitude, longitude point to the nearest harbor in Harbors within a predetermined range. The actual range used in our experiments will be discussed in Section 6.4. The label and most specific type of this harbor is then returned, e.g. ‘place_DryBulk4’ and ‘ptype_DryBulkHarbor’. Similarly, the other service, Intersection, returns a set of (label, most
specific type) pairs corresponding to the regions in *AnchoragesAndClearways* that intersect with a given point. Both web services were implemented in SWI-Prolog using the Space package (van Hage et al. (2010)).

We represent the geographical domain knowledge in RDF using the GeoNames and GeoRSS ontologies. Besides these ontologies, the SWI-Prolog Space package also supports other geolocation representations in RDF, like those used by DBpedia, Freebase, and LinkedGeoData (OpenStreetMap). Therefore the services that match the trajectories to geographical features could just as well use RDF from these sources by directly loading them over the web. The main reason to use RWS sea maps and manually converted Port of Rotterdam harbor types is that currently there are hardly any maritime polygons to be found on the web. Moreover, the maritime features that do exist are not of the suitable level of abstraction. In the case of GeoNames, the lowest level of abstraction is often not low enough, as discussed before, while in LinkedGeoData the existing levels of abstractions are too low or inappropriate. For example, harbors are categorized as leisure areas for angling, and each separate trash bin is listed as such.

### 6.2.2 Enriching Trajectories

In this chapter we consider trajectories $T$ as sequences of points, as defined in Definition 2.2.1. The trajectories are taken from the datasets described in Section 2.3.1. These trajectories have already been cut at the stops and are therefore all moves. They are delimited by the vessel entering the area of observation or starting (from being stopped) and the vessel leaving the area of observation or stopping.

Using the Intersection service we create a sequence of sets of geo-labels $T_{lab}$ as defined in Definition 6.2.1.

**Definition 6.2.1.** A sequence of sets of geo-labels $T_{lab}$ for a trajectory $T$ is defined as: $T_{lab} = L_1, \ldots, L_{|T|}$, where a set of geo-labels $L_i = \{(name_i, type_i) \ldots, (name_m, type_m)\}$. Each $L_i$ is a set of pairs where name$_i$ is the label of the region such that $T(i)$ is contained in the polygon that defines that region and type$_i$ is the type of that region. Let $T_{lab}(i) = L_i$.

Note that $L_i$ is a set because a point can be in multiple regions. Furthermore, $L_i$ can be empty since the defined regions do not cover everything.

Next to a trajectory as a sequence of sets of geo-labels $T_{lab}$ as a separate object, we also define the combination of the sequence of points $T$ and $T_{lab}$ as $T_{traj,lab}$ in Definition 6.2.2.

---

3 [http://dbpedia.org](http://dbpedia.org)

4 [http://freebase.com](http://freebase.com)

5 [http://linkedgeodata.org](http://linkedgeodata.org)
Definition 6.2.2. The combination of a trajectory $T$ and its corresponding sequence of sets of geo-labels $T_{lab}$ is defined as $T_{traj\_lab} = (T(1), T_{lab}(1)), \ldots, (T(n), T_{lab}(n))$. Let $T_{traj\_lab}(i) = (T(i), T_{lab}(i))$.

We treat the start $T(1)$ and end $T(|T|)$ of a trajectory with special interest and define special start and end objects for these in Definition 6.2.3.

Definition 6.2.3. For a trajectory $T$ we define the start and end object $T_{start}$ and $T_{end}$ as $T_{start} = (\text{stopped}, L_{start})$ and $T_{end} = (\text{stopped}, L_{end})$. stopped is a boolean value indicating whether the vessel is stopped. $L_{start}$, respectively $L_{end}$, is a set of pairs (name, type).

To add domain knowledge about whether a vessel is docked at a port we use the NearestHarbor service to find the geographically closest harbor (name, type) to the point $T(1)$, respectively $T(|T|)$. If this service returns a harbor and the vessel is also stopped, then we put this pair in $L_{start}$, respectively $L_{end}$. If the vessel is stopped but there is no harbor close, then we use the Intersection service to find the regions that $T(1)$, respectively $T(|T|)$, is in, and add those to $L_{start}$, respectively $L_{end}$. We do the same if the vessel is not stopped. Thus, we are interested in harbors if a vessel is docked there, where docked is defined as being close to that harbor and stopped. We could have also added the start and stop harbors to the sequence of geo-labels $T_{lab}$. However, by treating these separately we have more flexibility in weighing the importance of the start and stop places.

So, for each trajectory we have five objects, the trajectory itself, $T$, a sequence of sets of geo-labels, $T_{lab}$, the combination of $T$ and $T_{lab}$ into $T_{traj\_lab}$, and start and end information, $T_{start}$ and $T_{end}$.

The above labeling process has similarities with the work by Alvares et al. (2007). However, we label not only the stops (or starts and ends in our terminology), but also the moves. Furthermore we use RDF based web services instead of a geographic database.

6.3 Enriched Trajectory Kernels

Like trajectories $T$, sequences of sets of geo-labels $T_{lab}$ and their combinations $T_{traj\_lab}$ can be compared using the same alignment methods as those used in Chapter 4. However, they require different substitution functions, which we will define below. After these definitions we look into how we create kernels for the different objects defined above. An overview of the kernels defined in this chapter is given in Table 14.
Table 14: List of kernels defined in this chapter.

<table>
<thead>
<tr>
<th>Shorthand</th>
<th>Parameters</th>
<th>Description</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{\text{start}}$</td>
<td>none</td>
<td>Kernel for start of trajectory domain knowledge</td>
<td>6.3.3</td>
</tr>
<tr>
<td>$K_{\text{end}}$</td>
<td>none</td>
<td>Kernel for end of trajectory domain knowledge</td>
<td>6.3.3</td>
</tr>
<tr>
<td>$K_{\text{lab}}$</td>
<td>$\beta, g$</td>
<td>Kernels for sequences of sets of geo-labels</td>
<td>6.3.4</td>
</tr>
<tr>
<td>$K_{\text{traj,lab}}$</td>
<td>$\beta, g, w_1, w_2$</td>
<td>Kernels for trajectories combined with sets of geo-labels</td>
<td>6.3.5</td>
</tr>
<tr>
<td>$K_{\text{all}}_1$</td>
<td>$w_1, w_2, w_3, w_4$</td>
<td>Kernel combining $K_{\text{traj}}$, $K_{\text{lab}}$, $K_{\text{start}}$ and $K_{\text{end}}$</td>
<td>6.3.6</td>
</tr>
<tr>
<td>$K_{\text{all}}_2$</td>
<td>$w_1, w_2, w_3, w_4, w_5$</td>
<td>Kernel combining $K_{\text{traj,lab}}$, $K_{\text{start}}$ and $K_{\text{end}}$</td>
<td>6.3.7</td>
</tr>
</tbody>
</table>

6.3.1 Substitution Functions for Sets of Geo-Labels

For sequences of sets of geo-labels $T_{\text{lab}}$, the substitution function $\text{sub}_{\text{lab}}$, given in Definition 6.3.1, expresses how many labels two sets of labels $L_i$ and $L_j$ have in common.

**Definition 6.3.1.** For two sets of labels $L_i, L_j$, we define the substitution function $\text{sub}_{\text{lab}}$ as:

$$\text{sub}_{\text{lab}}(L_i, L_j) = -1 + \frac{|N_i \cap N_j| + |M_i \cap M_j|}{\sqrt{|N_i| + |M_i|)(|N_j| + |M_j|)},$$

where $N_m = \{l_1 \mid l \in L_m\}$ and $M_m = \{l_2 \mid l \in L_m\}$, for $m = i, j$.

Note that $l_1$ indicates the first element of the (name, type) pair, i.e. the name, and $l_2$ the second element, i.e. the type. Thus, we count the number of names and type labels that both sets have in common and divide this by the square root of the multiplied lengths of the sets. We add $-1$ to create a function that ranges from $0$ to $-1$, which is $0$ when two objects are equal, similar to the negative of the Euclidean distance (i.e. $\text{sub}_1$). We use this range since the alignment measures in Chapter 4 are defined using the max function.

The function $\text{sub}_{\text{lab}}$ can be expressed as the dot products of vectors that have a length equal to the total number of possible name and
type labels. In this vector we use the value 1 if a particular label is in the set, and 0 otherwise. Let \( n \) be this vector representation of \( L_i \) and \( m \) the vector representation of \( L_j \). Then Definition 6.3.1 can be represented as:

\[
\text{sub}_{\text{lab}}(n, m) = -1 + \frac{n \cdot m}{\sqrt{n \cdot n \cdot m \cdot m}}.
\]

This is a positive semi-definite kernel, where \( \sqrt{n \cdot n \cdot m \cdot m} \) is the regular kernel normalization. This means that \( \exp(\text{sub}_{\text{lab}}) \), which is used in the soft-max kernels, is also a positive definite kernel. Besides a substitution function we require a gap penalty if we use edit distance, we fix this to the minimum value of \( \text{sub}_{\text{lab}} \), which is \(-1\).

We can combine the substitution functions for position, \( \text{sub}_1 \) and \( \text{sub}_2 \), with the above substitution function for sets of labels such that we can do alignments on the combined trajectories of Definition 6.2.2. This leads to Definition 6.3.2.

**Definition 6.3.2.** Let \( S_{\text{traj}_{\text{lab}}} \) and \( T_{\text{traj}_{\text{lab}}} \) be two combinations of trajectories and sequences of sets of geo-labels. Furthermore, \((S_i, L_i) = S_{\text{traj}_{\text{lab}}}(i) \) and \((T_j, L_j) = T_{\text{traj}_{\text{lab}}}(j)\). Then

\[
\text{sub}_{\text{traj}_{\text{lab}}}(w_1, w_2)(S_i, L_i, T_j, L_j) = w_1 \gamma \text{sub}_m(S_i, T_j) + w_2 \text{sub}_{\text{lab}}(L_i, L_j),
\]

where \( m = 1, 2 \).

This function combines the substitution functions for trajectories and for sequences of set of labels into one function, where the index \( m \) determines if \( \text{sub}_1 \) or \( \text{sub}_2 \) from Chapter 4 is used. The weights \( w_1 \) and \( w_2 \) are used to determine the influence of the position and domain knowledge information. We furthermore use the parameter \( \gamma \) to get the position substitution function and the label substitution function on the same scale. This can also be done directly by incorporating \( \gamma \) into \( w_1 \). However, this would result in the weights \( w_1 \) and \( w_2 \) not summing to 1, which we consider to be less clear. We experiment with \( w_1 \) and \( w_2 \) and fix \( \gamma \) to an appropriate value. In the case of the \( \text{sub}_{\text{traj}_{\text{lab}}} \) functions, the gap penalty \( g \) is also \(-1\), since this is the minimum value that these substitution functions are designed to take, which is achieved by tuning the \( \gamma \) parameter.

### 6.3.2 Kernels

We have not discussed similarity or substitution functions for start and end objects yet. This similarity is straightforward; it can immediately be put into kernel form, given in Definition 6.3.3.

**Definition 6.3.3.** For all \((\text{stopped}_i, L_i)\) and \((\text{stopped}_j, L_j)\) in a set of start/end objects, we compute a kernel matrix as:

\[
K_{\phi}(i, j) = 1 + [\text{stopped}_i = \text{stopped}_j] + \text{sub}_{\text{lab}}(L_i, L_j),
\]
where $\phi = \text{start}$ for a set of start objects and $\phi = \text{end}$ for a set of end objects.

Thus the similarity between two start/end objects is determined by whether the vessel is stopped or not and how much labels there are in common. Furthermore we add $+1$ to compensate for the $-1$ in the definition of $\text{sub}_\text{lab}$. Using Definition 6.3.3 we get a kernel $K_{\text{start}}$ for the start objects and a kernel $K_{\text{end}}$ for the end objects. These kernels are positive semi-definite, since the function $\text{sub}_\text{lab}$, the boolean function $[\text{stopped}_i = \text{stopped}_j]$ and the constant $1$ are PSD.

Chapter 4 defines different alignment kernels in which we use the substitution functions defined above to get kernels for the sequence of sets of geo-labels $T_{\text{lab}}$ and the combined version $T_{\text{traj\_lab}}$. To avoid unnecessary reduplication of definitions, we give an abridged definition for kernels for sequences of sets of geo-labels in Definition 6.3.4.

**Definition 6.3.4.** $K_{\text{lab}}$ kernels for sequences of sets of geo-labels $T_{\text{lab}}$ are computed by using the $\text{sub}_\text{lab}$ substitution function in $\text{sim}_{\text{max}}$ (Definition 4.2.7), $\text{sim}_{\text{maxnorm}}$ (Definition 4.2.8) and $\text{sim}_{\text{softmax}}$ (Definition 4.2.9). Then, Definitions 4.2.10 and 4.2.11 are used to compute kernels.

An example of such a kernel is $K_{\text{lab, max, DTW}}$ which is the kernel that uses the regular non normalized dynamic time warping measure applied to the sets of labels sequence.

For the kernels for $T_{\text{traj\_lab}}$ we do almost the same as for the $K_{\text{lab}}$ kernels above; see Definition 6.3.5.

**Definition 6.3.5.** $K_{\text{traj\_lab}}$ kernels for sequences of sets of geo-labels combined with normal trajectories $T_{\text{traj\_lab}}$ are computed by using the $\text{sub}_\text{traj\_lab}$ substitution function in $\text{sim}_{\text{max}}$ (Definition 4.2.7) and $\text{sim}_{\text{maxnorm}}$ (Definition 4.2.8). The $\text{sub}_\text{traj\_lab}$ substitution function is used in $\text{sim}_{\text{softmax}}$ (Definition 4.2.9). Then, Definitions 4.2.10 and 4.2.11 are used to compute kernels. The superscripts $w_1, w_2$ are used for the weights of the substitution function.

In this chapter, we give the subscript $\text{traj}$ to kernels for normal trajectories $T$, i.e. the kernels as studied in Chapter 4.

For explanatory purposes, we work out $\text{sim}_{\text{traj\_lab, softmax, DTW}}^{\beta, w_1, w_2}$ i.e. the similarity function on which $K_{\text{traj\_lab, softmax, DTW}}$ is based:

\[
\text{sim}_{\text{traj\_lab, softmax, DTW}}^{\beta, w_1, w_2}(S_{\text{traj\_lab}}(\pi_1), T_{\text{traj\_lab}}(\pi_2)) = \sum_{\pi \in \Pi(S, T)} \exp(\beta \sum_{i=1}^{||\pi||} \text{sub}_{\text{traj\_lab}}^{w_1, w_2}(S_{\text{traj\_lab}}(\pi_1(i)), T_{\text{traj\_lab}}(\pi_2(i))))
\]

\[
= \sum_{\pi \in \Pi(S, T)} \prod_{i=1}^{||\pi||} \exp(\beta \text{sub}_{\text{traj\_lab}}^{w_1, w_2}(S_{\text{traj\_lab}}(\pi_1(i)), T_{\text{traj\_lab}}(\pi_2(i))))
\]

(6.2)
where (with $S_i$ as shorthand for $S_{\text{traj}\_\text{lab}}(\pi_1(i))$ and $T_i$ as shorthand for $T_{\text{traj}\_\text{lab}}(\pi_2(i))$),

\[
\exp(\beta \sub_{\text{traj}\_\text{lab}}^{w_1,w_2}(S_i,T_i)) = \exp(\beta w_1 \gamma \sub_{\text{lab}}((S_i)_1,(T_i)_1)) \\
+ w_2 \sub_{\text{lab}}((S_i)_2,(T_i)_2)) \\
= \exp(\beta w_1 \gamma \sub_{\text{lab}}((S_i)_1,(T_i)_1)) \\
\cdot \exp(\beta w_2 \sub_{\text{lab}}((S_i)_2,(T_i)_2))
\] (6.3)

Both $\exp(\sub)$ and $\exp(\sub_{\text{lab}})$ are positive semi-definite kernels. Hence we are multiplying PSD kernels, and therefore the use of the substitution function $\sub_{\text{traj}\_\text{lab}}$ leads to a positive semi-definite softmax kernel.

We combine the kernels defined above together by taking weighted sums in two ways. The first way, in Definition 6.3.6, uses separate kernels for the trajectories $T$ and the geo-labels $T_{\text{lab}}$.

**Definition 6.3.6.** Let $K_{\text{traj}}$ be an alignment kernel for trajectories, $K_{\text{lab}}$ an alignment kernel for sequences of sets of geo-labels, and $K_{\text{start}}$ and $K_{\text{end}}$ kernels for start and end objects, then

\[
K_{\text{all,1}} = w_1 K_{\text{traj}} + w_2 K_{\text{lab}} + w_3 K_{\text{start}} + w_4 K_{\text{end}} ,
\]

with $w_1 + w_2 + w_3 + w_4 = 1$.

The other kernel uses the alignment on the trajectories combined with sequences of geo-labels $T_{\text{traj}\_\text{lab}}$, which is given in Definition 6.3.7.

**Definition 6.3.7.** Let $K_{\text{traj}\_\text{lab}}$ be an alignment kernel for trajectories combined with their sequence of sets of geo-labels, and $K_{\text{start}}$ and $K_{\text{end}}$ kernels for start and end objects, then

\[
K_{\text{all,2}} = w_3 K_{\text{traj}\_\text{lab}}^{w_1,w_2} + w_4 K_{\text{start}} + w_5 K_{\text{end}} ,
\]

with $w_1 + w_2 = 1$ and $w_3 + w_4 + w_5 = 1$.

Depending on the alignment kernels that are used for $T$, $T_{\text{lab}}$ and $T_{\text{traj}\_\text{lab}}$ these kernels are positive semi-definite (PSD). The weighted kernels are inspired by work in computational biology on combined kernels for comparing protein sequences and DNA (Cuturi (2010)).

The $K_{\text{start}}$ and $K_{\text{end}}$ sub-kernels are cheap to compute. For $K_{\text{traj}}$, $K_{\text{lab}}$ and $K_{\text{traj}\_\text{lab}}$ we use the dynamic programming approach, mentioned earlier in this thesis.

### 6.4 Experiments

Like we did in the previous two chapters, the goal of our experiments is to investigate the performance of the defined kernels, which incorporate geographical domain knowledge, on the three typical machine learning tasks of clustering, classification and outlier detection. We are interested to see if performance increases over using only trajectory information and under what settings.
6.4 EXPERIMENTS

6.4.1 Experimental Set-Up

In Chapter 4 we saw that for none of the tasks the best performance was achieved in the no compression setting. Therefore, we will not investigate this no compression setting in the experiments here. Furthermore, the number of repeated labels for each sequence of sets of geo-labels $T_{\text{lab}}$ would be very large in this setting, which seems useless. All trajectories are compressed using $\text{pls}_{E_{cn}}\epsilon = 50\text{m}$, which is the parameter setting that achieves the highest performance in the clustering and outlier detection task. Furthermore, visual inspection of some of the labeled trajectories suggests that no regions are missed under this setting in the labeling process. For completeness, in the classification task we will also include the $\epsilon = 1000\text{m}$ setting, since under this setting the best results are obtained in that experiment. We use the time weight setting of $w = 0$; thus time is ignored.

Contrary to the previous two chapters, the datasets used in all experiments are from the same area. In this area the AnchoragesAndClearways ontology contains the names and polygons for approximately 50 regions of 6 different types. There are around 90 different harbors in Harbors that are distinguished into 7 different types. For each trajectory in the respective dataset we created a sequence of sets of geo-labels $T_{\text{lab}}$, the combination with the raw trajectory $T_{\text{traj,lab}}$, a start object $T_{\text{start}}$ and an end object $T_{\text{end}}$. The threshold used in the NearestHarbor service is set to 100m. This threshold range was determined manually and is suitable given the size of the vessels, docks and clearways, and is of a larger order of magnitude than the GPS errors.

The different weight settings for the kernels from Definition 6.3.6 and 6.3.7 that we experiment with are different per task. Hence, we will mention them there. This also holds for the $\beta$ parameter when applicable.

6.4.2 Clustering

The gold standard clustering dataset that we used in the previous two chapters is not well suited to evaluate the performance of clustering using the domain knowledge kernels that we defined in this chapter, since the gold standard is located in an area for which not a lot of geographical domain knowledge exists. We use another dataset of 1917 trajectories here, which contains data from the same area as the classification (Section 2.3.3) dataset. The difference with the classification set is that all vessel types are included. As we mentioned above, for this dataset there is domain knowledge. Since we do not have a gold standard, we cannot evaluate the clustering results quantitatively. Our evaluation is qualitative in nature, and consists of show-
adding three illustrative clustering examples for three different settings of the weights in Definition 6.3.6. After some experimentation, we chose as the $K_{\text{traj}}$ kernel the edit distance variant: $K_{\text{traj,maxnorm,ED}}$, with $g = -0.01$, and as $K_{\text{lab}}$ kernel we took $K_{\text{lab,maxnorm,ED}}$. The first setting for $K_{\text{all}}$, is $w_1 = 1/7, w_2, w_3, w_4 = 1/5$, the resulting kernel being $K_{\text{comb}}$. This setting weighs the trajectory information and the ontological information equally. There is also a setting for just the trajectory information, $w_1 = 1, w_2, w_3, w_4 = 0$, $K_{\text{traj}}$, and one for just the ontological information $w_1 = 0, w_2, w_3, w_4 = 1/3$, $K_{\text{onto}}$. We define these three distinct settings to investigate the effect of using low-level trajectory information and domain knowledge in clustering.

These kernels are used as input for the weighted kernel k-means algorithm of Section 2.1.1.2. For each kernel, clustering is done 100 times with random initializations. This process is repeated 10 times. Because we have no gold standard clustering that we wish to achieve or a specific criterion that we want to optimize, it is difficult to determine a good value for the number of clusters parameter $k$. Therefore, we manually experimented with different values and finally selected $k = 40$, which gave cluster examples that show the differences between the three kernels well. We could have also used a clustering algorithm that has no parameter $k$, e.g. density based ones are popular in combination with moving object trajectories. However, these algorithms have other parameters that need to be determined, which also is a manual process. Moreover, we are not sure that the combined kernel that we have defined induces a feature space in which density based algorithms work well.

**Examples**

The three examples we give below illustrate behavior clusters of vessels that arise in the combined setting, i.e. using kernel $K_{\text{comb}}$. For each example we will also show the clusters from the other two settings ($K_{\text{traj}}$ and $K_{\text{onto}}$) that resemble the behavior the most. All figures show the trajectories in one cluster in black against a background of all trajectories in gray. For the trajectories in a cluster, the start of a trajectory is indicated by a dot and the end by an asterisk.

Figure 14 illustrates the behavior of vessels anchoring in a specified anchoring area. In Figure 14A we show a cluster resulting from the combined information kernel $K_{\text{comb}}$. We see that all the tracks end up in one anchoring area. If we use only the trajectory information, i.e. $K_{\text{traj}}$, we get the result in Figure 14B. In this case there are a number of other trajectories that do not end in the anchoring area. For the clustering with only the ontological information, $K_{\text{onto}}$, we see something different (Figure 14C). Here there is another track of a vessel anchoring in another anchoring area. So, the combination of trajectory and ontological information results in the discovery of the behavior “anchoring in a specific anchoring area”.
The cluster in Figure 15A shows the docking behavior of vessels in a certain part of the harbor. There is some noise in the cluster; not all trajectories go to that part. This cluster is a result of clustering with the combined kernel $K_{\text{comb}}$. We see something similar in Figure 15B. However, here we have used the kernel $K_{\text{traj}}$, which uses only the trajectory information. The result is that the cluster also contains trajectories starting from anchoring areas. This differs from the combined setting, where we only have trajectories coming from outside the observation area, i.e. the open sea. In Figure 15C, the ontology only setting, $K_{\text{onto}}$, we also have trajectories going to the harbor from outside the observation area, but, all trajectories stop in the deep water lane, not on a dock. This is somewhat odd and is the result of only considering ontological information. In the combined case we have stopped in the deep water lane, but also in the adjacent docks. Thus, the combined case shows the behavior of “docking in a certain part of the harbor, coming directly from the open sea”.

The trajectories in Figure 16A, on which we zoom in in Figure 16B, are a result of clustering with the combined kernel. The figures show trajectories that do not stop and continue on the river to the land behind. These vessels are smaller, and in Figure 16B we see that none of them pass through the deep water lane. Figure 16C is a cluster from clustering with the trajectory only kernel $K_{\text{traj}}$. The trajectories in this cluster both stop and go on, and some of them go through the deep water lane and some of them do not. Because no domain knowledge is used the deep water lane and non-deep water lane trajectories are difficult to separate, since they do not differ much in shape. The comparable cluster for the $K_{\text{onto}}$ kernel, Figure 16D, shows trajectories that go in different directions and some noise. Using only domain knowledge does not guarantee that trajectories that go in different directions are not clustered. The combined kernel discovers the be-
Adding domain knowledge

Figure 15: Example of a cluster of trajectories showing docking behavior. The start of a trajectory is indicated by a dot, the end by an asterisk. Figure A shows a cluster generated with the combined kernel. Figure B shows the most similar cluster from clustering with the trajectory information only kernel, and figure C shows the most similar cluster from clustering with the domain knowledge only kernel.

behavior of “smaller ships coming from sea and continuing directly to the land behind”.

The above examples show that a combination of low-level trajectory information and geographical domain knowledge in one similarity measure can lead to the discovery of interesting vessel behavior patterns that are indeed due to a combination of these two information sources.

6.4.3 **Classification**

In the classification task we use the dataset from Section 2.3.3, which contains trajectories of the four most common vessel types. From this set we randomly select 200 trajectories for each type, creating a 800 trajectories dataset. Note that this set is larger than the one we used in Chapter 4. Due to the fact that we do not have a ‘no compression’ setting, we can easily run the experiments with more data.

The first stage of the experiments is to discover the best performance settings for each of the basic kernels: $K_{lab}$ and $K_{traj\_lab}$. For $K_{traj}$ we already know these settings from Chapter 4. The kernels $K_{start}$ and $K_{end}$ do not have any settings.

For $K_{lab}$ we test the different variants of dynamic time warping and edit distance. The soft-max kernels parameter $\beta$ is varied over: $\frac{1}{1024}, \frac{1}{256}, \frac{1}{64}, \frac{1}{16}, \frac{1}{4}, 1$.

Besides the parameters of the different variants of DTW and edit distance, the $K_{traj\_lab}$ kernel has the parameters $w_1$ and $w_2$, for which we test the combinations: $(0.9, 0.1), (0.7, 0.3), (0.5, 0.5), (0.3, 0.7)$ and $(0.1, 0.9)$. The $\gamma$ parameter is set to 100, which is the ratio between the best gap penalty ($-0.01$) for the $K_{traj}$ kernel and the gap penalty ($-1$) in the $K_{lab}$ kernel. Furthermore, for the soft-max alignments we vary $\beta$ over: $\frac{1}{1024}, \frac{1}{256}, \frac{1}{64}, \frac{1}{16}, \frac{1}{4}, 1$. 

Figure 16: Example of a cluster of trajectories continuing through the harbor, not going through the deep water lane. The start of a trajectory is indicated by a dot, the end by an asterisk. Figures A and B shows a cluster generated with the combined kernel. Figure C shows the most similar cluster from clustering with the trajectory information only kernel, and figure D shows the most similar cluster from clustering with the domain knowledge only kernel.
With the best performing basic kernels we create variants of \( K_{\text{all,1}} \) and \( K_{\text{all,2}} \) with different weight settings, to cover different combinations of the basic kernels. We give the specific settings in the Results section.

The classification algorithm set-up is the same as in the classification experiment of Chapter 4, we use the C-SVC Support Vector Machines from LibSVM; see Section 2.1.1.1. We use a 10-fold cross validation set-up to evaluate the kernel performance, with inner 10-fold cross validation to optimize the \( C \) parameter.

## 6.4.3.1 Results

In the results we present the mean classification accuracy over 10 folds. To statistically compare accuracies for two kernels we use a two-tailed paired t-test with \( p < 0.05 \). The tables give results for two different compression settings: \( \epsilon = 50,1000 \text{m} \). If a kernel has parameter settings we give the results for the best parameter settings and also give the mean, maximum and minimum accuracies.

Table 15 presents the results for the different basic kernels. For reference we give the performance of the best \( K_{\text{traj}} \) kernels. The performance of the \( K_{\text{start}} \) and \( K_{\text{end}} \) kernels is almost the same, and obviously the same for both \( \epsilon \) settings.

The \( K_{\text{lab}} \) kernels are relatively close together in performance, with the DTW max and normalized max performing the worst. The best performance overall (60.5\%) is for the \( K_{\text{lab,softmax,ED}} \) kernel, this however does not differ significantly from the best performing DTW kernel (60.13\%).

In case of the \( K_{\text{traj,lab}} \) kernels, the best performance (75.25\%) is clearly for the \( K_{\text{traj,lab,maxnorm,ED}} \) kernel. It does not differ significantly from the best \( K_{\text{traj}} \) score (75.63\%). Also, there is no significant difference from the score (73.38\%) for the \( K_{\text{traj,lab,max,ED}} \) kernel. However, there is a significant difference with the best DTW variant score (65.25\%) and the best edit distance soft-max score (67.38\%).

Based on Table 15 we take the kernel \( K_{\text{lab,softmax,ED}} \) for the sequences of geo-labels, with \( \beta = \frac{1}{16} \). We combine this kernel with \( K_{\text{traj,maxnorm,ED}} \) with \( g = -0.01 \) and the start and end object kernels to create a kernel incorporating domain knowledge using Definition 6.3.6.

The results for different weight settings are shown in Table 16. For each score we indicate whether there is a significant difference with the accuracy achieved by the \( K_{\text{traj}} \) kernel for that \( \epsilon \) setting. A \( ^{+} \) indicates a significant positive difference, whereas a \( ^{-} \) indicates a significant negative difference. We see that for both \( \epsilon \) settings there are weight settings that significantly outperform the \( K_{\text{traj}} \) kernel.

From Table 15 we see that \( K_{\text{traj,lab,maxnorm,ED}} \) has the best performance of the \( K_{\text{traj,lab}} \) kernels. Thus we take this kernel with its best settings and combine it with the start and end object kernels, as in Definition 6.3.7. Results are shown in Table 17. The combined kernels
| Table 15: Mean classification accuracy (%) for different basic kernels. |
|------------------|------------------|
| $K_{\text{traj, maxnorm, ED}}$ | $K_{\text{start}}$ |
| $\epsilon = 50$ | $\epsilon = 1000$ |
| 73.50 | 75.63 |
| 63.63 | 63.63 |
| 62.88 | 62.88 |
| $K_{\text{lab, max, DTW}}$ | $K_{\text{end}}$ |
| 45.38 | 44.75 |
| 50.13 | 51.0 |
| $K_{\text{lab, softmax, DTW}}$ | $K_{\text{lab, max, maxnorm, DTW}}$ |
| best ($\beta = \frac{1}{4}$) | 60.13 |
| mean$^{\max}$ | 59.38 |
| 58.67 & 58.73 |
| 59.26 & 59.38 |
| $K_{\text{lab, max, ED}}$ | $K_{\text{lab, maxnorm, ED}}$ |
| 52.75 | 56.13 |
| 58.75 | 57.75 |
| $K_{\text{lab, softmax, ED}}$ | $K_{\text{lab, max, maxnorm, ED}}$ |
| best ($\beta = \frac{1}{4}$) | 58.25 |
| mean$^{\max}$ | 60.5 |
| 58.52 & 58.45 |
| 60.13 & 59.75 |
| $K_{\text{traj, lab, max, DTW}}$ | $K_{\text{traj, lab, maxnorm, DTW}}$ |
| best ($w_1 = 0.9, w_2 = 0.1$) | 45.88 |
| mean$^{\max}$ | 52.0 |
| 46.85 & 45.88 |
| 51.65 & 51.25 |
| $K_{\text{traj, lab, softmax, DTW}}$ | $K_{\text{traj, lab, max, maxnorm, DTW}}$ |
| best ($w_1 = 0.5, w_2 = 0.5$) | 54.38 |
| mean$^{\max}$ | 57.0 |
| 54.45 & 54.25 |
| 55.75 & 54.0 |
| $K_{\text{traj, lab, softmax, DTW}}$ | $K_{\text{traj, lab, max, maxnorm, DTW}}$ |
| best ($w_1 = 0.5, w_2 = 0.5, \beta = \frac{1}{4}$) | 64.88 |
| mean$^{\max}$ | 65.25 |
| 62.14 & 59.13 |
| 62.26 & 58.88 |
| $K_{\text{traj, lab, max, ED}}$ | $K_{\text{traj, lab, maxnorm, ED}}$ |
| best ($w_1 = 0.7, w_2 = 0.3$) | 73.38 |
| mean$^{\max}$ | 72.63 |
| 71.37 & 73.38 |
| 71.15 & 70.88 |
| $K_{\text{traj, lab, softmax, ED}}$ | $K_{\text{traj, lab, maxnorm, ED}}$ |
| best ($w_1 = 0.7, w_2 = 0.3$) | 74.5 |
| mean$^{\max}$ | 75.25 |
| 72.17 & 74.5 |
| 73.87 & 71.38 |
| $K_{\text{traj, lab, softmax, ED}}$ | $K_{\text{traj, lab, maxnorm, ED}}$ |
| best ($w_1 = 0.9, w_2 = 0.1, \beta = 1$) | 59.38 |
| mean$^{\max}$ | 67.38 |
| 61.63 & 64.88 |
| 62.59 & 58.88 |
Table 16: Mean classification accuracy (%) for different $K_{all_1}$ kernels.

<table>
<thead>
<tr>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
<th>$w_4$</th>
<th>$\epsilon = 50$</th>
<th>$\epsilon = 1000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>73.50</td>
<td>75.63</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>58.25</td>
<td>60.5</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>63.63$^+$</td>
<td>63.63$^-$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>62.88$^-$</td>
<td>62.88$^-$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>1/2</td>
<td>71.0</td>
<td>71.0</td>
</tr>
<tr>
<td>0</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td>73.38</td>
<td>72.63</td>
</tr>
<tr>
<td>1/2</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td>73.63</td>
<td>73.38$^-$</td>
</tr>
<tr>
<td>7/10</td>
<td>3/10</td>
<td>0</td>
<td>0</td>
<td>74.0</td>
<td>73.88</td>
</tr>
<tr>
<td>9/10</td>
<td>1/10</td>
<td>0</td>
<td>0</td>
<td>74.13</td>
<td>74.63</td>
</tr>
<tr>
<td>1/3</td>
<td>0</td>
<td>1/3</td>
<td>1/3</td>
<td>75.88</td>
<td>77.25</td>
</tr>
<tr>
<td>1/2</td>
<td>0</td>
<td>1/4</td>
<td>1/4</td>
<td>76.50$^+$</td>
<td>78.38$^+$</td>
</tr>
<tr>
<td>3/4</td>
<td>0</td>
<td>1/2</td>
<td>1/2</td>
<td>76.13$^+$</td>
<td>78.75$^+$</td>
</tr>
<tr>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>76.38$^+$</td>
<td>77.0</td>
</tr>
<tr>
<td>1/2</td>
<td>1/6</td>
<td>1/6</td>
<td>1/6</td>
<td>76.88$^+$</td>
<td>78.25</td>
</tr>
<tr>
<td>2/3</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td>75.63</td>
<td>77.88</td>
</tr>
</tbody>
</table>

Table 17: Mean classification accuracy (%) for different $K_{all_2}$ kernels.

<table>
<thead>
<tr>
<th>$w_3$</th>
<th>$w_4$</th>
<th>$w_5$</th>
<th>$\epsilon = 50$</th>
<th>$\epsilon = 1000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>74.5</td>
<td>75.25</td>
</tr>
<tr>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td>76.63</td>
<td>77.38</td>
</tr>
<tr>
<td>1/2</td>
<td>1/4</td>
<td>1/4</td>
<td>76.63$^+$</td>
<td>77.38</td>
</tr>
<tr>
<td>3/4</td>
<td>1/2</td>
<td>1/2</td>
<td>75.38</td>
<td>77.38</td>
</tr>
<tr>
<td>3/4</td>
<td>1/2</td>
<td>1/2</td>
<td>75.12</td>
<td>77.13</td>
</tr>
</tbody>
</table>

show better performance than the $K_{traj}$ kernel. However, the difference is only significant for one setting.

6.4.3.2 Discussion

In the results above we see that kernels combining geographical domain knowledge and regular trajectory information can achieve significantly better classification accuracy than kernels that only consider the regular trajectory information. The highest performance increase is somewhat upward of 3%. We also see that kernels that only consider domain knowledge are outperformed by kernels that only consider trajectory information.

We saw earlier in Chapter 4 that the start and end of a trajectory already contain a lot of information about the type of vessel and we see the same here. The kernel that just takes this type of information
into account \((w_3, w_4 = \frac{1}{2})\) achieves a score of 71.0\%. We also see that the best performing kernel type for classification with only trajectory information, the regular normalized max edit distance kernels, also performs well for the trajectories combined with domain knowledge \(T_{\text{traj,lab}}\) and almost the best for the sequences of geo-labels \(T_{\text{lab}}\). Thus, for the three types of data used here, edit distance seems to be a good choice for classification.

### 6.4.4 Outlier Detection

For our outlier detection experiment we use the dataset from Section 2.3.4, which we also used in the previous two chapters. This set consists of 786 trajectories, 39 of which are outliers.

Like we did in the classification task, the first stage of the experiment is to discover the best performing basic kernels. We already know from Chapter 4 that the best performing \(K_{\text{traj}}\) kernels are the DTW soft-max variants. For the other kernels that have parameters, we use the same settings as in the classification experiment. The weights \(w_1\) and \(w_2\) are varied over the combinations: \((0.9, 0.1), (0.7, 0.3), (0.5, 0.5), (0.3, 0.7)\) and \((0.1, 0.9)\). And for the \(\beta\) parameter we take the settings: \(\frac{1}{1024}, \frac{1}{256}, \frac{1}{64}, \frac{1}{16}, \frac{1}{4}\) and 1.

Using the best performing kernels we create variants of \(K_{\text{all}_1}\) and \(K_{\text{all}_2}\), covering different combinations of the basic kernels and hence different combinations of domain knowledge.

The algorithmic set-up for the outlier detection experiment is the same as for the experiments in Chapter 4 and 5. We use a one-class Support Vector Machine. The 747 normal trajectories in the dataset are randomly split into a training set of \(\frac{2}{3}\) and a test set of the rest, to which we add the outlying trajectories. This split is done 10 times per kernel, and for each split the one-class SVM is optimized on the train set using 10-fold cross validation.

#### 6.4.4.1 Results

We present the mean precision@39 results over 10 folds. We statistically compare the results for two kernels using a two-tailed Student t-test with \(p < 0.05\). The presentation of the results follows the same set-up as before.

In Table 18 we see the results for the different basic kernels. The best performance (0.82) is achieved by the \(K_{\text{traj,lab,softmax,DTW}}\) kernel, which does not differ significantly from the best performance (0.80) for the \(K_{\text{traj,softmax,DTW}}\) kernel. It does differ significantly from the best score (0.76) for the \(K_{\text{traj,lab,ED}}\) kernels and the best \(K_{\text{lab}}\) scores. Moreover, for all the \(K_{\text{lab}}\) and \(K_{\text{traj,lab}}\) kernels, the soft-max kernels achieve a significantly better score than the non soft-max kernels.
Table 18: Mean precision@39 for different basic kernels.

<table>
<thead>
<tr>
<th>Kernel Description</th>
<th>$\epsilon = 50$</th>
<th>$\epsilon = 50$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{traj,softmax,DTW}$ best ($\beta = 64$)</td>
<td>0.80</td>
<td></td>
</tr>
<tr>
<td>$K_{start}$</td>
<td>0.08</td>
<td></td>
</tr>
<tr>
<td>$K_{end}$</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td>$K_{lab,max,DTW}$</td>
<td>0.29</td>
<td></td>
</tr>
<tr>
<td>$K_{lab,maxnorm,DTW}$</td>
<td>0.22</td>
<td></td>
</tr>
<tr>
<td>$K_{lab,softmax,DTW}$ best ($\beta = 1$)</td>
<td>0.67</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.52</td>
<td>0.44</td>
</tr>
<tr>
<td>$K_{lab,max,ED}$</td>
<td>0.29</td>
<td></td>
</tr>
<tr>
<td>$K_{lab,maxnorm,ED}$</td>
<td>0.26</td>
<td></td>
</tr>
<tr>
<td>$K_{lab,softmax,ED}$ best ($\beta = 1$)</td>
<td>0.61</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.48</td>
<td>0.42</td>
</tr>
<tr>
<td>$K_{traj_lab,max,DTW}$ best ($w_1 = 0.9, w_2 = 0.1$)</td>
<td>0.41</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.38</td>
<td>0.39</td>
</tr>
<tr>
<td>$K_{traj_lab,maxnorm,DTW}$ best ($w_1 = 0.1, w_2 = 0.9$)</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.19</td>
<td>0.18</td>
</tr>
<tr>
<td>$K_{traj_lab,softmax,DTW}$ best ($w_1 = 0.5, w_2 = 0.5, \beta = 1$)</td>
<td>0.82</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.65</td>
<td>0.46</td>
</tr>
<tr>
<td>$K_{traj_lab,max,ED}$ best ($w_1 = 0.7, w_2 = 0.3$)</td>
<td>0.43</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.42</td>
<td>0.40</td>
</tr>
<tr>
<td>$K_{traj_lab,maxnorm,ED}$ best ($w_1 = 0.1, w_2 = 0.9$)</td>
<td>0.47</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.44</td>
<td>0.42</td>
</tr>
<tr>
<td>$K_{traj_lab,softmax,ED}$ best ($w_1 = 0.9, w_2 = 0.1, \beta = 1$)</td>
<td>0.76</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.57</td>
<td>0.44</td>
</tr>
</tbody>
</table>
Table 19: Mean precision@39 for different $K_{\text{all}_1}$ kernels.

<table>
<thead>
<tr>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
<th>$w_4$</th>
<th>$\epsilon = 50$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.80</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.67−</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.15−</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0.24−</td>
</tr>
<tr>
<td>0</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>0.33−</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>0</td>
<td>0.69−</td>
</tr>
<tr>
<td>$\frac{7}{10}$</td>
<td>$\frac{3}{10}$</td>
<td>0</td>
<td>0</td>
<td>0.71−</td>
</tr>
<tr>
<td>$\frac{9}{10}$</td>
<td>$\frac{1}{10}$</td>
<td>0</td>
<td>0</td>
<td>0.75−</td>
</tr>
<tr>
<td>$\frac{1}{3}$</td>
<td>0</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>0.37−</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{4}$</td>
<td>0.38−</td>
</tr>
<tr>
<td>$\frac{3}{4}$</td>
<td>0</td>
<td>$\frac{1}{8}$</td>
<td>$\frac{1}{8}$</td>
<td>0.42−</td>
</tr>
<tr>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{4}$</td>
<td>0.33−</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{6}$</td>
<td>$\frac{1}{6}$</td>
<td>$\frac{1}{6}$</td>
<td>0.39−</td>
</tr>
<tr>
<td>$\frac{2}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>0.43−</td>
</tr>
</tbody>
</table>

Table 20: Mean precision@39 for different $K_{\text{all}_2}$ kernels.

<table>
<thead>
<tr>
<th>$w_3$</th>
<th>$w_4$</th>
<th>$w_5$</th>
<th>$\epsilon = 50$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.82</td>
</tr>
<tr>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>0.35−</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{4}$</td>
<td>$\frac{1}{4}$</td>
<td>0.36−</td>
</tr>
<tr>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{8}$</td>
<td>$\frac{1}{8}$</td>
<td>0.38−</td>
</tr>
<tr>
<td>$\frac{3}{4}$</td>
<td>$\frac{1}{8}$</td>
<td>$\frac{1}{8}$</td>
<td>0.40−</td>
</tr>
</tbody>
</table>

For $K_{\text{all}_1}$, we take from Table 18 the $K_{\text{lab},\text{softmax,DTW}}$ kernel as basic kernel, with $\beta = 1$. We combine this with the $K_{\text{traj},\text{softmax,DTW}}$ kernel for regular trajectories, and the start and end object kernels.

The results for different weight settings for $K_{\text{all}_1}$ are shown in Table 19. For each score a significant difference with the $K_{\text{traj},\text{softmax,DTW}}$ kernel is indicated with a $+$ if it is positive and a $-$ if it is negative. All the combined kernels show a significantly worse performance than the $w_1 = 1$ kernel, i.e. $K_{\text{traj},\text{softmax,DTW}}$.

$K_{\text{traj}_\text{lab},\text{softmax,DTW}}$ has the best performance of the $K_{\text{traj}_\text{lab}}$ kernels. Thus we plug this one into Definition 6.3.7. We give the results in Table 20. Again all combinations are significantly outperformed by the $K_{\text{traj},\text{softmax,DTW}}$ kernel, as indicated by a $-$. 
6.4.4.2 Discussion

The only combination of trajectory information and geographical domain knowledge that performs similarly to the trajectory information only kernel $K_{\text{traj,softmax,DTW}}$, is the kernel $K_{\text{traj_lab,softmax,DTW}}$, since performance under their best settings shows no significant difference. The incorporation of the start and end information has a clear negative effect on outlier detection performance.

Like in Chapter 4, we see again that soft-max kernels are well suited for outlier detection, since the soft-max variants that include domain knowledge outperform the non-soft-max versions.

6.5 Conclusions & Future Work

In this chapter we investigated the incorporation of geographical domain knowledge into the alignment similarity kernels of Chapter 4. To achieve this, geographical domain knowledge, i.e. types and labels of places and regions, was added to the regular trajectories in two forms. One form was to create a separate sequence of sets of geo-labels, and the other form was a combined version of the regular trajectory and the sequence of sets of geo-labels. Furthermore, we created special objects to represent the information about the start and the end of the trajectories.

In the clustering experiment the kernels with added domain knowledge discovered examples of interesting vessel behavior. The classification task of predicting vessel types showed that adding domain knowledge can increase classification accuracy significantly. In outlier detection no significantly better performance could be achieved.

For future work in clustering we would like to let domain experts label interesting vessel behavior. However, doing this labeling while incorporating domain knowledge can be quite difficult. It is also interesting to see what outliers are more dependent on geographical domain knowledge and how our method works for these cases.

Furthermore, we are interested in applying this kind of similarity in other domains where comparable domain knowledge exists. The domain knowledge is essentially a parameter of the similarity measure, which can be varied, so the quality and content of this information is of direct influence on the similarity. So, next to applying this similarity in other domains we should also consider other knowledge, i.e. ontologies, in the same domain.

In all three tasks, more complicated weighting schemes are possible for creating kernels. In the classification and outlier detection setting finding the optimal weights can be done automatically. However, for clustering this is more difficult, and playing around with weight settings is something left to a domain expert or end-user.
CONCLUSION

In this chapter we look at the questions posed in the Introduction and answer them using the research presented in Chapters 3 to 6.

7.1 ANSWERS TO RESEARCH QUESTIONS

Before we can answer the main research question, that asked how we can efficiently and effectively assign semantic labels to vessel trajectories, we look at the subsidiary questions that we posed.

The first sub-question posed was the following.

1. How do we reduce the large volume and redundancy of vessel trajectories, and make them suitable for higher level reasoning?

Chapter 3 deals with this question. The large amount of data is compressed using a variant of a piecewise linear segmentation algorithm, dubbed 2stage-pls, which removes redundant data points from a trajectory. This algorithm adds an extra stage, compared to regular piecewise linear segmentation. In this stage compression is done using only the speed of the vessel along the trajectory. As shown by the experiments, this extra step ensures that the stop and move information in a vessel trajectory is retained better than when using regular piecewise linear segmentation.

Based on this two-stage compression, trajectory segments are created, which are the basic movement events in the event ontology called Simple Event Model (SEM). With this model extra knowledge can be added, which is illustrated by reasoning from these segments to the high level concept of ferry behavior.

Chapters 4 and 5 answer the sub-question below.

2a. How do we compare vessel trajectories to be able to apply machine learning techniques?

In these chapters we consider two alignment similarities: edit distance and dynamic time warping, and a family of integral over time measures. For all these measures we used the regular versions and their corresponding positive semi-definite kernel variants. The best results are achieved by alignment based measures, which significantly outperform the integral based measures. Regular variants of edit distance achieve the highest scores in the tasks of clustering and classification. However, their is no significant difference with the best scores for regular versions of dynamic time warping. The positive semi-definite soft-max version of dynamic time warping gives the best results in the outlier detection task.
However, a full answer to Question 2a also depends on the answer to Question 2b below.

2b. What is the influence of applying trajectory compression on comparing vessel trajectories for machine learning?

The effect of compression is studied in Chapter 4. We see that for the three tasks, alignment measures perform better in these tasks on compressed data. Moreover, compression reduces the quadratic computation time of alignment measures drastically. Thus the influence of compression on the machine learning tasks is positive. The performance on the two tasks of clustering and classification does not indicate that piecewise linear segmentation is a more effective form of compression than resampling the trajectory at a low sample rate. However, in the outlier detection task significantly better results are achieved using piecewise linear segmentation.

The last research question posed was:

3. How do we incorporate geographical domain knowledge to improve machine learning for vessel trajectories?

We answer this question in Chapter 6. In this chapter, compressed vessel trajectories are enriched with name and type labels of geographical locations. The same alignment techniques as in Chapter 4 are used for these enriched trajectories, but with a different substitution function, which is defined for sets of geo-labels instead of points in geographical space. This results in similarity measures and kernels that incorporate geographical domain knowledge. Application of these measures shows a significantly better performance in our classification task and also leads to interesting examples of clustering results. No significant improvements are achieved in the outlier detection task.

Having answered the subsidiary questions, we come back to the main research question.

How do we, effectively and efficiently, assign semantic labels to vessel trajectory data?

The answer to this question is in the sub-questions above. The combination of 2stage-pls piecewise linear segmentation with alignment based similarity measures achieves good results in the three machine learning tasks that we defined. These three tasks represent different kinds of semantic labels that can be assigned to vessel trajectories. The use of compression makes computation of the alignment measures around a factor 100 faster, significantly increasing efficiency. The combination of alignments and compression shows better performance, in all three tasks, than the integral over time based similarity measures, most of which are slower to compute. Furthermore, we can include domain knowledge in the form of types and labels of
geographical locations to discover more complex clusters of behavior and to increase classification accuracy, adding to the effectiveness of the methods.

7.2 DISCUSSION

The experiments in this thesis are all performed on vessel trajectory data derived from AIS information. Therefore the conclusions are limited to this type of data. One of the main properties of vessel trajectory data is that it is highly regular, especially for the larger vessel types. For other moving object data with this property we expect to achieve similar results. However, for data in which the sample rate is low compared to change in behavior, such as humans tracked by cellphones, and animals, integral over time similarities might give better results. So, for that type of data the positive semi-definite kernels that we defined for the integral over time measures might prove useful.

Apart from the fact that this thesis only investigates one type of data, it also only considers three types of tasks. Using vessel trajectory data to solve other tasks is also possible, such as route prediction and frequent pattern discovery. Moreover, other variants of the tasks that we do consider are possible, e.g. outlier detection on fishing vessels, instead of cargo ships and tankers. Also, the classification task turned out to be too easy on the one hand, and too difficult on the other hand. Relatively good task performance could be achieved considering only the start and end points of a trajectory. It was quite difficult to improve these results, though some significant performance gain could be achieved by incorporating geographical domain knowledge. Nonetheless, a classification task with more variation in trajectories seems useful, to potentially better discriminate the different trajectory similarity kernels.

In our experiments we have seen a phenomenon that is commonly witnessed among practitioners of kernel methods. It is the phenomenon that indefinite, or non positive semi-definite, kernels often perform well. In the three chapters that have experiments with kernels, we see that the the best kernels for the clustering and classification tasks are, in five of the six cases, indefinite. However, in the outlier detection task the best performance is achieved by positive semi-definite kernels, for all three chapters. It is interesting to see that this difference holds for both alignment measures and integral measures. This seems to imply that there is something about the PSD kernels that is useful for outlier detection. This can either be the positive semi-definite property itself, or the fact that the PSD kernels are constructed by taking into account all possible alignments and time-shfits, respectively.

Apart from the fact that alignment similarities show better performance than the integral based ones, we also prefer them because
compression makes their computation very fast. However, for both alignment measures and integral based measures, solutions exist to speed up computation that we have not considered. For example, for dynamic time warping and edit distance there is the possibility of using a constrained warping window.

7.3 **FUTURE WORK**

Lee et al. (2007, 2008b,a) present a partition-and-group data-mining framework for trajectories that covers the same tasks as this thesis. However, their approach focuses on solving clustering, classification and outlier detection for subtrajectories. Furthermore, in terms of the two broad types of spatio-temporal data-mining (see the chapter by Nanni et al. (2008) and Section 2.2), the work in this thesis is of another type than that by Lee et al. We use generic machine learning algorithms for the three tasks, putting all knowledge about the application domain in the similarity measure. The Lee et al. framework integrates knowledge about the moving object domain more directly in their methods. These two issues complicate a direct comparison. First, our datasets are not yet ideally suited for a subtrajectory focused approach, but this can likely be adapted. Second, because our framework differs a lot from the framework by Lee et al. there are a lot of parameters to consider to make a good comparison, i.e. it is not just a matter of plugging in another similarity measure.

For all three tasks, good performance was obtained using a simple similarity measure on highly compressed data. This suggests that, for vessel trajectory data, further studying similarities purely for raw trajectories might not be very useful. Far more interesting tasks and problems can be tackled when we incorporate domain knowledge, of which we saw examples in Chapter 3, and in Chapter 6, which especially showed interesting clustering results.

One option to approach learning with trajectories and domain knowledge more generically is via their representation as RDF. As mentioned, RDF represents a graph and it would be interesting to see if graph similarities/kernels can be directly applied to the trajectories, with domain knowledge, as RDF graphs. In this way we would integrate the domain knowledge directly in the way it is represented, instead of creating a new representation in terms of sequences of sets of labels. For instance, this direct graph representation can deal with a subtype hierarchy and does not repeat labels if multiple trajectory points are in the same region, instead all points are linked to the same region concept in the graph. This might result in better performance on the three tasks that we defined.

To implement the results of this thesis in an actual Maritime Safety and Security (MSS) system some steps are still required. For the work in Chapter 3 we have implemented an online variant that was part
of a demo MSS system. However, the clustering, classification and outlier detection solutions that we presented are not suitable for a real system. More work is needed, especially to make the methods amenable to online application. For instance, their exist solutions to make Support Vector Machines a lot faster during classification time, which is needed in online systems, since this an expensive step; see e.g. the work by Burges and Schölkopf (1997). Moreover, in an actual system, interaction with the user is required, i.e. to set weights in the clustering algorithm or to label potential outliers.

In the future, moving object trajectory data will become even more ubiquitous. As a result, research in the field of spatio-temporal data-mining will surge further. The presented investigations into the performance of various trajectory similarity kernels are a small contribution in maturing this field.
STOP RETENTION FULL RESULTS

In the following we will present the results of the experiment in Chapter 3 that were not given there. For the different $\epsilon$ settings used, we refer to that chapter. The graphs plot stop retention on the vertical axis versus compression rate on the horizontal axis. Each graph shows results for one setting of the stop threshold $\theta$ (vertically).

Figure 17 shows the stop retention performance for the baselines that we defined. $E_{\mu=2}$ and $E_t$ show the best performance. However, $E_t$ is incomplete, since it does not consider the spatial dimension.

The performance of the variant of 2stage-pls with $E_t$ in the place of $E_v$ is shown in Figure 18. The results are worse than for regular 2stage-pls.

The results for the two variant using the $\land$ construction are given in Figure 19. The graphs look a bit strange, which is due to how we have implemented the necessary max function, since we define the size of the error as a fraction of the value for $\epsilon$.

Algorithm 2stage-pls with the speed and spatial steps reversed performs worse than regular 2stage-pls, as we can see from the results in Figure 20.
Figure 17: Results of the baselines under different experimental settings. Results for $\text{plse}_{\mu=2}$ are repeated for reference.
Figure 18: Results of 2stage-pls with $E_1, E_2$ and $E_1, E_{\mu=2}$ under different experimental settings. Results for $\text{pls}_{E_{\mu=2}}$ are included for reference.
Figure 19: Results of $E_2 \wedge E_v$ and $E_{\mu=2} \wedge E_v$, with recomputed SoG, under different experimental settings. Results for $\text{plS}_{E_{\mu=2}}$ are included for reference.
Figure 20: Results of 2stage-pls with $E_2$ first, and 2stage-pls with $E_{\mu=2}$ first, with recomputed SoG, under different experimental settings. Results for pls$_{E_{\mu=2}}$ are included for reference.
EDIT DISTANCE SOFT-MAX KERNEL PROOF

To prove that the \( \text{sim}_{\text{softmax}, \text{ED}} \) function is positive semi-definite, we construct it as a mapping kernel (see Definition 2.1.2 and the paper by Shin and Kuboyama (2008)). Because of the use of the mapping kernel, this proof is different than the one by Vert et al. (2004). The proof considers sequences in general, not specifically trajectories. Furthermore, we assume that we have a conditionally positive definite substitution function sub, e.g. the negative of the Euclidean distance as used in Chapter 4. We also assume \( \beta > 0 \) and a value for \( g \). Because \( \text{sub} \) is conditionally positive definite, \( \text{exp}(\beta \text{sub}) \) is positive semi-definite (Berg et al. (1984)).

First we define the set of all non-contiguous subsequences of a sequence \( S \) of length \( n \):

\[
\text{subseq}(S, n) = \{(S', x) \mid S' = S(i_1), \ldots, S(i_n), \ \ 1 \leq i_1 < \ldots < i_n \leq |S|, \ x = |S| - n\}.
\]

The integer \( x \) represents the number of unused sequence elements, i.e. \(|S| - n\). Thus, this set contains pairs of subsequences and integers.

For a set of sequences \( S \) and \( n \geq 0 \) we define the kernel below on the set \( \bigcup_{S \in S} \text{subseq}(S, n) \),

\[
k_n^{\text{subseq}}((S', x), (T', y)) = \exp(\beta g \cdot (x + y)) \prod_{i=1}^{n} \exp(\beta \text{sub}(S'(i), T'(i))).
\]

The part \( \exp(\beta g \cdot (x + y)) \) is positive semi-definite, since it can be written as \( \exp(\beta gx) \cdot \exp(\beta gy) \).\(^1\) We know that \( \exp(\beta \text{sub}) \) is positive semi-definite. Hence, \( k_n^{\text{subseq}} \) is positive semi-definite, since it is the product of positive semi-definite kernels.

For the set \( \bigcup_{S \in S} \text{subseq}(S, n) \) we create the straightforward transitive map:

\[
M_n(S, T) = \text{subseq}(S, n) \times \text{subseq}(T, n).
\]

With the map \( M_n \) we create the mapping kernel:

\[
k_n^{\text{fixed}}(S, T) = \sum_{((S', x), (T', y)) \in M_n(S, T)} k_n^{\text{subseq}}((S', x), (T', y)).
\]

We create the following positive semi-definite kernel, by closure under pointwise limit:

\[
k(S, T) = \sum_{i=0}^{\infty} k_i^{\text{fixed}}(S, T).
\]

\(^1\) It is a basic string kernel defined by Vert et al. (2004).
This kernel considers all possible length subsequences of $S$ and $T$.

The kernel $k_{all}$ is equivalent to $\text{sim}_{\text{softmax,ED}}$. To see this we construct the map $M^n$ in another way, via the definition of the edit distance alignment. First, consider the set of all alignments of length $n$ for all edit distance alignments $\Pi(S,T)$ between two sequences $S$ and $T$ (see Definition 4.2.3):

$$\Pi^n(S,T) = \{ \pi \in \Pi(S,T) \mid |\pi| = n \} . \quad \text{(B.6)}$$

Using this set we define the map:

$$N^n(S,T) = \{(S', x), (T', y)\} \mid S' = S(\pi_1(1)), \ldots, S(\pi_1(n)),
T' = T(\pi_2(1)), \ldots, T(\pi_2(n)),
\pi \in \Pi^n(S,T), x = |S| - n, y = |T| - n \} . \quad \text{(B.7)}$$

This map is equal to the map $M^n$ as in the following lemma.

**Lemma B.0.1.** For two sequences $S$ and $T$:

$$M^n(S,T) = N^n(S,T) .$$

**Proof.** To prove Lemma B.0.1 we assume towards contradiction that the equality does not hold. This gives two cases:

$$\exists ((S', |S'| - n), (T', |T'| - n)) \in M^n(S,T) \wedge ((S', |S'| - n), (T', |T'| - n)) \notin N^n(S,T) \quad (1)$$

and (2)

$$\exists ((S', |S'| - n), (T', |T'| - n)) \in N^n(S,T) \wedge ((S', |S'| - n), (T', |T'| - n)) \notin M^n(S,T) . \quad (2)$$

(1) The pair $((S', |S'| - n), (T', |T'| - n))$ is not in $N^n(S,T)$. Thus there is no alignment $\pi \in \Pi^n(S,T)$ for which $S' = S(\pi_1(1)), \ldots, S(\pi_1(n))$ and $T' = T(\pi_2(1)), \ldots, T(\pi_2(n))$. Thus one of the constraints of Definition 4.2.3 does not hold. Without loss of generality we assume that this is the first constraint: $1 \leq \pi_1(1) < \pi_1(2) < \ldots < \pi_1(n) \leq |S|$. Thus either $\pi_1(1) < 1, \pi_1(n) > |S'|$, or $\pi_1(i + 1) < \pi_1(i)$, all of which violate Equation B.1. Hence we arrive at a contradiction.

(2) We assume that the pair $((S', |S'| - n), (T', |T'| - n))$ is not in $M^n(S,T)$. Since $M^n(S,T)$ is defined as the full cross product, we can assume without loss of generality that $(S', |S'| - n) \notin \text{subseq}(S', n)$. Thus the constraint $1 \leq i_1 < \ldots < i_n \leq |S|$ is violated. Like in case (1), this leads to a contradiction with the definition of an edit distance alignment (Definition 4.2.3). □
Thus, we can rewrite the kernel $k_{\text{all}}$, using the shorthand $S_{\pi_1}$ for $S(\pi_1(1)), \ldots, S(\pi_1(|\pi|))$ and $T_{\pi_2}$ for $T(\pi_2(1)), \ldots, T(\pi_2(|\pi|))$:

$$k_{\text{all}}(S, T) = \sum_{i=0}^{\infty} \sum_{(S', x), (T', y) \in N^i(S, T)} k^i_{\text{subseq}}((S', x), (T', y))$$  \hspace{1cm} (B.8)

$$= \sum_{i=0}^{\infty} \sum_{\pi \in \Pi^i(S, T)} k^i_{\text{subseq}}([S_{\pi_1}(\pi_1 - i)], [(T_{\pi_2}(\pi_2 - i)]) \hspace{1cm} (B.9)

$$= \sum_{\pi \in \Pi(S, T)} \prod_{i=1}^{\pi} \exp(\beta \text{sub}(S(\pi_1(i)), T(\pi_2(i)))) \cdot \exp(\beta g \cdot (|S| - |\pi|) + (|T| - |\pi|))$$  \hspace{1cm} (B.10)

$$= \sum_{\pi \in \Pi(S, T)} \exp(\beta \sum_{i=1}^{\pi} \text{sub}(S(\pi_1(i)), T(\pi_2(i)))$$

$$+ g \cdot (|S| - |\pi|) + (|T| - |\pi|))$$  \hspace{1cm} (B.11)

$$= \text{sim_{softmax,ED}}(S, T) \cdot$$  \hspace{1cm} (B.12)

For line B.8 we use Equation B.4 and plug in the map $N^n$. We use Equation B.7 to get line B.9. With Equation B.2 and the fact that $\bigcup_{i=0}^{\infty} \Pi^i = \Pi$ we derive line B.10. Finally we put the product in the exponent to get line B.11. This line is equal to the definition of $\text{sim_{softmax,ED}}$. Thus, we have proven that $\text{sim_{softmax,ED}}$ is positive semidefinite.
SUMMARY

With the ubiquity of GPS sensors, tracking people and objects in geographical space has become trivial in the past decade. This has introduced a new kind of data called the moving object trajectory. These trajectories are usually stored as is, without any labels indicating, for instance, what the person or object is doing. Automatically adding such labels to the movement data can help end-users and operators in interpreting the trajectories.

This thesis presents research on automatically assigning classes to vessel trajectories. To do this we apply techniques from the fields of machine learning and data-mining to solve three typical tasks: clustering, classification and outlier detection. Clustering vessel trajectories discovers groups of similar movement behavior. In the classification task we predict the type of vessel belonging to a trajectory. Outlier detection tries to find deviant vessel trajectories among a set of normal trajectories.

The presented research was carried out in the context of the Poseidon project, which aimed at developing a Maritime Safety and Security (MSS) system. The Poseidon project provided the datasets used to evaluate the performance on the three machine learning tasks. These datasets consist of vessel trajectories around the Dutch coast captured using the Automatic Identification System (AIS).

The main element in solving these three tasks is computing a similarity measure between trajectories. These similarity measures are defined as kernels, so that they can be used in kernel based machine learning algorithms. Kernel methods provide a unified framework in which the kernel is the central component for performing different tasks. For classification and outlier detection we use Support Vector Machines, and we use kernel k-means for clustering.

The volume of vessel trajectory data is huge and often quite regular, thus it makes sense to apply a trajectory compression technique to make the data more manageable. In moving object analysis, the concepts of stop and move are essential movement primitives. We show in Chapter 3 that standard trajectory compression techniques, based on line-simplification, are not ideally suited to retain the stop and move information. Therefore we propose a simple two-stage extension to this method that looks at speed along a trajectory first. We evaluate standard trajectory compression and our technique on the task of stop retention using a dataset of vessel trajectories with stops. From this evaluation we conclude that our method generally has better stop retention at the same compression rate. Furthermore,
we show how we can use this method to create the semantic building blocks for higher level reasoning about vessel behavior.

In Chapter 4 we apply two alignment measures, dynamic time warping and edit distance, in the three tasks of clustering, classification, and outlier detection. The alignment measures are defined as kernels in three different variants. We investigate the performance of these alignment kernels in combination with the trajectory compression method of Chapter 3. The experiments indicate that compression has a positive effect on the performance in the three tasks. The regular kernels, based on taking the best alignment, perform best in the clustering and classification tasks, whereas the soft-max kernels, based on summing over all alignments, show the best performance in the outlier detection task.

Chapter 5 studies similarity measures for vessel trajectories that are based on taking the integral over time of the distance between two trajectories. For these similarities we define kernel variants in an approach similar to Chapter 4. We test these kernels in the tasks of clustering, classification and outlier detection. These experiments indicate that these kernel variants work, but they are outperformed by the alignment methods from the previous chapter. As in the previous chapter the soft-max kernel variants perform well in outlier detection.

In Chapter 6 we investigate similarity measures that combine low-level trajectory information with geographical domain knowledge to compare vessel trajectories. These similarity measures are largely based on the alignment techniques of Chapter 4. In a clustering experiment we show how these measures can be used to discover behavior concepts in vessel trajectories that are dependent both on the low-level trajectories and the domain knowledge. We also apply these measures in a classification and outlier detection task. In the classification experiment we show that significantly better classification accuracy can be achieved by combining trajectory information and geographical domain knowledge. The outlier detection experiment shows no significant increase in performance due to the added domain knowledge.

The main conclusion from the presented research is that the combination of alignment similarities with two-stage compression achieves the best results in the three defined machine learning tasks. The use of compression makes computation of the alignment measures around a factor 100 faster, significantly increasing efficiency. The alignment and compression combination shows better performance than the integral over time based similarity measures, most of which are slower to compute. Furthermore, we can include domain knowledge in the form of types and labels of geographical locations to discover more complex clusters of behavior and to increase classification accuracy, adding to the effectiveness of the methods.
An interesting direction for future research is better incorporation of (geographical) domain knowledge. One option is to investigate a more general method to integrate trajectories with geographical domain knowledge using graphs instead of sequences as in Chapter 6. The implementation of the results of this thesis in an actual MSS system still requires some steps, especially to make the methods usable online.
SAMENVATTING

Het volgen van mensen en objecten in de geografische ruimte is de laatste tien jaar steeds eenvoudiger geworden. Dit komt vooral doordat tegenwoordig allerlei dingen zijn uitgerust met een GPS sensor. Deze ontwikkeling heeft gezorgd voor een nieuw soort data: Trajecten van Bewegende Objecten. Deze trajecten worden meestal opgeslagen zonder veel verdere informatie, zoals een label wat de persoon of het object aan het doen is. Het automatisch toevoegen van dit soort informatie kan eindgebruikers helpen bij het interpreteren van deze traject data.

Dit proefschrift beschrijft onderzoek naar het automatisch toewijzen van classes aan trajecten van schepen. Hiervoor gebruiken we technieken uit de onderzoeksvelden van machine learning en datamining om drie gebruikelijk taken, clustering, classificatie en het detecteren van uitbijters, uit te voeren. Clustering gebruiken we om groepen met vergelijkbaar bewegingsgedrag te vinden. Classificatie doen we om het type van een schip behorende bij een bepaald traject te bepalen. En het detecteren van uitbijters doen we om te bepalen welke schepen afwijkend gedrag vertonen.

Het gepresenteerde onderzoek is gedaan binnen het Poseidon project. Dit project had tot doel het ontwikkelen van een zogenaamd Maritiem Veiligheidssysteem. Binnen het project was er toegang tot datasets van scheepstrajecnten die we gebruikt hebben om de prestatie op de drie verschillende taken te meten. Deze trajecten zijn verzameld rond de Nederlandse (kust)wateren met behulp van het Automatic Identification System (AIS).

Het belangrijkste onderdeel dat nodig is om de drie taken uit te kunnen voeren is een vergelijkingsmaat tussen trajecten. Deze vergelijkingsmaten worden gedefinieerd als zogenaamde kernels, zodat ze gebruikt kunnen worden in kernel gebaseerde machine learning algoritmen. De kernel methoden leveren een uniform raamwerk om verschillende taken uit te voeren, met als centraal element de kernel. Voor classificatie en uitbijter detectie gebruiken we de bekende Support Vector Machine techniek, en voor clustering gebruiken we kernel k-means.

De hoeveelheid scheepstrajecnt data is groot, maar de bewegingen zijn over het algemeen vrij regulier, dus kan het zinnig zijn om een traject compressie techniek toe te passen. Bij het analyseren van object bewegingsdata is het herkennen van stops essentieel. In Hoofdstuk 3 laten we zien dat standaard traject compressie technieken deze stop informatie niet voldoende behouden. Daarom introduceren we een eenvoudige twee staps uitbreiding, die in de eerste stap alleen naar
De snelheid gedurende het traject kijkt. We vergelijken onze uitzending met de standaard techniek in het behouden van stop informatie. Uit deze evaluatie blijkt dat onze methode in het algemeen stop informatie beter behoudt als de compressie ratio hetzelfde is. Bovendien kunnen we onze methode gebruiken om semantische bouwstenen te vormen, die hoger niveau redeneringen over scheepsgedrag mogelijk maken.

In Hoofdstuk 4 passen we twee vergelijkingsmaten toe die gebruik maken van het oplijnen van elementen in een sequentie. Deze maten, dynamic time warping en edit distance, gebruiken we voor de clustering, classificatie en uitbijter detectie taken. We definieren drie kernel varianten van deze maten. Van deze kernels onderzoeken we de prestaties in de drie taken in combinatie met het comprimeren van de scheepstrajecten. Uit de experimenten blijkt dat de compressie een positief effect heeft op de prestatie in de drie taken. De kernels gebaseerd op het nemen van alleen de best oplijning doen het het beste in de clustering en classificatie taken, en de kernels gebaseerd op het sommeren over alle oplijningen geven de beste resultaten in de uitbijter detectie taak.

Hoofdstuk 5 bestudeert vergelijkingsmaten die gebaseerd zijn op het berekenen van de integraal over de tijd van de afstandsfunctie tussen twee trajecten. Voor deze maten definieren we kernels volgens een aanpak analoog aan die uit Hoofdstuk 4. Ook deze kernels testen we in de clustering, classificatie en uitbijter detectie taken. Uit deze experimenten blijkt dat de kernels gebaseerd op de oplijningsmaten van het vorige hoofdstuk beter werken.

Vergelijkingsmaten die de laag niveau traject informatie combineren met geografische domein kennis zijn het onderwerp van Hoofdstuk 6. Deze maten zijn grotendeels gebaseerd op de maten in Hoofdstuk 4. In een clustering experiment laten we zien hoe deze maten gebruikt kunnen worden om concepten te ontdekken die gebaseerd zijn op zowel de laag niveau informatie als de domein kennis. We gebruiken de maten ook in een classificatie en uitbijter detectie taak. In het classificatie experiment zien we dat de combinatie maat significant betere resultaten kan behalen dan wanneer we slechts laag niveau informatie gebruiken. In de uitbijter taak zien we geen significant betere prestaties voor de gecombineerde maat.

De belangrijkste conclusie van het gepresenteerde onderzoek is dat oplijningsmaten gecombineerd met onze twee staps compressie methode de beste resultaten behalen in de drie taken. Het gebruik van compressie maakt het berekenen van de oplijningsmaten in de orde van een factor 100 sneller. De oplijningsmaten laten significant betere resultaten zien dan de maten gebaseerd op het berekenen van de integraal over tijd, die grotendeels ook meer rekentijd kosten. Bovendien kunnen we eenvoudig domein kennis in de vorm van type en label informatie over geografische locaties toevoegen, waardoor
we nog complexere clusters kunnen ontdekken en de classificatie nauwkeurigheid kunnen verbeteren.

In toekomstig onderzoek zou het zeer interessant zijn om te kijken hoe (geografische) domein kennis beter geïntegreerd kan worden. Een mogelijkheid om dit te generaliseren is de verrijking met domein kennis te doen door middel van grafen in plaats van de sequenties uit Hoofdstuk 6. Voor het gebruik van de resultaten uit dit proefschrift in een echt Maritiem Veiligheidssysteem zijn er nog een aantal stappen nodig, de belangrijkste daarvan is het beter bruikbaar maken van de methodes in een on line context.


Martin Ester, Hans-Peter Kriegel, Joerg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.


IJCNN 2008. Proceedings of the International Joint Conference on Neural Networks, IJCNN 2008, part of the IEEE World Congress on Computational Intelligence, WCCI 2008, Hong Kong, China, June 1-6, 2008, 2008. IEEE.


Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proc. VLDB Endow.*, 1(1):1081–1094, August 2008b. ISSN 2150-8097. doi: http://dx.doi.org/10.1145/1453856.1453972.


<table>
<thead>
<tr>
<th>Year</th>
<th>Author(s)</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1998</td>
<td>Johan van den Akker (CWI)</td>
<td>DEGAS - An Active, Temporal Database of Autonomous Objects</td>
</tr>
<tr>
<td>1998</td>
<td>Floris Wiesman (UM)</td>
<td>Information Retrieval by Graphically Browsing Meta-Information</td>
</tr>
<tr>
<td>1998</td>
<td>Ans Steuten (TUD)</td>
<td>A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective</td>
</tr>
<tr>
<td>1998</td>
<td>Dennis Breuker (UM)</td>
<td>Memory versus Search in Games</td>
</tr>
<tr>
<td>1998</td>
<td>E.W. Oskamp (RUL)</td>
<td>Computerondersteuning bij Straftoemeting</td>
</tr>
<tr>
<td>1999</td>
<td>Mark Sloof (VU)</td>
<td>Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products</td>
</tr>
<tr>
<td>1999</td>
<td>Rob Potharst (EUR)</td>
<td>Classification using decision trees and neural nets</td>
</tr>
<tr>
<td>1999</td>
<td>Don Beal (UM)</td>
<td>The Nature of Minimax Search</td>
</tr>
<tr>
<td>1999</td>
<td>Jacques Penders (UM)</td>
<td>The practical Art of Moving Physical Objects</td>
</tr>
<tr>
<td>1999</td>
<td>Aldo de Moor (KUB)</td>
<td>Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems</td>
</tr>
<tr>
<td>1999</td>
<td>Niek J.E. Wijngaards (VU)</td>
<td>Re-design of compositional systems</td>
</tr>
<tr>
<td>1999</td>
<td>David Spelt (UT)</td>
<td>Verification support for object database design</td>
</tr>
<tr>
<td>1999</td>
<td>Jacques H.J. Lenting (UM)</td>
<td>Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation</td>
</tr>
<tr>
<td>2000</td>
<td>Frank Niessink (VU)</td>
<td>Perspectives on Improving Software Maintenance</td>
</tr>
<tr>
<td>2000</td>
<td>Koen Holtman (TUE)</td>
<td>Prototyping of CMS Storage Management</td>
</tr>
<tr>
<td>2000</td>
<td>Carolien M.T. Metselaar (UvA)</td>
<td>Sociaal-organisatorische gevolgen van kennis technologie; een procesbenadering en actangeperspectief</td>
</tr>
<tr>
<td>2000</td>
<td>Geert de Haan (VU)</td>
<td>ETAG, A Formal Model of Competence Knowledge for User Interface Design</td>
</tr>
<tr>
<td>2000</td>
<td>Ruud van der Pol (UM)</td>
<td>Knowledge-based Query Formulation in Information Retrieval</td>
</tr>
<tr>
<td>2000</td>
<td>Rogier van Eijk (UU)</td>
<td>Programming Languages for Agent Communication</td>
</tr>
<tr>
<td>2000</td>
<td>Niels Peek (UU)</td>
<td>Decision-theoretic Planning of Clinical Patient Management</td>
</tr>
<tr>
<td>2000</td>
<td>Veerle Coup (EUR)</td>
<td>Sensitivity Analysis of Decision-Theoretic Networks</td>
</tr>
<tr>
<td>2000</td>
<td>Florian Waas (CWI)</td>
<td>Principles of Probabilistic Query Optimization</td>
</tr>
<tr>
<td>2000</td>
<td>Jonas Karlsson (CWI)</td>
<td>Scalable Distributed Data Structures for Database Management</td>
</tr>
</tbody>
</table>
2001

2001-1 Silja Renooij (UU)  
Qualitative Approaches to Quantifying Probabilistic Networks

2001-2 Koen Hindriks (UU)  
Agent Programming Languages: Programming with Mental Models

2001-3 Maarten van Someren (UvA)  
Learning as problem solving

2001-4 Evgueni Smirnov (UM)  
Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets

2001-5 Jacco van Ossenbruggen (VU)  
Processing Structured Hypermedia: A Matter of Style

2001-6 Martijn van Welie (VU)  
Task-based User Interface Design

2001-7 Bastiaan Schoenage (VU)  
Diva: Architectural Perspectives on Information Visualization

2001-8 Pascal van Eck (VU)  
A Compositional Semantic Structure for Multi-Agent Systems Dynamics

2001-9 Peter Jan ’t Hoen (RUL)  
Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes

2001-10 Maarten Sierhuis (UvA)  
Modeling and Simulating Work Practice, BRAHMS: a multiagent modeling and simulation language for work practice analysis and design

2001-11 Tom M. van Engers (VU)  
Knowledge Management: The Role of Mental Models in Business Systems Design

2002

2002-1 Nico Lassing (VU)  
Architecture-Level Modifiability Analysis

2002-2 Roelof van Zwol (UT)  
Modelling and searching web-based document collections

2002-3 Henk Ernst Blok (UT)  
Database Optimization Aspects for Information Retrieval

2002-4 Juan Roberto Castelo Valdueza (UU)  
The Discrete Acyclic Digraph Markov Model in Data Mining

2002-5 Radu Serban (VU)  
The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents

2002-6 Laurens Mommers (UL)  
Applied legal epistemology; Building a knowledge-based ontology of the legal domain

2002-7 Peter Boncz (CWI)  
Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications

2002-8 Jaap Gordijn (VU)  
Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas

2002-9 Willem-Jan van den Heuvel (KUB)  
Integrating Modern Business Applications with Objectified Legacy Systems

2002-10 Brian Sheppard (UM)  
Towards Perfect Play of Scrabble

2002-11 Wouter C.A. Wijngaards (VU)  
Agent Based Modelling of Dynamics: Biological and Organisational Applications

2002-12 Albrecht Schmidt (UvA)  
Processing XML in Database Systems

2002-13 Hongjing Wu (TUE)  
A Reference Architecture for Adaptive Hypermedia Applications

2002-14 Wieke de Vries (UU)  
Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems

2002-15 Rik Eshuis (UT)  
Semantics and Verification of UML Activity Diagrams for Workflow Modelling
<table>
<thead>
<tr>
<th>Year</th>
<th>Author</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>2002-17</td>
<td>Stefan Manegold (UvA)</td>
<td>Understanding, Modeling, and Improving Main-Memory Database Performance</td>
</tr>
<tr>
<td>2003-1</td>
<td>Heiner Stuckenschmidt (VU)</td>
<td>Ontology-Based Information Sharing in Weakly Structured Environments</td>
</tr>
<tr>
<td>2003-2</td>
<td>Jan Broersen (VU)</td>
<td>Modal Action Logics for Reasoning About Reactive Systems</td>
</tr>
<tr>
<td>2003-3</td>
<td>Martijn Schuemie (TUD)</td>
<td>Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy</td>
</tr>
<tr>
<td>2003-4</td>
<td>Milan Petkovic (UT)</td>
<td>Content-Based Video Retrieval Supported by Database Technology</td>
</tr>
<tr>
<td>2003-5</td>
<td>Jos Lehmann (UvA)</td>
<td>Causation in Artificial Intelligence and Law - A modelling approach</td>
</tr>
<tr>
<td>2003-6</td>
<td>Boris van Schooten (UT)</td>
<td>Development and specification of virtual environments</td>
</tr>
<tr>
<td>2003-7</td>
<td>Machiel Jansen (UvA)</td>
<td>Formal Explorations of Knowledge Intensive Tasks</td>
</tr>
<tr>
<td>2003-8</td>
<td>Yongping Ran (UM)</td>
<td>Repair Based Scheduling</td>
</tr>
<tr>
<td>2003-9</td>
<td>Rens Kortmann (UM)</td>
<td>The resolution of visually guided behaviour</td>
</tr>
<tr>
<td>2003-10</td>
<td>Andreas Lincke (UvT)</td>
<td>Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture</td>
</tr>
<tr>
<td>2003-11</td>
<td>Simon Keizer (UT)</td>
<td>Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks</td>
</tr>
<tr>
<td>2003-12</td>
<td>Roeland Ordelman (UT)</td>
<td>Dutch speech recognition in multimedia information retrieval</td>
</tr>
<tr>
<td>2003-13</td>
<td>Jeroen Donkers (UM)</td>
<td>Nosce Hostem - Searching with Opponent Models</td>
</tr>
<tr>
<td>2003-14</td>
<td>Stijn Hoppenbrouwers (KUN)</td>
<td>Freezing Language: Conceptualisation Processes across ICT-Supported Organisations</td>
</tr>
<tr>
<td>2003-15</td>
<td>Mathijs de Weerdt (TUD)</td>
<td>Plan Merging in Multi-Agent Systems</td>
</tr>
<tr>
<td>2003-16</td>
<td>Menzo Windhouwer (CWI)</td>
<td>Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses</td>
</tr>
<tr>
<td>2003-18</td>
<td>Levente Kocsis (UM)</td>
<td>Learning Search Decisions</td>
</tr>
<tr>
<td>2004-1</td>
<td>Virginia Dignum (UU)</td>
<td>A Model for Organizational Interaction: Based on Agents, Founded in Logic</td>
</tr>
<tr>
<td>2004-2</td>
<td>Lai Xu (UvT)</td>
<td>Monitoring Multi-party Contracts for E-business</td>
</tr>
<tr>
<td>2004-3</td>
<td>Perry Groot (VU)</td>
<td>A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving</td>
</tr>
<tr>
<td>2004-4</td>
<td>Chris van Aart (UvA)</td>
<td>Organizational Principles for Multi-Agent Architectures</td>
</tr>
<tr>
<td>2004-5</td>
<td>Viara Popova (EUR)</td>
<td>Knowledge discovery and monotonicity</td>
</tr>
<tr>
<td>2004-7</td>
<td>Elise Boltjes (UM)</td>
<td>Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes</td>
</tr>
<tr>
<td>2004-8</td>
<td>Joop Verbeek (UM)</td>
<td>Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise</td>
</tr>
</tbody>
</table>
2004-9 Martin Caminada (VU)  
For the Sake of the Argument; explorations into argument-based reasoning

2004-10 Suzanne Kabel (UvA)  
Knowledge-rich indexing of learning-objects

2004-11 Michel Klein (VU)  
Change Management for Distributed Ontologies

2004-12 The Duy Bui (UT)  
Creating emotions and facial expressions for embodied agents

2004-13 Wojciech Jamroga (UT)  
Using Multiple Models of Reality: On Agents who Know how to Play

2004-14 Paul Harrenstein (UU)  
Logic in Conflict. Logical Explorations in Strategic Equilibrium

2004-15 Arno Knobbe (UU)  
Multi-Relational Data Mining

2004-16 Federico Divina (VU)  
Hybrid Genetic Relational Search for Inductive Learning

2004-17 Mark Winands (UM)  
Informed Search in Complex Games

2004-18 Vania Bessa Machado (UvA)  
Supporting the Construction of Qualitative Knowledge Models

2004-19 Thijs Westerveld (UT)  
Using generative probabilistic models for multimedia retrieval

2004-20 Madelon Evers (Nyenrode)  
Learning from Design: facilitating multidisciplinary design teams

2005

2005-1 Floor Verdenius (UvA)  
Methodological Aspects of Designing Induction-Based Applications

2005-2 Erik van der Werf (UM)  
AI techniques for the game of Go

2005-3 Franc Grootjen (RUN)  
A Pragmatic Approach to the Conceptualisation of Language

2005-4 Nirvana Meratnia (UT)  
Towards Database Support for Moving Object data

2005-5 Gabriel Infante-Lopez (UvA)  
Two-Level Probabilistic Grammars for Natural Language Parsing

2005-6 Pieter Spronck (UM)  
Adaptive Game AI

2005-7 Flavius Frasincar (TUE)  
Hypermedia Presentation Generation for Semantic Web Information Systems

2005-8 Richard Vdovjak (TUE)  
A Model-driven Approach for Building Distributed Ontology-based Web Applications

2005-9 Jeen Broekstra (VU)  
Storage, Querying and Inference for Semantic Web Languages

2005-10 Anders Bouwer (UvA)  
Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments

2005-11 Elth Ogston (VU)  
Agent Based Matchmaking and Clustering - A Decentralized Approach to Search

2005-12 Csaba Boer (EUR)  
Distributed Simulation in Industry

2005-13 Fred Hamburg (UL)  
Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen

2005-14 Borys Omelayenko (VU)  
Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics

2005-15 Tibor Bosse (VU)  
Analysis of the Dynamics of Cognitive Processes

2005-16 Joris Graaumans (UU)  
Usability of XML Query Languages

2005-17 Boris Shishkov (TUD)  
Software Specification Based on Re-usable Business Components
2005-18  Danielle Sent (UU)  Test-selection strategies for probabilistic networks
2005-19  Michel van Dartel (UM)  Situated Representation
2005-20  Cristina Coteanu (UL)  Cyber Consumer Law, State of the Art and Perspectives
2005-21  Wijnand Derks (UT)  Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics

2006
2006-1  Samuil Angelov (TUE)  Foundations of B2B Electronic Contracting
2006-2  Cristina Chisalita (VU)  Contextual issues in the design and use of information technology in organizations
2006-3  Noor Christoph (UvA)  The role of metacognitive skills in learning to solve problems
2006-4  Marta Sabou (VU)  Building Web Service Ontologies
2006-5  Cees Pierik (UU)  Validation Techniques for Object-Oriented Proof Outlines
2006-7  Marko Smiljanic (UT)  XML schema matching – balancing efficiency and effectiveness by means of clustering
2006-8  Eelco Herder (UT)  Forward, Back and Home Again - Analyzing User Behavior on the Web
2006-9  Mohamed Wahdan (UM)  Automatic Formulation of the Auditor’s Opinion
2006-10  Ronny Siebes (VU)  Semantic Routing in Peer-to-Peer Systems

2006-12  Bert Bongers (VU)  Interactivation - Towards an ecology of people, our technological environment, and the arts
2006-13  Henk-Jan Lebbink (UU)  Dialogue and Decision Games for Information Exchanging Agents
2006-14  Johan Hoorn (VU)  Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change
2006-15  Rainer Malik (UU)  CONAN: Text Mining in the Biomedical Domain
2006-16  Carsten Riggelsen (UU)  Approximation Methods for Efficient Learning of Bayesian Networks
2006-17  Stacey Nagata (UU)  User Assistance for Multitasking with Interruptions on a Mobile Device
2006-18  Valentin Zhizhkun (UvA)  Graph transformation for Natural Language Processing
2006-20  Marina Velikova (UvT)  Monotone models for prediction in data mining
2006-21  Bas van Gils (RUN)  Aptness on the Web
2006-22  Paul de Vrieze (RUN)  Fundaments of Adaptive Personalisation
2006-23  Ion Juvina (UU)  Development of Cognitive Model for Navigating on the Web
2006-24  Laura Hollink (VU)  Semantic Annotation for Retrieval of Visual Resources
2006-25  Madalina Drugan (UU)  Conditional log-likelihood MDL and Evolutionary MCMC
2006-26  Vojkan Mihailović (UT)  Score Region Algebra: A Flexible Framework for Structured Information Retrieval
2006-27 Stefano Bocconi (CWI)
Vox Populi: generating video documentaries from semantically annotated media repositories

2006-28 Borkur Sigurðbjörnsson (UvA)
Focused Information Access using XML Element Retrieval

2007

2007-1 Kees Leune (UvT)
Access Control and Service-Oriented Architectures

2007-2 Wouter Teepe (RUG)
Reconciling Information Exchange and Confidentiality: A Formal Approach

2007-3 Peter Mika (VU)
Social Networks and the Semantic Web

2007-4 Jurriaan van Diggelen (UU)
Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach

2007-5 Bart Schermer (UL)
Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance

2007-6 Gilad Mishne (UvA)
Applied Text Analytics for Blogs

2007-7 Natasa Jovanovic’ (UT)
To Whom It May Concern - Addressee Identification in Face-to-Face Meetings

2007-8 Mark Hoogendoorn (VU)
Modeling of Change in Multi-Agent Organizations

2007-9 David Mobach (VU)
Agent-Based Mediated Service Negotiation

2007-10 Huib Aldewereld (UU)
Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols

2007-11 Natalia Stash (TUE)
Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System

2007-12 Marcel van Gerven (RUN)
Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty

2007-13 Rutger Rienks (UT)
Meetings in Smart Environments; Implications of Progressing Technology

2007-14 Niek Bergboer (UM)
Context-Based Image Analysis

2007-15 Joyca Lacroix (UM)
NIM: a Situated Computational Memory Model

2007-16 Davide Grossi (UU)
Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems

2007-17 Theodore Charitos (UU)
Reasoning with Dynamic Networks in Practice

2007-18 Bart Orriens (UvT)
On the development and management of adaptive business collaborations

2007-19 David Levy (UM)
Intimate relationships with artificial partners

2007-20 Slinger Jansen (UU)
Customer Configuration Updating in a Software Supply Network

2007-21 Karianne Vermaas (UU)
Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005

2007-22 Zlatko Zlatev (UT)
Goal-oriented design of value and process models from patterns

2007-23 Peter Barna (TUE)
Specification of Application Logic in Web Information Systems

2007-24 Georgina Ramírez Camps (CWI)
Structural Features in XML Retrieval

2007-25 Joost Schalken (VU)
Empirical Investigations in Software Process Improvement
<table>
<thead>
<tr>
<th>Year</th>
<th>Author</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>2008</td>
<td>Katalin Boer-Sorbán</td>
<td>Agent-Based Simulation of Financial Markets: A modular, continuous-time approach</td>
</tr>
<tr>
<td>2008</td>
<td>Vera Hollink</td>
<td>Optimizing hierarchical menus: a usage-based approach</td>
</tr>
<tr>
<td>2008</td>
<td>Ander de Keijzer</td>
<td>Management of Uncertain Data - towards unattended integration</td>
</tr>
<tr>
<td>2008</td>
<td>Bela Mutschler</td>
<td>Modeling and simulating causal dependencies on process-aware information systems from a cost perspective</td>
</tr>
<tr>
<td>2008</td>
<td>Arjen Hommersom</td>
<td>On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective</td>
</tr>
<tr>
<td>2008</td>
<td>Peter van Rosmalen</td>
<td>Supporting the tutor in the design and support of adaptive e-learning</td>
</tr>
<tr>
<td>2008</td>
<td>Janneke Bolt</td>
<td>Bayesian Networks: Aspects of Approximate Inference</td>
</tr>
<tr>
<td>2008</td>
<td>Christof van Nimwegen</td>
<td>The paradox of the guided user: assistance can be counter-effective</td>
</tr>
<tr>
<td>2008</td>
<td>Wauter Bosma</td>
<td>Discourse oriented summarization</td>
</tr>
<tr>
<td>2008</td>
<td>Vera Kartseva</td>
<td>Designing Controls for Network Organizations: A Value-Based Approach</td>
</tr>
<tr>
<td>2008</td>
<td>Jozsef Farkas</td>
<td>A Semiotically Oriented Cognitive Model of Knowledge Representation</td>
</tr>
<tr>
<td>2008</td>
<td>Caterina Carraciolo</td>
<td>Topic Driven Access to Scientific Handbooks</td>
</tr>
<tr>
<td>2008</td>
<td>Arthur van Bunningen</td>
<td>Context-Aware Querying; Better Answers with Less Effort</td>
</tr>
<tr>
<td>2008</td>
<td>Henriette van Vugt</td>
<td>Embodied agents from a user’s perspective</td>
</tr>
<tr>
<td>2008</td>
<td>Martin Op ’t Land</td>
<td>Applying Architecture and Ontology to the Splitting and Allying of Enterprises</td>
</tr>
<tr>
<td>2008</td>
<td>Guido de Croon</td>
<td>Adaptive Active Vision</td>
</tr>
<tr>
<td>2008</td>
<td>Henning Rode</td>
<td>From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search</td>
</tr>
<tr>
<td>2008</td>
<td>Rex Arendsen</td>
<td>Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven</td>
</tr>
<tr>
<td>2008</td>
<td>Krisztian Balog</td>
<td>People Search in the Enterprise</td>
</tr>
<tr>
<td>2008</td>
<td>Henk Koning</td>
<td>Communication of IT-Architecture</td>
</tr>
<tr>
<td>2008</td>
<td>Stefan Visscher</td>
<td>Bayesian network models for the management of ventilator-associated pneumonia</td>
</tr>
<tr>
<td>2008</td>
<td>Zharko Aleksovski</td>
<td>Using background knowledge in ontology matching</td>
</tr>
<tr>
<td>2008</td>
<td>Marijn Huijbregts</td>
<td>Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled</td>
</tr>
</tbody>
</table>
2008-27 Hubert Vogten (OU)  
Design and Implementation Strategies for IMS Learning Design

2008-28 Ildiko Flesch (RUN)  
On the Use of Independence Relations in Bayesian Networks

2008-29 Dennis Reidsma (UT)  
Annotations and Subjective Machines - Of Annotators, Embodied Agents, Users, and Other Humans

2008-30 Wouter van Atteveldt (VU)  
Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content

2008-31 Loes Braun (UM)  
Pro-Active Medical Information Retrieval

2008-32 Trung H. Bui (UT)  
Toward Affective Dialogue Management using Partially Observable Markov Decision Processes

2008-33 Frank Terpstra (UvA)  
Scientific Workflow Design; theoretical and practical issues

2008-34 Jeroen de Knijf (UU)  
Studies in Frequent Tree Mining

2008-35 Ben Torben Nielsen (UvT)  
Dendritic morphologies: function shapes structure

2009-1 Rasa Jurgelenaite (RUN)  
Symmetric Causal Independence Models

2009-2 Willem Robert van Hage (VU)  
Evaluating Ontology-Alignment Techniques

2009-3 Hans Stol (UvT)  
A Framework for Evidence-based Policy Making Using IT

2009-4 Josephine Nabukunya (RUN)  
Improving the Quality of Organisational Policy Making using Collaboration Engineering

2009-5 Sietse Overbeek (RUN)  
Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality

2009-6 Muhammad Subianto (UU)  
Understanding Classification

2009-7 Ronald Poppe (UT)  
Discriminative Vision-Based Recovery and Recognition of Human Motion

2009-8 Volker Nannen (VU)  
Evolutionary Agent-Based Policy Analysis in Dynamic Environments

2009-9 Benjamin Kanagwa (RUN)  
Design, Discovery and Construction of Service-oriented Systems

2009-10 Jan Wielemaker (UvA)  
Logic programming for knowledge-intensive interactive applications

2009-11 Alexander Boer (UvA)  
Legal Theory, Sources of Law & the Semantic Web

2009-12 Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin)  
Operating Guidelines for Services

2009-13 Steven de Jong (UM)  
Fairness in Multi-Agent Systems

2009-14 Maksym Korotkiy (VU)  
From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)

2009-15 Rinke Hoekstra (UvA)  
Ontology Representation - Design Patterns and Ontologies that Make Sense

2009-16 Fritz Reul (UvT)  
New Architectures in Computer Chess

2009-17 Laurens van der Maaten (UvT)  
Feature Extraction from Visual Data

2009-18 Fabian Groffen (CWI)  
Armada, An Evolving Database System

2009-19 Valentin Robu (CWI)  
Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets
2009-20 Bob van der Vecht (UU)  
Adjustable Autonomy: Controlling Influences on Decision Making  

2009-21 Stijn Vanderlooy (UM)  
Ranking and Reliable Classification  

2009-22 Pavel Serdyukov (UT)  
Search For Expertise: Going beyond direct evidence  

2009-23 Peter Hofgesang (VU)  
Modelling Web Usage in a Changing Environment  

2009-24 Annerieke Heuvelink (VU)  
Cognitive Models for Training Simulations  

2009-25 Alex van Ballegooij (CWI)  
"RAM: Array Database Management through Relational Mapping"  

2009-26 Fernando Koch (UU)  
An Agent-Based Model for the Development of Intelligent Mobile Services  

2009-27 Christian Glahn (OU)  
Contextual Support of social Engagement and Reflection on the Web  

2009-28 Sander Evers (UT)  
Sensor Data Management with Probabilistic Models  

2009-29 Stanislav Pokraev (UT)  
Model-Driven Semantic Integration of Service-Oriented Applications  

2009-30 Marcin Zukowski (CWI)  
Balancing vectorized query execution with bandwidth-optimized storage  

2009-31 Sofiya Katrenko (UvA)  
A Closer Look at Learning Relations from Text  

2009-32 Rik Farenhorst (VU) and Remco de Boer (VU)  
Architectural Knowledge Management: Supporting Architects and Auditors  

2009-33 Khiet Truong (UT)  
How Does Real Affect Affect Affect Recognition In Speech?  

2009-34 Inge van de Weerd (UU)  
Advancing in Software Product Management: An Incremental Method Engineering Approach  

2009-35 Wouter Koelewijn (UL)  
Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling  

2009-36 Marco Kalz (OUN)  
Placement Support for Learners in Learning Networks  

2009-37 Hendrik Drachsler (OUN)  
Navigation Support for Learners in Informal Learning Networks  

2009-38 Riina Vuorikari (OU)  
Tags and self-organisation: a metadata ecology for learning resources in a multilingual context  

2009-39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin)  
Service Substitution – A Behavioral Approach Based on Petri Nets  

2009-40 Stephan Raaijmakers (UvT)  
Multinomial Language Learning: Investigations into the Geometry of Language  

2009-41 Igor Berezhnyy (UvT)  
Digital Analysis of Paintings  

2009-42 Toine Bogers (UvT)  
Recommender Systems for Social Bookmarking  

2009-43 Virginia Nunes Leal Franqueira (UT)  
Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients  

2009-44 Roberto Santana Tapia (UT)  
Assessing Business-IT Alignment in Networked Organizations  

2009-45 Jilles Vreeken (UU)  
Making Pattern Mining Useful  

2009-46 Loredana Afanasiev (UvA)  
Querying XML: Benchmarks and Recursion  

2010  

2010-1 Matthijs van Leeuwen (UU)  
Patterns that Matter  

2010-2 Ingo Wassink (UT)  
Work flows in Life Science
2010-3 Joost Geurts (CWI)
A Document Engineering Model and Processing Framework for Multimedia documents

2010-4 Olga Kulyk (UT)
Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments

2010-5 Claudia Hauff (UT)
Predicting the Effectiveness of Queries and Retrieval Systems

2010-6 Sander Bakkes (UvT)
Rapid Adaptation of Video Game AI

2010-7 Wim Fikkert (UT)
Gesture interaction at a Distance

2010-8 Krzysztof Siewicz (UL)
Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments

2010-9 Hugo Kielman (UL)
A Politie gegevensverwerking en Privacy. Naar een effectieve waarborging

2010-10 Rebecca Ong (UL)
Mobile Communication and Protection of Children

2010-11 Adriaan Ter Mors (TUD)
The world according to MARP: Multi-Agent Route Planning

2010-12 Susan van den Braak (UU)
Sensemaking software for crime analysis

2010-13 Gianluigi Folino (RUN)
High Performance Data Mining using Bio-inspired techniques

2010-14 Sander van Splunter (VU)
Automated Web Service Reconfiguration

2010-15 Lianne Bodenstaff (UT)
Managing Dependency Relations in Inter-Organizational Models

2010-16 Sicco Verwer (TUD)
Efficient Identification of Timed Automata, theory and practice

2010-17 Spyros Kotoulas (VU)
Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications

2010-18 Charlotte Gerritsen (VU)
Caught in the Act: Investigating Crime by Agent-Based Simulation

2010-19 Henriette Cramer (UvA)
People’s Responses to Autonomous and Adaptive Systems

2010-20 Ivo Swartjes (UT)
Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative

2010-21 Harold van Heerde (UT)
Privacy-aware data management by means of data degradation

2010-22 Michiel Hildebrand (CWI)
End-user Support for Access to Heterogeneous Linked Data

2010-23 Bas Steunebrink (UU)
The Logical Structure of Emotions

2010-24 Dmytro Tykhonov
Designing Generic and Efficient Negotiation Strategies

2010-25 Zulfiqar Ali Memon (VU)
Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective

2010-26 Ying Zhang (CWI)
XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines

2010-27 Marten Voulon (UL)
Automatisch contracteren

2010-28 Arne Koopman (UU)
Characteristic Relational Patterns

2010-29 Stratos Idreos (CWI)
Database Cracking: Towards Auto-tuning Database Kernels

2010-30 Marije van Erp (UvT)
Accessing Natural History - Discoveries in data cleaning, structuring, and retrieval

2010-31 Victor de Boer (UvA)
Ontology Enrichment from Heterogeneous Sources on the Web
2010-32 Marcel Hiel (UvT)  
An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems

2010-33 Robin Aly (UT)  
Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval

2010-34 Teduh Dirgahayu (UT)  
Interaction Design in Service Compositions

2010-35 Dolf Trieschnigg (UT)  
Proof of Concept: Concept-based Biomedical Information Retrieval

2010-36 Jose Janssen (OU)  
Paving the Way for Lifelong Learning; Facilitating competence development through a learning path specification

2010-37 Niels Lohmann (TUE)  
Correctness of services and their composition

2010-38 Dirk Fahland (TUE)  
From Scenarios to components

2010-39 Ghazanfar Farooq Siddiqui (VU)  
Integrative modeling of emotions in virtual agents

2010-40 Mark van Assem (VU)  
Converting and Integrating Vocabularies for the Semantic Web

2010-41 Guillaume Chaslot (UM)  
Monte-Carlo Tree Search

2010-42 Sybren de Kinderen (VU)  
Needs-driven service bundling in a multi-supplier setting - the computational e-service approach

2010-43 Peter van Kranenburg (UU)  
A Computational Approach to Content-Based Retrieval of Folk Song Melodies

2010-44 Pieter Bellekens (TUE)  
An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain

2010-45 Vasilios Andrikopoulos (UvT)  
A theory and model for the evolution of software services

2010-46 Vincent Pijpers (VU)  
e3alignment: Exploring Inter-Organizational Business-ICT Alignment

2010-47 Chen Li (UT)  
Mining Process Model Variants: Challenges, Techniques, Examples

2010-48 Milan Lovric (EUR)  
Behavioral Finance and Agent-Based Artificial Markets

2010-49 Jahn-Takeshi Saito (UM)  
Solving difficult game positions

2010-50 Bouke Huurnink (UvA)  
Search in Audiovisual Broadcast Archives

2010-51 Alia Khairia Amin (CWI)  
Understanding and supporting information seeking tasks in multiple sources

2010-52 Peter-Paul van Maanen (VU)  
Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention

2010-53 Edgar Meij (UvA)  
Combining Concepts and Language Models for Information Access

2011-1 Botond Cseke (RUN)  
Variational Algorithms for Bayesian Inference in Latent Gaussian Models

2011-2 Nick Tinnemeier (UU)  
Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language

2011-3 Jan Martijn van der Werf (TUE)  
Compositional Design and Verification of Component-Based Information Systems

2011-4 Hado van Hasselt (UU)  
Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference learning algorithms
2011-5 Base van der Raadt (VU)  
Enterprise Architecture Com-  
ing of Age - Increasing the Per-  
formance of an Emerging Disci-  
pline.

2011-6 Yiwen Wang (TUE)  
Semantically-Enhanced Rec-  
ommendations in Cultural  
Heritage

2011-7 Yujia Cao (UT)  
Multimodal Information Pre-  
sentation for High Load Hu-  
man Computer Interaction

2011-8 Nieske Vergunst (UU)  
BDI-based Generation of Ro-  
bust Task-Oriented Dialogues

2011-9 Tim de Jong (OU)  
Contextualised Mobile Media  
for Learning

2011-10 Bart Bogaert (UvT)  
Cloud Content Contention

2011-11 Dhaval Vyas (UT)  
Designing for Awareness: An  
Experience-focused HCI Pers-  
pective

2011-12 Carmen Bratosin (TUE)  
Grid Architecture for Distri-  
buted Process Mining

2011-13 Xiaoyu Mao (UvT)  
Airport under Control. Multi-  
agent Scheduling for Airport  
Ground Handling

2011-14 Milan Lovric (EUR)  
Behavioral Finance and Agent-  
Based Artificial Markets

2011-15 Marijn Koolen (UvA)  
The Meaning of Structure: the  
Value of Link Evidence for In-  
formation Retrieval

2011-16 Maarten Schadd (UM)  
Selective Search in Games of  
Different Complexity

2011-17 Jiyin He (UvA)  
Exploring Topic Structure: Co-  
herence, Diversity and Relat-  
edness

2011-18 Mark Ponsen (UM)  
Strategic Decision-Making in  
complex games

2011-19 Ellen Rusman (OU)  
The Mind’s Eye on Personal  
Profiles

2011-20 Qing Gu (VU)  
Guiding service-oriented soft-  
ware engineering - A view-  
based approach

2011-21 Linda Terlouw (TUD)  
Modularization and Specifi-  
cation of Service-Oriented Sys-  
tems

2011-22 Junte Zhang (UvA)  
System Evaluation of Archival  
Description and Access

2011-23 Wouter Weerkamp (UvA)  
Finding People and their Ut-  
erances in Social Media

2011-24 Herwin van Welbergen (UT)  
Behavior Generation for In-  
terpersonal Coordination with  
Virtual Humans On Specifying,  
Scheduling and Realizing Multi-  
modal Virtual Human Behavior

2011-25 Syed Waqar ul Qounain Jaffry (VU)  
Analysis and Validation of  
Models for Trust Dynamics

2011-26 Matthijs Aart Pontier (VU)  
Virtual Agents for Human  
Communication - Emotion  
Regulation and Involvement-  
Distance Trade-Offs in Em-  
bodied Conversational Agents  
and Robots

2011-27 Aniel Bhulai (VU)  
Dynamic website optimization  
through autonomous manage-  
ment of design patterns

2011-28 Rianne Kaptein (UvA)  
Effective Focused Retrieval by  
Exploiting Query Context and  
Document Structure

2011-29 Faisal Kamiran (TUE)  
Discrimination-aware Classifi-  
cation

2011-30 Egon van den Broek (UT)  
Affective Signal Processing  
(ASP): Unraveling the mystery  
of emotions

2011-31 Ludo Waltman (EUR)  
Computational and Game-  
Theoretic Approaches for Mod-  
eling Bounded Rationality

2011-32 Nees-Jan van Eck (EUR)  
Methodological Advances in
Bibliometric Mapping of Science

2011-33 Tom van der Weide (UU)
Arguing to Motivate Decisions

2011-34 Paolo Turrini (UU)
Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations

2011-35 Maaie Harbers (UU)
Explaining Agent Behavior in Virtual Training

2011-36 Erik van der Spek (UU)
Experiments in serious game design: a cognitive approach

2011-37 Adriana Burlutiu (RUN)
Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference

2011-38 Nyree Lemmens (UM)
Bee-inspired Distributed Optimization

2011-39 Joost Westra (UU)
Organizing Adaptation using Agents in Serious Games

2011-40 Viktor Clerc (VU)
Architectural Knowledge Management in Global Software Development

2011-41 Luan Ibraimi (UT)
Cryptographically Enforced Distributed Data Access Control

2011-42 Michal Sindlar (UU)
Explaining Behavior through Mental State Attribution

2011-43 Henk van der Schuur (UU)
Process Improvement through Software Operation Knowledge

2011-44 Boris Reuderink (UT)
Robust Brain-Computer Interfaces

2011-45 Herman Stehouwer (UvT)
Statistical Language Models for Alternative Sequence Selection

2011-46 Beibei Hu (TUD)
Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work

2011-47 Azizi Bin Ab Aziz (VU)
Exploring Computational Models for Intelligent Support of Persons with Depression

2011-48 Mark Ter Maat (UT)
Response Selection and Turn-taking for a Sensitive Artificial Listening Agent

2011-49 Andreea Niculescu (UT)
Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality

2012

2012-1 Terry Kakeeto (UvT)
Relationship Marketing for SMEs in Uganda

2012-2 Muhammad Umair (VU)
Adaptivity, emotion, and Rationality in Human and Ambient Agent Models

2012-3 Adam Vanya (VU)
Supporting Architecture Evolution by Mining Software Repositories

2012-4 Jurriaan Souer (UU)
Development of Content Management System-based Web Applications

2012-5 Marijn Plomp (UU)
Maturing Interorganisational Information Systems

2012-6 Wolfgang Reinhardt (OU)
Awareness Support for Knowledge Workers in Research Networks

2012-7 Rianne van Lambalgen (VU)
When the Going Gets Tough: Exploring Agent-based Models of Human-like Performance under Demanding Conditions

2012-8 Gerben Klaas Dirk de Vries (UvA)
Kernel Methods for Vessel Trajectories