Data mining in dynamically evolving systems via diffusion methodologies

Thesis submitted for the degree of “Doctor of Philosophy”

by

Neta Rabin

This thesis was carried out under the supervision of Prof. Amir Averbuch

Submitted to the Senate of Tel Aviv University March, 2010
Abstract

In this thesis, we describe a new approach for detecting and tracking the behavior of dynamical systems. The goal is to identify trends that deviate from normal behavior. A high-dimensional dataset, which describes the measured/observed parameters of a dynamical system, is embedded into a lower-dimension space by the application of the diffusion maps to it. Then, the sought after anomalies are detected in this lower-dimension embedded space. To achieve it, the diffusion maps methodology was extended to provide hierarchal (multi-scale) processing via the construction of super graphs. The frequency appearance of each point in the embedded space is quantitatively measures the system’s state at each given time point. In addition, the data was reformulated to extract its hidden underlying oscillatory behavior which turns out to be a major source for failure of dynamical systems. Most of the presented algorithms have two sequential steps: training and detection. Graph alignment was developed to signal when the current computed training baseline data has to be updated. The classification of the status of each newly arrived data point as normal or abnormal depends on its location coordinates. These coordinates are determined by the application of a multi-scale Gaussian approximation procedure.

In a second part of the thesis, we propose three novel algorithms for the detection of vehicles based on their recordings. The algorithms use different methods like wavelet packets and random projections to extract spatio-temporal characteristic features from the recordings. These form a unique acoustic signature for each of the recordings. The feature extraction procedure is followed by a dimensionality reduction step for reducing the size of the signature. The methods that are utilized are diffusion maps, classification trees and principal component analysis. A new recording can be classified by employing similar steps in the low dimensional structures, which were constructed on a training set, are extended to the newly collected acoustic recordings.
The proposed algorithms are generic since they fit a large number of related problems where abnormal behavior of dynamically evolving system has to be predicted based upon some knowledge of its typical (normal) behavior.
Acknowledgments

I owe my deepest gratitude and appreciation to my advisor Prof. Amir Averbuch for his constant encouraging, for his invaluable academic guidance, patience, generosity and for his friendship.

I am much obliged to Prof. Raphy Coifman from Yale University for inviting me to Yale, for his insights and inspirations that impacted this work and for his kindness.

Many thanks to Prof. Yehuda Roditty for his help throughout my work.

I am grateful to my husband, my children, my parents and the whole family for their love and support and for encouraging me to follow this path.

Many thanks my friend and colleague Gil David, for his advice throughout this research and for being a great friend.

I appreciate the fruitful discussions with my lab members and I thank them for the time we spent together.

During the course of this work I was supported by the Eshkol Fellowship, administered by the Israeli Ministry of Science.
Contents

I Introduction and background 11

1 Introduction 12
  1.1 Background and motivation ........................................ 12
  1.2 Structure of the thesis .............................................. 17

2 Mathematical background 20
  2.1 The diffusion maps framework ................................. 20
     2.1.1 Diffusion maps ............................................... 20
     2.1.2 Diffusion kernels ............................................. 22
     2.1.3 Choosing the scale parameter ε ............................ 23
     2.1.4 The geometric harmonics .................................... 24
     2.1.5 The multi-scale pyramid approximation and interpolation method 26
  2.2 Wavelet and wavelet packet transforms ....................... 28

II Hierarchical data mining approach for detection and prediction of anomalies in dynamically evolving systems 32

3 Introduction 33
  3.1 Background and motivation ...................................... 33
  3.2 Novelty and contribution ......................................... 35

4 Evaluation dataset - performance of a monitored application 39
  4.1 Implementation of the learning phase ......................... 40
### CONTENTS

4.1.1 Hierarchical three level embedding model ........................................ 41
4.1.2 Application of the diffusion maps to a single parameter - bottom level processing ................................................................. 43
4.1.3 Application of the diffusion maps to a group of parameters: middle level processing ................................................................. 53
4.1.4 Creating a super-embedding graph: top level processing .................. 56
4.2 The online detection phase using the multi-scale pyramid interpolation .. 57
4.3 Tracking and detection of anomalies (problems) .............................. 64
4.4 Comparisons with other dimensionality reduction methods ............... 72
  4.4.1 Characterizing the geometry and structure of the dataset .............. 73
  4.4.2 Comparison with PCA ....................................................... 74
  4.4.3 Comparison with MDS ....................................................... 80
  4.4.4 Comparing between DM, PCA and MDS to track and detect system failures ................................................................. 86
4.5 Discussions ......................................................................................... 87
4.6 Summary ............................................................................................ 88

5 Updating the training dataset ................................................................. 89
  5.1 Introduction ....................................................................................... 89
  5.2 Updating the training set of the operational system ......................... 90
    5.2.1 Merging embedding graphs ...................................................... 91
    5.2.2 Comparing between embedded datasets .................................. 101
5.3 Discussions ......................................................................................... 105
5.4 Summary ............................................................................................ 106

III Dimensionality reduction for detection of moving vehicles .............. 107

6 Introduction ......................................................................................... 108
  6.1 Background and Motivation ............................................................ 108
  6.2 Related work ................................................................................... 109
  6.3 Contributions and novelty .............................................................. 110
7 Algorithm I: Detection of moving vehicles via dimensionality reduction 112
  7.1 Dimensionality reduction via random projections . . . . . . . . . . . . . 113
  7.2 The learning phase . . . . . . . . . . . . . . . . . . . . . . . . . . . . 114
  7.3 The classification phase . . . . . . . . . . . . . . . . . . . . . . . . . . 117
  7.4 Experimental results . . . . . . . . . . . . . . . . . . . . . . . . . . . 119
  7.5 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 129

8 Algorithm II: Wavelet based acoustic detection of moving vehicles 131
  8.1 The structure of the acoustics signals . . . . . . . . . . . . . . . . . . . . 132
  8.2 Formulation of the approach . . . . . . . . . . . . . . . . . . . . . . . . 135
    8.2.1 Outline of the approach . . . . . . . . . . . . . . . . . . . . . . . 136
  8.3 Description of the algorithm and its implementation . . . . . . . . . . . . 137
    8.3.1 Implementation: extraction of characteristic features . . . . . . . 138
    8.3.2 Building the classification and regression trees (CARTs) . . . . . 141
    8.3.3 Identification of an acoustic signal . . . . . . . . . . . . . . . . . 142
  8.4 Experimental results . . . . . . . . . . . . . . . . . . . . . . . . . . . . 144
    8.4.1 Detection experiments . . . . . . . . . . . . . . . . . . . . . . . 145
  8.5 Random search for a near optimal footprint (RSNOFP) scheme . . . . . 152

9 Algorithm III: A diffusion framework for detection of moving vehicles 157
  9.1 The learning step . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 158
  9.2 The classification step . . . . . . . . . . . . . . . . . . . . . . . . . . . . 160
  9.3 Experimental results . . . . . . . . . . . . . . . . . . . . . . . . . . . . 162

10 Summary and comparison 168
## List of Algorithms

<table>
<thead>
<tr>
<th></th>
<th>Algorithm</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Multi-scale pyramid approximation</td>
<td>27</td>
</tr>
<tr>
<td>2</td>
<td>Multi-scale pyramid extension for defining the coordinates of a newly ar-</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>rived data point</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Merge embedded graphs</td>
<td>92</td>
</tr>
<tr>
<td>4</td>
<td>Graph alignment (GA)</td>
<td>92</td>
</tr>
<tr>
<td>5</td>
<td>Compare graph structures</td>
<td>102</td>
</tr>
<tr>
<td>6</td>
<td>Learning acoustic features using random projections</td>
<td>115</td>
</tr>
<tr>
<td>7</td>
<td>Classification of acoustic features using random projections</td>
<td>118</td>
</tr>
<tr>
<td>8</td>
<td>Learning acoustic features by wavelets analysis</td>
<td>159</td>
</tr>
<tr>
<td>9</td>
<td>Classification of acoustic features by wavelet analysis</td>
<td>161</td>
</tr>
</tbody>
</table>
Part I

Introduction and background
Chapter 1

Introduction

1.1 Background and motivation

The thesis describes a framework for detecting and tracking the behavior in datasets that were captured from dynamically evolving systems. This framework, which includes theory and algorithms that are based upon diffusion processes and dimensionality reduction, is introduced. These methodologies are applied to two large and heterogeneous datasets from different fields.

Data mining via dimensionality reduction occupies a central role in many fields such as compression and coding, statistics, machine learning, image and video processing, sampling of networking systems, fault diagnosis, performance analysis and many more. In essence, the goal is to change the representation of the captured datasets, originally in a form involving a large number of independent variables (features), which dynamically changing, into a low-dimensional description using only a small number of free parameters. The relevant low-dimensional features enable us to gain insight into our understanding of the phenomenon that generated and governs the inspected data.

Application of data mining and machine learning algorithms to high dimensional data, which is sampled from a time-series data stream, can impose some challenges. Here is a list of some:

1. *True representation of the dataset* such that the outcome from the application of a machine learning algorithm to a time-series dataset captures the underlying process
that generates and characterizes the data.

2. *Meaningful embedding of the dataset* such that machine learning algorithms, which are based on dimensionality reduction, will embed the data in a faithful manner by preserving local mutual distances in the low-dimensional space.

3. *Rescaling the dataset* since the sampled data is assumed to be heterogeneous and the input parameters may have different scales. In order to combine and fuse the recorded information, the dataset has to be rescaled. The rescaling should not change the meaning of the processed dataset.

4. *Classification of newly arrived data points* by assigning them correctly to the previously embedded clusters.

5. *Dynamical changes of the system behavior* happen when many dynamical applications collect real data. Then, the data volume may grow with time and new phenomenon emerges, which is governed by external forces or by change of environment. This has to be added to the known system phenomenon that was captured in the training phase. For maintaining a reliable learning system, the outcome from a machine learning algorithm has to be updated to incorporate the new detected behavior.

6. *Learning large volume of data* by choosing the right sub-sampling points from the large volume while extending faithfully the computation from the smaller subspace to the whole data volume via out-of-sample extension.

7. *Identification of a system problem* is done in a feature space, which is usually of a low dimension. Problems are identified in the feature space. Often, it is needed to go back to the original space to rate the parameters that caused the abnormal system behavior. Preferably, the method should have in it a way to rate the parameters, which caused the anomaly in the original space.

The methodology, which is presented in this thesis, addresses these challenges.

Classical dimensionality reduction techniques like principal component analysis (PCA) [43, 47] or multidimensional scaling (MDS) [26, 49] fail on datasets that have a
complex geometric structure with a low intrinsic dimensionality. Recently, a great deal of attention has been directed to the so-called “kernel methods” such as local linear embedding [64], Laplacian eigenmaps [9, 10], Hessian eigenmaps [32] and local tangent space alignment [79]. These algorithms exhibit two major advantages over classical methods: they are nonlinear and they preserve locality. The first aspect is essential since most of the time, the data points in their original form do not lie on linear manifolds. The second point is the expression of the fact that in many applications, distances of points, which are far apart, are meaningless. Therefore, they do not have to be preserved. In the diffusion framework [21, 22], which is used in this work, the ordinary Euclidean distance in the embedding space measures intrinsic diffusion metrics in the original high-dimensional data such that points, which are located far from the main cluster in the embedded space, were also isolated in the original space. The method is non-linear and it finds efficiently distances on non-linear manifolds. The eigenfunctions of Markov matrices and associated family of diffusion distances are used to construct diffusion maps [21] that generate efficient representations of complex geometric structures. In particular, the first top eigenfunctions provide a low-dimensional geometric embedding of the set so that the ordinary Euclidean distance in the embedding space measures intrinsic diffusion metrics in the data.

The diffusion maps framework [21] unifies ideas arising in a variety of contexts such as machine learning, spectral graph theory and kernel methods. Diffusion maps have been applied to clustering and classification problems [51, 73]. In [50, 48], diffusion maps were used for data fusion and multicue data matching. Image processing and signal de-noising applications were implemented by diffusion methods in [69, 72]. Recently, tomographic reconstruction was achieved in [25] by utilizing diffusion maps. There are only several published examples of utilizing diffusion maps to time series data. Classification of fMRI Time Series was done in [68, 54]. Anomaly detection and classification in hyper-networks by diffusion methodologies was presented in [28]. Recently, diffusion maps were used in [15] to model hurricane tracks.

This thesis shows how to utilize the diffusion maps methodology to process in an unsupervised way high dimensional time series datasets, which dynamically change all the time. The unsupervised algorithms have two major steps:

1. Training step in which the system’s profile is generated studied and analyzed from a training dataset via dimensionality reduction. This step also performs data rescaling
and normalization if needed. The embedding of the training dataset forms a low-
dimensional manifold/s, which represents faithfully the high dimensional source
dataset.

2. The detection step uses the training profile and the embedding coordinates to find
whether each newly arrived data point resides in or outside one of the profile’s clus-
ters.

Additional algorithms, which capture long term changes in the system’s behavior, are
introduced.

We focus on two types of related applications whose datasets are dynamically change.
The first type fits high dimensional datasets that describe the behavior of a monitored sys-
tem. For example, this task appears in monitoring of network activities, in fault diagnosis
of a unit system and in performance monitoring. To achieve this goal, the system’s behav-
ior is studied in an off-line manner while detection and prediction of the system’s faults
are done online. The constructed algorithms implement an hierarchical (multi-scale) ap-
proach, which embeds the system into a lower-dimension space that allows us to track
the failures sources. The second type of applications, which the presented methodologies
are applied to, are problems that arise in signal processing applications. In these types of
problems, the data can be very noisy with different types of noises. The goal is to produce
a good feature extractor in order to classify the recorded data into different predetermined
classes. This task often involves the application of dimensionality reduction.

The acoustic processing problems are good examples how different components from
machine learning, data mining, dimensionality reduction and classification are joined to-
gether to provide generic coherent solutions.

The first applicative example, which demonstrates the hierarchical data mining ap-
proach, detects and predicts anomalies in a dataset that was assembled from a performance
monitor. Many performance monitors application systems today are rule based and depend
on manual tuning of thresholds. These systems suffer from two fundamental weaknesses:
First, each rule or threshold usually checks a simple connection (combination) between a
limited number of input parameters. Second, rules are created by the system developers
based on their experience and assumptions regarding the system’s behavior. These rule
based systems often miss new anomalies since these linear methods cannot capture all the
significant connections among the input parameters. Rule based systems are inflexible to adapt themselves dynamically to constant changes in the extracted parameters. Since the proposed method is data driven, the user does not need to rate the importance of each input parameter in the training phase and there is no need to define thresholds. It is all done automatically. Non-linear connections between all the input parameters are examined simultaneously without introducing our own bias and can be visualized in a low dimensional space. This allows us to detect the emergence of new unknown anomalies.

The performance monitor example, which is presented in this thesis as a running example to illustrate the theory, collects a dataset that consists of several parameters. The parameters measure different components in the monitored system. They are measured in different scales, therefore, they have to be rescaled. The scaling procedure involves the application of the diffusion maps to each parameter followed by a construction of a frequency score function. The rescaling process transforms the inputs into a probability space. Their probabilities are measured in a reliable low-dimensional space, which describes the parameter’s dynamical behavior. The same process is applied again to groups of parameters, which measure similar system components and finally to all of the input parameters. This application constructs an hierarchical tree structure. The bottom level in the hierarchy describes the behavior of a single system parameter, the intermediate hierarchy level describes the joint behavior of groups of parameters and the top level of the hierarchy holds a low-dimensional embedding that characterizes the entire system. New points can be dynamically added to the hierarchical structure and abnormal behavior can be detected on-line. The hierarchical structure allows us to track the parameters and the corresponding system components that caused a system failure.

The second applicative example are acoustic datasets. These datasets contain recordings of vehicles and the surrounding environmental noises in various conditions. The goal is to automatically detect the presence of a vehicle. The dataset is varied. The recorded vehicles travel at different speeds, on different types of roads and many types of background noises like wind, planes and speech are present. In addition, the recordings were sampled in a low-resolution. The challenge is to extract characteristic features for discriminating between vehicles and non-vehicles in these unstable varied conditions. In this thesis, we present three solutions for this problem. Wavelet packets [27, 74, 53] are used for the signature construction of signals in two algorithms. These algorithms also employ
The proposed framework in this thesis is generic since it fits a large number of related problems where abnormal behavior of dynamically evolving system has to be predicted based upon some knowledge of its typical (normal) behavior.

1.2 Structure of the thesis

The organization of the thesis and the relationships among its main parts are illustrated by the flowcharts in Figs. 1.2.1 and 1.2.2.

Part I provides an overview of the subject and describes the main mathematical tools that are used. These tools include diffusion maps, geometric harmonics, pyramidal multi-scale approximation and wavelet packets.

Part II describes the hierarchal data mining approach for dynamically evolving data where the focus is on performance monitoring of transaction based systems. Chapter 3 overviews the challenges of tracking and detecting problems in a monitored transaction based system. It briefly describes the proposed learning method and explains the applicative example and its associated dataset. Chapter 4 introduces the hierarchical off-line learning scheme and its on-line extension to new data points. The application of the hierarchical learning method to track a specific system failure is described in Section 4.3. Chapter 5 uses spectral graph matching schemes to solve problems which occur when time series systems that capture large data volumes are examined. The main addressed issue is when the training dataset, which was created in the learning phase for ongoing analysis of the activities of the system, has to be updated because its current profile does not fit the system’s behavior. The hierarchical learning structure, which is constructed in Chapter 4 and applied to a dataset from a performance monitor application, is used to demonstrate the algorithms that are presented in Chapter 5. Figure 1.2.1 links between the content and the structure of Part II with the mathematical tools that are used in different chapters.
Part III describes three signal processing algorithms that were applied to acoustic data, which contains acoustic recordings of moving vehicles. Chapter 7 presents an algorithm that uses random projections for extracting features from the acoustic signals. Chapter 8 describes a wavelet packet based algorithm for vehicle classification. The third acous-
tic detection algorithm, which uses wavelet packets and diffusion maps, is presented in Chapter 9. The flowchart in Fig. 1.2.2 shows the organization of this part.

Figure 1.2.2: Organization of Part III
Chapter 2

Mathematical background

2.1 The diffusion maps framework

2.1.1 Diffusion maps

This section describes the diffusion framework that was introduced in [21]. Diffusion maps and diffusion distances provide a method for finding meaningful geometric descriptions in datasets. In most cases, the dataset consists of points in $\mathbb{R}^n$ where $n$ is large. The diffusion maps construct coordinates that parameterize the dataset and the diffusion distance provides a local preserving metric for this dataset. A non-linear dimensionality reduction, which reveals global geometric information, is constructed by local overlapping structures. The following provides a more formal description.

Let $\Gamma$ be a set of points in $\mathbb{R}^n$. In order to study the intrinsic geometry of the set $\Gamma$, we construct a graph $G = (\Gamma, W)$ with a kernel $W \triangleq w(x, y)$. The kernel, which is a weight function, measures the pairwise similarity between the points that satisfies the following properties: symmetry: $w(x, y) = w(y, x)$, positive-preserving: $w(x, y) \geq 0$ for all $x$ and $y$ in $\Gamma$, and positive semi-definite: for all real-valued bounded function $f$ defined on $\Gamma$ $\sum_{x \in \Gamma} \sum_{y \in \Gamma} w(x, y) f(x) f(y) \geq 0$.

Choosing a kernel that satisfies the above properties allows us to normalize $W$ into a Markov transition matrix $P$ which is constructed as
\[ P \triangleq p(x, y) = \frac{w(x, y)}{d(x)}, \quad d(x) = \sum_{y \in \Gamma} w(x, y) \quad (2.1.1) \]

where \( d(x) \) is the degree of the node \( x \). In spectral graph theory, this normalization is known as the weighted graph Laplacian normalization \([20]\). Since \( P \) consists of nonnegative real numbers, where each row is summed to 1, the matrix \( P \) can be viewed as the probability to move from \( x \) to \( y \) in one time step. These probabilities measure the connectivity of the points within the graph. Raising \( P \) to the power \( t \), denoted by \( P_t \), measures the connectivity between the points after \( t \) time steps. If the graph of the data is connected, then the stationary distribution of the Markov chain is proportional to the degree of \( x \) in the graph. In other words, \( \lim_{t \to \infty} p_t(x, y) = \phi_0(y) \) where \( \phi_0(y) = \frac{d(y)}{\sum_{z \in \Gamma} d(z)} \).

Moreover, the Markov chain is reversible since it satisfies \( \phi_0(x)p(x, y) = \phi_0(y)p(y, x) \).

The transition matrix \( P \) is conjugate to a symmetric matrix \( A \) given by

\[ a(x, y) = \sqrt{d(x)p(x, y)} \frac{1}{\sqrt{d(y)}}. \quad (2.1.2) \]

In a matrix notation, \( A \triangleq D^{\frac{1}{2}}PD^{-\frac{1}{2}} \), where \( D \) is the diagonal matrix with the values \( \sum_y w(x, y) \) on its diagonal. The symmetric matrix \( A \) has \( n \) real eigenvalues \( \{\lambda_k\}_{k=0}^{n-1} \) and a set of orthogonal eigenvectors \( \{v_k\} \) in \( \mathbb{R}^n \), thus, it has the following spectral decomposition

\[ a(x, y) = \sum_{k \geq 0} \lambda_k v_k(x)v_k(y). \quad (2.1.3) \]

Since \( P \) is conjugate to \( A \), the eigenvalues of both matrices are identical. In addition, if \( \{\phi_k\} \) and \( \{\psi_k\} \) are the corresponding left and right eigenvectors of \( P \), then we have

\[ \phi_k = D^{\frac{1}{2}}v_k, \quad \psi_k = D^{-\frac{1}{2}}v_k. \quad (2.1.4) \]

From the orthonormality of \( \{v_k\} \) and Eq. (2.1.4), it follows that \( \{\phi_k\} \) and \( \{\psi_k\} \) are biorthonormal which means that \( \langle \phi_l, \psi_k \rangle = \delta_{lk} \). Combining Eqs. (2.1.3) and (2.1.4) with the biorthogonality of \( \{\phi_k\} \) and \( \{\psi_k\} \) lead to the following eigendecomposition of the
transition matrix $P$
\[ p_t(x, y) = \sum_{k \geq 0} \lambda_k^t \psi_k(x) \phi_k(y). \quad (2.1.5) \]

Because of the fast decay of the spectrum, only a few terms are required to achieve sufficient accuracy in the sum. The family of diffusion maps $\{\Psi_t\}$, which are defined by
\[ \Psi_t(x) = (\lambda_1^t \psi_1(x), \lambda_2^t \psi_2(x), \lambda_3^t \psi_3(x), \cdots), \quad (2.1.6) \]
embeds the dataset into Euclidean space. This embedding is a new parametrization of the data into a lower dimension space. We also recall the diffusion distance as was defined in [59, 21]. The diffusion distance between two data points $x$ and $y$ is the weighted $L^2$ distance
\[ D_t^2(x, y) = \sum_{z \in \Gamma} \frac{(p_t(x, z) - p_t(z, y))^2}{\phi_0(z)}. \quad (2.1.7) \]

This distance reflects the geometry of the dataset as the value of $\frac{1}{\phi_0(x)}$ depends on the points’ density. Two data points are close if there is a large number of paths that connect them. In this case, the diffusion distance is small. Substituting Eq. (2.1.5) in Eq. (2.1.7) together with the biorthogonality property, then, the diffusion distance with the right eigenvectors of the transition matrix $P$ is expressed as
\[ D_t^2(x, y) = \sum_{k \geq 1} \lambda_k^2 \psi_k(x) - \psi_k(y))^2. \quad (2.1.8) \]

In these new coordinates, the Euclidean distance between two points in the embedded space represents the distances between the two high-dimensional points as defined by a random walk.

### 2.1.2 Diffusion kernels

Section 2.1.1 described the construction of diffusion maps and diffusion distance. The formation of this new coordinate set for the data and the metric defined on it may vary according to the way the kernel is defined. We look at kernels of the form $w_c(x, y) = h(\frac{1}{2\sigma^2} ||x-y||^2)$ that are direction independent. A common choice is the Gaussian kernel.
2.1. THE DIFFUSION MAPS FRAMEWORK

The diffusion maps framework was introduced in [23]. This kernel can now be normalized in a number of ways. A general form of a kernel, with a parameter $\alpha$ that controls the normalization type, given by

$$
\begin{align*}
  w^{(\alpha)}_{\epsilon}(x, y) &= \frac{w_{\epsilon}(x, y)}{q^{\alpha}(x) q^{\alpha}(y)}, \quad q(x) = \sum_{y \in \Gamma} w_{\epsilon}(x, y)
\end{align*}
$$

was introduced in [23]. Then, the transition matrix is defined by

$$
\begin{align*}
  p^{\alpha}_{\epsilon}(x, y) &= \frac{w^{(\alpha)}_{\epsilon}(x, y)}{d^{(\alpha)}_{\epsilon}(y)}
\end{align*}
$$

where

$$
\begin{align*}
  d^{(\alpha)}_{\epsilon}(y) &= \sum_{x \in \Gamma} w^{(\alpha)}_{\epsilon}(x, y).
\end{align*}
$$

The value of the parameter $\alpha$ defines the effect of the density in the diffusion kernel. The asymptotic behavior, which is obtained as $\epsilon \rightarrow 0$, generates different infinitesimal operator for different values of $\alpha$. Coifman et. al. showed in [21, 58, 23] that the eigenfunction of the Markov matrix $P^{\alpha}_{\epsilon}$ approximates a symmetric Schrödinger operator. Three interesting values of $\alpha$ are used:

1. $\alpha = 0$ is the classical normalized graph Laplacian [20];

2. $\alpha = 1$ together with the assumption that the points of $\Gamma$ lie on a manifold in $\mathbb{R}^n$ generate a diffusion matrix that approximates the Laplace-Beltrami operator [21];

3. For $\alpha = \frac{1}{2}$ the diffusion of the Fokker-Planck equation is approximated [58].

The choice of the parameter $\alpha$ varies from one application to another according to the data type and the a-priori knowledge regarding its behavior.

2.1.3 Choosing the scale parameter $\epsilon$

The diffusion maps reveals and extracts global information in local structures. The scale parameter $\epsilon$ determines the size of the local neighborhood of each point. In this thesis, we used the method that was presented in [65, 66] for setting the value of $\epsilon$. A large $\epsilon$ defines a wide neighborhood and thus produces a coarse analysis of the data since most of
the neighborhoods will contain a large number of data points. In contrast, for a small \( \epsilon \), many neighborhoods will contain a single data point. Clearly, an appropriate \( \epsilon \) should be between these two cases and should stem from the data. For a dataset \( \Gamma \), which contains \( m \) data points, the pairwise Euclidean distance matrix between data points in \( \Gamma \) is defined as \( \bar{D} = \{d_{ij}\}_{i,j=1...m} \). The median determines \( \epsilon \) as

\[
\epsilon = \text{median}\{d_{ij}\}_{i,j=1,...,m}.
\] (2.1.12)

The median of \( \bar{D} \) provides an estimate to the average pairwise distance that is robust to outliers.

### 2.1.4 The geometric harmonics

Geometric harmonics (GH) \[22\] is a method that extends the low dimensional embedding for new data points. Let \( \Gamma \) be a set of points in \( \mathbb{R}^n \) and \( \Psi_t \) be its diffusion embedding map. Let \( \bar{\Gamma} \) be a set in \( \mathbb{R}^n \) such that \( \Gamma \subseteq \bar{\Gamma} \). The GH scheme extends \( \Psi_t \) into a new dataset \( \bar{\Gamma} \). We first overview the Nyström extension \[60, 29, 63\], which is a common method for the extension of functions from a training set to accommodate the arrival of new samples \[11, 75, 36\]. The eigenvectors and eigenvalues of a Gaussian kernel on the training set \( \Gamma \) with \( \epsilon \) are computed by

\[
\lambda_l \varphi_l(x) = \sum_{y \in \Gamma} e^{-\frac{\|x-y\|^2}{2 \epsilon}} \varphi_l(y), \ x \in \Gamma.
\] (2.1.13)

If \( \lambda_l \neq 0 \), the eigenvectors in Eq. 2.1.13 can be extended to any \( x \in \mathbb{R}^n \) by

\[
\bar{\varphi}_l(x) = \frac{1}{\lambda_l} \sum_{y \in \Gamma} e^{-\frac{\|x-y\|^2}{2 \epsilon}} \varphi_l(y), \ x \in \mathbb{R}^n.
\] (2.1.14)

This is known as the Nyström extension. The extended distance of \( \bar{\varphi}_l \) from the training set is proportional to \( \epsilon \). Let \( f \) be a function on the training set \( \Gamma \). In our case, we are interested in extending each of the coordinates of the embedding function \( \Psi_t = (\lambda_1 \psi_1(x), \lambda_2 \psi_2(x), \lambda_3 \psi_3(x), \ldots) \). The eigenfunctions \( \{\varphi_l\} \) are the outcome of the spectral decomposition of a symmetric positive matrix, thus, they form a basis in \( \mathbb{R}^n \). Conse-
2.1. THE DIFFUSION MAPS FRAMEWORK

subsequently, any function \( f \) can be written as a linear combination

\[
f(x) = \sum_l \langle \varphi_l, f \rangle \varphi_l(x), \ x \in \Gamma
\]  

(2.1.15)

of this basis. Using the Nyström extension, as given in Eq. 2.1.14, \( f \) can be defined for any point in \( \mathbb{R}^n \) by

\[
\tilde{f}(x) = \sum_l \langle \varphi_l, f \rangle \varphi_l(x), \ x \in \mathbb{R}^n.
\]  

(2.1.16)

The above extension allows us to decompose each diffusion maps coordinate \( \psi_i \) in the same way using

\[
\psi_i(x) = \sum_l \langle \varphi_l, \psi_i \rangle \varphi_l(x), \ x \in \Gamma.
\]  

(2.1.17)

In addition, the embedding of a new point \( \bar{x} \in \overline{\Gamma} \setminus \Gamma \) can be evaluated in the embedding coordinate system by

\[
\bar{\psi}_i(\bar{x}) = \sum_l \langle \varphi_l, \psi_i \rangle \bar{\varphi}_l(\bar{x}).
\]  

(2.1.18)

Nyström extension, which is given in Eq. 2.1.14, has two major drawbacks:

1. The extended distance is proportional to the value of \( \varepsilon \) used in the kernel. Numerically this extension vanishes beyond this distance.

2. The scheme is ill conditioned since \( \lambda_l \to 0 \) as \( l \to \infty \).

The second issue can be solved by cutting off the sum in Eq. 2.1.16 keeping the eigenvalues (and the corresponding eigenfunctions) satisfying \( \lambda_l \geq \delta \lambda_0 \)

\[
\tilde{f}(x) = \sum_{\lambda_l \geq \delta \lambda_0} \langle \varphi_l, f \rangle \varphi_l(x), \ x \in \mathbb{R}^n.
\]  

(2.1.19)

The result is an extension scheme with a condition number \( \delta \). In this new scheme, \( f \) and \( \tilde{f} \) do not coincide on \( \Gamma \), but they are relatively close. The value of \( \varepsilon \) controls this error. An iterative method for modifying the value of \( \varepsilon \) with respect to the function to be extended is introduced in [22, 50]. The outline of the algorithm is as follows:
2.1. THE DIFFUSION MAPS FRAMEWORK

1. Determine an acceptable error for relatively big value of \( \varepsilon \) which are denoted by \( \text{err} \) and \( \varepsilon_0 \), respectively;

2. Build a Gaussian kernel using \( \varepsilon_0 \) such that \( k_{\varepsilon_0}(x, y) = e^{(||x-y||^2/2\varepsilon_0)} \);

3. Compute the set of eigenvalues for this kernel. Denote this set by \( \varphi_l(x) \) and write \( f \) as a linear combination of this basis as \( f(x) = \sum_l \langle \varphi_l, f \rangle \varphi_l(x) \);

4. Compute the sum

\[
\sum_{\lambda_l > \delta \lambda_0} \sqrt{|\langle \varphi_l, f \rangle|^2};
\]

5. If \( s_{\text{err}} < \text{err} \) then expand \( f \) with this basis as explained in Eq. 2.1.19. Otherwise, reduce the value of \( \varepsilon_0 \) and repeat the process.

The sum, which was computed in Step 4 of the algorithm, consists of only large coefficients. This quantity grows with the number of oscillations the function has. This means that \( \varepsilon \) will become smaller with respect to the behavior of \( f \). Since our interest is in expanding the smallest diffusion maps coordinates, which are likely to be smooth functions, we expect to find a large enough value for \( \varepsilon \) that will allow us to extend them to new data points.

2.1.5 The multi-scale pyramid approximation and interpolation method

This sections presents a method for approximation and extension of a function. In the context of this work, the functions to expand will be the diffusion maps coordinates. The method enables us to perform efficiently out-of-sample extension. It provides coordinates to existing and to newly-arrived data points in an embedded space. Multi-scale Gaussian approximation, also known as the pyramid method, was used for image processing applications [16, 17]

Let \( \Gamma \in \mathbb{R}^N \) be a set of points and \( f \) is a function that is defined on this data. The values of \( f \) on the set \( \Gamma \) are known. The goal is to approximate \( f(\bar{x}) \) where \( \bar{x} \in \mathbb{R}^N \setminus \Gamma \) is a newly-arrived data point. In our applications, we are interested in the case where \( f \)
is an embedding coordinate that was generated from the application of the diffusion maps to the data. The multi-scale extension of \( f \) replaces the geometric harmonics methodology [22], that is introduced in Section 2.1.4. The idea is to approximate \( f \) with a basis which is the spectral decomposition of a wide Gaussian kernel. If the approximation error on the set \( \Gamma \) is too big, then, this approximation is refined. The refinement is achieved by approximating the error with a basis which is the spectral decomposition with a narrower Gaussian kernel. This procedure continues until the error becomes sufficiently small. The detailed algorithm goes as follows: the first iteration is explained next, and it can be repeated recursively to generate better accurate results.

Algorithm 1 Multi-scale pyramid approximation

\textbf{Input}: Training data \( \Gamma \), function \( f \) on the data, Gaussian with width \( \sigma \), multiplication factor \( T \), admissible error \( \text{err} \)

\textbf{Output}: Multi-scale approximation of \( f \) such that \( f \approx \sum_{n=1}^{k} F_n \).

1: Set \( k = 1 \).
2: Construct the Gaussian kernel \( P^k(x, y) = e^{-\frac{\|x-y\|^2}{2T\epsilon}} \) from the data. \( T\epsilon \) determines the Gaussian width and \( T \) should be relatively big for \( k = 1 \).
3: Compute the eigendecomposition of \( P^k \) that was constructed in Step 2. \( P^k(x, y) = \sum_{l\geq 0} \lambda_l \psi_l^k(x) \psi_l^k(y) \) where \( \{\lambda_l\} \) and \( \{\psi_l\} \) are the eigenvalues and eigenvectors of \( P^k \), respectively.
4: If \( k = 1 \) then \( f \) is approximated by \( F_1 = \sum_{l: \lambda_l \geq \delta \lambda_0} \lambda_l \psi_l^1(x) \psi_l^1(y) < f, \psi_l^1 > \psi_l^1 \), where \( \delta \) is a condition number (see Eq. (2.1.19)). For \( k > 1 \), \( F_k \) is approximated by \( F_k = \sum_{l: \lambda_l \geq \delta \lambda_0} < f - \sum_{n=1}^{k-1} F_n, \psi_l^k > \psi_l^k \).
5: \( F_{k+1} \triangleq f - \sum_{n=1}^{k} F_n \) is the approximated error computed on \( \Gamma \).
6: If \( \|F_{k+1}\| \leq \text{err} \) then \( \sum_{n=1}^{k} F_n \) approximates \( f \) and the procedure ends. Otherwise, go to Step 2 and repeat the process with \( k = k + 1 \) and \( T = \frac{T}{2} \).

Once \( f \) is approximated by \( f \approx \sum_{n=1}^{k} F_n \), then it is used in Algorithm 2, which is applied to a newly arrived point \( \bar{x} \in \mathbb{R}^N \setminus \Gamma \) to perform out-of-sample extension of \( f \) to \( \bar{x} \). The outcome of Algorithm 2 is diffusion maps coordinates for the newly arrived data point \( \bar{x} \) in the embedded space.
2.2. WAVELET AND WAVELET PACKET TRANSFORMS

Algorithm 2 Multi-scale pyramid extension for defining the coordinates of a newly arrived data point

**Input:** Training data \( \Gamma \), function \( f \) on the data, multi-scale approximation \( \{ F_k \}_{k \geq 1} \) from Algorithm 1, newly arrived data point \( \bar{x} \in \mathbb{R}^N \setminus \Gamma \) to evaluate its coordinates

**Output:** \( f(\bar{x}) \)

1: For each \( k \), the eigenvectors \( \{ \psi^k_l \}_{\lambda_l \geq \delta \lambda_0} \), which belong to \( F_k \), can be extended to any point in \( \mathbb{R}^N \) by

\[
\psi^k_l(\bar{x}) = \frac{1}{N} \sum_{z \in \Gamma} e^{-\frac{\|\bar{x} - z\|^2}{4\epsilon}} \psi^k_l(z). \tag{2.1.20}
\]

\( F_k \) is extended to the newly arrived data point \( \bar{x} \) by

\[
F_k(\bar{x}) = \sum_{l: \lambda_l \geq \delta \lambda_0} <F_k, \psi^k_l(\bar{x})> \psi^k_l(\bar{x}). \tag{2.1.21}
\]

2: \( f \) is extended to the newly arrived data point \( \bar{x} \) by \( f(\bar{x}) = \sum_k F_k(\bar{x}) \).

2.2 Wavelet and wavelet packet transforms

By now the wavelet and wavelet packet transforms are widespread and have been described comprehensively in the literature [27, 74, 53]. Therefore, we restrict ourselves to mention only relevant facts that are necessary to understand the construction of the algorithm.

The result from the application of the wavelet transform to a signal \( f \) of length \( n = 2^J \) is a set of \( n \) correlated coefficients of the signal with scaled and shifted versions of two basic waveforms – the father and mother wavelets. The transform is implemented through iterated application of a conjugate pair of low– (\( H \)) and high– (\( G \)) pass filters followed by down-sampling. In the first decomposition step, the filters are applied to \( f \) and, after down-sampling, the result has two blocks \( w^1_0 \) and \( w^1_1 \) of the first scale coefficients, each of size \( n/2 \). These blocks consist of the correlation coefficients of the signal with 2-sample shifts of the low frequency father wavelet and high frequency mother wavelet, respectively. The block \( w^1_0 \) contains the coefficients necessary for the reconstruction of the low-frequency component of the signal. Similarly, the high frequency component of the signal can be reconstructed from the block \( w^1_1 \). In this sense, each decomposition block is linked to a certain half of the frequency domain of the signal.
While block $w_1^1$ is stored, the same procedure is applied to block $w_1^1$ in order to generate the second level (scale) of blocks $w_0^2$ and $w_1^2$ of size $n/4$. These blocks consist of the correlation coefficients with 4-sample shifts of the two times dilated versions of the father and mother wavelets. Their spectra share the low frequency band previously occupied by the original father wavelet. Then, $w_0^2$ is decomposed in the same way and the procedure is repeated $m$ times. Finally, the signal $f$ is transformed into a set of blocks $f \rightarrow \{w_0^m, w_1^m, w_1^{m-1}, w_1^{m-2}, \ldots, w_2^2, w_1^1\}$ up to the $m$-th decomposition level. This transform is orthogonal. One block is remained at each level (scale) except for the last one. Each block is related to a single waveform. Thus, the total number of waveforms involved in the transform is $m + 1$. Their spectra cover the whole frequency domain and split it in a logarithmic form. Each decomposition block is linked to a certain frequency band (not sharp) and, since the transform is orthogonal, the $l_2$ norm of the coefficients of the block is equal to the $l_2$ norm of the component of the signal $f$ whose spectrum occupies this band.

Through the application of the wavelet packet transform, many more waveforms, namely, $2^j$ waveforms at the $j$-th decomposition level are involved. The difference between the wavelet packet and wavelet transforms begins in the second step of the decomposition. Now both blocks $w_1^1$ and $w_1^1$ are stored at the first level and at the same time both are processed by pair of $H$ and $G$ filters which generate four blocks $w_0^2, w_1^2, w_2^2, w_3^2$ in the second level. These are the correlation coefficients of the signal with 4-sample shifts of the four libraries of waveforms whose spectra split the frequency domain into four parts. All of these blocks are stored in the second level and transformed into eight blocks in the third level, etc. The involved waveforms are well localized in time and frequency domains. Their spectra form a refined partition of the frequency domain (into $2^j$ parts in scale $j$). Correspondingly, each block of the wavelet packet transform describes a certain frequency band.

Flow of the wavelet packet transform is given by Figure 2.2.1. The partition of the frequency domain corresponds approximately to the location of blocks in the diagram.
There are many wavelet packet libraries. They differ from each other by their generating filters $H$ and $G$, the shape of the basic waveforms and their frequency content. In Fig. 2.2.2 we display the wavelet packets after decomposition into three levels generated by the spline of 6-th order. While the splines do not have a compact support in time domain, they are localized fairly. They produce perfect splitting of the frequency domain.

There is a duality in the nature of the wavelet coefficients of a certain block. On one hand, they indicate the presence of the corresponding waveform in the signal and measure its contribution. On the other hand, they evaluate the contents of the signal inside the
related frequency band. We may argue that the wavelet packet transform bridges the gap between time-domain and frequency-domain representations of a signal. As we advance to coarser level (scale), we see a better frequency resolution at the expense of time domain resolution and vice versa. In principle, the transform of a signal of length \( n = 2^J \) can be implemented up to the \( J \)-th decomposition level. At this level there exist \( n \) different waveforms, which are close to the sine and cosine waves with multiple frequencies. In Fig. 2.2.3, we display a few wavelet packets after decomposition into six levels generated by spline of the 6-th order. The waveforms resemble the windowed sine and cosine waves, whereas their spectra split the Nyquist frequency domain into 64 bands.

Figure 2.2.3: Spline-6 wavelet packets of sixth scale (left) and their spectra (right).
Part II

Hierarchical data mining approach for detection and prediction of anomalies in dynamically evolving systems
Chapter 3

Introduction

3.1 Background and motivation

The reliability of detection and tracking of failures in computerized systems and in transaction oriented systems in particular, influences the quality of service for short and long terms. Some expected failures can be fixed online by providing an advance warning through tracking of the emerged problem. In other cases, post-processing of the problem can reveal the importance of each parameter to the operation of the entire system and to the relation between the parameters. Classification of failures is also possible since the parameters that caused the problem can be singled out.

A new approach for detecting and tracking of failures in the behavior of a computerized system, is described. The approach is demonstrated on an un-labeled, high-dimensional dataset, which is constantly generated by a performance monitor. Representing and understanding the monitored system is complex. There is no guarantee that the given parameters fully capture the system’s behavior and phenomena, most likely the given dataset is not the optimal one. In addition, the normal state of the system is composed of a set of normal system behaviors. The transaction system’s behavior changes in different times of the day, the week and the month.

The dataset is embedded into a lower-dimension space by the application of diffusion methodologies to track and detect system failures. This is done by identifying trends in this embedded lower-dimension space that deviate from normal behavior. This un-
supervised algorithm first studies the system’s profile from a training dataset by extracting its parameters and then reduces its dimensions. The embedding of the training dataset forms a low-dimensional manifold. The performance monitor produces online the newly arrived data points. Then, the coordinates are determined by the application of the iterative multi-scale pyramid procedure. This determines whether the newly arrived data point, which does not belong to the training dataset, is abnormal since it deviates from the normal embedded manifold. To achieve this, an hierarchial processing of the incoming data is introduced.

Although network performance monitoring and automatic fault detection and prediction is somewhat related to the field of network intrusion detection, only a few research papers have been published in this field during the last few years. An automatic system for detection of service and failure anomalies in transaction based networks was developed in AT&T [39, 40, 42, 41]. The transaction-oriented system performs tasks that belonged to a few service classes, which are mutually dependent and correlated. Probability functions, which describe the average process time of the system’s transactions, were a basis for learning the system’s normal traffic intensities. Dynamic thresholds are generated and an anomaly is detected by deviation from these thresholds.

In this work, we provide a framework that is based upon diffusion processes for finding meaningful geometric descriptions in large heterogeneous datasets that were captured by a performance monitor. The eigenfunctions of the underlying Markov matrices and the associated family of diffusion distances are used to construct diffusion maps that generate efficient representations of complex high dimensional geometric structures. The capabilities and the performance of the proposed algorithm are demonstrated on a system that handles transactions for approval in which the data is constantly collected and processed. This system has in it a performance monitor device that records many different parameters from various components of the system at each predetermined time unit. System downtime is expensive and unaffordable. Generally, a breakdown of the system is not sudden but there is an underlying buildup process that caused it. Minimizing downtime periods is done by identifying and tracking of this buildup process ahead of time - problem avoidance. In addition, the algorithm saves useful information for analyzing and classifying the problem while identifying the influence and the connections among its measured parameters. It rates the importance and the contribution of each parameter to the overall behavior.
of the system. It helps in identifying more specifically the problem.

The high-dimensional data is embedded into a low-dimensional space. Tracking of buildups is done by identifying trends that deviate from normal behavior in this low-dimensional space. The proposed algorithm, which is classified as an unsupervised learning algorithm, has two steps:

1. **Training** (learning): Study and analysis of the datasets that were collected. Then, they are embedded onto low-dimensional space. The outcome of this process is clusters of data that represent typical (normal) behavior of the system.

2. **Prediction**: An automatic tool that enables to track and predict online (real time) when problems are about to occur (buildup) early enough to enable to apply certain measures to prevent it from being fatal while providing constant and steady quality of service (QoS). The online process saves effective information for post-processing and analysis of possible system failures and it can help to determine which parameters originated system failures.

This work extends and enhances the basic graph construction in the diffusion maps methodology. It builds a super-graph that provides an hierarchical (multi-scale) representation of the inspected data. This hierarchical representation provides better understanding of the data, more options how to group the data, more options for scaling the data, which is a critical pre-processing step. The two-phased unsupervised learning algorithm is a generic process that forms a base for studying complex dynamic systems of different types.

### 3.2 Novelty and contribution

Representation and understanding of real world datasets are the essence of machine learning and data mining methodologies. In this chapter, we introduce a novel framework for modeling high-dimensional heterogenous datasets. This framework addresses the following questions:

1. How to represent and process heterogenous dataset?

2. How to understand and query the data? How to fuse the data? How to find anomalies in data?
The thesis offers new methodologies to track anomalies and their classification. Anomaly detection identifies patterns in a given dataset that do not conform to an established normal behavior that are called anomalies and often translated to critical and actionable information in several application domains since they deviate from their normal behavior. Anomalies are also referred to as outliers, deviation, peculiarity, etc.

The core technology of the thesis is based upon diffusion processes, diffusion geometries and other methodologies for tracking and detecting meaningful geometric descriptions of geometric patterns that deviate from normality in the inspected data. The main core of proposed methodology is based upon training the system to extract heterogeneous features to cluster the normal behavior and then detect patterns that deviate from it. The thesis offers behavioral analysis of heterogeneous complex data to maintain and preserve systems’ health.

The challenges the thesis solves are: How to extract features (parameters) from high-dimensional data of measurements and sensors. How to combine (fuse) them into a coherent understanding of the data. The data can be heterogeneous: huge database, e-mails, logs, communication packets and diverse protocols, computers’ data, electronic data, intelligence data, etc. The data do not have the same scale. How to cluster and segment high-dimensional data from different scales where some can be nominal (i.e. categorical) while others are numeric. How to expose and represent clusters/segmentations in high-dimensional space. How to track and find deviation from normal behavior. How to process it to find abnormal patterns in ocean of data. How to find distances in high-dimensional data using diffusion geometries. How can we determine whether a data point belongs to a cluster/segment or not. How we treat huge high dimensional data that is dynamically and constantly changes such as communication streams and monitoring data. How to classify newly arrived data point to decide to which cluster it belongs to.

The main challenges robust anomaly detection algorithms face are: achieving low false alarm rate, clustering into normal and abnormal regions, detection of anomalies that are distributed by malicious adversaries, treating dynamic and evolving data, portability of the techniques between different domains and applications, availability of labeled data for training/validation and noisy data that tend to be similar to real anomalies. Usually, in addition to the challenge of tracking and detecting anomalies in a dataset, the analyzed data are also high-dimensional and noisy. The algorithms in this thesis meet these challenges.
The thesis is focused on unsupervised anomaly detection techniques that track and detect anomalies in an unlabeled dataset under the assumption that majority of the instances in the dataset are normal while achieving low false alarm rate. This thesis builds an efficient and coherent normal baseline that on one hand enables to track and detect anomalies but on the other hand keeps a very low rate of false alerts. Defining a normal region, which encompasses every possible normal behavior, is difficult. In addition, the boundary between normal and anomalous behavior is often imprecise. Thus, an anomalous observation, which lies close to the boundary, can actually be normal and vice-versa. The thesis handles normal behavior that keeps evolving and a current notion of normal behavior might not be sufficiently representative in the future. In addition, the exact notion of anomaly is different for different application domains. The thesis also handles high-dimensional data where local neighborhood in high dimensions is no longer local in the lower dimension space. In addition, this thesis handles noisy data that tend to be similar to real anomalies and hence it is difficult to track and detect. As more dimensions are added, which is good for the reliability of the extracted features, the noise is amplified.

Transforming a heterogenous dataset, which consists of a number of parameters (extracted from measurements and sensors) that are measured in different scales into a uniform workable setup, is done by the application of the diffusion maps. The diffusion maps algorithm, which was originally proposed as a dimensionality reduction scheme, is used to transform each parameter or sensor, into a new set of coordinates that reflect its geometric structure. This novel step of using the diffusion coordinates as a re-scaling method allows us to construct a reliable and comparable representation of the dataset.

The task of understanding the data is addressed in the thesis by two directions. First, we formulate a task related function on the data. The task is to track and detect anomalies as was explained above. The thesis provides different ways to accomplish it. One way is to process each parameter separately and then fuse the outputs to provide an answer whether the processes data point is anomalous. Another way is to fuse the data and then provide an answer. The constructed functions, which are later referred to as score functions, emphasize the properties that are of interest in the scope of the task. In general, one can construct a number of score functions on a given dataset. A decision will be made based on their fused scores. The second direction, which was developed in the thesis, is a novel method for fusing the data in its new representation. An hierarchical tree was introduced for mod-
eling the structure of the dataset. A tree node holds a part of the dataset and a parent node fuses between the data of its child nodes. In the application we deal with in Chapters 4 and 5, the score functions, which question the data, were constructed in each tree node and the hierarchical structure fused the constructed score functions rather the data. Fusing the score functions instead of the data reduces the impact of the phenomena in the data that are not task related. If the dataset is sampled from a system or a process, then the flexible structure of the tree allows us to analyze the subsystems separately and then to analyze the entire system. This structure makes failures classification easier.

The contribution of this chapter is the introduction of a representation method for heterogeneous datasets by using a flexible tree structure, which fuses the data in a bottom-up manner while constructing score functions on the data to help us understand its phenomena and how it is related to the task at hand.
Chapter 4

Evaluation dataset - performance of a monitored application

This section describes the application of the hierarchial diffusion maps framework to process high-dimensional data that was collected from a performance monitor (at specified time intervals), which resides inside a monitored system that handles non-stop incoming transactions to be either approved or disapproved. The dimensionality reduction of the data into a lower dimension space creates an informative and reliable learning set that separates between normal and abnormal events in the system behavior. An online prediction of an emerged anomaly is done by extending the current embedding to new points via the application of the multi-scale pyramid algorithm that is described in Algorithms 1 and 2 in Section 2.1.5.

The data, which was collected from an internal performance monitor, consists of parameters of different types and scales. The current monitor saves the data approximately once every minute. This time interval is a parameter that can be modified. For this specific analysis, 13 measured parameters were used (the monitor can be configured to produce more parameters and the algorithm can handle any number of available parameters). They have different scales. These 13 parameters can be separated into three groups. The first group of parameters measures the average process time of different types of transactions that were executed in the system during the last time interval. Another group of parameters measures the percentage of running transactions that are waiting for different system
resources such as I/O and database access. This group of parameters represents the distribution of queues among the system’s resources. The third group consists of two parameters that measure the percentage of memory usage. The introduced algorithm is general and the number of input parameters can grow as needed to improve its robustness.

4.1 Implementation of the learning phase

The learning phase is carried out in a bottom up approach. A similar embedding process is applied to a single input parameter and then the input parameters are grouped to include the entire set of input data. This application forms an hierarchical embedding tree structure, which describes the system and its sub-systems. The idea is to embed the sub-system parameters that each node describes by using the diffusion maps. Next, a frequency score function is constructed on the embedding result. This function defines a measure for the probability of each data point value in the group to be abnormal. The value of the score function of each data point in the embedded space is calculated as a summation of distances to its nearest neighbors in the embedded space. The Euclidean distances in the embedded space are equivalent to the diffusion distances in the original space (see Chapter 2 Eqs. (2.1.7) and (2.1.8)). The frequency score functions are used as input to their parent’s node in the tree. The bottom level tree nodes, which process a single input parameter, have one additional pre-processing step in them. A single parameter, which is one dimensional data captured over some time frame, is formed to be vectorial input of \( \mu \) dimensions. The diffusion maps is then applied to this vectorial form of the input parameter. This application also bypasses the need to re-scale the input parameters in a traditional manner like ordinal or linear re-scaling. Usually, finding the right scaling mechanism is complex. The proposed algorithm bypasses this difficulty.

The diffusion processes in each tree node are constructed with a Gaussian kernel \( w_c(x, y) = e^{-\frac{|x-y|^2}{2\epsilon}} \). The kernel is normalized as described in Section 2.1.2. Setting \( \alpha = 1 \) in Eq. (2.1.9) yields the best experimental results.
4.1. IMPLEMENTATION OF THE LEARNING PHASE

4.1.1 Hierarchical three level embedding model

The learning phase implementation can be represented by an hierarchical three level embedding model that is shown in Fig. 4.1.1. The low-dimensional embedding process is modeled as a tree, where each bottom level node describes the embedding process of a single parameter (feature) - see Section 4.1.2. The intermediate (middle) level nodes fuse the data from the bottom level. The diffusion process is applied to an intermediate level. The top level fuses the data from the intermediate level. The application of the diffusion to the top level node forms an embedding, denoted as a super-graph, which describes the entire system behavior in a low-dimensional space. The diffusion process, which is applied to the bottom level tree nodes, goes as follows: First, each feature is formulated to emphasize its dynamic. Next, the diffusion maps is applied to it. The diffusion coordinates are saved. The diffusion maps coordinates form a reliable space for representation of the behavior of each parameter. Finally, a frequency score function is calculated from the diffusion maps coordinates. These score functions are the input for the next level of the tree nodes. The whole process is explained in Section 4.1.2. The nodes that belong to the middle and top level tree nodes go through a similar diffusion process - see Section 4.1.3. The steps, which are applied to the tree nodes at different levels, are described in Fig. 4.1.2.
Figure 4.1.1: Hierarchical three level embedding model that describes the complete learning phase. The labels on the tree nodes are defined in Sections 4.1.2, 4.1.3 and 4.1.4.

Figure 4.1.2 illustrates the diffusion embedding process that is applied to the tree nodes at different levels in the tree.
4.1. IMPLEMENTATION OF THE LEARNING PHASE

4.1.2 Application of the diffusion maps to a single parameter - bottom level processing

This section describes the activities that take place in the bottom part of Fig. 4.1.2.

Construction of the low dimensional embedding space, which is used in the training phase, begins by studying the dynamic behavior of each parameter. The sought after problems in this transaction based system are characterized by an oscillatory behavior of the system that appears a few minutes prior to a fatal failure. The input data is formulated in such a way that it helps to reveal these oscillations in order to predict future system failures. This is done by creating overlapping sequences from the input and saving them as the new input. This formulation provides a dynamic representation of the data. The diffusion maps algorithm is applied to each input parameter in its dynamic representation. The outcome produces an embedding which expresses the dynamic behavior of every input parameter. Abnormal behavior of a single parameter is detected this way. The distance of a point from its nearest neighbors in the embedding space gives a value that reflects the point’s probability to be abnormal.
Let $\vec{P} = \{P_1, P_2, \ldots, P_K\}$ (in our examples $K = 13$) be the data collected by the performance monitor over a predetermined number of time intervals $t = N$. $\vec{P}$ is a matrix of size $N \times K$. For the processed training data, it can be assumed that $\vec{P}$ has a fixed size. Each parameter $P_i \in \vec{P}$, $1 \leq i \leq K$, which is a column and holds data that measures a specific activity in the system, is defined as $P_i \triangleq (p_{i1}^1, p_{i2}^2, \ldots, p_{iN}^N)^T$. By using a sliding window of length $\mu$, which moves along each parameter $P_i$, we form $N - \mu + 1$ dynamic-paths $P_i^r$

$$P_i^r = (p_i^r, p_i^{r+1}, \ldots, p_i^{r+\mu-1}) \quad r = 1, \ldots, N - \mu + 1, \quad i = 1, \ldots, K \quad (4.1.1)$$

that are associated with $P_i$. The dynamic-paths $P_i^r$, which belong to $P_i$, capture the parameters behavior during the next $\mu$ time steps. Oscillatory behavior is expressed by an exceptional dynamic-path. Finally, a dynamic-matrix $\tilde{P}_i$ is constructed, which consists of $r$ dynamic-paths as its rows, is defined as

$$\tilde{P}_i = \begin{pmatrix} P_i^1 \\ \vdots \\ P_i^{N-\mu+1} \end{pmatrix} \in \mathbb{R}^{(N-\mu+1) \times \mu}. \quad (4.1.2)$$

For example, the parameter

$$P_i = \begin{pmatrix} 34 & 38 & 41 & 34 & 41 & 38 \end{pmatrix}^T$$

is represented as $\tilde{P}_i = \begin{pmatrix} 34 & 38 & 41 \\ 38 & 41 & 34 \\ 41 & 34 & 41 \\ 34 & 41 & 38 \end{pmatrix}$.

The diffusion maps algorithm is applied to each of the dynamic matrices $\{\tilde{P}_i\}_{i=1}^K$. The outcome produces embedding coordinates for the dynamic behavior of every input parameter. Usually, two diffusion maps coordinates of each parameter are sufficient to describe its dynamic-matrix activity. The embedding coordinates of an input parameter in the dynamic-matrix $\tilde{P}_i$ are denoted by $\Psi_i = \{\psi_{i1}, \psi_{i2}\}$, $i = 1, \ldots, K$. After the parameter is embedded, a frequency score function is defined on the points in the embedded space.
For a set of embedding coordinates \( \{ \Psi(x) \} \) the frequency score function is defined by

\[
D(x) = \sum_{y \in S} \| \Psi(x) - \Psi(y) \|, \quad S = \{ \eta \text{ nearest neighbors of } x \text{ in } \Psi(x) \}. \tag{4.1.3}
\]

The value of the score function for each point in the embedded space is calculated as the sum of distances to its \( \eta \) closest neighbors. Points that lie in dense areas in the embedding graph have low values in this score function and isolated points have a large value. Denote the frequency score functions of the embeddings \( \{ \Psi_i \}_{i=1}^K \) by \( \{ D_i \}_{i=1}^K \).

In our case, \( K = 13 \). The application of the diffusion maps begins with the first six parameters \( P_1, \ldots, P_6 \). These parameters measure the average response time from different transaction types in the previous time step. The measured scale is in minutes while the precision is ten seconds long. The six dynamic-matrices \( \tilde{P}_1, \ldots, \tilde{P}_6 \) with a path length \( \mu = 3 \) are the inputs to six diffusion processes. The outcomes are the embeddings \( \Psi_1, \ldots, \Psi_6 \) with six score functions \( D_1, \ldots, D_6 \), which are calculated in the embedded space. Figures 4.1.3 and 4.1.4 show the application of the learning process to the dynamic behavior (dynamic matrix) of the parameters \( P_1, \ldots, P_6 \).
4.1. IMPLEMENTATION OF THE LEARNING PHASE

(a) The embedding $\Psi_1$ of $\tilde{P}_1$
(b) The frequency score function $D_1$

(c) The embedding $\Psi_2$ of $\tilde{P}_2$
(d) The frequency score function $D_2$

(e) The embedding $\Psi_3$ of $\tilde{P}_3$
(f) The frequency score function $D_3$

Figure 4.1.3: The outcome of the learning phase of the three parameters $P_1, P_2, P_3$ that were marked in Fig. 4.1.1. Images (a) and (b) show the diffusion maps embedding $\Psi_1$ and the score function $D_1$ that was calculated from $\Psi_1$, respectively. The points of $\Psi_1$ are colored according to their frequency score. Normally behaved points, which correspond to normal rows of the dynamic matrix $\tilde{P}_1$, are colored blue while abnormal points are colored red. Similarly (c) and (d) and (e) and (f) show the outcome of the learning process of the parameters $P_2$ and $P_3$, respectively.
4.1. IMPLEMENTATION OF THE LEARNING PHASE

Figure 4.1.4: The outcome of the learning phase of the three parameters $P_4, P_5, P_6$ that were marked in Fig. 4.1.1. (a) and (b) show the diffusion maps embedding $\Psi_4$ and the score function $D_4$ that was calculated from $\Psi_4$, respectively. The points of $\Psi_4$ are colored according to their frequency score. Normally behaved points, which correspond to normal rows of the dynamic matrix $\tilde{P}_4$, are colored blue while abnormal points are colored red. Similarly, images (c), (d) and images (e) and (f) show the outcome of the learning process of the parameters $P_5$ and $P_6$, respectively.

The second group of parameters $P_7, \ldots, P_{11}$ measures the percentage of executed transactions that wait for a specific system’s resource. At each time step, the performance
monitor tracks all the running transactions in the system and calculates the percentage of the running transactions that wait for a particular system’s resource like I/O, database access, etc. The embeddings $\Psi_7, \ldots, \Psi_{11}$ of the dynamic-matrices $\tilde{P}_7, \ldots, \tilde{P}_{11}$ resemble a parabola. Figures 4.1.5 and 4.1.6 show the embeddings $\Psi_7, \ldots, \Psi_{11}$ and their associated frequency score functions $D_7, \ldots, D_{11}$. 
4.1. IMPLEMENTATION OF THE LEARNING PHASE

Figure 4.1.5: The outcome of the learning phase of the three parameters $P_7, P_8, P_9$ that were marked in Fig. 4.1.1. Images (a) and (b) show the diffusion maps embedding $\Psi_7$ and the score function $D_7$ that was calculated from $\Psi_7$, respectively. The points of $\Psi_7$ are colored according to their frequency score. High density areas, which contain normal behaved points, are colored blue. Abnormal points, which indicate that the queue may overflow, are located on the edges of the main cluster and are colored red. Similarly, images (c), (d) and images (e) and (f) show the outcome of the learning process of the parameters $P_8$ and $P_9$, respectively.
4.1. IMPLEMENTATION OF THE LEARNING PHASE

(a) The embedding $\Psi_{10}$ of $\tilde{P}_{10}$
(b) The frequency score function $D_{10}$

(c) The embedding $\Psi_{11}$ of $\tilde{P}_{11}$
(d) The frequency score function $D_{11}$

Figure 4.1.6: The outcome of the learning phase of the two parameters $P_{10}, P_{11}$ that were marked in Fig. 4.1.1. Images (a) and (b) show the diffusion maps embedding $\Psi_{10}$ and the score function $D_{10}$ that was calculated from $\Psi_{10}$, respectively. The points of $\Psi_{10}$ are colored according to their frequency score. High density areas, which contain normal behaved points, are colored blue. Abnormal points, which indicate that the queue may overflow, are located on one of the edges of the main cluster, and are colored red. Similarly, images (c) and (d) show the outcome of the learning process for the parameter $P_{11}$.

Parameters $P_{12}$ and $P_{13}$ constitute the third group. The dynamic-matrices $\tilde{P}_{12}$ and $\tilde{P}_{13}$ hold the dynamics of two parameters that capture the capacity (in percentage) of two different memories that the system uses. Their embeddings $\Psi_{12}$ and $\Psi_{13}$ with their score functions $D_{12}$ and $D_{13}$ are shown in Fig. 4.1.7. Similarly to the parameters that hold the queue distribution of the system’s resources, the embeddings have a parabola shape and the abnormal points (in red) lie on the edge.
4.1. IMPLEMENTATION OF THE LEARNING PHASE

(a) The embedding $\Psi_{12}$ of $\tilde{P}_{12}$

(b) The frequency score function $D_{12}$

(c) The embedding $\Psi_{13}$ of $\tilde{P}_{13}$

(d) The frequency score function $D_{13}$

Figure 4.1.7: The outcome of the learning phase of the two parameters $P_{12}, P_{13}$ that were marked in Fig. 4.1.1. Images (a) and (b) show the diffusion maps embedding $\Psi_{12}$ and the score function $D_{12}$ that was calculated from $\Psi_{12}$, respectively. The points of $\Psi_{12}$ are colored according to their frequency score. Points that more likely to appear are colored blue. Abnormal points, which indicate that the memory capacity is high or the trend is abnormal, are located on one of the edges or inside the parabola, are colored red. Similarly, images (c) and (d) show the outcome of the learning process on the parameter $P_{11}$.

The embeddings $\Psi_1, \Psi_2, \ldots, \Psi_{13}$ provide each parameter an informative description. More specifically, the appearance frequency of each point in the embedded space can be used as a measurement to predict and detect system’s anomalies. The original parameter set $P = \{P_1, P_2, \ldots, P_{13}\}$ is replaced with a set of score functions $D = \{D_1, D_2, \ldots, D_{13}\}$. The set $D$ can be thought of as a feature space of the data, where the features describe the dynamic behavior of the input parameters. These frequency score functions are the inputs to the second level nodes of the hierarchial tree.

In order to summarize this step, the learning process, described in this section, is demonstrated on the input parameter $P_9$. Figure 4.1.8 describes the learning process on a bottom level of the tree node that is seen in the bottom part of Fig. 4.1.2. Figure 4.1.8 (a)
4.1. IMPLEMENTATION OF THE LEARNING PHASE

displays the values of this parameter as they were recorded from the system’s monitoring device. Next, the dynamic matrix $\tilde{P}_9$ is constructed. Since the path length in this application is 3, the dynamic paths can be plotted as points in $\mathbb{R}^3$. Figure 4.1.8 (b) displays the rows of the dynamic matrix $\tilde{P}_9$. The application of the diffusion maps algorithm to $\tilde{P}_9$ generates the embedding $\Psi_9$, which is seen in Fig. 4.1.8 (c). Finally, $D_9$, which is the frequency score function of the embedding, is calculated. $D_9$ is presented in Fig. 4.1.8 (d).

Figure 4.1.8: Application of the diffusion maps to a single parameter $P_9$. This describes the process that is applied to the bottom level nodes in the hierarchical tree shown in Fig. 4.1.1. This process is also explained the bottom level of Fig. 4.1.2.
4.1. IMPLEMENTATION OF THE LEARNING PHASE

4.1.3 Application of the diffusion maps to a group of parameters: middle level processing

This section describes the activities in the middle level tree nodes shown in Fig. 4.1.2. The output from the last step (Section 4.1.2) is a set of functions $D = \{D_1, D_2, \ldots, D_{13}\}$ that replaces the original input and describes the performance of the system’s parameters. The score functions were separated into three groups, according to the parameter type. Each group holds score functions that were derived from the embeddings of the parameters of the same type and have the same scale. The diffusion maps algorithm was applied to the set of score functions of each group. The result from this step is a set of new embeddings that describe the dependency and the correlation among parameters in the same group. This grouping process correlates between different types of system failures and the groups of parameters that cause these failures. Problems like system load can be detected in the embedding of one group while deadlocks emerges in another group.

The inputs to this phase (second level in Fig. 4.1.1) are the following three groups: $G_1 = \{D_1, \ldots, D_6\}$ holds the frequency score functions of the parameters that measure the average process time of different transactions types. $G_2 = \{D_7, \ldots, D_{11}\}$ gathers the frequency score functions of the embedding of the parameters that measure the percentage of running transactions that wait for system resources and $G_3 = \{D_{12}, D_{13}\}$ contains the frequency score functions of the embeddings of the two parameters that capture the memory usage of the two different memories used by the system.

Now, the diffusion maps algorithm is applied to $G_1, G_2$ and $G_3$ to produce three embeddings that are denoted by $\Phi_1, \Phi_2$ and $\Phi_3$, respectively. Points in the embedded spaces of $\Phi_1, \ldots, \Phi_3$ correspond to rows in the dynamic matrices $\{\tilde{P}_i\}_{i=1}^{13}$ that comprise $G_1, \ldots, G_3$. The embedding of the first group $G_1$, which is in the middle level of the hierarchial tree to a three dimensional space $\Phi_1 = \{\phi_{1,1}, \phi_{1,2}, \phi_{1,3}\}$, is presented in Fig. 4.1.9 (a). The frequency score function of this embedding, denoted by $D_{\Phi_1}$, is shown in Fig. 4.1.9 (b).
4.1. IMPLEMENTATION OF THE LEARNING PHASE

The learning process is applied to the group $G_2$, which is located in the middle level of the hierarchical tree and combines five bottom level parameters of the same type (see Fig. 4.1.1). Figure 4.1.10 (a) shows the joint three dimensional embedding $\Phi_2 = \{\phi_{2,1}, \phi_{2,2}, \phi_{2,3}\}$ of the second group $G_2$ of frequency score functions that measures the percentage of running transactions that wait for different system resources. The frequency score function of $\Phi_2$, denoted by $D_{\Phi_2}$ is presented in Fig. 4.1.10 (b).
4.1. IMPLEMENTATION OF THE LEARNING PHASE

(a) Embedding \( \Phi_2 \) of \( G_2 \)

(b) Score function \( D_{\Phi_2} \) of \( \Phi_2 \)

Figure 4.1.10: (a) Embedding of the dynamics of the second group of frequency score functions \( G_2 \). This input group gathers the dynamic behavior of five parameters that measure the percentage of running transactions that wait for five different resources. The combination of these parameters expresses the distribution of the running transaction between different resources. Points that deviate from this profile indicate a possible buildup towards a failure. Abnormal points (colored red) are located further out or in between the main clusters that is described by the three-dimensional graph \( \Phi_2 = \{ \phi_{2,1}, \phi_{2,2}, \phi_{2,3} \} \). (b) The frequency score function \( D_{\Phi_2} \) of \( \Phi_2 \).

The last group from the middle level in the hierarchical tree is \( G_3 \) that holds the two bottom level parameters. Its location in the hierarchical tree is shown in Fig. 4.1.1. The application of the diffusion maps algorithm to the input group of the frequency score functions \( G_3 \) creates the embedding \( \Phi_3 = \{ \phi_{3,1}, \phi_{3,2}, \phi_{3,3} \} \) that is shown in Fig. 4.1.11 (a). \( D_{\Phi_3} \), which describes the appearance frequency of each point in \( \Phi_3 \), is shown in Fig 4.1.11 (b).
4.1. IMPLEMENTATION OF THE LEARNING PHASE

(a) Embedding $\Phi_3$ of $G_3$

(b) Score function $D_{\Phi_3}$ of $\Phi_3$

Figure 4.1.11: (a) Embedding of the memory frequency score functions that were gathered in $G_3$. The system saves two parameters that capture the capacity (in percentage) of the two different memories. $\Phi_3 = \{\phi_{3,1}, \phi_{3,2}, \phi_{3,3}\}$ expresses the dynamic behavior of this group. (b) $D_{\Phi_3}$ is the frequency score function of $\Phi_3$.

4.1.4 Creating a super-embedding graph: top level processing

This section describes the construction of the top level in Figs. 4.1.1 and 4.1.2. In many cases, two-levels group embeddings are sufficient for prediction, tracking and detection of failures. There are situations where we want to see the entire representation of the input data embedded in a low dimensional space. The functions $D_{\Phi_1}, D_{\Phi_2}$ and $D_{\Phi_3}$ are taken to be the new input for another application of the diffusion algorithm, which produces a super-embedding graph (top level in Figure 4.1.1). Denote this new super-graph by $V = \{v_1, v_2, v_3\}$ and the super-graph’s frequency score function by $D_V$. This super-embedding graph describes the combined high dimensional dynamic system as embedded in a low dimensional space. Abnormal behavior can be detected in this super-embedding graph. For problem classification, we go back to either groups embeddings or to a single parameter’s embedding, to find the parameters that caused this anomaly. Figure 4.1.12 (a) shows the embedding of the system into a super-graph $V$. The frequency score function $D_V$ of this embedding (see Fig. 4.1.12 (b)), can be used as a measure for abnormality trends of the system’s behavior.
4.2. THE ONLINE DETECTION PHASE USING THE MULTI-SCALE PYRAMID INTERPOLATION

After the completion of the training phase, different behavioral aspects of the observed system are described by embedding matrices and their associated frequency score functions. To process each newly arrived data point, which did not participate in the learning phase, in order to determine if it deviates from normal behavior (represented by an embedding space), the coordinates of this point in the embedded space have to be determined.

The multi-scale pyramid interpolation method, which was described in Section 2.1.5, provides the coordinates to approximate the newly arrived data points. The multi-scale pyramid interpolation replaces the geometric harmonics [22] method that uses the Nyström extension.

The multi-scale pyramid interpolation is done in two steps:

1. A pre-processing that is done off-line at the end of the learning phase.

2. An online classification of newly arrived data points.

Pre-processing approximation of the hierarchial embeddings: The hierarchy of low-dimensional embeddings, which were described in Sections 4.1.2, 4.1.3 and 4.1.4,
is approximated with multi-scaled Gaussian kernels.

The application of Algorithm 1 in Section 2.1.5 is slightly different for each level in the tree. For each bottom level nodes, the embedding $\Psi_i$, $i = 1, \ldots, 13$, is approximated via the application of Algorithm 1, where $\Gamma = \tilde{P}_i$ is the training set and $f = \Psi_i$ is the function to approximate. In the middle level, each of the embeddings $\Phi_1, \Phi_2$ and $\Phi_3$ is approximated by Algorithm 1 in Section 2.1.5. Algorithm 1 is applied to the training set $\Gamma = G_i$ and $f = \Phi_i$. Finally, the super-graph top level is approximated by the application of Algorithm 1 with $\Gamma = \{D\Phi_1, D\Phi_2, D\Phi_3\}$ and $f = V$.

The pre-processing, which approximates the hierarchial tree node’s embeddings by the application of Algorithm 1, is demonstrated on several tree nodes that are marked in Fig. 4.2.1

![Pre-processing approximation step of the hierarchical tree embeddings](image)

Figure 4.2.1: The application of the multi-scale pyramid approximation (Algorithm 1) is demonstrated in Fig. 4.2.2 on the five selected nodes seen here. The small arrow in each tree node indicates that the node’s embedding has been approximated.

Figure 4.2.2 shows the embeddings and the results from the application of Algorithm 1 to nodes that were marked in Fig. 4.2.1
4.2. THE ONLINE DETECTION PHASE USING THE MULTI-SCALE PYRAMID INTERPOLATION

![Image](image.png)

Figure 4.2.2: (a),(c),(e),(g),(i) and (k) show the embeddings $\Psi_1$, $\Psi_8$, $\Phi_1$, $\Phi_2$, $\Phi_3$ and $V$, respectively. (b),(d),(f),(h),(j) and (l) show the result from the multi-scale pyramid interpolation (Algorithm 1) to $\Psi_1$, $\Psi_8$, $\Phi_1$, $\Phi_2$, $\Phi_3$ and $V$, respectively. (a) and (c) are the bottom level tree nodes embeddings, (e), (g) and (i) are the intermediate level node embeddings and (k) is the top level node embedding.

**Online processing of the newly arrived data points:** The created approximations are used to process the newly arrived data point. The newly arrived data point is separated into its 13 components, which first extends the embeddings of the bottom tree level nodes and then extends the embeddings of the middle and top level nodes. Let $p^t = (p^t_1, p^t_2, \ldots, p^t_{13})$ be a newly arrived data point that holds the 13 measured parameters that were collected by the performance monitor at time $T = t$. The newly
arrived data point $p^t$ undergoes similar steps to those described in the learning phase (see Section 4.1). The following steps are applied online to a newly arrived data point:

1. Construct the dynamic behavior of each parameter of the newly arrived data point: The dynamic behavior is constructed from the last $\mu$ newly arrived data points. In our application, $\mu = 3$. We begin with three newly arrived consecutive data points from times $t$, $t-1$ and $t-2$ such that

$$
p^t = (p^t_1 \ p^t_2 \ p^t_3 \ p^t_4 \ \ldots \ p^t_{13}),
$$

$$
p^{t-1} = (p^{t-1}_1 \ p^{t-1}_2 \ p^{t-1}_3 \ p^{t-1}_4 \ \ldots \ p^{t-1}_{13}),
$$

$$
p^{t-2} = (p^{t-2}_1 \ p^{t-2}_2 \ p^{t-2}_3 \ p^{t-2}_4 \ \ldots \ p^{t-2}_{13}).
$$

The three points are reformulated to 13 dynamic matrices of size $1 \times 3$,

$$
\tilde{p}^t_1 = \begin{pmatrix} p^t_1 \\ p^{t-1}_1 \\ p^{t-2}_1 \end{pmatrix}^T, \tilde{p}^t_2 = \begin{pmatrix} p^t_2 \\ p^{t-1}_2 \\ p^{t-2}_2 \end{pmatrix}^T, \ldots \tilde{p}^t_{13} = \begin{pmatrix} p^t_{13} \\ p^{t-1}_{13} \\ p^{t-2}_{13} \end{pmatrix}^T.
$$

These dynamic matrices $\tilde{p}^t_i$, $i = 1, \ldots, 13$, are new data ‘points’ that will be embedded in the bottom level embeddings.

2. Application of Algorithm 2 (Section 2.1.5), which uses the results from Algorithm 1, to each newly constructed dynamic-matrix $\tilde{p}^t_i$, $i = 1, \ldots, 13$, to produce the embedding values that are denoted by $a^t_i = \Psi_i(\tilde{p}^t_i)$, $i = 1, \ldots, 13$.

3. The frequency score of each newly constructed dynamic matrix are calculated and denoted by $d^t_i = D_i(a^t_i)$, $i = 1, \ldots, 13$.

4. The new scores $d^t_1, d^t_2, \ldots, d^t_{13}$, which are the outputs from Step 3, become the inputs for extending the hierarchial tree’s middle level embeddings (see Figs. 4.1.1 and 4.1.2). The new scores are gathered by groups. Algorithm 2 is applied to each group. Algorithm 2 assigns coordinates to the newly constructed scores points for each of the middle level embeddings $\Phi_1, \Phi_2$ and $\Phi_3$. Denote the new extended values by $b^t_1, b^t_2$ and $b^t_3$ where $b^t_1 = \Phi_1(d^t_1, \ldots, d^t_6)$,
4.2. THE ONLINE DETECTION PHASE USING THE MULTI-SCALE PYRAMID INTERPOLATION

\[ b'_2 = \Phi_2(d'_7, \ldots, d'_{11}) \quad \text{and} \quad b'_3 = \Phi_3(d'_{12}, d'_{13}). \]

5. The corresponding values of the hierarchial tree’s middle level frequency score functions are computed and denoted by \( \tilde{d}'_1 = D_{\Phi_1}(b'_1), \tilde{d}'_2 = D_{\Phi_2}(b'_2) \) and \( \tilde{d}'_3 = D_{\Phi_3}(b'_3). \)

6. The output from the previous step, which is the three newly computed frequency scores \( \tilde{d}'_1, \tilde{d}'_2 \) and \( \tilde{d}'_3 \), become the new input for extending the supergraph. Algorithm 2 is applied to \((\tilde{d}'_1, \tilde{d}'_2, \tilde{d}'_3)\) and the newly computed supergraph coordinates are denoted by \( c' = V(\tilde{d}'_1, \tilde{d}'_2, \tilde{d}'_3). \)

7. The value of the frequency score function of the super graph is calculated and denoted by \( \hat{d}' = D_V(c'). \)

The application of Algorithm 2 is also demonstrated on the selected nodes that were marked in Fig. 4.2.1. Figure 4.2.3 shows the extension of two bottom level node embeddings to a new point. Figure 4.2.4 shows the extension of the middle level nodes and Fig. 4.2.5 shows the extension of the super-graph.
4.2. THE ONLINE DETECTION PHASE USING THE MULTI-SCALE PYRAMID INTERPOLATION

Figure 4.2.3: (a) and (c): Applications of Algorithm 2 to $\Psi_1$ and to $\Psi_8$ with the newly arrived inputs $\tilde{p}_1^t$ and $\tilde{p}_8^t$, respectively. The computed coordinates $a_1^t = \Psi_1(\tilde{p}_1^t)$ and $a_8^t = \Psi_8(\tilde{p}_8^t)$ are colored in red. (b) and (d): The frequency scores of the newly computed data points $d_1^t = D_1(a_1^t)$ and $d_8^t = D_1(a_8^t)$ are computed according to the distances of the embedded point from its nearest neighbors in $\Psi_1$ and $\Psi_8$, respectively. The new frequency scores $d_1^t$ and $d_8^t$ are plotted at the end of $D_1$ and $D_8$ and are colored black.
4.2. THE ONLINE DETECTION PHASE USING THE MULTI-SCALE PYRAMID INTERPOLATION

Figure 4.2.4: (a),(c) and (e): Application of Algorithm 2 to $\Phi_1$, $\Phi_2$ and $\Phi_3$ with the newly arrived inputs $(d_{t1}^1, d_{t2}^1, \ldots, d_{t6}^1)$, $(d_{t7}^2, d_{t8}^2, \ldots, d_{t11}^2)$ and $(d_{t12}^3, d_{t13}^3)$, respectively. The computed coordinates $b_t^1 = \Phi_1(d_{t1}^1, d_{t2}^1, \ldots, d_{t6}^1)$, $b_t^2 = \Phi_2(d_{t7}^2, d_{t8}^2, \ldots, d_{t11}^2)$ and $b_t^3 = \Phi_3(d_{t12}^3, d_{t13}^3)$ are colored in red. (b),(d) and (f): The frequency scores of the newly computed data points $\tilde{d}_{t1}^1 = D_{\Phi_1}(b_t^1)$, $\tilde{d}_{t2}^2 = D_{\Phi_2}(b_t^2)$ and $\tilde{d}_{t3}^3 = D_{\Phi_3}(b_t^3)$ are computed according to the distances of the embedded point from its nearest neighbors in $\Phi_1$, $\Phi_2$ and $\Phi_3$, respectively. The new frequency scores $\tilde{d}_{t1}^1$, $\tilde{d}_{t2}^2$ and $\tilde{d}_{t3}^3$ are plotted at the end of $D_{\Phi_1}$, $D_{\Phi_2}$ and $D_{\Phi_3}$ and are colored black.
4.3. TRACKING AND DETECTION OF ANOMALIES (PROBLEMS)

In this section, an illustrative example, which was detected on data that belongs to the training dataset, shows how to track a buildup that ends in a deadlock, which caused the system to crash. As apposed to failures that occur because of system overload, which are easy to detect because of extremely high parameter values, deadlocks failures are characterized by small values that can easily be mistaken for normal behavior. A deadlock can begin with one system component that stops to function, and it may take some time until this affects other parts of the system that eventually cease to function as well.

The status of the system is analyzed according to the values of the frequency score functions. The frequency score of each point is calculated by summing the points distances from its nearest neighbors. Normal behaved points, which lie in the main cluster of the embedding, have low score values. Abnormal points, which lie on the edges of the main embedding cluster, or classified as outliers, have a large score value. The score function is colored according to its values. When the measured parameter, or group of parameters, behave normally, the frequency scores are small and their color is blue. Deviation from normal behavior is expressed by higher frequency scores and the color changes from blue towards red.

Figure 4.2.5: (a) Application of Algorithm 2 to $V$ with the newly arrived inputs $(\tilde{d}_1, \tilde{d}_2, \tilde{d}_3)$. The computed super-graph coordinates $c^t = V(\tilde{d}_1, \tilde{d}_2, \tilde{d}_3)$ are colored in red. (b) The frequency scores of the newly computed data point $\hat{d}^t = D_V(c^t)$ are computed according to the distances of the embedded point from its nearest neighbors in the super-graph $V$. The new frequency score $\hat{d}^t$ is plotted at the end of $D_V$ and is colored black.
Tracking a buildup of a problem is done by looking at the embedding and frequency score function of the top-level and middle-level nodes in the hierarchical tree. The evolution (buildup) of problems is likely to appear in at least one of these nodes. When the score function values of a top-level or middle-level node become high, we continue down the tree to the bottom-level nodes for detecting the parameters that caused the failure.

In this example, a problem that caused the system to crash is tracked. We look back 40 time steps prior to the crash to see if the buildup is detected. Figures 4.3.1 - 4.3.9 track the system behavior during the 40 time steps prior to its crash. The values of the frequency score functions are presented on the right images in each figure. The horizontal axis is the time and the vertical axis is the score. The corresponding embedding values of the 40 time steps prior to its crash are circled in black in the embeddings that are presented in the left images of each figure.

In Figs. 4.3.1 - 4.3.9, which illustrate this example, a line was drawn at the height of three standard divination. This line is used as a measure to indicate if the next arrived point is normal or abnormal. A new frequency score is above the 3-standard divination line, is considered abnormal.

Figures 4.3.1, 4.3.2, 4.3.3 and 4.3.4 track the system behavior of the super-graph, \( G_1 \), \( G_2 \) and \( G_3 \). Figures 4.3.5, 4.3.6, 4.3.7, 4.3.8 and 4.3.9 show the last 40 values of the frequency score functions of the bottom-level nodes.

It can be seen in Fig. 4.3.1 that the values of the score function were high (red) approximately 30, 20 and 10 time steps before the system crashed. The last three values are clearly an alarm that the system is in critical condition.
4.3. TRACKING AND DETECTION OF ANOMALIES (PROBLEMS)  66

Figure 4.3.1: Left: The embedding $V$ of the super-graph colored by score values. Right: Forty values of the super-graph’s density function $D_V$ that were measured prior to the system crash. Points above the line indicate abnormal behavior.

Going down the tree in Figs. 4.3.2, 4.3.3 and 4.3.4, explains the origin of these peaks in the top level super-graph nodes. Each of them originated from abnormal behavior in one of the intermediate nodes. Approximately 30 time steps prior to the system crash, the deviation in $D_{\Phi_1}$ showed unusual behavior. This is the beginning of the failure’s buildup process. In Fig. 4.3.3, the dark red point in the score function $D_{\Phi_2}$ occurred 10 time steps before the system crashed. It indicates of untypical behavior in this group. Most likely that this is related to the abnormal behavior in $D_{\Phi_1}$, which was seen a few minutes earlier. Figure 4.3.4 shows the behavior of $G_3$. In this group, most of the values are normal and only a few points in the score function reach values that are over the 3 standard deviations from the average density values of $D_{\Phi_3}$. Together with the score functions of $D_{\Phi_1}$ and $D_{\Phi_2}$, these small deviations influenced the value of $D_V$

Figure 4.3.2: Left: The embedding $\Phi_1$. Right: The values of the score function $D_{\Phi_1}$ of $G_1$ (middle level in Fig. 4.1.1) that were measured prior to the system crash.
Figure 4.3.3: Left: The embedding $\Phi_2$. Right: Measures from the score function $D_{\Phi_2}$ of $G_2$ (middle level in Fig. 4.1.1) 30 minutes before the system crashed.

Figure 4.3.4: Left: The embedding $\Phi_3$. Right: The score function $D_{\Phi_3}$ of $G_3$ (middle level in Fig. 4.1.1).

Meaningful information about the source of the system’s failure can be seen when going further down the tree, as shown in Figs. 4.3.5 - 4.3.9. In Fig. 4.3.5, the parameters $P_1$ and $P_3$ reach high density values. In fact, 15 minutes prior to the crash, $P_1$ showed extremely high values followed by very low values. $P_1$ measures the average process time of a certain transaction during the last minute. These values show that 15 minutes prior to the crash, the transaction’s process time was unusually long. After that, the sharp drop to zero implies that this transaction ceased to be processed.
4.3. TRACKING AND DETECTION OF ANOMALIES (PROBLEMS)

(a) The embedding $\Psi_1$ of $P_1$

(b) The score function $D_{P_1}$ of $P_1$

(c) The embedding $\Psi_2$ of $P_2$

(d) The score function $D_{P_2}$ of $P_2$

(e) The embedding $\Psi_3$ of $P_3$

(f) The score function $D_{P_3}$ of $P_3$

Figure 4.3.5: Left: The embedding $\Psi_1$, $\Psi_2$ and $\Psi_3$ (bottom level in Fig. 4.1.1), which belong to parameters $P_1$ – $P_3$ colored by their score. Right: The values of the last 40 time steps before the system crashed of the frequency score functions $D_1$, $D_2$ and $D_3$. In $D_1$ and $D_3$, exceptional values can be seen 30 time steps and 20 time steps before the system crashed. The last values of $D_1$, which change from very high to zero, implies that this transaction stop from functioning.
4.3. TRACKING AND DETECTION OF ANOMALIES (PROBLEMS)

Figure 4.3.6: Left: The embedding $\Psi_4$, $\Psi_5$ and $\Psi_6$ (bottom level in Fig. 4.1.1), which belong to parameters $P_4 - P_6$ colored by their score. Right: The last 40 values of the score functions $D_4$, $D_5$ and $D_6$ before the system crashed.

The score function values of group $G_2$ are interesting and give a justification to the hierarchical structure. In each of the functions in Figs. 4.3.7 and 4.3.8, the values are normal but their merge (Fig. 4.3.3) clearly confirms that the mutual relations between these parameters reached uncommon low probability combinations.
4.3. TRACKING AND DETECTION OF ANOMALIES (PROBLEMS) 70

(a) The embedding $\Psi_7$ of $P_7$

(b) The score function $D_{P_7}$ of $P_7$

(c) The embedding $\Psi_8$ of $P_8$

(d) The score function $D_{P_8}$ of $P_8$

(e) The embedding $\Psi_9$ of $P_9$

(f) The score function $D_{P_9}$ of $P_9$

Figure 4.3.7: Left: Embeddings $\Psi_7$, $\Psi_8$ and $\Psi_9$ of the parameters $P_7 - P_9$ (bottom level in Fig. 4.1.1) that belong to $G_2$. Right: The values of the score functions $D_7$-$D_9$ from the last 40 time steps before the system crashed.
4.3. TRACKING AND DETECTION OF ANOMALIES (PROBLEMS)

Figure 4.3.8: Left: The Embeddings $\Psi_{10}$ and $\Psi_{11}$ of the parameters $P_{10} - P_{11}$ that belong to $G_2$ (bottom level in Fig. 4.1.1). Forty values of the score functions $D_{10}$ and $D_{11}$ that were measured prior to the system crash.

For the group $G_3$, the fact that the individual trends of the two parameters in Fig. 4.3.9 are not completely correlated, can explain the rise of $G_3$’s frequency score function (in Fig. 4.3.4) to a high abnormal value.
4.4 Comparisons with other dimensionality reduction methods

In this section, the diffusion maps method, which was used in the learning algorithm in Section 4.1, is replaced by principal component analysis (PCA) and multidimensional scaling (MDS). First, the dataset complexity and structure is characterized by the applications of three different dimensionality reduction algorithms to the data. This step motivates the necessity to use a non-linear dimensionality reduction method like diffusion maps, which preserves the local distances that unfold the geometric structure of the data. The dimensionality reduction step of the proposed learning algorithm from Section 4.1 is done by
the application of PCA and MDS in Sections 4.4.2 and 4.4.3, respectively. Section 4.4.4 compares the reliability of the score functions, which were constructed, when the diffusion maps, PCA and MDS are used for detecting and tracking system failures.

4.4.1 Characterizing the geometry and structure of the dataset

This section analyzes the geometric structure of the high-dimensional dataset, which was captured from a performance monitor application, and was used as the running example throughout Chapter 4. The data, which consist of 13 parameters and \( N = 3577 \) points, is reduced to a lower dimension by the application of diffusion maps, PCA and MDS. The parameters \( \{ P_i \}, i = 1, \ldots, 13 \) are re-scaled by dividing each parameter by its norm. Each re-scaled parameter is denoted by \( P^\star_i = \frac{P_i}{\|P_i\|}, i = 1, \ldots, 13 \). Next, a dynamic-matrix is constructed on the set \( P^\star = \{ P^\star_1, \ldots, P^\star_{13} \} \). A row in the dynamic-matrix consists of \( \mu \) consecutive rows from the matrix \( P^\star \). Similarly to the construction in Section 4.1.2, \( \mu \) is set to 3. Figure 4.4.1 show 3 images of the embedding of the dataset to the first three diffusion maps coordinates. The images plot the same dataset, but are rotated for allowing a better view of the embedded data. The data is colored by time. It can be seen that the data lies on a non-linear manifold and that the diffusion coordinates organize the different system behaviors.

![Figure 4.4.1: Embedding of the dynamic-matrix \( P^\star \) to a 3-dimensional space by using the top three diffusion coordinates. The three images show different rotations of the diffusion coordinates. The points, which are colored by time, are organized by the diffusion coordinates in a way that enables to track both the system’s normal and abnormal behavior.](image)

Figure 4.4.2 shows the application of PCA and MDS to the dynamic-matrix \( P^\star \). The data points are colored according to the time evolution. Both methods do not capture the
4.4. COMPARISONS WITH OTHER DIMENSIONALITY REDUCTION METHODS

non-linear high-dimensional structure of the data. Although outliers are seen, it is difficult
to characterize the system’s normal dynamic behavior.

Figure 4.4.2: Left: Projection of the dynamic-matrix $P^*$ to a 3-dimensional space by using
the first three principal components. Right: Embedding of the dynamic-matrix $P^*$ to a 3-
dimensional space by the application of MDS

Comparing the results from the three dimensionality reduction algorithms show the
advantage of using the diffusion maps, which unfolds the data’s geometric structures, for
analyzing the given dataset.

4.4.2 Comparison with PCA

Principal component analysis (PCA) [47] is a standard method for dimensionality reduc-
tion. In PCA, the dimension of a given dataset is reduced while retaining most of the
original variability in the data. The dimensionality reduction step, which is applied to the
data of each node that belongs to the hierarchical three level embedding model, is per-
formed by PCA. The bottom level processing includes the application of PCA to each of
the dynamic matrices $\{\tilde{P}_i\}$, $i = 1 : 13$ and then constructing a frequency score function
(see Eq. (4.1.3)) in the embedded space. Figures 4.4.3 and 4.4.4 show the PCA projections
and the frequency score functions for parameters $P_1, \ldots, P_6$. 
4.4. COMPARISONS WITH OTHER DIMENSIONALITY REDUCTION METHODS

Figure 4.4.3: The outcome from the learning phase using PCA for representing the three parameters $P_1, P_2, P_3$ that were marked in Fig. 4.1.1. The projection of each parameter, which is seen in (a), (c) and (e), is obtained by projecting its data onto the first two principal components. The corresponding frequency score functions are seen in (b), (d) and (f).
4.4. COMPARISONS WITH OTHER DIMENSIONALITY REDUCTION METHODS

4.4.4 The outcome from the learning phase using PCA for representing the three parameters $P_4, P_5, P_6$ that were marked in Fig. 4.1.1. The projection of each parameter, which is seen in (a), (c) and (e), is obtained by projecting its data onto the first two principal components. The corresponding frequency score functions are seen in (b), (d) and (f).

Figure 4.4.5 shows the PCA projections and the frequency score functions for parameters $P_7, \ldots, P_{11}$. 
4.4. COMPARISONS WITH OTHER DIMENSIONALITY REDUCTION METHODS

Figure 4.4.5: The PCA projections of the five parameters $P_7,\ldots, P_{11}$, which were marked in Fig. 4.1.1, are plotted on the left. The corresponding frequency score functions are plotted on the right.
The PCA projections and the frequency score functions for parameters $P_{12}$ and $P_{13}$ are displayed in Fig. 4.4.6.

Figure 4.4.6: The PCA projections of the parameters $P_{12}$ and $P_{13}$, which were marked in Fig. 4.1.1, are plotted on the left. The corresponding frequency score functions are plotted on the right.

The middle level and top level processing are carried out as was described in Section 4.1.3 and 4.1.4. The results are displayed in Fig. 4.4.7.
4.4. COMPARISONS WITH OTHER DIMENSIONALITY REDUCTION METHODS

Figure 4.4.7: The left images (a),(c) and (e) show the PCA projections of the three middle level nodes $G_1$, $G_2$ and $G_3$, which were displayed in Fig. 4.1.1. Their frequency score functions are seen in images (b), (d) and (f). The projection of the top level node is displayed in (g) and the corresponding frequency score function is seen in (h).

Substituting the diffusion maps with PCA in the learning phase does not produce similar results. The non-linear structures of the data are not preserved when the data is projected onto the principal components.
4.4.3 Comparison with MDS

The goal of multidimensional scaling (MDS) is to provide a visual representation of proximities pattern such as similarities or distances among a set of objects. MDS [26, 49] represents a collection of techniques that maps the high dimensional data representation into a low-dimensional representation while retaining the pairwise distances between the data points as much as possible. The dimensionality reduction method in the learning algorithm, which was described in Section 4.1, is replaced by the classical MDS, which uses the pairwise Euclidean distances between data points as its input. Figures 4.4.8 - 4.4.12 display the results of applying MDS to the 13 dynamic matrices that belong to the bottom level parameters, which were displayed in Fig. 4.1.1.

Figures 4.4.8 and 4.4.9 present the embedding of the dynamic behavior of the six parameters $P_1$-$P_6$ by the application of MDS.
4.4. COMPARISONS WITH OTHER DIMENSIONALITY REDUCTION METHODS

Figure 4.4.8: The outcome from the learning phase using MDS for representing the three parameters $P_1, P_2, P_3$ that were marked in Fig. 4.1.1. In (a), (c) and (e), each parameter is embedded into two dimensions. The corresponding frequency score functions are displayed in (b), (d) and (f).
4.4. COMPARISONS WITH OTHER DIMENSIONALITY REDUCTION METHODS

Figure 4.4.9: The outcome from the learning phase using MDS for representing the three parameters \( P_4, P_5, P_6 \) that were marked in Fig. 4.1.1. In (a), (c) and (e), each parameter is embedded into two dimensions. The corresponding frequency score functions are seen in (b), (d) and (f).

The MDS embeddings and frequency score functions for parameters \( P_7, \ldots, P_{11} \) are displayed in Fig. 4.4.10.
4.4. COMPARISONS WITH OTHER DIMENSIONALITY REDUCTION METHODS

Figure 4.4.10: The MDS embeddings of the five parameters $P_7, \ldots, P_{11}$, which were marked in Fig. 4.1.1, are plotted on the left. The corresponding frequency score functions are plotted on the right.
The MDS embeddings and the frequency score functions for parameters $P_{12}$ and $P_{13}$ are displayed in Fig. 4.4.11.

Figure 4.4.11: The MDS embeddings of the parameters $P_{12}$ and $P_{13}$, which were marked in Fig. 4.1.1, are plotted on the left. The corresponding frequency score functions are plotted on the right.

The middle level and the top level processing are carried out as was described in Section 4.1.3 and 4.1.4. The results are displayed in Fig. 4.4.7.
4.4. COMPARISONS WITH OTHER DIMENSIONALITY REDUCTION METHODS

Figure 4.4.12: The left figures (a),(c) and (e) show the MDS embeddings of the three middle level nodes $G_1$, $G_2$ and $G_3$, which were displayed in Fig. 4.1.1. Their frequency score functions are displayed in figures (b), (d) and (f). The embedding of the top level node is displayed in (g) and the corresponding frequency score function is displayed in (h).

The application of MDS to the dataset produces similar results to those of the PCA. Although the application of MDS to single parameters produce reasonable results, embed-
4.4. COMPARISONS WITH OTHER DIMENSIONALITY REDUCTION METHODS

4.4.4 Comparing between DM, PCA and MDS to track and detect system failures

Sections 4.1, 4.4.2 and 4.4.3 apply different dimensionality reduction algorithms within the hierarchical learning algorithm. The application of diffusion maps organized the data in a way that reveals its dynamically evolving process. Section 4.3 focused on finding a system failure that was detected by the score functions. The failure, which was a part of the training set, was the most severe failure in the training set, since it caused the system to crash. We assume that the failure was caused by a deadlock. The reason that it was difficult to detect is because the system’s behavior prior to the crash was not characterized by high, abnormal values in each of its parameters. Figure 4.4.13 compares the frequency score function that were constructed on the top level node following by the application of diffusion maps, PCA and MDS. The failure, which was tracked in Section 4.3, is marked in red. Even though the three methods detected the anomaly shortly before the system crashed, the frequency score function that was constructed from the diffusion coordinates, indicated the existence of anomalous behavior sometime before the crash. This advantage is also seen for the anomaly that is marked in green. In the green ellipse, an anomaly was detected by the three methods, but there is a build-up of unstable behavior towards the failure in the diffusion maps based score function. The purple ellipse marks a system trend that indicates the existence of burden and load in the system. This behavior, which is also seen before and after the green ellipse and before the red ellipse, is a normal phenomena that is not expressed by the PCA and MDS low dimensional representations.
4.5 Discussions

This section elaborates how the parameters throughout this chapter were chosen.

The first parameter $\mu$ is the length of the dynamic path. It defines the length of the time trajectories that is used as input to the diffusion maps algorithm. The performance monitor was tuned to record every one minute (it can be tuned differently). This produces a coarse resolution of the system behavior. In an optimal setting, a finer recording resolution would allow to track the system behavior in more detail. Since system failures were characterized by oscillations, the parameter $\mu$ was set to 3, which is the smallest value that can capture an oscillation.

The application of diffusion maps requires to set the scaling parameter $\epsilon$ for the reconstruction of the diffusion kernel. The median (see Section 2.1.3) was chosen and it determines the size of the local neighborhood of the input data at different levels in the hierarchical tree.

Figure 4.4.13: Comparison between the frequency score functions that were constructed on the top level node using the following applications: DM, PCA and MDS. The failure, which was tracked in Section 4.3, is marked in red. An anomaly was detected by the three methods (marked by the green ellipse). A build-up of unstable behavior towards the failure is only seen in the diffusion maps based score function.
The number of data points, which were used in the training step, was $N = 3577$. The frequency score functions, which were constructed on each embedding, used $\eta = 10$ nearest neighbors. This parameter was set empirically. Setting $\eta$ to be smaller than 5 introduces noise in the score functions. Taking $\eta$ to be larger than 30 smoothes the score functions.

4.6 Summary

In this chapter, we introduced a general un-supervised data mining approach for detection and prediction of anomaly that are suitable for a wide range of dynamically evolving systems. First, a training step is carried out on a designated training dataset. The high-dimensional dynamic data is embedded into a low-dimensional space by using an hierarchical tree structure. This system is decomposed into sub-systems with different resolutions. The top node of the hierarchical tree describes the entire system, middle level nodes describe different sub-systems and bottom level node describe separately the dynamic behavior of each input parameter. Diffusion maps algorithm is applied to each tree node to embed the sub-system into a lower dimension space. This embedded space is described by the diffusion maps coordinates. A frequency score function, which is defined on the points in their embedded space, provides a measure to identify input data points as normal or abnormal. Next, a method that evaluates the status of newly arrived data points is introduced. The embedding, which were constructed in the training step, are extended online to embed the newly arrived data points. This allows us to detect online if the newly arrived data points are normal or abnormal. Finally, an illustrative example of a system crash shows how the proposed approach can be used to predict that the system state is unstable.
Chapter 5

Updating the training dataset

5.1 Introduction

The need for graph matching and manifold alignment arises in many real life applications. Many efforts have been channeled into finding reliable machine learning and data mining algorithms for understanding the data. These algorithms have to be able to cope with large amounts of new data. Sometimes, the training dataset may no longer match and reflect the activities and nature of the newly arrived data. For these cases, we need to perform an occasional check whether the training set has to be updated. A reliable way to compare between the newly collected data chunks and the training datasets, is to match between these datasets in a space that best represents them. Once the datasets are compared, a decision can be made whether or not to store and use the newly arrived data chunks as the new training datasets.

Over the last years, several works, which dealt how to update the training dataset, have emerged. Local tangent space to align multiple datasets and describe them with global coordinates is described in [78]. ISOMAPs were used in [8] to align between two graphs in an embedded space. The second eigenvector of the Laplacian matrix together with the edit distance [56] were used in [61] to match between graphs. Recently, a method based on the distributions of the diffusion distances was used in [14] for finding shape similarities.

The method, which is described in this chapter, combines the non linear multicue data
5.2 Updating the training set of the operational system

Sections 4.1.2 - 4.2 describe an hierarchical methodology for learning and online detecting of data that was collected from a dynamic system. The methodologies were demonstrated on data that was collected from a performance monitor of a transaction based system. Since the system is dynamic, it changes with time, and new phenomena, which neither captured nor appeared in the training dataset, may appear. For maintaining a reliable detection system, the training dataset needs to be updated occasionally to fit the newly arrived data. In this section, a method, which determines whether or not the current training dataset has to be updated, is introduced.

Denote the training dataset, which was collected for the performance monitor application as $P_A$. The set $P_A$ records each time unit 13 parameters of different types and from different components in the transaction based system. The dataset $P_A$ goes through the learning process, which was described in Sections 4.1.2-4.1.4. The learning process generates an hierarchical embedding tree. This tree is used for online detection and prediction of the system behavior, as described in Sections 4.2-4.3. Over time, a new dataset is accumulated. Denote the new dataset by $P_B$. The hierarchical learning process, which was applied to $P_A$, is applied to the set $P_B$. The comparison between the original training dataset $P_A$ and the new dataset $P_B$ is performed by comparing their hierarchical embedding trees. Figure 5.2.1 shows two hierarchical embedding trees that were generated by the application of the learning process from Section 4.1.
5.2. UPDATING THE TRAINING SET OF THE OPERATIONAL SYSTEM

Figure 5.2.1: Two embedding trees that were generated by the application of the learning process in Section 4.1. The blue and the red trees are the hierarchial embedding of the original training dataset $P_A$ and of the newly collected data $P_B$, respectively.

The embedding tree that was generated from $P_B$ (the red tree in Fig. 5.2.1) may hold new phenomena. These new behaviors may be seen in all the tree nodes, or they may be relevant to just some of the system components and seen only in a few tree nodes. The algorithm introduced in this section, compares between the two embedding trees by checking pairs of corresponding nodes. The algorithm has two steps:

1. Merge embedding graphs: The embedding graphs of two corresponding nodes from the two trees are merged into one.

2. Compare the embedded datasets: Once the data points, which belong to $P_A$ and $P_B$, are located in the same embedding graph, the two datasets are compared in this embedded space.

Next, the two steps are explained in detail.

### 5.2.1 Merging embedding graphs

The algorithm for merging two datasets, denoted as $P_A$ and $P_B$, in their embedding space is based on least squares minimization. The idea is to form a new dataset, denoted by $P_{AB}$, which holds sampled data points from $P_A$ and $P_B$. The embedding of $P_{AB}$ is computed...
via the applications of the diffusion maps as was described in Section 4.1.2. Next, the embeddings of $P_A$ and $P_B$ are aligned with the embedding of $P_{AB}$ by an affine (linear) transformation. Algorithms 3 and 4 describe the process of merging the embeddings of two datasets into one.

Algorithm 3 Merge embedded graphs

**Input:** Two datasets $P_A = \{x_1, \ldots, x_m\}$ and $P_B = \{y_1, \ldots, y_n\}$, where $x_i, y_j \in \mathbb{R}^d$

**Output:** Embedding coordinates $\Psi_{AB}$, which embed $P_A$ and $P_B$ into a merged graph.

1: Compute the diffusion maps embedding coordinated for $P_A$ and $P_B$. Denote these embeddings by $\Psi_A = \{\psi_{A,1}, \ldots, \psi_{A,q}\}$ and $\Psi_B = \{\psi_{B,1}, \ldots, \psi_{B,q}\}$, $q > 3$.

2: Construct the set $P_{AB}$, which holds approximately half of the points from $P_A$ and $P_B$. For example: $P_{AB} = \{x_{2k}, y_{2l}\}$, $x_{2k} \in P_A$, $y_{2l} \in P_B$, $k = 1, \ldots, \frac{m}{2}$, $l = 1, \ldots, \frac{n}{2}$.

3: Apply the diffusion maps to $P_{AB}$ and save $q$ embedding coordinates $\Psi_{AB} = \{\psi_{AB,1}, \ldots, \psi_{AB,q}\}$.

4: Align $\Psi_A$ to $\Psi_{AB}$ by executing the Graph Alignment (Algorithm 4) with the group $C_A = \{x_{2k}\}$, $k = 1, \ldots, \frac{m}{2}$ as the landmark indices of $P_A$ and their corresponding indices from $P_{AB}$.

5: Align $\Psi_B$ to $\Psi_{AB}$ by executing the Graph Alignment (Algorithm 4) with the group $C_B = \{y_{2l}\}$, $l = 1, \ldots, \frac{n}{2}$ as the landmark indices of $P_B$ and their corresponding indices from $P_{AB}$.

Algorithm 4 Graph alignment (GA)

**Input:** Two datasets $P_A$ and $P_{AB}$, their embedding graphs $\Psi_A$ and $\Psi_{AB}$. A group of landmark points $C$, such that $C \subset P_A$ and $C \subset P_{AB}$.

**Output:** A function $\tilde{f}$ that aligns between the embeddings $\Psi_A$ and $\Psi_{AB}$ based on the landmarks $C$.

1: Set $q > 3$.

2: Compute the $q$-dimensional affine function with the least square error between the sets $\Psi_A(C)$ and $\Psi_{AB}(C)$. The function $f : \{\psi_{A,1}, \psi_{A,2}, \ldots, \psi_{A,q}\} \rightarrow \{\psi_{AB,1}, \psi_{AB,2}, \ldots, \psi_{AB,q}\}$ satisfies $f(\Psi_A(C)) \approx \Psi_{AB}(C)$.

3: Use $f$ to align the points $P_A\setminus C$ into the coordinates of $\Psi_{AB}$ by applying $f(\Psi_A(P_A\setminus C))$.

4: For aligning embeddings in $\mathbb{R}^3$ (or $\mathbb{R}^2$), use $\tilde{f}$, the projection of $f$ into $\mathbb{R}^3$ (or $\mathbb{R}^2$).

The applications of Algorithms 3 and 4 is demonstrated on several pairs of corresponding nodes from the two hierarchial trees in Fig. 5.2.1. We begin with $P_3$. Denote by $P^A_3$
5.2. UPDATING THE TRAINING SET OF THE OPERATIONAL SYSTEM

the collected data from the parameter $P_3$ in the training dataset (the blue) and by $P_3^B$ the newly arrived captured data from the parameter $P_3$ (red tree).

The following steps are carried out when Algorithms 3 and 4 are applied:

1. The embedding $\Psi_3^A$ of $P_3^A$ and $\Psi_3^B$ of $P_3^B$ are computed as was described in Section 4.1.2. $q > 3$ embedding coordinates from $\Psi_3^A$ and $\Psi_3^B$ are saved.

2. Half of the points from $P_3^A$, denoted by $C_3^A$, are marked as landmarks.

3. Half of the points from $P_3^B$, denoted by $C_3^B$, are marked as landmarks.

4. The diffusion maps embedding $\Psi_3^{AB}$ of $C_3^{AB} = \{C_3^A, C_3^B\}$ is computed and $q > 3$ embedding coordinates from $\Psi_3^{AB}$ are saved.

Figure 5.2.2 shows the embedding $\Psi_3^A$ of $P_3^A$ (in blue), the embedding $\Psi_3^B$ of $P_3^B$ (in red) and the embedding $\Psi_3^{AB}$ of the landmark group $\{C_3^A, C_3^B\}$ (in blue and red). The embedding $\Psi_3^{AB}$ contains half of the points of $P_3^A$ and half of the points of $P_3^B$.

Figure 5.2.2: Left: The embedding $\Psi_3^A$ of $P_3^A$. Center: The embedding $\Psi_3^{AB}$, which consists of half of the points from $P_3^A$ and half of the points from $P_3^B$. Right: The embedding $\Psi_3^B$ of $P_3^B$.

Next, Algorithm 4 is applied twice. The first application aligns between $\Psi_3^A$ and $\Psi_3^{AB}$. It is based on the set of landmark points $C_3^A$. The second application finds a function that aligns between $\Psi_3^B$ and $\Psi_3^{AB}$. It is based on the set of landmark points $C_3^B$. Figure 5.2.3 shows the results of the two alignments. The blue points are the embedding of the landmarks $C_3^A$ in $\Psi_3^{AB}$. The red points are the embedding of the landmarks $C_3^B$ in $\Psi_3^{AB}$.
The light blue are the points $P_3^A \setminus C_3$, which are aligned from $\Psi_3^A$ to $\Psi_3^{AB}$ by the application of Algorithm 4. The pink points are the points $P_3^B \setminus C_3$, which are aligned from $\Psi_3^B$ to $\Psi_3^{AB}$ by the application of Algorithm 4. It can be seen that in the embedded space $\Psi_3^{AB}$, the points that belong $P_A$ and the points that belong to $P_B$ have similar behavior.

Figure 5.2.3: The graph $\Psi_3^{AB}$, which embeds the points of $P_3^A$ and $P_3^B$. This embedding merges the embedding $\Psi_3^A$ and $\Psi_3^B$ that are shown in Fig. 5.2.2. The red and blue circles are the points of $C_3$. The light blue stars are the points of $P_3^A \setminus C_3$, the pink stars are the points of $P_3^B \setminus C_3$.

The application of the Algorithm 3 is demonstrated on another bottom level node which is the node that embeds the parameter $P_{13}$. Algorithm 3 merges between the embeddings of the corresponding marked tree nodes.

Figures 5.2.4 and 5.2.5 show the application of Algorithm 3 to two datasets $P_{13}^A$ and $P_{13}^B$, which were collected from the parameter $P_{13}$. In Figure 5.2.4, the embedding $\Psi_{13}^A$ of $P_A$ is colored blue and the embedding $\Psi_{13}^B$ of $P_B$ is colored red. The embedding $\Psi_{13}^{AB}$, which embedded half of the points of $P_A$ and half of the points of $P_B$, is colored blue and red. In can be seen that the right section of the graph $\Psi_{AB}$ contains points only from $P_B$. This implies that these points capture the new behavior in this parameter, which was not seen in the training data $P_3^A$.
5.2. UPDATING THE TRAINING SET OF THE OPERATIONAL SYSTEM

Figure 5.2.4: Left: The embedding $\Psi^A_{13}$ of $P^A_{13}$. Center: The embedding $\Psi^{AB}_{13}$, which consists of half of the points from $P^A_{13}$ and half of the points from $P^B_{13}$. Right: The embedding $\Psi^B_{13}$ of $P^B_{13}$.

Figure 5.2.5: $\Psi^{AB}_{13}$, which embeds the points of $P^A_{13}$ and $P^B_{13}$. This embedding merges the embedding of the blue and red nodes $P^A_{13}$ and $P^B_{13}$ that were marked in Fig. 5.2.1. The red and pink points on the right part of the graph, which belong to the red tree node ($P^B_{13}$), have new behavior that was not seen in the blue point set ($P^A_{13}$).

The application of the Algorithms 3 and 4 is demonstrated on the middle level tree nodes and on the super-graph node. In Figure 5.2.1, we denote the data from the group $G_1$ in the blue tree by $G^A_1$ and the data from the group $G_1$ in the red tree by $G^B_1$. The embedding of the groups $G^A_1$ and $G^B_1$ are denoted as $\Phi^A_1$ and $\Phi^B_1$, respectively. These embeddings were constructed by the group score functions of the bottom level of the tree nodes as was described in Section 4.1.3. The graph merge Algorithm 3 is applied to $G^A_1$
and $G^B_1$ together using $\Phi^A_1$ and $\Phi^B_1$. Algorithm 3 constructs the group $C_{G_1}$, which takes half the points from $G^A_1$ and half the points from $G^B_1$, and generates an embedding $\Phi^{AB}_1$. Figure 5.2.6 shows $\Phi^A_1$, $\Phi^B_1$ and $\Phi^{AB}_1$.

![Figure 5.2.6](image1.png)

(a) (b) (c)

Figure 5.2.6: Left: In blue, the embedding $\Phi^A_1$ of $G^A_1$. Center: The embedding $\Phi^{AB}_1$ of the joint points from $G^A_1$ (in blue) and $G^B_1$ (in red). Right: In red, the embedding $\Phi^B_1$ of $G^B_1$.

The points from $G^A_1$ and $G^B_1$ that do not belong to $C_{G_1}$ are aligned to $\Phi^{AB}_1$ by the application of Algorithm 4. This is presented in Fig. 5.2.1. The new accumulated dataset $G^B_1$ holds in it the new behavior, which is expressed by areas that are red and pink (without blue) in $\Phi^{AB}_1$.

![Figure 5.2.7](image2.png)

Figure 5.2.7: $\Phi^{AB}_1$, which embeds the points of $C_{G_1}$, in blue and red circles. The blue circles are points that belonged to $G^A_1$ and the red circles belong to $G^B_1$. The light blue stars are the points from $G^A_1 \setminus C_{G_1}$ and pink stars are the points $G^B_1 \setminus C_{G_1}$. The light blue and pink points were aligned to $\Phi^{AB}_1$ by the application of Algorithm 4.
5.2. UPDATING THE TRAINING SET OF THE OPERATIONAL SYSTEM

In a similar manner, the middle level node $G_2^A$ is merged with new samples from $G_2^B$. Figure 5.2.8 presents the embedding $\Phi_{2}^{AB}$, which was constructed by the application of Algorithm 3, and embeds the points $C_{G_B}$ that were sampled from $G_2^A$ and $G_2^B$. In addition, the embedding $\Phi_{2}^{A}$ of $G_2^A$ and the embedding $\Phi_{2}^{B}$ of $G_2^B$ are presented. Notice how in $\Phi_{2}^{AB}$ the behavior of the new dataset (in red) has the same structure has the embedding of training dataset (in blue) while there is a shift in the embedding position.

Figure 5.2.8: Left: The embedding $\Phi_{2}^{A}$ of $G_2^A$. Center: The embedding $\Phi_{2}^{AB}$ of $C_{G_2}$. Right: The embedding $\Phi_{2}^{B}$ of $P_2^B$.

Embedding of the points $G_2^A$ and $G_2^B$ in $\Phi_{2}^{AB}$ is presented in Fig. 5.2.9. The red points, which belong to $G_2^B$, are separated from the blue points which belong to $G_2^A$. The structure is similar.
5.2. UPDATING THE TRAINING SET OF THE OPERATIONAL SYSTEM

Figure 5.2.9: $\Phi_{2}^{AB}$, which was created by $C_{G_2}$. The blue circles are the points of $C_{G_2}$ that belong to $G_2^A$ and the red circles are the points from $C_{G_2}$ that belong to $G_2^B$ (in red). The light blue stars are the points from $G_2^A \setminus C_{G_2}$ and pink stars are the points $G_2^B \setminus C_{G_2}$. The light blue and pink points were aligned to $\Phi_{2}^{AB}$ by the application of Algorithm 4.

The embeddings of the last middle-level node are merged to one by the application of Algorithm 3. Figure 5.2.1 marks the node $G_3$ in both trees, $G_3^A$ in the blue tree and $G_3^B$ in the red tree. The embedding $\Phi_{2}^{AB}$, which is seen in Fig. 5.2.10, embeds the group $C_{G_3}$, which consists of half of the points from $G_3^A$ and half of the points from $G_3^B$.

Figure 5.2.10: (a) The embedding $\Phi_{3}^{A}$ of $G_3^A$. Center: The embedding $\Phi_{3}^{AB}$ of $C_{G_3}$. Right: The embedding $\Phi_{3}^{B}$ of $G_3^B$.

The result of Algorithm 4 is the points from $G_3^A$ and $G_3^B$ embedded in $\Phi_{3}^{AB}$. It is presented in Fig. 5.2.11. The behavior of the blue and light blue points (that belong $G_3^A$)
5.2. UPDATING THE TRAINING SET OF THE OPERATIONAL SYSTEM

in $\Phi^A_3B$ is quite similar to the behavior of the red and pink points (these belong to $G^B_3$).

Figure 5.2.11: $\Phi^A_3B$, which was created by $C_{G_3}$. The blue circles are the points of $C_{G_3}$ that belong to $G^A_3$ and the red circles are the points from $C_{G_3}$ that belong to $G^B_3$ (in red). The light blue stars are the points from $G^A_3 \setminus C_{G_3}$ and pink stars are the points $G^B_3 \setminus C_{G_3}$. The light blue and pink points were aligned to $\Phi^A_3B$ by the application of Algorithm 4.

Algorithm 3 is applied to the top level-tree node in the training dataset embedding tree and in the new dataset embedding tree. The embedding of the top level tree node was denoted as the super-graph in Section 4.1.4. It embeds the complete system behavior to a low-dimensional space. The two super-graph nodes, which embed the system, are marked in Figure 5.2.1.

The super-graph embeds the score function of the middle level nodes. Denote the input for the super-graph, which belongs to the training dataset (the blue tree in Fig. 5.2.1), by $S^A$ and denote by $S^B$ the input for the super-graph of the new dataset (the red tree in Fig. 5.2.1). Figure 5.2.12 shows the embedding of $S^A$, denoted by $V_A$ on the left, and the embedding of $S^B$, denoted by $V_B$ on the right. The embedding of the group $C_S$ that holds half the points from $S^A$ together with half of points from $S^B$ is denoted by $V_{AB}$ and presented in Fig. 5.2.12 (center).
5.2. UPDATING THE TRAINING SET OF THE OPERATIONAL SYSTEM

Figure 5.2.12: Left: The embedding $V^A$ of $S^A$. Center: The embedding $V^{AB}$ of the landmark points $C_S$ that belong to $S^A$ and $V^B$. Right: The embedding $V^B$ of $S^B$.

Application of Algorithm 4 aligns the points from $V_A$ and $V_B$ to $V_{AB}$. Figure 5.2.13 shows the super-graph embedding $V_{AB}$. The blue points and the red circles are the points from $C_A$. The blue points belong to $S_A$ and the red points to $S_B$. The light blue and pink stars are the rest of the points from the groups $S_A$ and $S_B$, respectively, which were aligned to $V_{AB}$. Although some new behavior is seen in a couple of the middle level and bottom level tree nodes, Fig. 5.2.12 shows that the super-graph, which embeds the training dataset and the super-graph that embeds the new dataset, have similar structures. This implies that the overall behavior of the transaction-based system did not change much.

Figure 5.2.13: The embedding $V_{AB}$, which embeds the super-graph points from the original training dataset (blue and light blue points) and the new dataset (red and pink points). The blue and the red are the top-level nodes in Fig. 5.2.1.
5.2.2 Comparing between embedded datasets

Section 5.2.1 described the first step of an algorithm that checks if the training dataset has to be updated. The training dataset is compared with a newer dataset that was generated by the same dynamic system at a later time. Both sets are first embedded by an hierarchical structure as was described in Section 4.1 and displayed in Fig. 5.2.1. Then, the embeddings, which belong to corresponding pairs of hierarchical tree-nodes, are merged. The merging process was explained in Section 5.2.1. Once the embeddings of two corresponding nodes are merged into one new embedding, we want to check in the newly created embedding space whether the new data contains new behavior (trend), which was not captured in the original training dataset.

This section describes an algorithm that compares between two datasets that lie in the same embedding space. New behavior in data points, which belong to the new dataset, can be expressed in many ways. The obvious way is when a region in the merged embedding contains only data points from the new dataset. New behavior can also be expressed by a different distribution of the data points in the same embedding structure. The introduced algorithm is scale free and compares between two graphs based on their geometries. The average time to reach every node in each graph is calculated. We denote this as the characteristic relaxation time or the characteristic diffusion time of the graph. Since the characteristic relaxation time describes the geometry the dataset, it can distinguish between two datasets that have different structures. The characteristic relaxation time was used as a measure for image segmentation in [1]. There, the criteria for merging two segments from a given image was based on the similarity of the segment’s relaxation time.

The comparison method is described in Algorithm 5. It employs two sets of data points: data points from the training dataset and data points from the new dataset, which are embedded by the same low-dimensional coordinates. The algorithm computes the characteristic diffusion time for the embedded training data points, for the embedded new points and for a subset of data points that are drawn from both of these groups. The characteristic diffusion time, as was defined in [57], is computed by $1/(1-\lambda_2)$, where $\lambda_2$ is the second eigenvalue from the diffusion maps. A threshold is defined to determine whether the values of the three calculated characteristic diffusion times are too different. In case they are, we conclude that the new dataset contains a new behavior. The training dataset
5.2. UPDATING THE TRAINING SET OF THE OPERATIONAL SYSTEM

should be updated and replaced with a merged dataset, which holds data points from the new dataset as well as from the original one.

Updating the dataset of a given node implies that the pre-processing step of the multi-scale approximation and extension method, which was described in Section 4.2, needs to be re-computed. The frequency score functions, which are easily computed, can be updated based on the joint embedding to capture the new statistics of the new data. This is done independently of the result of Algorithm 5.

**Algorithm 5** Compare graph structures

**Input:** Embedding coordinates $\Psi_{AB}$ that embed two sets of points $P_A$ and $P_B$. A threshold.

**Output:** $\delta_A$ - the characteristic diffusion time of $P_A$ in $\Psi_{AB}$, $\delta_B$ - the characteristic diffusion time of $P_B$ in $\Psi_{AB}$ and $\delta_{AB}$ - the characteristic diffusion time of $P_{AB}$ in $\Psi_{AB}$

1: Denote the values of the points $\Psi_{AB}(P_A)$ and $\Psi_{AB}(P_B)$ by $I_A$ and $I_B$, respectively.
2: Denote a sampled set of the points in $\Psi_{AB}$ as $I_{AB}$, where half of the points in $I_{AB}$ originated from $P_A$ and the other half from $P_B$.
3: Apply the diffusion maps to $I_A$, $I_B$ and $I_{AB}$. Denote the diffusion maps by $\Upsilon^A = \{\tau_1^A v_1^A, \tau_2^A v_2^A, \ldots\}$, $\Upsilon^B = \{\tau_1^B v_1^B, \tau_2^B v_2^B, \ldots\}$ and $\Upsilon_{AB} = \{\tau_1^{AB} v_1^{AB}, \tau_2^{AB} v_2^{AB}, \ldots\}$, respectively.
4: Compute $\delta_A = \frac{1}{1-\tau_1^A}$, $\delta_B = \frac{1}{1-\tau_1^B}$ and $\delta_{AB} = \frac{1}{1-\tau_2^{AB}}$.
5: If $\max\{\frac{\delta_A}{\delta_{AB}}, \frac{\delta_B}{\delta_{AB}}, \frac{\delta_{AB}}{\delta_A}, \frac{\delta_{AB}}{\delta_B}\} > \text{threshold}$ then, the joint embedding $\Psi_{AB}$ should replace $\Psi_A$ or $\Psi_B$.

Algorithm 5 is applied to the merged bottom-level tree nodes in Fig. 5.2.1. The threshold for the transaction-based system application was set to 2. Tables 5.2.1, 5.2.2 and 5.2.3 show the results from Algorithm 5. For each bottom-level node, which embeds the parameter’s training data, denoted by $P_i^A$, $i = 1, \ldots, 13$, new data, denoted as $P_i^B$, $i = 1, \ldots, 13$ were collected. The application of Algorithms 3 and 4 was demonstrated in Section 5.2.1 and it computed the new embedding coordinates $\Psi_{AB}$, which embed the points of $P_i^A \cup P_i^B$. Algorithm 5 computes the characteristic diffusion times $\delta_A$, $\delta_B$ and $\delta_{AB}$ for each parameter $P_i$. A significant difference between these values implies that the training set needs to be updated. For example, the points of the embedding graph, which were presented in Fig. 5.2.3, are the input to Algorithm 5. Algorithm 5 applies the dif-
fusion maps procedure to the set $I_3^A$, which consists of the blue and light blue points of Fig. 5.2.3 and to the set $I_3^B$, which consists of the red and pink points of Fig. 5.2.3. It also applies the diffusion maps procedure to the set $I_3^{AB}$, which is a sampled set of $I_3^A$ and $I_3^B$. The characteristic diffusion times of $I_3^A$, $I_3^B$ and $I_3^{AB}$ are seen in the third column in Table 5.2.1. The results show that for this parameter, the training dataset does not need to be updated. This is also seen in Fig. 5.2.3 as the blue and red points lay in the same space with a similar distribution.

Table 5.2.1 shows characteristic diffusion times that were calculated for the parameters $P_1, \ldots, P_6$. When the threshold is 2, only the first parameter, $P_1$, should be updated. Table 5.2.2 shows the characteristic diffusion times which are the output of Algorithm 5 for the parameters $P_7, \ldots, P_{11}$. For threshold 2, none of these parameters have to be updated, although the results for $P_7$ and $P_9$ are close to 2, which indicates that there was a new behavior in the new dataset. Table 5.2.3 shows the characteristic diffusion times for the parameters $P_{12}$, $P_{13}$. For parameter $P_{13}$, which is displayed in Fig. 5.2.5, the differences between the sets $P_{13}^A$ and $P_{13}^B$ are reflected by a difference in the $\delta$ values of the last column of Table 5.2.3. The training data for $P_{13}$ should be updated.

<table>
<thead>
<tr>
<th></th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
<th>$P_5$</th>
<th>$P_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_A = \frac{1}{1-\tau_A^2}$</td>
<td>5.6854</td>
<td>1.3461</td>
<td>1.0226</td>
<td>1.0226</td>
<td>1.0226</td>
<td>1.0226</td>
</tr>
<tr>
<td>$\delta_B = \frac{1}{1-\tau_B^2}$</td>
<td>1.4448</td>
<td>1.5469</td>
<td>1.0303</td>
<td>1.0303</td>
<td>1.0303</td>
<td>1.0303</td>
</tr>
<tr>
<td>$\delta_{AB} = \frac{1}{1-\tau_{AB}^2}$</td>
<td>4.8837</td>
<td>1.3183</td>
<td>1.8746</td>
<td>1.8746</td>
<td>1.8746</td>
<td>1.8746</td>
</tr>
</tbody>
</table>

Table 5.2.1: The characteristic diffusion time measures for nodes $P_1, \ldots, P_6$. $\delta_A$ measures the diffusion time of the points $P_1^A, \ldots, P_6^A$, which were used in the training phase. $\delta_B$ measures the diffusion time of the new values of $P_1^B, \ldots, P_6^B$ that were captured from the parameters $P_1, \ldots, P_6$. $\delta_{AB}$ measures the diffusion time of the joint sets $P_{1AB}, \ldots, P_{6AB}$. 
5.2. Updating the Training Set of the Operational System

<table>
<thead>
<tr>
<th></th>
<th>$P_7$</th>
<th>$P_8$</th>
<th>$P_9$</th>
<th>$P_{10}$</th>
<th>$P_{11}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_A = \frac{1}{1-\tau_A}$</td>
<td>19.2189</td>
<td>6.1109</td>
<td>7.6600</td>
<td>1.5744</td>
<td>8.7514</td>
</tr>
<tr>
<td>$\delta_B = \frac{1}{1-\tau_B}$</td>
<td>11.1870</td>
<td>9.1275</td>
<td>14.3281</td>
<td>1.9060</td>
<td>6.0832</td>
</tr>
<tr>
<td>$\delta_{AB} = \frac{1}{1-\tau_{AB}}$</td>
<td>14.2667</td>
<td>10.7431</td>
<td>15.2940</td>
<td>1.9138</td>
<td>5.0946</td>
</tr>
</tbody>
</table>

Table 5.2.2: The characteristic diffusion time measures for the nodes $P_7, \ldots, P_{11}$. $\delta_A$ measures the diffusion time for data points that were used in the training phase, $P_7^A, \ldots, P_{11}^A$. $\delta_B$ measures the diffusion time of the new captured data $P_7^B, \ldots, P_{11}^B$. $\delta_{AB}$ measures the diffusion time of the merged sets $P_{7}^{AB}, \ldots, P_{11}^{AB}$.

<table>
<thead>
<tr>
<th></th>
<th>$P_{12}$</th>
<th>$P_{13}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_A = \frac{1}{1-\tau_A}$</td>
<td>9.3920</td>
<td>16.0633</td>
</tr>
<tr>
<td>$\delta_B = \frac{1}{1-\tau_B}$</td>
<td>9.1977</td>
<td>2.3344</td>
</tr>
<tr>
<td>$\delta_{AB} = \frac{1}{1-\tau_{AB}}$</td>
<td>7.4924</td>
<td>4.2980</td>
</tr>
</tbody>
</table>

Table 5.2.3: The characteristic diffusion time measures for the nodes $P_{12}, P_{13}$. $\delta_A$ measures the diffusion time of $P_{12}^A, P_{13}^A$, which are the points that were used for $P_{12}$ and $P_{13}$ in the training phase. $\delta_B$ measures the diffusion time of the new values that were captured, denoted as $P_{12}^B, P_{13}^B$. $\delta_{AB}$ measures the diffusion times of the sets $P_{12}^{AB}, P_{13}^{AB}$.

After updating the bottom-level nodes, Algorithm 3 is applied to the middle and top-level nodes of the hierarchical tree. The process checks if the existing dataset, which embeds the groups $G_1^A, G_2^A$ and $G_3^A$, has to be updated. The embedding of the groups’ training datasets are merged with the embeddings of the groups’ newly collected data, as was described in Section 5.2.1. The merged graphs of the middle-level nodes are seen in Figs. 5.2.1, 5.2.9 and 5.2.11. The values of the characteristic diffusion time of each group are given in Table 5.2.4. We see that the training dataset of the group $G_1$ does not have to be updated if the threshold is 2. The different values of $\delta$ for second group $G_2$, imply that this group should be updated. The new behavior, which was seen in parameter $P_{13}$ (Figure 5.2.5), reflects the group $G_3$. Although the structure of the new embedding manifold is quite the same for the datasets $G_3^A$ and $G_3^B$ (see Fig. 5.2.10), the points are distributed differently in the joint embedding in Fig. 5.2.11 and the training dataset has to be updated.
5.3. DISCUSSIONS

Last, the top node of the hierarchial tree is checked. The general structure of the manifolds $V^A$ and $V^B$ is similar. Even if the middle level nodes were updated, the top node shows that the system behavior did not change much. The $\delta$ measures in the last column of Table reflect this.

<table>
<thead>
<tr>
<th></th>
<th>$G_1$</th>
<th>$G_2$</th>
<th>$G_3$</th>
<th>Super-graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_A$</td>
<td>$\frac{1}{1-\lambda_2^G}$</td>
<td>19.8798</td>
<td>4.0429</td>
<td>14.3480</td>
</tr>
<tr>
<td>$\delta_B$</td>
<td>$\frac{1}{1-\lambda_2^B}$</td>
<td>11.5664</td>
<td>5.7582</td>
<td>13.4190</td>
</tr>
<tr>
<td>$\delta_{AB}$</td>
<td>$\frac{1}{1-\lambda_2^{G+B}}$</td>
<td>9.6618</td>
<td>2.3042</td>
<td>4.9755</td>
</tr>
</tbody>
</table>

Table 5.2.4: The characteristic diffusion time measures for the nodes $G_1, G_2, G_3$ and the super-graph. $\delta_A$ measures the diffusion time of the points that were used for $G_1^A, G_2^A, G_3^A$ and for the super-graph in the training phase, denoted as $S^A$. $\delta_B$ measures the diffusion time of the new values that were captured from the parameters $G_1^B, G_2^B, G_3^B$ and the new constructed super-graph, denoted as $S^B$. $\delta_{AB}$ measures the diffusion time of the joint set.

5.3 Discussions

This section used a graph alignment approach to merge the low-dimensional representations of two high-dimensional datasets that were captured by the same sensor. In this application, the low-dimensional embeddings were described by two or three coordinates. In Algorithm 3, the parameter $q$ was set to 10 such that the first $q = 10$ diffusion coordinates were used to merge and align each pair of graphs. In Algorithm 4, the size of the landmark group $C$ was taken to be half of the size of $P_A$. The size of the group $C$ can be smaller, but as it is reduced, we still have to assure that the landmark points are well spread along the embeddings to be merged.

Algorithm 5 discusses how to compare between two graph structures. Three characteristic diffusion times that originated from three groups of points of equal size, which lie in the same space, are calculated. A threshold is set to determine if one data point set is distributed differently from the other. The value of this threshold should not be too close to one since this will require updating all or most of the dataset. On the other hand, setting the threshold to a number, which is much larger than 1, may badly affect the online exten-
5.4. SUMMARY

This chapter describes a method to compare between two given datasets. In the context of dynamically evolving data, the method is used to check whether the training dataset should be updated. Behavioral profile from the training has to be updated if it does not represent well anymore the regular activities of the system. The comparison is done in a low-dimensional space to where both datasets were embedded. Once the dataset was embedded into a lower dimension space, quantitative measures are calculated. The diffusion maps is applied to each embedded set and the characteristic diffusion time of the set is calculated. The characteristic diffusion time is a constant and it is calculated by the second eigenvalue of the diffusion maps and it equals to $\frac{1}{1-\lambda_2}$. The characteristic diffusion time is also calculated for a sampled dataset from the low-dimensional embedding space that contains data points from the two datasets. Given the three diffusion based characteristic quantities, a decision is made regarding the similarity between these two datasets.

The introduced algorithm is needed for the learning and for the detecting phases as was described in Chapter 4. Its application is demonstrated on the same dataset that was used in Chapter 4, which collects data from a performance monitor that records the activities in a transaction-based system. Although the method is demonstrated on dynamic data, it can be used as a general method for comparing between two given datasets of the same type.
Part III

Dimensionality reduction for detection of moving vehicles
Chapter 6

Introduction

6.1 Background and Motivation

Automatic acoustic-based vehicle detection is a common task in security and surveillance systems. Usually, a recording device is placed in a designated area and a hardware/software system processes the sounds that are intercepted by this recording device to identify vehicles only as they pass by. Moving vehicles produce typical sounds that are mainly influenced by their engine vibrations and the friction between the tires and the road. Airplanes, helicopters, wind, steps and speech create sounds that have different acoustic features when compared to vehicles. Similar vehicles types produce similar sounds, however, it is not a trivial task to identify similar vehicles that travel in diverse speeds, in various distances from the recording device and on different types of roads (land, asphalt, etc). The goal is to separate between vehicles and non-vehicles sounds by analyzing their dynamic acoustic sounds.

The proposed algorithms introduce several methods for feature extraction. They all construct some dynamic representation of the features before using a classifier. The dynamic representation is achieved by looking at overlapped data sequences, each sequence captures a few seconds from the signal. These capture the underlying dynamic structures that differentiate between vehicles and non-vehicles. Three different algorithms for automatic detection of vehicles are proposed. The first scheme (Algorithm I in Chapter 7) uses
dimensionality reduction methodologies such as random projections instead of using tradi-
tional signal processing methods to extract features. It uncovers characteristic features
of the recorded sounds without any assumptions about the structure of the signal. The
set of features is classified by the application of PCA. The second algorithm (Algorithm
II, Chapter 8) combines a construction of a training database of acoustic signatures sig-
nals emitted by vehicles using the distribution of energies among blocks of wavelet packet
coefficients with a procedure of random search for a near-optimal footprint (RSNOFP).
The third algorithm (Algorithm III, Chapter 9) uses the wavelet-packet transform in order
to extract spatio-temporal characteristic features from the recordings where the underly-
ing assumption is that these features constitute a unique acoustic signature for each of the
recordings. The feature extraction procedure is followed by the application of the diffusion
maps (DM) to achieve dimensionality reduction algorithm which further reduces the size
of the signature. A new recording is classified by employing the wavelet-packet feature
extraction and embedding the result in the reduced-dimension space via the application of
the geometric harmonics (GH) algorithm which is the out-of-sample extension algorithm
that complements the DM algorithm.

The introduced algorithms are generic and can be applied to various signal types for
solving different detection and classification problems.

6.2 Related work

Several papers have dealt with the problem of separating between vehicle and non-vehicle
sounds. Most of them describe systems in a military context.

Extraction of acoustic features by using the discrete wavelet transform is described in
[19]. The feature vectors were compared to reference vectors in a database using statistical
pattern matching to determine vehicle type from which the signal originated. The discrete
cosine transform was applied in [35] to signals and a time-varying autoregressive model-
ing approach was used for their analysis. A system, which is based on wavelet packets
coefficients in order to discriminate between different vehicles types, is described in [2].
Classification and regression trees were used for the classification of new unknown signals.
In a later paper [3], the same authors used similar methods with a multi-scale local cosine
transform applied to the frequency domain of the acoustic signal. The classifier was based on the “parallel coordinates” [44, 45] methodology. The “eigenfaces method” [70], which was originally used for human face recognition, to distinguish between different vehicle sound signatures, was used in [76]. The data was decomposed into a series of short-time frames. Then, each frame is transformed into the frequency domain. Classification is done by projecting the new frames on the principal components that were calculated for a known training set. Comparison between several speech recognition techniques for classification of vehicle types was presented in [55]. These methods were applied to short-time Fourier transform of the vehicles’ acoustic signatures. Different types of moving vehicles in a wireless environment, which includes acoustic and seismic sensors, were classified in [34]. Each sensor extracted features by the application of the FFT. The averaged low frequencies values are saved. A local classifier like \( K \)-nearest neighbors, maximum likelihood or SVM classified the signal at each sensor. Then, a global fusion process classifies the final signal. A remote netted acoustic detection system for detection and classification of targets at tactical ranges was described in [71]. Hamming window and FFT were applied to windows at each sensor. Uniformly spaced beams were formed and frequency peaks were marked. The signal was classified according to harmonic lines that were generated from the frequency peaks. Multiple hypothesis tracking and Kalman filter algorithms were used for real time target tracking.

Due to the fact that these applications have military contexts, most of the publications are classified and there is no universal benchmark dataset that is commonly used. Therefore, comparisons among different works is difficult. The datasets, which were used in different papers, were taken at different sample rates and the experimental conditions were not alike. In several papers, the settings are different, the vehicles are classified by using an array of sensors rather than a single one. In this case, the classification task is easier.

### 6.3 Contributions and novelty

This chapter introduces three algorithms that operate on real recorded acoustic data. This contains a large variety of scenarios and conditions. In addition, the sample rate of the
given dataset is typically lower than what is used in most of the related works that are cited in Section 6.2.

This part proposes new ways to define and classify acoustic signatures, which perform well on low-resolution data. This is its main contribution. The first algorithm, which is introduced in Chapter 7, performs well. It is based on Fourier transform, random projections and PCA. In addition, its computational complexity implementation is very low in terms of CPU time and storage space. Chapter 8 uses the wavelet packets as a basis for generating a characteristic acoustic signature. Although wavelet packets have widely been used in many fields over the years, their use for vehicle detection in these applications has not been explored much. Its advantage is the ability to separate between different frequency classes that characterize different parts of the vehicle’s acoustic. The algorithm in Chapter 9 takes a different approach. It uses the geometric structure of the data that was revealed by the application of dimensionality reduction methods. The acoustic signature is generated from the application of wavelet packet. The classification is based on the way the data is organized by the diffusion maps.

The proposed methods contribute to the field of acoustic detection and classification. They overcome several difficulties and therefore they are generic and fit wide range of applications.
Chapter 7

Algorithm I: Detection of moving vehicles via dimensionality reduction

Every sound emitting device can be characterized according to the acoustic features of the sounds it produces. These characteristic features are referred to as acoustic signatures and are used to differentiate between vehicles and non-vehicles. Traditionally, these signatures are analyzed by signal processing. This scheme [62] uses ideas that come from compressed sensing [31, 30, 18] to uncover dominating features of an unknown acoustic signal. The short-term dynamics of the acoustic signal is treated as a point \( x \in \mathbb{R}^m \). It is correlated with approximately \( \log N \) random vectors in \( \mathbb{R}^m \), were \( N \) is the total number of points. The outcome of this process is a set of features that is further processed to obtain the acoustic signature.

The algorithm contains two phases: an offline training phase and an online detection phase. In the training phase, the data, which consists of vehicle and non-vehicle recordings, is analyzed and features that characterize it are extracted to produce acoustic signatures. These signatures are used during the online detection phase, which processes a new acoustic signal in order to determine whether or not it is a vehicle.

The learning phase analyzes a test sample set of recordings \( TSS = \{ s_i \}_{i=1}^\tau \) whose classifications are known a-priori, where \( s_i \) is a recording of length \( |s_i| \) and \( \tau \) is the number of signals in the training set. The signals do not necessarily have the same size. Each signal
7.1 Dimensionality reduction via random projections

$s_i$ is decomposed into overlapping segments $W_i = \{w_i^j\}$ that are referred to as windows. A window size is $l = 2^r, r, l \in \mathbb{N}$. The windows are grouped into a single set $\Omega = \bigcup_{i=1}^r W_i$. For notational convenience, a single index is used in $w_i^j$ and the output is denoted by $\Omega = \{w_j\}_{j=1}^{n_w}$ where the total number of windows resulting from the decomposition of all the signals is $n_w \triangleq |\Omega|$.

Following the decomposition, features are extracted from every window by the application of random projections. The classification phase does not process the entire signal in a batch mode but a short segment at a time. This fact along with the high efficiency of the algorithm render this classification to be suitable for real-time applications.

7.1 Dimensionality reduction via random projections

The applicability of dimensionality reduction via random projections was proved in [46]. Specifically, it was shown that $N$ points in $N$ dimensional space can almost always be projected into a space of dimension $C \log N$ where the ratio between distances and error (distortion) is controlled. Bourgain [12] showed later that any metric space with $N$ points can be embedded by a bi-Lipschitz map into an Euclidean space of $\log N$ dimension with a bi-Lipschitz constant of $\log N$. Various randomized versions of this theorem are used for protein mapping [52] and for the reconstruction of frequency-sparse signals [30, 31, 18]. Random projection plays a role in a machine learning application. It can replace traditional feature extraction methods to go from high dimensional to low dimensional space. In this case, the projected data becomes the feature space and these features are classified by some classification algorithm. Manifolds construction for learning, which is based on the random projections, is given in [38]. Random projections were used in [77] to extract features from face images. In addition, random projection can be added as a dimensionality reduction step to algorithms that select features in different ways. For example, noisy speech signals in [37] were classified and random projections were used as a tool to reduce the data dimension to get faster computational results.

It is assumed that the acoustic data signals have some sparse representation in an orthonormal basis such as wavelets or Fourier. The goal is to find the most important coefficients, which contain information that will discriminate between the input classes. In
order to reduce the dimensionality of a dataset \( \Gamma = \{x_1, x_2, \ldots, x_n\} \), where \( x_i \) is a row vector of length \( m \), by using random projections, a random matrix is generated. Denote it by \( \Upsilon = (\rho_{ij}) \), where \( q \) is the dimension of the target reduced space. Two common choices for generating a random matrix are:

1. The columns of \( \Upsilon \) are uniformly distributed on the \( q \) dimensional unit sphere.

2. The column elements of \( \Upsilon \) are chosen from a Bernoulli +1/-1 distribution and the columns are \( l_2 \) normalized to have length 1.

The embedding \( \bar{x}_i \) of \( x_i \) into a low dimensional space is obtained by

\[
\bar{x}_i \triangleq (\Upsilon \cdot x_i^T)^T, \quad i = 1, \ldots, n
\]

where \( T \) denotes the transpose of a vector/matrix and \( \cdot \) is an inner product.

Following the random projection stage, the classifier concatenates every \( \mu \) consecutive window projections and further reduces the dimensionality by applying principal component analysis (PCA) to this concatenation. PCA, which is common way for dimensionality reduction of high dimensional data, projects the data onto the direction where the variance of the data is maximal. The classification is done in the dimension-reduced space. Thus, two dimensionality reduction steps are applied to the data. This assures a better compaction of the data than if a single dimensionality reduction technique had been used.

### 7.2 The learning phase

The learning phase uses random projections in order to extract features from every input window \( w_j \in \Omega \). The dynamics of the dataset is captured by looking at paths of consecutive windows. Algorithm 6 outlines the main steps in the learning phase.
7.2. THE LEARNING PHASE

Algorithm 6 Learning acoustic features using random projections

1. Every signal is decomposed into overlapping windows.

2. The windows are transformed into the frequency domain via the application of the Fourier transform.

3. Dimensionality reduction via random projections: each window is projected onto a given number \( n_{RM} \) of random generated bases.

4. Paths from the random projections are constructed. A path contains the random projections of \( \mu \) consecutive windows where \( \mu \) is a given parameter.

5. Dimensionality reduction of the paths via the application of principal component analysis (PCA).

In the following, we describe Steps 2-5 in details.

**Step 2:** Following the assumption that the acoustic signature of a signal is better seen in its frequency domain, the fast Fourier transform (FFT) is applied to each window \( w_j \) from Step 1. The magnitudes of the frequencies are saved. Furthermore, the dynamic range is reduced by taking the logarithm of the magnitudes (a small constant is added to avoid taking the logarithm of zero-magnitude frequencies). The output of this step is denoted by \( U \triangleq \{ u_j \}_{j=1}^{n_w}, u_j \in \mathbb{R}^h, h = \frac{l}{2} \), where \( l \) is the window size. For the experimental results that are presented in Section 7.4 the TSS was sampled at 1000 Sps and the application of the FFT improved the results. When the TSS is given in a better resolution, this step can be skipped and the random projections can be applied directly to the windows.

**Step 3:** A number of random matrices \( RM = \{ \Upsilon^i \}_{i=1}^{n_{RM}} \) are generated where \( \Upsilon^i \) is the \( i^{th} \) matrix of size \( r \times h \). The dimension of the set \( U \) is reduced by projecting it using every matrix in \( RM \), as described in Eq. 7.1.1. Every projection using \( \Upsilon^i \) produces a single embedding into a dimension-reduced space. The random projection of \( U \) onto a random basis \( \Upsilon^i \) is denoted by \( \tilde{U}^i \triangleq \{ \tilde{u}^i_j \}_{j=1}^{n_w} \) where \( \tilde{u}^i_j \in \mathbb{R}^r \). Each projection \( \tilde{u}^i_j \) describes the acoustic signature of window \( w_i \). A single projection is referred to as a dimension-reduced-window (DRW) and the set of all random projections on \( RM \)
7.2. THE LEARNING PHASE

is denoted by $\widetilde{U} = \left\{ \widetilde{U}^i \right\}_{i=1}^{n_{RM}}$.

**Step 4:** Given a random projection $\widetilde{U}^i = \left\{ \widetilde{u}^i_j \right\}_{j=1}^{n_w}$, all sequences of $\mu$ consecutive DRWs are constructed. These sequences are referred to as paths. A path captures the short-term dynamics of a signal at a specific time. Furthermore, a path is more robust to local noise (such as a wind gust) than a single window since the duration of the dynamics it captures is longer than that of a single window. This construction is done separately for each subset of DRWs according to the original signal classification. Specifically, every vector $\widetilde{u}^i_j$ is labeled according to the class of its corresponding signal $s_k$. As mentioned above, the classifications of the signals, which are analyzed during the learning phase, are known a-priori, so a label is associated with each signal. $\widetilde{U}^i$ is separated according to the labels of the DRWs and the paths are constructed in each set by concatenating $\mu$ sequential DRWs. Denote the paths constructed from the DRW of all the signals, which were obtained by the random matrix $\Upsilon^i$, by $P^i \triangleq \left\{ \tilde{p}^i_j \right\}_{j=1}^{n_w-\mu+1}$, where $\tilde{p}^i_j \in \mathbb{R}^{r^i\mu}$. The output of this step is the set $P = \left\{ P^i \right\}_{i=1}^{n_{RM}}$ that consists of $n_{RM}$ learning-sets that contain the short-term dynamic paths of the acoustic signatures. These sets are organized according to the classification (labels) of the paths.

**Step 5:** Let $P^i$ be the paths constructed from the DRW of all the signals via the random matrix $\Upsilon^i$. The paths in $P^i$ are shifted to be centered around the origin. Principal component analysis (PCA) is applied to the set $P^i$. The projection of the dataset $P^i$ onto the first $k$ principal components yields $Q^i \triangleq \left\{ \tilde{q}^i_j \right\}_{j=1}^{n_w-\mu+1}$, where $\tilde{q}^i_j \in \mathbb{R}^k$. This step is performed for every set of paths $P^i$, $i = 1, \ldots, n_{RM}$, that was produced in Step 3. Thus, $n_{RM}$ low-dimensional learning-sets, $Q = \left\{ Q^i \right\}_{i=1}^{n_{RM}}$, are created by projecting the paths, which were created in Step 3, onto the PCA bases that were constructed in this step.

The flow of the learning algorithm is presented in Fig. 7.2.1.
7.3 The classification phase

The classification phase is performed online. There is no need to wait for the entire signal to be received. In order to classify the signal at time $t$, the algorithm only needs the path that ends at time $t$, i.e., the $\mu$ consecutive overlapping windows of size $l$ that immediately preceded time $t$. The values of $\mu$ and $l$ are the same as those used in the learning phase.

Let $\sigma_t = \langle \sigma(t-\nu+1), \sigma(t-\nu+2), \ldots, \sigma(t) \rangle$ be the sequence of $\nu$ signal values that were received up to time $t$ where $\sigma(x)$ is the signal’s value that was captured at time $x$. $\sigma_t$ is decomposed into $\mu$ overlapping windows $\{\omega_j\}_{j=1}^\mu$ of size $l$. In order to classify $\{\omega_j\}_{j=1}^\mu$, an algorithm, which is similar to Algorithm 6 in Section 7.2, is employed. Algorithm 7
outlines the steps for classifying $\sigma_t$.

**Algorithm 7** Classification of acoustic features using random projections

1. Application of FFT.
2. Application of dimensionality reduction using random projections via the matrices that were generated in Step 3 in Algorithm 6.
3. Construction of a path from the output of Step 2.
4. Application of dimensionality reduction using the principal components that were calculated in Step 5 of Algorithm 6.
5. Classification of the new sample according to its nearest neighbor in the reduced space produced by the PCA.

Here is a detailed description of each step.

**Step 1:** The FFT is applied to each window in $\{\omega_j\}_{j=1}^\mu$. As in Step 2 of Algorithm 6, the logarithm magnitudes of the frequencies are saved and the result is denoted by $\{\upsilon_j\}_{j=1}^\mu$.

**Step 2:** The dimensionality of $\{\upsilon_j\}_{j=1}^\mu$ is reduced by random projecting it using all the random matrices $RM = \{\Upsilon^i\}_{i=1}^{nRM}$ that were generated in Step 3 of Algorithm 6. The projection via a single matrix $\Upsilon^i$ produces a set of dimension reduced vectors $\{\tilde{\upsilon}^i_j\}_{j=1}^\mu$.

**Step 3:** For each single matrix $\Upsilon^i$, the vectors $\{\tilde{\upsilon}^i_j\}_{j=1}^\mu$ are concatenated into a path $\phi^i$. Thus, the output of this step is a set of $nRM$ paths $\{\phi^i\}_{i=1}^{nRM}$.

**Step 4:** The set of paths $\{\phi^i\}_{i=1}^{nRM}$ is projected on the first $k$ principal components that were calculated in Step 5 of Algorithm 6. These embeddings are denoted by $\{\psi^i\}_{i=1}^{nRM}$.

**Step 5:** Let $Q^i$ be a low dimensional learning set that was generated in Algorithm 6 by using the random matrix $\Upsilon^i$ and let $\varphi^i$ be the new embedded signal that was produced by using the same random matrix $\Upsilon^i$. The $\delta$ nearest neighbors of $\varphi^i$ from the set $Q^i$ are found and their labels are saved. The classification of the new arrived
signal is determined according to the label with the highest number of occurrences within the group of nearest neighbors that are gathered from the entire learning set \( Q = \{Q^i\}_{i=1}^{n_B} \).

7.4 Experimental results

Forty two recordings were used in the learning phase: 21 of them were recordings of vehicles that include cars, trucks and vans and the rest contains non-vehicle recordings such as airplanes, helicopters, silence, speech, wind, steps, etc. The recordings were sampled at 2000Sps. The following parameters were used for the learning and classification phases: the window size, \( l \), was set to 1024 and the overlap between consecutive windows was 50\%. The windows size after the Fourier transform is 512. The number of random matrices, which were generated in Algorithm 6 Step 3, was \( RM = 3 \). The number of random vectors in each random matrix was \( r = 30 \). By using Johnson Lindenstrauss lemma [46] for random projections, the dimension is reduced to \( O(\frac{\ln(512)}{\epsilon^2}) \), \( 0 < \epsilon < 1 \). Setting \( \epsilon \) to 0.5 implies that the dimension can be reduced to approximately 30. Figure 7.4.1 shows how the embedding of the training set changes according to different values of \( r \). Raising \( r \) from 5 to 15 and then to 30 improves the separation. Raising it beyond 30, to 45 and 60 does not improve much. This implies about the dimension of the feature space.
7.4. EXPERIMENTAL RESULTS

Figure 7.4.1: The embedding of the training set via the first two PCA coordinates, after random projecting the dataset to different number of random matrices, $r$. The red points embed windows that belong to vehicle recordings and the green points embed windows that belong to background recordings. It can be seen that the separation improved as $r$ is raised from 5 to 30, and stays quite stable for $r = 30, 45$ and 60.

To capture the dynamics in a certain time frame, one can choose to set a larger window size $l$ and a shorter path length $\mu$, or construct a longer path from shorter windows. In this application, the dynamics of a few time seconds was constructed by setting the windows size $l$ to 1024. This way, the frequency domain of each window is well captured and the path length $\mu$ was set to 5. The number of principal components used in Step 5 in Algorithm 6 was $k = 5$. Figure 7.4.2 shows the spectrum of the PCA eigenvalues. It can be seen that the first two PCA coordinates are the most important, but setting $k = 5$ improved the results. This parameters was determined empirically after testing various values of $k$ on a set of vehicles recordings that were taken in convenient environmental conditions.
A new data point, which is embedded online by Algorithm 7 into principal components, is classified according to its $\delta = 7$ nearest neighbors. This parameter was determined empirically. The “car probability” of new data points is defined as

$$\frac{\# \text{ cars in data point’s } \delta \text{ nearest neighbors}}{\delta}.$$ 

It was tested on a set of vehicles that were easy to detect. Figure 7.4.3 shows the classification of a car and a truck that pass by at $t = 40$ and $t = 50$ seconds with three different values of $\delta$. 
7.4. EXPERIMENTAL RESULTS

Figure 7.4.3: Classification of a car and a truck that pass by at $t = 40$ and $t = 50$ seconds with three different values of $\delta$. It can be seen that when $\delta$ is set to be 5 in (a), the vehicle classification percentage is smaller than 1 at all times. On the other hand, setting $\delta$ to be 11 in (c) generates a false-positive classification at the beginning of the recording. Setting $\delta$ to 7 (in (b)) yields the best results for this recording and for a larger set of simple test recordings.

The classification phase was tested on recordings that were taken in various road and wind conditions. Fifty new recordings were classified on line. Most of the recordings contained at least one vehicle and background noises. In more than half of the recordings, the recording device was located approximately 10 meters from a paved road and 10 from a dirt road. In some of the remaining, the recording device was located about 50 meters from a paved road and in a few recordings the recording device was located 100 meters from a sand road. The correct classification percentage was 85. Ten percent of the misclassifications were false negatives, the majority of these happened in the recordings that
took place on the sand road. The false positive occurred when load noises like speech or wind were heard close to the recording device. These were seen as short positive spikes. These false positives can be minimized by defining that a vehicle is detected only if the detection is positive for a period of time. The algorithm behavior was also validated by taking the TSS to be the test set. All of the recordings in the training set were classified correctly.

For every recording, which was used in the classification phase, we show: (a) A plot of the original signal; (b) a graph showing the vehicle classification probability that the online classification Algorithm 7 produced. A vehicle is classified as detected if the probability is above 50% for $M$ seconds. The value of $M$ may change according to the distance of the recording device from the road. In these experiments, $M$ was set to 5 seconds.

The recordings in Figs. 7.4.4 - 7.4.9 were not part of the TSS that was used in the training phase. The classifier successfully filters out background noises.
Figure 7.4.4: (a): Recordings that contain sounds emitted by a car at $t = 130$ sec. The recording device was located 10 meters from the road. (b) Vehicle detection probability. The vehicle was classified correctly.
Figure 7.4.5: (a): Recordings that contain sounds emitted by a plane, which are heard strongest at the beginning of the recording. Speech sounds are heard as well. A truck passes on the road between $t = 20$ and $t = 35$ sec. The recording device was located 50 meters from an asphalt road. (b) Vehicle detection probability. The truck was classified correctly.
7.4. EXPERIMENTAL RESULTS

Figure 7.4.6: (a): Recordings that contain sounds emitted by a truck driving in first gear between $t = 100$ and $t = 140$ seconds. Sounds emitted by birds are heard and there were strong winds towards the end. The recording device was located 50 meters from the a sand road. (b) Vehicle detection probability. The vehicle was classified correctly, the classification percentage drops at $t = 105$ when the truck was relatively close and the recording device was saturated.
7.4. EXPERIMENTAL RESULTS

Figure 7.4.7: (a): Recordings that contain sounds emitted by a truck driving in second gear between \( t = 50 \) and \( t = 90 \) seconds. Sounds of wind and birds were present throughout the recording. The recording device was located 50 meters from the a sand road. (b) Vehicle detection probability. The truck was classified correctly.
Figure 7.4.8: (a): This recording contain sounds of speech that was taking place close to the recording device. (b) Vehicle detection probability. There is a short positive spike at \( t = 12 \), but since \( M \) was defined to be 5 seconds, this is not classified as a vehicle.
7.5 CONCLUSIONS

Figure 7.4.9: An example of a false detection. (a): Recordings that contain strong sounds of wind and a jeep that drove on a sand road between $t = 480$ and $t = 520$. The recording device was located 100 meters from the sand road. (b) Vehicle detection probability. The vehicle was missed, all of the detection values are below 50 percent.

The experimental results showed good performance for recordings that were taken in environmental conditions that were similar to most of the training set.

7.5 Conclusions

We presented a two-phase algorithm that detects vehicles according to their acoustic characterizations. Every acoustic signal was decomposed into overlapping windows and dominating features were extracted from each window by using random projections. Short term dynamic paths were then constructed from sequences of features that were extracted from consecutive windows. In order to detect the vehicles, these paths were embedded into
7.5. CONCLUSIONS

a lower dimension space using PCA. The online classification of newly arrived acoustic signals was obtained by employing similar steps.

The results, which were presented in Section 7.4, were based on a relatively small training set. The results were satisfying despite the fact that some of the recordings, which were tested in the online detection algorithm, were recorded in environmental conditions that were different from those that were present during the recording of the training set.

The experimental results indicate that the accuracy of the classification is affected by a number of factors:

*The size of the training set:* Using a larger number of recordings during the learning phase provides a more reliable training set which results in a more accurate detection.

*Coverage of the test sample set:* In order to detect vehicles that run on dirt roads, the training set should include a number of recordings of this nature. In addition, the training set should include a large variety of typical background noises. Otherwise discrimination between background noises, which are not included in the training set, and vehicles is not guaranteed.

*The sample rate of the recordings:* The proposed algorithm was tested on acoustic signals that were sampled at 1000Sps. The algorithm produced similar results for the same data at sample rates of 500Sps and 2000Sps. However, using sample rates of 6000SPS and above improves the results since the extracted features are more accurate.
Chapter 8

Algorithm II: Wavelet based acoustic detection of moving vehicles

The second algorithm [7] proposes robust method to detect the arrival of a vehicle of arbitrary type when other noises are present. It is done via analysis of its acoustic signature against an existing database of recorded and processed acoustic signals to detect the arrival of a vehicle of arbitrary type when other noises are present. To achieve it with minimum number of false alarms, the training database is constructed by using the energy distribution of the acoustic signatures signals among blocks of wavelet packet coefficients with a procedure of random search for a near-optimal footprint (RSNOFP). The number of false alarms in the detection is minimized even under severe conditions such as: the signals emitted by vehicles of different types differ from each other, whereas the set of non-vehicle recordings (the training database) contains signals emitted by planes, helicopters, wind, speech, steps etc.

A successful detection depends on the constructed acoustics signatures that were built from characteristic features. These signatures enable us to discriminate between vehicle (V) and non-vehicle (N) classes. Acoustics signals emitted by vehicles have quasi-periodic structure. It stems from the fact that each part of the vehicle emits a distinct acoustic signal which contains in the frequency domain only a few dominating bands. As the vehicle moves, the conditions are changed and the configuration of these bands may vary, but the
8.1. THE STRUCTURE OF THE ACOUSTICS SIGNALS

general disposition remains. Therefore, we assume that the acoustic signature for the class of signals emitted by a certain vehicle is obtained as a combination of the inherent energies in the blocks of the wavelet packet coefficients of the signals, each of which is related to a certain frequency band. This assumption has been corroborated in the detection and identification of a certain type of vehicles ([5, 4]). The experiments presented here demonstrate that a choice of distinctive characteristic features, which discriminate between vehicles and non-vehicle classes, can be derived from blocks of wavelet packet coefficients. Extraction of characteristic features (parameters) is a critical task in the training phase of the process. The wavelet packet transform preserves temporal locality, which is an advantage over Fourier analysis. Discontinuities in the signal, which may occur when using low-resolution sampled data, cannot be processed by Fourier analysis to extract meaningful information.

In order to identify the acoustic signatures, in the final phase of the process the outputs from two classifiers is combined. One is the well known Classification and Regression Trees classifier [13]. The other classifier is based on the comparison of distances between the test signal and sets of pattern signals from the V and N classes.

8.1 The structure of the acoustics signals

The recordings were taken under very different conditions in different dates. The recordings sampling rate (SR) was 48000 samples per second (SPS). It was down-sampled to SR of 1000 SPS and 600 SPS.

Data fragments were extracted from the set of recordings that were used for training the algorithm. The fragments that contain sounds emitted by vehicles were stored as the V-class signals. Recorded fragments that did not contain vehicles sounds were stored as the N-class signals. Both classes were highly variable. Recordings in the V-class were taken from different types of vehicles during different field experiments under various surrounding conditions. In particular, the velocities of the vehicles and their distances from the recording device were varied. Moreover, the vehicles traveled on either various paved (asphalt) or unpaved roads, or on a mixture of paved and unpaved roads. Recordings in N-class comprised of sounds emitted by planes, helicopters, sometimes strong wind and
speech nearby the receiver, to name a few.

Figures 8.1.1 - 8.1.4 display the acoustic samples from the V-class and the N-class. These samples include the original recording in the time domain in seconds on the left and the associating Fourier transform, in frequencies, on the right.

Figure 8.1.1 displays portions of acoustic signals emitted by two cars with their Fourier transforms.

Figure 8.1.1: Fragments of two car recordings and their spectra. Frames from left to right: First car; its spectrum; second car; its spectrum.

Figure 8.1.2 displays portions of acoustic signals emitted by a truck and a van with their Fourier transforms.

Figure 8.1.2: Fragments of a truck and a van recordings and their spectra. Frames from left to right: Truck; its spectrum; van; its spectrum.

The spectra of different cars differs from each other. It is even more apparent in the spectra of other vehicles. Figure 8.1.3 displays portions of acoustic signals emitted by a
plane and a helicopter with their Fourier transforms, whereas Fig. 8.1.4 does the same for speech and wind patterns.

Figure 8.1.3: Fragments of a plane and a helicopter recordings and their spectra. Frames from left to right: Plane; its spectrum; helicopter; its spectrum.

Figure 8.1.4: Fragments of a speech and a wind recordings and their spectra. Frames from left to right: Wind; its spectrum; speech; its spectrum.

For the constructed training set, even within the same class (V or N), the signals differ significantly from each other. The same is true for their Fourier transforms. However, there are some common properties to all these acoustic signals that were recorded from moving vehicles. First, these signals are quasi-periodic in the sense that there exist some dominating frequencies in each signal. These frequencies may vary as motion conditions are changed. However, for the same vehicle, these variations are confined in narrow frequency bands. In addition, the relative locations of the frequency bands are stable (invariant) to some extent for signals that belong to the same vehicle.
Therefore, we conjectured that the distribution of the energy (or some energy-like parameters) of acoustics signals that belong to some class over different areas in the frequency domain, may provide a reliable characteristic signature for this class.

8.2 Formulation of the approach

Wavelet packet analysis (see Section 2.2) is a highly relevant tool for adaptive search for valuable frequency bands in a signal or a class of signals. Once implemented, the wavelet packet transform of a signal yields a huge (redundant) variety of different partitions of the frequency domain. The transform is computationally efficient. Due to the lack of time invariance in the multi-scale wavelet packet decomposition, we use the whole blocks of wavelet packet coefficients rather than individual coefficients and waveforms. The collection of energies in blocks of wavelet packet coefficients can be regarded as an averaged version of the Fourier spectrum of the signal, which provides more sparse and more robust representation of signals compared to the Fourier spectrum. This can be seen, for example, in Fig. 8.2.1. This figure displays the energies in the blocks of wavelet packet coefficients of the sixth level of a car acoustics signal. Here, the wavelet packet transform based on the orthogonal splines of sixth order is applied.
8.2. FORMULATION OF THE APPROACH

Figure 8.2.1: Top: Fourier spectrum of the car signal in Fig. 8.1.1. Bottom: Energies in the blocks of wavelet packet coefficients of the sixth level of the wavelet packet transform that uses the orthogonal spline wavelets of sixth order.

Variation on the best basis Algorithm [24, 74] that searches for a few blocks that mostly discriminate a certain vehicle from other vehicles and the background was used in [2, 3]. Here, this approach did not prove to be robust and efficient because of the variability in vehicles types in Class V and kinds of the background in Class N recordings. Therefore, another way to utilize the wavelet packet coefficients blocks was chosen. This method can be characterized as a random search for the near-optimal footprint (RSNOFP) of a class of signals. This is close to the compressed sensing ([30, 33, 18]) idea.

In order to enhance the robustness of the algorithm, we implement three different versions of RSNOFP that validate each other (described in Section 8.5).

The sample signals for the training phase and the online signals in the detection phase are formed by imposing a comparatively short window on each input signal followed by a shift of this window along the signal so that adjacent windows have some overlap.

8.2.1 Outline of the approach

The complete process has three sequential steps:

Training phase: The training dataset signals with known membership. These signals
are sliced into the overlapped fragments of length $L$ (typically, $L = 1024$). The fragments are subjected to the wavelet packet transform.

The wavelet packet transform bridges the gap between time domain and frequency domain representations of a signal. The coefficients from the upper levels (finer scales) correspond to basic waveforms, which are well localized in time domain but their spectra occupy wide frequency bands. The coefficients from the deeper levels (coarser scales) correspond to waveforms, which have narrow spectra but wide supports in time domain. We establish experimentally that the coefficients from the sixth level provide an optimal tradeoff between time and frequency domain representations of the 1024-samples acoustic fragments signals we are dealing with.

The blocks energies are calculated and three versions of RSNOFP are applied. As a result, each fragment is represented by three different vectors of length $l \ll L$ (typically, $l = 12$ or $l = 8$). The components of these vectors are the characteristic features of the fragments. These vectors are used as pattern data sets and also are utilized to produce three versions of classification trees.

**Identification – features extraction phase:** The newly acquired signal is sliced to overlapped fragments of length $L$. Then, the wavelet packet transform is applied followed by energies calculations of the blocks of coefficients. Then, three different transforms that are determined by three versions of RSNOFP are applied. As a result, each fragment is represented by the three different vectors of length $l$.

**Identification – decision making phase:** These vectors are submitted to the corresponding versions of the tree classifiers. In addition, the vectors are tested by a second classifier that calculates the distances of the vectors from the pattern data sets associated with V and N classes. The final decision on membership of the fragment is derived by combining the answers for all the three vectors from both classifiers.

### 8.3 Description of the algorithm and its implementation

The algorithm is centered around three basic phases:
8.3. DESCRIPTION OF THE ALGORITHM AND ITS IMPLEMENTATION

I. Extraction of characteristic features from V and N classes. It contains the following steps:

1. The analyzing wavelet filters are chosen.
2. The training sets of signals are constructed by slicing the training signals into overlapped segments.
3. The wavelet packet transform is applied to segments.
4. The energies in the blocks of the wavelet packet coefficients are calculated.
5. RSNOFP (see Section 8.5) is called. It results in the embedding of the training sets of signals into low-dimensional reference sets that contain its characteristic features.

II. Building the classification trees.

III. Identifying whether the new signal belongs to either V or N class:

1. The new signal is sliced into overlapped segments.
2. The wavelet packet transform is applied to these segments.
3. The energies in the blocks of the wavelet packet coefficients are calculated.
4. The set of blocks energies of each segment is embedded into a low-dimensional vector that contains its characteristic features.
5. The distances of the vector that contains characteristic features from the reference sets of V and N classes are calculated.
6. The vector is tested by the tree classifier.
7. Decision whether the vector belongs to either V or N class is made.

8.3.1 Implementation: extraction of characteristic features

Choice of the analyzing waveforms: By now, a broad variety of orthogonal and biorthogonal filters, which generate wavelet packets coefficients, are available ([27, 53, 6, 5]). In this algorithm, the 6-th order spline wavelet is used. This filter
8.3. DESCRIPTION OF THE ALGORITHM AND ITS IMPLEMENTATION

reduces the overlap between frequency bands associated with different decomposition blocks. At the same time, the transform with this filter provides a variety of waveforms that have a fair time-domain localization. For details see Section 2.2.

**Signal preparation for training the algorithm:** Initially, many recordings are gathered for V and N classes, which have to be separated. Then, for each selected recording, which belongs to a certain class, a number of overlapped slices are created. The slices are of length \( L = 2^J \) samples, and shifted by \( S \ll L \) samples with respect to each other. Altogether, \( M^v \) and \( M^n \) slices are created for the V and N classes, respectively. The slices are arranged to be rows of two matrices. Slices from V–signals form the matrix \( A^v = \{ A^v_{i,j} \} \), \( i = 1, \ldots, M^v \), \( j = 1, \ldots, L \) and slices from N–signals form the matrix \( A^n = \{ A^n_{i,j} \} \), \( i = 1, \ldots, M^n \), \( j = 1, \ldots, L \).

**Embedding the sets of slices into sets of energies:** First, the measure for the energy is specified. The normalized \( l_1 \) norms of the blocks are used. Then, the following operations are carried out:

1. The wavelet packet transform down to level \( m \) (typically \( m = 6 \) if \( L = 1024 \)) is applied to each slice of length \( L \) from both V and N classes. We take the coefficients from the coarsest scale \( m \). This scale contains \( L = 2^J \) coefficients that are arranged into \( 2^m \) blocks of length \( l = 2^{J-m} \), each of which is associated with a certain frequency band. These bands form a near-uniform partition of size \( 2^m \) of the Nyquist frequency domain.

2. The “energy” of each block is calculated using the chosen measure, in this case block “energy” = \( \sum_{i=1}^{l} \| i^{th} \) block coefficient \( \| \). We obtain, to some extent, the distribution of the “energies” of the slices \( A^v(i,:) \) from V–signals and \( A^n(i,:) \) from N–signals over various frequency bands of widths \( N_F/m \), where \( N_F \) is the Nyquist frequency. The “energies” are stored in the energy vectors \( \vec{E}_i^v \) and \( \vec{E}_i^n \) of length \( \lambda = 2^m = L/l \), respectively (typically, \( \lambda = 64 \)). The energy vectors are arranged to be rows of two matrices \( B^v = \{ B^v_{i,j} \} \), \( i = 1, \ldots, M^v \), \( j = 1, \ldots, \lambda \) and \( B^n = \{ B^n_{i,j} \} \), \( i = 1, \ldots, M^n \), \( j = 1, \ldots, \lambda \), where the \( i \)-th row of the matrix \( B^v \) is the vector \( \vec{E}_i^v \) and the \( i \)-th row of the matrix \( B^n \) is the vector \( \vec{E}_i^n \). The “energy” vectors \( B^v(i,:) \) and \( B^n(i,:) \) are considered to
be the averaged Fourier spectra of the slices $A^v (i,:)$ and $A^n (i,:)$, respectively, as it is seen in Fig. 8.2.1. We consider these vectors to be proxies of the slices. By the above operations the dimension of the database is reduced by factor $l = 2^{J-m}$.

**Embedding of sets of energies into the sets of features:** The subsequent operations yield a further reduction of the dimensionality in the compressed sensing [30, 33] spirit. It is achieved by the application of three versions of the RSNOFP scheme to the energy matrices $B^v$ and $B^n$. The RSNOFP scheme is described in Section 8.5. As a result, three pairs of the reference matrices: $D^v_{\text{rand}}$ & $D^n_{\text{rand}}$, $D^v_{\text{pca}}$ & $D^n_{\text{pca}}$ and $D^v_{\text{perm}}$ & $D^n_{\text{perm}}$ and the corresponding random matrices $\rho_{\text{rand}}$, $\rho_{\text{pca}}$ and $\rho_{\text{perm}}$ are constructed. Rows of these matrices represent the slices of training signals. These random features matrices will be utilized in the identification phase.

**Compaction of the feature matrices in V-class:** In order to refine the V-class of feature matrices, we test their rows. Recall that each row is associated with a segment of a signal that belongs to V-class. The Mahalanobis distances $d^v$ and $d^n$ of each row in the matrix $D^v_{\text{rand}}$ from the sets $D^v_{\text{rand}}$ and $D^n_{\text{rand}}$ is calculated. For a segment $x \in \{D^v_{\text{rand}} \cup D^n_{\text{rand}}\}$

$$d^v = (x - \mu_v)\Sigma_v^{-1}(x - \mu_v)$$  \hspace{1cm} (8.3.1)

and

$$d^n = (x - \mu_n)\Sigma_n^{-1}(x - \mu_n),$$  \hspace{1cm} (8.3.2)

where $\mu_v$ and $\mu_n$ are the mean segments in $D^v_{\text{rand}}$ and $D^n_{\text{rand}}$, respectively, and $\Sigma_v$ and $\Sigma_n$ are the covariance matrices of $D^v_{\text{rand}}$ and $D^n_{\text{rand}}$, respectively. If for some row $d^v > d^n$, then, this row is removed from the matrix $D^v_{\text{rand}}$. The same is done for the matrices $D^v_{\text{pca}}$ and $D^v_{\text{perm}}$.

**Conclusion:** As a result of the above operations, the dimensionality of the training set was substantially reduced. Ostensibly, this part of the process looks computationally expensive, especially if, for better robustness, large training sets are involved. This procedure is called once and it is done off-line before the detection phase that is done on-line. Altogether, formation of three pairs of reference matrices requires 2-3
As seen in Figure 8.5.1 in Section 8.5, the signal that of dimension 1024 was reduced to dimension 64 by the wavelet packet transform. By using random projection (the Johnson - Lindenstrauss lemma [46]), the dimension is reduced to \( O\left(\frac{\ln (64)}{\epsilon^2}\right) \), \( 0 < \epsilon < 1 \). The random projections reduced the signal to a 20 dimensions by using the RSNOFP scheme. The most valuable components from these 20 dimensions were saved to preserve most of the information.

Figure 8.3.1 displays one row from matrix \( D_{\text{perm}}^v \) and one row from matrix \( D_{\text{perm}}^n \). These are sets of features are the output from the application of the RSNOFP scheme to a segment in the V-class and to a segment in the N-class.

8.3.2 Building the classification and regression trees (CARTs)

After constructing the three pairs of reference matrices: \( D_{\text{rand}}^v \& D_{\text{rand}}^n \), \( D_{\text{pca}}^v \& D_{\text{pca}}^n \) and \( D_{\text{perm}}^v \& D_{\text{perm}}^n \), the classifiers are built. For this purpose the vectors, which consist of rows in the reference matrices are used. The construction of the tree is done by a binary split of the set of input patterns \( X \rightarrow \{X_1 \cup X_2 \cup \ldots \cup X_r\} \), so that, once a vector appeared in the subset \( X_k \), its membership could be predicted with a sufficient reliability. The answer is the class label the vector is assigned to and the probability of this assignment. The basic idea behind the split is that the data in each descendant subset is “purer” than the data in the parent subset. The scheme is described in full details in the monograph [13]. A brief outline is given in [2].
8.3.3 Identification of an acoustic signal

An acoustic signal to be identified is preprocessed by the same operations that were used for the training signals.

Preprocessing of a new acoustic recording 1. The recording is sliced to $M$ overlapped segments of length $L = 2^J$ samples each, shifted with respect to each other by $S$ samples. The wavelet packet transform up to scale $m$ is applied to each slice. We take the coefficients from the coarsest scale $m$ that are arranged into $2^m$ blocks of length $l = 2^{J-m}$. The “energy” of each block is calculated with the chosen measure. Thus, each $i$-th slice is embedded into an energy vector $\vec{E}_i$ of length $\lambda = 2^m = L/l$. The energy vectors are arranged in the matrix $B = \{B_{i,j}\}$, $i = 1, ..., M$, $j = 1, ..., \lambda$, where the $i$-th row of the matrix $B$ is the vector $\vec{E}_i$.

2. In order to embed the energy matrix $B$ into the features spaces, it is multiplied subsequently by the random matrices $\rho_{rand}$, $\rho_{pca}$ and $\rho_{perm}$. These multiplications produce three features matrices: $D_{rand}$, $D_{pca}$ and $D_{perm}$, where the $i$-th row in each matrix is associated with the $i$-th segment of the processed signal.

Identification of a single segment To identify the $i$-th segment of a signal, three vectors, $\vec{v}_{rand}^i$, $\vec{v}_{pca}^i$ and $\vec{v}_{perm}^i$ are used. These vectors form the $i$-th rows of the matrices $D_{rand}$, $D_{pca}$ and $D_{perm}$, respectively.

1. These vectors are submitted to the respective versions $T_{rand}$, $T_{pca}$ and $T_{perm}$ of the classification tree. Once a vector is submitted to the tree, it is assigned to one of the subsets $X_k$ of the input space $X$. These trees produce three decisions $\tau_{rand}$, $\tau_{pca}$ and $\tau_{perm}$ together with the corresponding probabilities $p_{rand}$, $p_{pca}$ and $p_{perm}$. The decision $\tau(\cdot)$ determines the most probable membership of the segment. The value $\tau(\cdot) = 1$ if the classification tree assigns the segment to V-class and $\tau(\cdot) = 0$ otherwise.

2. The distances (for example, Mahalanobis or Euclidean) between the vectors $\vec{v}_{rand}^i$, $\vec{v}_{pca}^i$ and $\vec{v}_{perm}^i$ and the respective pairs of the reference sets: $D_{rand}^v \& D_{rand}^n$, $D_{pca}^v \& D_{pca}^n$ and $D_{perm}^v \& D_{perm}^n$ are calculated. This cal-
calculation produces three decisions $\tilde{\tau}_{\text{rand}}$, $\tilde{\tau}_{\text{pca}}$ and $\tilde{\tau}_{\text{perm}}$ together with the corresponding quasi-probabilities (QP) $\tilde{p}_{\text{rand}}$, $\tilde{p}_{\text{pca}}$ and $\tilde{p}_{\text{perm}}$ in the following way. Let $d^v$ and $d^n$ be the distances of a vector $\vec{v}_i$ from the respective pair of the reference sets: $D^v_i$ and $D^n_i$. If $d^v < d^n$ then the decision is $\tilde{\tau}_i = 1$ (the segment is assigned to the V-class), otherwise $\tilde{\tau}_i = 0$ (the segment is assigned to the N-class). If $d^v < d^n$ then the membership QP in the V-class is defined as $\tilde{p}_i = 1 - d^v / d^n$, otherwise $\tilde{p}_i = 0$. This classification scheme is a version to the Nearest Neighbor scheme. Denote this scheme as the minimal distance (MD) classifier.

3. Two thresholds values $t$ and $\tilde{t}$ are set and the results for the $i$-th segment are combined into three 3-component column vectors $\vec{y}_i^{\text{rand}}$, $\vec{y}_i^{\text{pca}}$ and $\vec{y}_i^{\text{perm}}$, where:

$$
y_i^1(1) = \begin{cases} 
1, & \text{if } p_i > t \\
0, & \text{otherwise},
\end{cases} \quad (8.3.3)
$$

$$
y_i^1(2) = \begin{cases} 
1, & \text{if } \tilde{p}_i > \tilde{t} \\
0, & \text{otherwise},
\end{cases}
$$

$$
y_i^1(3) = \tau_i \times \tilde{\tau}_i.
$$

These thresholds are meant to control the false alarm. They are set to be between $\frac{1}{2}$ and 1. When higher thresholds are set, we are less tolerant to false alarms. For example, if $t = 0.7$, then we assign a segment to class V only if the classification probability $p_i > 0.7$.

Identification of a recording

1. The vectors $\vec{y}_i^{\text{rand}}$, $\vec{y}_i^{\text{pca}}$ and $\vec{y}_i^{\text{perm}}$ are gathered into three matrices $Y_{\text{rand}}$, $Y_{\text{pca}}$ and $Y_{\text{perm}}$ of size $3 \times M$, where $M$ is the number of overlapping segments produced from the analyzed signal. The vectors $\vec{y}_i$ serve as the $i$-th columns in the respective matrices $Y$.

2. The rows of the matrices are processed by moving average.

3. The matrices $Y_{\text{rand}}$, $Y_{\text{pca}}$ and $Y_{\text{perm}}$ are combined into the matrix $Y$ in the following way. Each entry in $Y$ is defined as the median value of the respective
8.4. EXPERIMENTAL RESULTS

entries of the three matrices:

\[ Y(i, j) = \text{median} \left( Y_{\text{rand}}(i, j), Y_{\text{pca}}(i, j), Y_{\text{perm}}(i, j) \right). \quad (8.3.4) \]

**Conclusions** The matrix \( Y \) contains the results for the analyzed signal. Its first row contains the averaged answers, which have a significant probabilities, from the classifier. The value at each point is the number of positive (class V) answers in the vicinity of this point, which is divided by the length of the vicinity. It represents the “density” of the positive answers around the corresponding segment. The structure of the second row is similar to the structure of the first row with the difference that these answers come from the MD classifier instead of answers from the classifier. The third row of the matrix \( Y \) combines the answers from both classifiers. First, these answers are multiplied by each other. The combined answer is equal to one for segments where both classifiers produce answer one (V-class) and zero otherwise. Thus, the classifiers cross-validate each other. Then, the results are processed by the application of the moving average providing the “density” for the positive answers. This row in the matrix \( Y \), yields the most robust result from the detection process with minimal false alarm.

**Real time implementation** Above, we described an identification process of a full recording. However, a slight modification of the scheme provides a real time identification of an acoustic signal. To get it, we process separately each segment upon its arrival (of course, these segments are overlapped). This segment-wise processing takes a fraction of a second. Once several initial segments are identified, we start a moving average procedure. Thus, the arriving signal is identified with a negligible time delay.

**8.4 Experimental results**

A series of experiments to detect the arrival of vehicles of arbitrary type in the presence of surrounding noises were conducted.
Altogether, 200 recordings were available taken in five different areas. Many recordings contained sounds emitted by different vehicles combined with the sounds of wind, speech, aircrafts etc. The original sampling rate (SR) was 48000 samples per second (SPS). The signals were down-sampled into SR of 1000 SPS. The motion dynamics, the distances of vehicles from the receiver, the surrounding conditions were highly diverse.

8.4.1 Detection experiments

The first task was to form the reference database of signals with known membership (training) for building the classifiers. This database was derived mainly from the recordings made on a single date by clipping the corresponding fragments. A few recordings from other dates were also used. The CAR fragments were extracted from 10 recordings, 5 recordings were used for the TRUCK fragments and the same number for the VAN fragments. The diverse non-vehicle fragments were extracted from 35 recordings. Altogether, 38 recordings were involved in the training process (most of them contained sounds from different sources). Various families of wavelet packets and various norms for the feature extraction were tested. Several combinations of features presented to the MD and classification tree classifiers were tested as well. The best results were achieved with wavelet packet transform that uses the sixth order spline filters and the $l_1$ norm.

The reference signals were separated into two groups. One group (V class) contains all signals associated with vehicles and the other group (N class) contains all the non-vehicles signals. The signals were sliced into overlapped segments of length $L = 1024$ that were shifted with respect to one another by $S = 256$, thus, the overlap was $3/4$ of a fragment. Altogether, 21898 V–segments and 43816 N–segments were obtained. The characteristic features were extracted from the segments as explained in Section 8.3.1. Each segment was expanded by the wavelet packet transform down to 6th level (scale) and the $l_1$ norm was used as the “energy” measure for all the 64 blocks of the 6th level. As a result of the procedures that were described in Section 8.3.1, we selected various sets of discriminant blocks. These procedures produced three pairs of reference matrices: $D_v^{\text{rand}} \& D_n^{\text{rand}}$, $D_v^{\text{pca}} \& D_n^{\text{pca}}$ and $D_v^{\text{perm}} \& D_n^{\text{perm}}$ with the corresponding random matrices $\rho_{\text{rand}}$, $\rho_{\text{pca}}$ and $\rho_{\text{perm}}$. Each matrix has 12 columns according to the number of the characteristic features. These matrices were used for the MD classification and also were utilized for building
three classification trees $T_{\text{rand}}, T_{\text{pca}}$ and $T_{\text{perm}}$. For the MD classification, all the available features (12) were used, unlike building the classification trees, where better results were achieved with sets containing only 8 features.

All of the 200 available recordings were employed in the detection phase. A recording number $k$ was embedded into three features matrices $D_{\text{rand}}^k$, $D_{\text{pca}}^k$ and $D_{\text{perm}}^k$, where the $i$-th row of each matrix is associated with the $i$-th segment of the recording (see Section 8.3.3). Each row was tested with the MD and classification tree classifiers. The results were gathered into the $Y^k$ matrix. The Euclidean distance was used for the MD classification. Although the signals were recorded in very different conditions and many of them were corrupted by strong background noise, the arrival of vehicles was correctly detected in overwhelming majority of cases, while false alarm was minimal.

In Figs. 8.4.1–8.4.6 we present a few typical results from the experiments on detection of vehicles of arbitrary types. All the figures are identically organized. Each comprises four frames. The top frame depicts the original recording #1. The next three frames present three rows from the $Y^k$ matrix with respect to time scale. The second from the top frame presents the combined answers (the median from three answers) from tree classifiers processed by the moving average. The third from the top frame similarly displays the results from the MD classifiers. The bottom frame illustrates the combined results from both classifiers, that is the answers from both classifiers multiplied with each other and processed by the moving average.

All the displayed recordings did not participate in the training phase.

**Recording # 1:** Figure 8.4.1 displays the results from testing recordings # 1. In the beginning of the recording, sound from a remote vehicle is heard. Then, the jumpy vehicle passed by the receiver at around 70, 134 and 200 seconds from start. In its last passage it was followed by another car. All the events were correctly detected by the classification tree and MD classifiers. The MD classifier produced some false alarms, which were eliminated by combining the classifiers (bottom frame).
8.4. EXPERIMENTAL RESULTS

Figure 8.4.1: Results of testing the recording #1. In the beginning, sound from a remote vehicle is heard. The jumpy vehicle passed by the receiver at around 70 seconds, 134 seconds and 200 seconds, in its last passage it was followed by a car.

**Recording #2:** Figure 8.4.2 presents the results from testing recordings #2. In the beginning of the recording, a truck passed by the receiver followed by a tender. At around 70 seconds from the start of the recording a car followed by a truck passed by. In the end, a minibus and a car arrived. All the events were correctly detected by the classification tree and MD classifier for each sampling rate. The MD classifier produced some false alarms, which were reduced by combining the classifiers (bottom frame).
Figure 8.4.2: Results of testing the recording # 2. In the beginning of the recording a truck passed by the receiver followed by a tender. At around 70 seconds a car followed by a truck passed by. In the end, a minibus and a car arrived. The recording did not participate in the training phase.

**Recording # 3:** Figure 8.4.3 shows the results from testing recordings # 3. Two trucks passed by the receiver moving in opposite directions at around 50 seconds from the start. Strong wind is present. The event was correctly detected by the classification tree and MD classifier for each sampling rate. The MD classifier produced some false alarms, which were reduced by combining the classifiers (bottom frame).
8.4. EXPERIMENTAL RESULTS

Figure 8.4.3: Results of testing the recording #3. Two trucks passed by the receiver in opposite directions at around 50 sec. from the start, the strong wind were present.

**Recording #4:** Figure 8.4.4 shows the results from testing recordings #4. A truck passed by the receiver from 30 seconds to 190 seconds. Then, a strong sound of a plane dominated the scene until the end of the recording. The truck was correctly detected by the classification tree and MD classifier. The MD classifier produced some false alarms, which were reduced by combining the classifiers (bottom frame). The plane was not assigned to the V class.
8.4. EXPERIMENTAL RESULTS

Figure 8.4.4: Results of testing the recording # 4. A truck passed by the receiver from 30 to 190 seconds, then the strong sound of a plane dominated the scene until the end of the recording. Strong wind is present.

**Recording # 5:** Figure 8.4.5 displays the results from testing recordings # 5. A truck followed by a minibus passed by the receiver at around 40 seconds from the start and one more truck at around 65 seconds. Strong wind was present. The vehicles were correctly detected by the classification tree and MD classifier. The MD classifier produced some false alarms, which were eliminated by combining the classifiers (bottom frame).
8.4. EXPERIMENTAL RESULTS

Figure 8.4.5: Results of testing the recording # 5. A truck followed by a minibus passed by the receiver at around 40 seconds and one more truck at around 65 seconds. Strong wind is present at the scene.

**Recording # 6:** Figure 8.4.6 presents the results from testing recordings # 6. A sound from a truck was heard within the intervals of 15 to 50 seconds and 80 to 110 seconds from the start. Then, a plane appeared. It lasted until the end of the recording. The truck was correctly detected by the classification tree and MD classifier. The plane was not assigned to V class. The MD classifier performed worse.
8.5 Random search for a near optimal footprint (RSNOFP) scheme

**RSNOFP: version I** A random matrix $R_1$ of size $\lambda \times r$ where $r \ll \lambda$ (typically, $r = 20$) is created. Entries of the matrix $R_1$ are Gaussian random variables. The rows in the matrix are normalized. The matrix $B^v$ (defined in Section 8.3.1) is multiplied by the matrix $R_1$. As a result, we obtain a new matrix $C^v = B^v \cdot R_1 = \{C^v_{i,j}\}$ of size $M^v \times r$. Each row in $C^v$ is associated with the corresponding slice from $A^v$. To select the most valuable columns in the matrix $C^v$, we average this matrix along the columns

$$c^v = \frac{1}{M^v} \sum_{i=1}^{M^v} |C^v_{i,j}| = \{c^v_j\}, \quad j = 1, \ldots, r.$$  

Let $K$ be the set of indices $k < r$ of the largest coordinates of the vector $c^v$ (typically, $k = 12$). Then, we remove the columns, whose indices do not belong to $K$, from the matrix $C^v$ and obtain the matrix $D^v$ of size $M^v \times k$. This operation is equivalent
to multiplication of $B^v$ with the matrix $\rho$ of size $\lambda \times k$, which is derived from $R_1$ by removing the rows, whose indices do not belong to $K$. Thus, the initial matrix $A^v$ consisting of the V-class slices, whose size was, for example, $M^v \times 1024$, is reduced to the matrix $D^v$ of the random footprints of the slices. The size of $D^v$ is $M^v \times 12$. To produce a similar reduction for the matrix $A^n$ in N-class slices, we multiply the N-class energy matrix $B^n$ with the matrix $\rho$. As a result, we obtain the random footprints matrix $D^n = B^n \cdot \rho$ of size $M^n \times 12$. We consider the coordinates of the $i$-th row of the matrix $D^{v(n)}$ as the set of $k$ characteristic features of the $i$-th slice from the matrix $A^{v(n)}$.

Now, the Mahalanobis distances $\mu_i, i = 1, \ldots, M^v$ of each row in the V-class matrix $D^v$ from the matrix $D^n$ are computed. Then, the sequence $\{\mu_i\}$ is averaged

$$\Delta = \frac{1}{M^v} \sum_{i=1}^{M^v} \mu_i.$$  

The value $\Delta$ is considered to be distance between the sets of features $D^v$ and $D^n$. The matrices $D^v$, $D^n$, $\rho$ and the value $\Delta$ are stored and we proceed to optimize the features.

We iterate this procedure up to 500 times. For the second iteration, all the above operations are conducted using a random matrix $R_2$, whose structure is similar to the structure of the matrix $R_1$. As a result, we obtain the features matrices $D^v_2$ and $D^n_2$, the random matrix $\rho_2$ and the distance value $\Delta_2$. The distance value $\Delta_2$ is compared to the stored value $\Delta$. Assume, $\Delta_2 > \Delta$. This means that the features matrices $D^v_2$ and $D^n_2$ are better separated from each other than the stored matrices $D^v$ and $D^n$. In this case, we denote $D^v_2$, $D^n_2$, $\rho_2$ and the value $\Delta_2$ as $D^v$, $D^n$, $\rho$ and the value $\Delta$, respectively. They are stored while replacing the previous stored items. If $\Delta_2 \leq \Delta$ then the stored items are left intact.

We continue to iterate this process up to 500 times. We stored the features matrices $D^v$ and $D^n$ such that the “distance” $\Delta$ between them among all the iterations is maximal. The reduced random matrix $\rho$ and the pattern matrices $D^v$, $D^n$ will be used in the identification phase. These items are denoted as $D_{\text{rand}}^v$, $D_{\text{rand}}^n$ and $\rho_{\text{rand}}$. 
RSNOFP: Version II  This version is similar, to some extent, to Version I. The difference is that, instead of selecting the most valuable columns in the matrix $C^v$ of size $M^v \times r$, we apply to this matrix the principal component analysis (PCA). As a result, we obtain the matrix $P = \{P_{i,j}\}$ of size $r \times r$. Each column of $P$ contains coefficients for one principal component. The columns are arranged in decreasing component variance order. The size of $P$ is reduced to $r \times k$ by retaining only the first $k$ columns

$$P_k = \{P_{i,j}\}, \quad i = 1, \ldots, r, \quad j = 1, \ldots, k.$$  

We obtain the feature matrix $D^v$ for the V-class by multiplying $C^v$ by $P_k$:

$$D^v = C^v \cdot P_k = B^v \cdot R_1 \cdot P_k = B^v \cdot \rho,$$

where $\rho = R_1 \cdot P_k$. The size of the matrix $\rho$ is $\lambda \times k$. Similarly, we produce the feature matrix $D^n$ for the N-class: $D^n = B^n \cdot \rho$. Similarly to Version I, we measure the “distance” $\Delta$ between the feature sets $D^v$ and $D^n$. The matrices $D^v$, $D^n$, $\rho$ and the value $\Delta$ are stored and we proceed to optimize the features, which is identical Version I. The features matrices $D^v$ and $D^n$ and the matrix $\rho$ are stored. These items are denoted by $D^v_{\text{pca}}$, $D^n_{\text{pca}}$ and $\rho_{\text{pca}}$.

RSNOFP: version III  This version differs from versions I and II. Here, we do not multiply the energy matrix $B^v$ by a random matrix. Instead, we perform a random permutation of the columns and retain the first $r$ columns. Thus, we get the matrix $C^v$ of size $M^v \times r$. Note, that this transform can be presented as the multiplication of the matrix $B^v$ by a matrix $T$ of size $\lambda \times r$, $C^v = B^v \cdot T$, where each column consists of zeros except for one entry, which is equal to 1.

Example:  Assume that the matrix $T$ is of size $4 \times 3$ that executes the permutation $[1 \ 2 \ 3 \ 4] \rightarrow [3 \ 1 \ 4 \ 2]$ of the columns of a matrix of size $4 \times 4$ while retaining the three
first columns:

\[
T = \begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & 0 \\
1 & 0 & 0 \\
0 & 0 & 1
\end{pmatrix}.
\]

The other operations are similar to the operations in Version II. We apply to the matrix \( C^v \) the PCA algorithm, which results in the matrix \( P = \{P_{i,j}\} \) of size \( r \times r \) of coefficients of principal components. The size of \( P \) is reduced to \( r \times k \) by retaining only the first \( k \) columns,

\[ P_k = \{P_{i,j}\}, \ i = 1, \ldots, r, \ j = 1, \ldots, k. \]

We obtain the feature matrix \( D^v \) for the V-class by multiplying \( C^v \) by \( P_k \):

\[ D^v = C^v \cdot P_k = B^v \cdot R_1 \cdot P_k = B^v \cdot \rho, \text{ where } \rho = R_1 \cdot P_k. \]

The size of the matrix \( \rho \) is \( \lambda \times k \). Similarly, we produce the feature matrix \( D^n \) for the N-class: \( D^n = B^n \cdot \rho. \) We measure the “distance” \( \Delta \) between the sets of features \( D^v \) and \( D^n \). The matrices \( D^v, D^n, \rho \) and the value \( \Delta \) are stored and we proceed to optimize the features, which is identical to Versions I and II. The features matrices \( D^v \) and \( D^n \) and the matrix \( \rho \) are stored. These items are denoted by \( D^v_{perm}, D^n_{perm} \) and \( \rho_{perm} \).

We illustrate the RSNOFP procedures (version II) by the diagram in Figure 8.5.1.
8.5. RANDOM SEARCH FOR A NEAR OPTIMAL FOOTPRINT (RSNOFP)

SCHEME

Figure 8.5.1: RSNOFP procedures (version II).
Chapter 9

Algorithm III: A diffusion framework for detection of moving vehicles

This algorithm [67] proposes a third approach for separating land vehicles such as cars and trucks from non-land vehicles like planes and helicopters via their acoustic signatures. The recordings are analyzed by inspecting the short-term dynamics of the acoustic signals.

The first phase of the algorithm is the learning phase, in which the training data is decomposed into small segments. Following the decomposition, features are extracted from every segment, feature extraction is applied to segments via the wavelet packets transform.

The wavelet packet transform is an efficient tool that is commonly used for signal processing applications where frequency and the time of the signal are jointly investigated. In this way, the wavelet transform produces an expansion over the time-frequency (scale) space, unlike the frequency-domain expansion that Fourier analysis yields. Wavelet analysis improves the resolution of the time evolution of the dominating frequencies, which is an important characteristic when analyzing acoustic signals. In addition, wavelet packet coefficients cover the frequency domain and capture the underlying dominating frequency.

The proposed algorithm extracts features by applying the 6-th order spline wavelet packet, which is described in Section 2.2 and in more detail in [2]. The features are then classified by using the diffusion maps framework (see [21] and Section 2.1). Diffusion
9.1. THE LEARNING STEP

Maps and diffusion distances provide a method for finding meaningful geometric structures in datasets. In most cases, the dataset holds data points in a high-dimensional space $\mathbb{R}^n$. The diffusion maps algorithm constructs coordinates that parameterize the dataset and the diffusion distance provides a local preserving metric for this dataset. A non-linear dimensionality reduction, which reveals global geometric information, is constructed by local overlapping structures. The diffusion maps and geometric harmonics are described in Section 2.1. For further details on the diffusion maps and geometric harmonics, see [21, 22].

In order to classify a new recording, only a short segment at a time is required (not the entire signal as a batch). The new recording is divided into small segments, which are embedded into the low dimensional space that was created by the diffusion maps coordinates. The new segments are classified by using their nearest neighbors in the low dimensional space.

9.1 The learning step

The learning step analyzes a test sample set of recordings $TSS = \{r_i\}_{i=1}^T$ whose classifications are known a-priori where $r_i$ is a recording of length $|r_i|$ and $T$ is the number of recordings in the training set. The recordings do not necessarily have the same size. Each recording $r_i$ is decomposed into overlapping segments $S_i = \{s^i_j\}$. A segment size is denoted by $l = 2^z, l, z \in \mathbb{N}$. The segments are grouped into a single set $\Sigma = \bigcup_{i=1}^T S_i$. For notational convenience, a single index is used in $s^i_j$ and the output is denoted by $\Sigma = \{s_j\}_{j=1}^{n_s}$ where the total number of segments resulting from the decomposition of all the signals is $n_s \triangleq |\Sigma|$. Next, representative features are extracts the segments. Each signal is divided into segments. Finally, the dimensionality of the feature vectors is reduced via the application of the diffusion maps algorithm. The process of feature extraction and dimensionality reduction is summarized in Algorithm 8:
9.1. THE LEARNING STEP

Algorithm 8 Learning acoustic features by wavelets analysis

1. Application of the wavelet packet transform that uses spline wavelet of order 6 to each segment.

2. Feature selection: Calculation of the energy distribution of the wavelet packet coefficients by summing the energy of the coefficients in every frequency band in each block.

3. Every $\beta$ consecutive segments are averaged in order to reduce the noise.

4. The learning set is classified via the application of diffusion maps.

Below is a detailed description of each step in Algorithm 8:

Step 1: The sixth order spline wavelet packet is applied up to a scale $D \in \mathbb{N}$ to each segment $s_j \in \Sigma$. Typically, if $l = 2^{10} = 1024$, then $D = 6$ and if $l = 2^9 = 512$ then $D = 5$. The coefficients are taken from the last scale D. This scale contains $l = 2^z$ coefficients that are arranged into $2^D$ blocks of length $2^{z-D}$. Each block is associated with a certain frequency band. These bands form a near uniform partition of the Nyquist frequency domain into $2^D$ parts. The outcome is $U \triangleq \{u_j\}_{j=1}^{n_s}$, where $u_j \in \mathbb{R}^l$. At the end of this step, each segment $s_j \in \Sigma$ is substituted for the set of its spline wavelet coefficients.

Step 2: The acoustic signature of a certain vehicle is constructed by using the distribution of the energy among blocks which consist of wavelet packet coefficients. The energy is calculated by summing the coefficients in every block $u_j$ that was constructed in the previous step by using the $l_1$ norm: block “energy” = $\sum \|\text{block coefficients}\|$. The result of this step is denoted by $E = \{e_j\}_{j=1}^{n_s}$ where $e_j \in \mathbb{R}^{2^D}$. This operation reduces the dimension by a factor of $2^{z-D}$.

Step 3: This step is applied to reduce perturbations and noise. The vectors of the set $E = \{e_j\}_{j=1}^{n_s}$ are labeled according to the label of the signal $r_i$ from where they were originated. The set $E$ is separated according to these labels. This separation forms $T$ groups. Within each group, the average of every $\beta$ consecutive segments is
9.2. THE CLASSIFICATION STEP

Calculated. The output is denoted by $\tilde{E} = \{\tilde{e}_j\}_{j=1}^{na}$ where $na = n_s - T \cdot (\beta - 1)$. The number of averaged segments $\beta$ is given as a parameter.

**Step 4:** The diffusion maps is applied to the set $\tilde{E}$. First, a Gaussian kernel $k(\tilde{e}_i, \tilde{e}_j) = \exp\left(\frac{\|\tilde{e}_i - \tilde{e}_j\|^2}{2\varepsilon}\right)$ of size $na \times na$ is constructed. The kernel is normalized by $P(\tilde{e}_i, \tilde{e}_j) = \frac{k(\tilde{e}_i, \tilde{e}_j)}{d(\tilde{e}_i)}$ where $d(\tilde{e}_i) = \sum_{\tilde{e}_{il} \in \tilde{E}} k(\tilde{e}_i, \tilde{e}_{il})$. Now, $P$ is a Markov transition matrix that corresponds to a random walk on the data with the following eigen-decomposition $p(\tilde{e}_i, \tilde{e}_j) = \sum_{\zeta \geq 0} \lambda_\zeta \psi_\zeta(\tilde{e}_i) \phi_\zeta(\tilde{e}_j)$. The family of diffusion maps $\Psi(\tilde{e}_j) = (\lambda_1 \psi_1(\tilde{e}_j), \lambda_2 \psi_2(\tilde{e}_j), \lambda_3 \psi_3(\tilde{e}_j), \ldots)$, embeds the dataset $\tilde{E} = \{\tilde{e}_j\}_{j=1}^{na}$ into an Euclidean space. In these new coordinates, the Euclidean distance between two points is equal to the diffusion distance between the two corresponding high-dimensional points. The diffusion distance measures the connectivity between the two points within the dataset.

9.2 The classification step

Denote by $r_t = (r(t - \zeta + 1), r(t - \zeta + 2), \ldots, r(t))$ the sequence of $\zeta$ signal values that were received up to time $t$ and have to be classified. Each recording $r_t$ is decomposed into $\beta$ overlapping segments $\{\vartheta_j\}_{j=1}^{\beta}$ of size $l$. First, features are extracted from the elements of the set $\{\vartheta_j\}_{j=1}^{\beta}$. Then, the diffusion maps embedding, which was created in the learning step (Section 9.1), is extended to include the new data via the geometric harmonics out-of-sample extension algorithm. This extension was presented in Section 2.1.4. Finally, each test segment is labeled and assigned to a specific class. These steps are described in Algorithm 9.
9.2. THE CLASSIFICATION STEP

**Algorithm 9** Classification of acoustic features by wavelet analysis

1. Application of the wavelet packet transform (spline wavelet of order 6) to the segments.

2. Calculation of the energies in the blocks of the wavelet packet coefficients.

3. Averaging of $\beta$ consecutive segments.

4. Extension of the diffusion maps embedding via the geometric harmonics (see Section 2.1.4).

5. Classification of the new sample according to its nearest neighbors in the diffusion maps coordinates.

---

**Step 1:** The spline wavelet packet is applied to each segment $s_j$ up to level D. The output is denoted by $\left\{v_j^{(\beta)} \right\}_{j=1}^{\beta}$, $v_j \in \mathbb{R}^l$.

**Step 2:** The blocks of energy are calculated similarly to what was done in the learning step. The output is denoted by $\left\{\epsilon_j^{(\beta)} \right\}_{j=1}^{\beta}$ where $\epsilon_j \in \mathbb{R}^{2D}$.

**Step 3:** The $\beta$ consecutive energy maps $\left\{\epsilon_j^{(\beta)} \right\}_{j=1}^{\beta}$ are averaged to a single energy map that is denoted by $\tilde{\epsilon}$, $\tilde{\epsilon} \in \mathbb{R}^{2D}$.

**Step 4:** The diffusion maps embedding of the training set is extended to the new point $\tilde{\epsilon}$ that was calculated in Steps 1-3. The training set $\tilde{E}$ is extended to include the new point $\tilde{\epsilon}$ by applying the geometric harmonic out-of-sample extension algorithm. The result is given by $\Psi(\tilde{\epsilon}) = (\lambda_1\psi_1(\tilde{\epsilon}), \lambda_2\psi_2(\tilde{\epsilon}), \lambda_3\psi_3(\tilde{\epsilon}), \cdots)$.

**Step 5:** The $\delta$ nearest neighbors of the embedded point $\Psi(\tilde{\epsilon}) = (\lambda_1\psi_1(\tilde{\epsilon}), \lambda_2\psi_2(\tilde{\epsilon}), \lambda_3\psi_3(\tilde{\epsilon}), \cdots)$, from the set $\left\{\Psi(\tilde{\epsilon}_j)\right\}_{j=1}^{n_0}$ are found, where $\delta \geq 1$ is given as a parameter. The classification of $r_i$ is determined according to highest occurring label of its $\delta$ nearest neighbors.
9.3 Experimental results

The test sample set, \((TSS)\) contains 42 recordings and is described in Section 7.4. The following parameters were used for the learning and classification steps of the algorithm: \(l = 512\), the overlap between segments was 50\%, \(\beta = 5\), \(D = 6\) and the number of diffusion coordinates was set to be 9. The number of nearest neighbors \(\delta\) was set to 7. These parameters were determined empirically.

The classification step was tested on recordings that were obtained in various road and wind conditions. For every recording that was used in the classification step we show: (a) the plot of the original signal and (b) the classification probability from the classifier. Probability that is above 0.5 signifies a detection of a vehicle. This probability is calculated by the number of nearest neighbors that are labeled as vehicles. As we are concerned with the detection of vehicles, an appropriate algorithm should detect vehicles as soon as they appear in the recording. Furthermore, the number of false positives should be minimal. Once detected, it is of less importance that the probabilities will be above 0.5 throughout the duration the vehicle is heard in the recording. The figures below show that the proposed algorithm exhibits all the required properties.
Figure 9.3.1: (a) The original recording that contains sounds emitted by a car at $t = 40$ seconds and by a truck $t = 50$ seconds. (b) Probability for detecting a vehicle, both events were classified as a moving vehicle.
Figure 9.3.2: (a) The original recording that contains sounds emitted by a jumpy vehicle that passed by the receiver at around $t = 70$ seconds, $t = 134$ seconds and $t = 200$ seconds from start. In its last passage it was followed by another car. (b) Probability detecting, the three events were detected accurately.
9.3. EXPERIMENTAL RESULTS

Figure 9.3.3: (a) Original recording. In the beginning of the recording, $t = 15$ seconds, a truck passed by the receiver followed by a car at $t = 35$ seconds. At around $t = 70$ seconds from the start of the recording a car followed by a truck passed by. At the end, $t = 100$ seconds, a minibus and a car arrived. (b) Probability for detecting a vehicle. The vehicles that passed by at $t = 15$, $t = 75$ and $t = 100$. The classification of the car that passed by at $t = 35$ was not obvious.

Figure 9.3.4 contains loud speech (from the start till 100 seconds), sounds emitted by a car (at around 105 seconds) and sound of a plane (from 107 seconds till the end of the recording).
9.3. EXPERIMENTAL RESULTS

Figure 9.3.4: (a) The original recording, which contains loud speech (from the start till $t = 100$ seconds), sounds emitted by a car (at around $t = 105$ seconds) and sound of a plane (from $t = 107$ seconds until the end of the recording) (b) Vehicle detection probability: the events were detected accurately.

Figure 9.3.5 contains the results from recording in which there is only background noise. The algorithm succeed in classifying the events in this recording as background noise.
Figure 9.3.5: (a) The original recording: background noise only. (b) Probability for detecting a vehicle, which stays very low.
Chapter 10

Summary and comparison

This part of the thesis described three algorithms for acoustic signal processing. The task was to classify and detect the presence of vehicles based on their acoustic recordings. The algorithms shared a similar structure: A learning phase for off-line training to separate between vehicle and non-vehicle recordings and an online detection phase that detects and classifies every new recording that did not participate in the training phase.

Chapter 7 introduced an algorithm that uses ideas that come from compressed sensing and Johnson-Linderstrauuss lemma to extract features from an acoustic signal. The features that were extracted in the training phase are used for the online detection phase to process the newly arrived acoustic data. The second algorithm was described in Chapter 8. The training database was constructed by using energy distribution of the acoustic signatures signals among blocks of wavelet packet coefficients. Classification of the acoustic signatures was determined by combining the outputs from two classifiers: classification and regression trees and nearest neighbors of the test signal from the training classes. Chapter 9 introduced the third algorithm. Wavelet packets are used to extract the acoustic features. The dimension of the features are reduced by the application of diffusion maps. The classifications of new recordings is done in the low dimensional space by the nearest neighbors algorithm. These three algorithms are generic and can be applied to different detection and classification problems in signal/image processing.

The following summarizes the main advantages and disadvantages of the three pro-
posed algorithms when they are compared to each other. Algorithm I (Chapter 7) is fast and easy to implement on a compact field instruments with limited CPU power and energy. This is sought after by different applications. On the other hand, the results from Algorithm I are noisier than Algorithm II and III because the separation between vehicles and background noises is not optimal. Algorithm II (Chapter 8) benefits from a strong feature extraction method. In addition, the classification is robust since it combines two classifiers. On the other hand, its complexity regarding feature extraction and classification phase is more expensive than Algorithm I. Algorithm III (Chapter 9) uses a robust feature extraction method that is based on wavelet packets. The features are further improved by using the low dimensional embedding. The algorithm’s disadvantage is in the classification phase of new data points. The extension scheme to find the coordinates for the classification of new data points is based on geometric harmonics, which requires to save the original dataset.
Bibliography


