Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Detektion und Verfolgung von Zügen in OTDR Signalen

Ein robustes Framework zur Zuglokalisierung in Signalen von optischer Zeitbereichsreflektometrie mit GPGPU Beschleunigung

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Visual Computing

eingereicht von

Adam Papp, BSc
Matrikelnummer 1327381

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: O.Univ.Prof. Dipl.-Ing. Dr.techn. Walter G. Kropatsch
Mitwirkung: Dipl.-Ing. Dr. Martin Litzenberger

Wien, 25. November 2016

Adam Papp
Walter G. Kropatsch

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

Train Detection and Tracking in OTDR Signals

A Robust Framework for Train Localization in Optical Time Domain Reflectometry (OTDR) Signals using GPGPU Acceleration

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Visual Computing

by

Adam Papp, BSc
Registration Number 1327381

to the Faculty of Informatics

at the TU Wien

Advisor: O.Univ.Prof. Dipl.-Ing. Dr.techn. Walter G. Kropatsch
Assistance: Dipl.-Ing. Dr. Martin Litzenberger

Vienna, 25th November, 2016

Adam Papp

Walter G. Kropatsch

Erklärung zur Verfassung der Arbeit

Adam Papp, BSc
Friedlgasse 47/6, 1190 Wien, Österreich

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 25. November 2016

Adam Papp

Danksagung

Ich möchte mich bei meinem Betreuer Professor Walter Kropatsch und bei seiner Arbeitsgruppe an der Patter Recognition and Image Processing Group der Technischen Universität Wien für die Möglichkeit bei dieser Masterarbeit über dieses interessante Thema zu schreiben und für die ausgezeichnete Unterstützung während der letzten Jahren bedanken.

Der Großteil der praktischen Arbeit für diese Masterarbeit wurde bei der New Sensor Technologies Business Unit des Digital Safety and Security Departments im AIT Austrian Institute of Technology durchgeführt. Ich möchte mich bei Christoph Wiesmeyr, Martin Litzenberger und Heinrich Garn für die Bereitstellung der notwendigen Infrastruktur und für die ausgezeichnete Unterstützung während der Forschungsarbeit bedanken.

Die Daten die die Grundlage für diese Arbeit bildeten, wurden im Forschungsprojekt *Sensorsystem für Bahnstrecken* erhoben, gefördert durch die Österreichische Forschungsförderungsgesellschaft FFG unter der Vertragsnummer 840448. Ich möchte mich besonders bei Herrn Wolfgang Zottl (ÖBB Infrastruktur GmbH) für Genehmigung der Messungen und bei Herrn Günther Neunteufel (Fiber Cable Technologies GmbH) für die Bereitstellung der Daten bedanken.

Acknowledgements

I want to thank my advisor Professor Walter Kropatsch and his work group at the department Patter Recognition and Image Processing Group at the Technical University of Vienna for the possibility to write my thesis about this interesting topic and for the outstanding support during the last years.

The major part of the practical work for this thesis was done at the New Sensor Technologies Business Unit of the Digital Safety and Security Department at the AIT Austrian Institute of Technology. I want to thank Christoph Wiesmeyr, Martin Litzenberger and Heinrich Garn for providing me with the necessary facilities and their outstanding support during my research work.

The data used for the work presented in this paper have been collected in the research project *Sensorsystem für Bahnstrecken* funded by the Austrian Research Agency FFG under contract number 840448. I especially want to acknowledge Wolfgang Zottl (ÖBB Infrastruktur GmbH) for approving the measurements and Günther Neunteufel (Fiber Cable Technologies GmbH) for providing the data.

Kurzfassung

Diese Masterarbeit untersucht Algorithmen für die Nutzung von optischer Zeitbereichsreflektometrie (optical time domain reflectometry, OTDR) für Verbesserung der Eisenbahnsicherheit. OTDR, oft auch als Distributed Acoustical Sensing (DAS) bezeichnet, misst die Rayleigh-Rückstreuung eines Lichtimpulses in einem Lichtwellenleiter. Das resultierende Signal liefert Informationen über den lokalen Schalldruck in den einzelnen Segmenten, die Positionen entlang des Lichtwellenleiters entsprechen.

Mit Hilfe von optischer Zeitbereichsreflektometrie ist es möglich Vibrationen im Boden, die durch verschiedene Quellen verursacht werden können, mit hoher Genauigkeit in Zeit und Ort zu detektieren. In dieser Arbeit wird ein neues Verfahren für die Detektion von Boden-Vibrationen die durch Züge verursacht werden, mittels optischer Signale aus Lichtwellenleitern welche in wenigen Meter Entfernung längs der Eisenbahnstrecken installiert sind, vorgeschlagen. Das hier präsentierte Verfahren lernt die charakteristischen Muster der Vibration von Zügen in der Fourier Domäne mit Hilfe einer Support Vector Machine (SVM).Es wurde gezeigt, dass es mit einer General Purpose Graphical Processing Units (GPGPU) möglich ist, die Merkmalswerte in Echtzeit zu berechnen.

Es wurde weiter die Möglichkeit untersucht eindeutige Merkmale aus dem Signal zu extrahieren um Zügen wiederzuerkennen. Ergebnisse dazu, die mit Hilfe von Short-Time Fourier Transformation (STFT) erzielt wurden, werden präsentiert und die derzeitigen Einschränkungen werden diskutiert. Ansätze zur Zuordnung eines Lichtwellenleitersegments zu einer eindeutigen Position entlang der Zugstrecke werden vorgestellt und diskutiert.

Für die Verfolgung von Zügen wird ein Punkt-basierter kausaler Algorithmus vorgestellt. Das Verfahren zur Zugsverfolgung ist zweistufig um Fehler zu reduzieren und es wurde als Optimierungsproblem gelöst. Obwohl in der Literatur bereits mehrere Algorithmen für die Zugverfolgung mit OTDR Signale demonstriert wurden, wurde diese nie an längeren, realistischen Streckenabschnitten oder mit einer größeren Anzahl von Zügen getestet. In Gegensatz zum bisherigen Stand der Forschung, enthält der Streckenabschnitt der in dieser Arbeit untersucht wurde, vier Zughaltestellen und mehr als zehn Zugtrajektorien über eine Gesamtbeobachtungsdauer von zwei Stunden unter realistischen Bedingungen. Nach unserer Kenntnis wurden in der vorliegenden Arbeit Algorithmen für Zugdetektion- und tracking mit OTDR Signalen zum ersten Mal gegen Grundwahrheiten von Trajektorien aus einem konventionellen Zugtracking-System validiert.

Abstract

This thesis investigates the use of an Optical Time Domain Reflectometry (OTDR) device for railway safety improvement. OTDR sensing, often also termed Distributed Acoustical Sensing (DAS), measures the Rayleigh backscattering of a light pulse along an optical fiber. The resulting signal provides information on local acoustic pressure at linearly spaced segments, corresponding to positions, along the fiber.

Using optical time-domain reflectometry, vibrations in the ground caused by different sources can be detected with high accuracy in time and space. We propose a novel method for the detection of vibrations caused by trains in an optical fiber buried within a few meters from the railway track. The presented method learns the characteristic pattern in the Fourier domain using a Support Vector Machine (SVM) and it becomes robust to background noise in the signal. We show that using a General Purpose Graphical Processing Unit (GPGPU) it is possible to compute feature values relevant for train detection in real-time.

We further investigate the possibility to extract unique features to identify trains. We show our results obtained with Short-Time Fourier Transformation (STFT) and discuss the current limitations. We discuss our results for associating fiber cable positions to real-world locations.

For the tracking of trains, a point-based causal algorithm is presented. The tracking has two stages to minimize the influence of false classifications of the vibration detection and is solved as an optimization problem. While several algorithms have been demonstrated in the literature for train tracking using OTDR signals, they have neither been tested on longer recordings nor with a large number of train samples. In contrast to that, our data contain four railway stations and more than ten train trajectory crossings over two hours under realistic conditions. To our knowledge, the presented algorithm is the first one in the literature which is tested against ground truth of train trajectories from a conventional train tracking system.

Contents

Kurzfassung **xi**

Abstract **xiii**

Contents **xv**

1 Introduction **1**
1.1 Train Tracking for Railway Safety 2
1.2 Optical Time Domain Reflectometry 3
1.3 Aim of Thesis 3
1.4 Structure of Thesis 3

2 Problem Formulation **5**
2.1 Experimental Setup and Input Data 6
2.2 Online Processing of OTDR Data 10

3 Related Work **11**
3.1 Signal Processing in OTDR Signals 11
3.2 Descriptors for Time Series Signals 13
3.3 Motion Tracking 14

4 Basics of Signal Processing **15**
4.1 Fourier Transformation 15
4.2 Short-Time Fourier Transformation 16
4.3 Wavelet Transformation 18
4.4 Data Binning 18

5 Train Detection and Tracking **21**
5.1 Vibration Detection 22
5.2 Point-Based Tracking 25
5.3 Train Profile 32
5.4 Segment Calibration 33
5.5 GPU Accelerated Vibration Detection 38

6 Evaluation **41**
6.1 Vibration Detection . . . 41
6.2 Point-Based Tracking . . . 44
6.3 Train Profile . . . 48
6.4 Segment Calibration . . . 53
6.5 GPU Accelerated Vibration Detection . . . 53

7 Conclusion **57**
7.1 Future Work and Outlook . . . 58

List of Figures **59**

List of Tables **60**

List of Algorithms **60**

Bibliography **61**

CHAPTER 1

Introduction

Railway control and safety is an important task as millions of people and cargo are being transported every day. The development of new technologies has allowed high-speed trains to be installed and to operate. The aim of todays engineers are improved safety, efficiency, punctuality and personalization.

At the IEEE International Conference on Intelligent Rail Transport, the key speakers pointed out that the current challenge is to merge and process as much data as possible, as fast as possible. The vision of Keith Dierkx, a member of IBM's Industry Academy, is that in 2025, self-driving trains will be on the rail tracks. He explained that around 80 % of the data loses its information value within minutes, therefore rapid interpretation is required. He mentioned that in the USA there are drones constantly monitoring railway tracks. Currently, these drones do not process data, but the next step is to do on-line processing to reduce time delay. As an innovative personalization concept he was talking about the possibility of tickets on demand. Stations will recognize that a passenger has entered the building and in case he needs to wait at the station, the system could automatically offer an earlier ticket for the passenger through his smartphone.

During the conference many speakers were discussing train position monitoring next to other important topics (e.g. scheduling or condition monitoring). Most of these techniques are based on bidirectional communication between train and control stations. It is clear that each system has its advantages and drawbacks. The most common way to overcome drawbacks is to use different equipment simultaneously and merge their strengths. In this thesis, we investigate the use of an optical time domain reflectometry to locate and track trains. The major advantage of this method is the passive detection of trains, in the sense that trains do not need to communicate with the OTDR device, which can overcome the weaknesses of active communication. In the remaining part of the introduction, we will discuss the basic concept of train tracking in railway safety and optical time domain reflectometry in brief.

1.1 Train Tracking for Railway Safety

During the development of railway safety, companies have implemented different types of systems to monitor train movements. Later, in the European Union, these systems were standardized into two categories, the European Train Control System (ETCS) and the Communications-based Train Control (CBTC). ETCS is applied on mainline sections, while CBTC is used in urban environments. The specifications of the European Train Control System (ETCS) defines four level of safety. The most common method for mainline train tracking is the installation of so called "balise" along the railway track. These sensors detect a train when it passes by and transmit a signal to the control station. Other sensor types like axle counters or track circuits can also be installed along the railway track. These tracking methods are included in the first three levels (0,1 and 2) of the ECTS specifications. Level 2 already includes continuous data transmission between trains and control station through GSM communication. The installation of such systems is often very costly, therefore it is often limited to high priority tracks. Examples for such systems can be found in [ALZ13]. The last level (currently, Level 3) departs from classic block signaling with track side equipment and defines the "moving block" signaling systems, see Figure 1.1. This new system defines in real-time safety blocks around moving trains by using some kind of end-of-train device, for example a GPS receiver. The position information from the GPS receiver is then transferred from the train to the station by a GSM antenna. Virtual "balise" can defined using GPS coordinates, which allows to overcome weak signals from GPS or GSM communication. The next goal is to collect more properties from moving trains like speed or acceleration to define shorter blocks of moving trains. [Wil16]

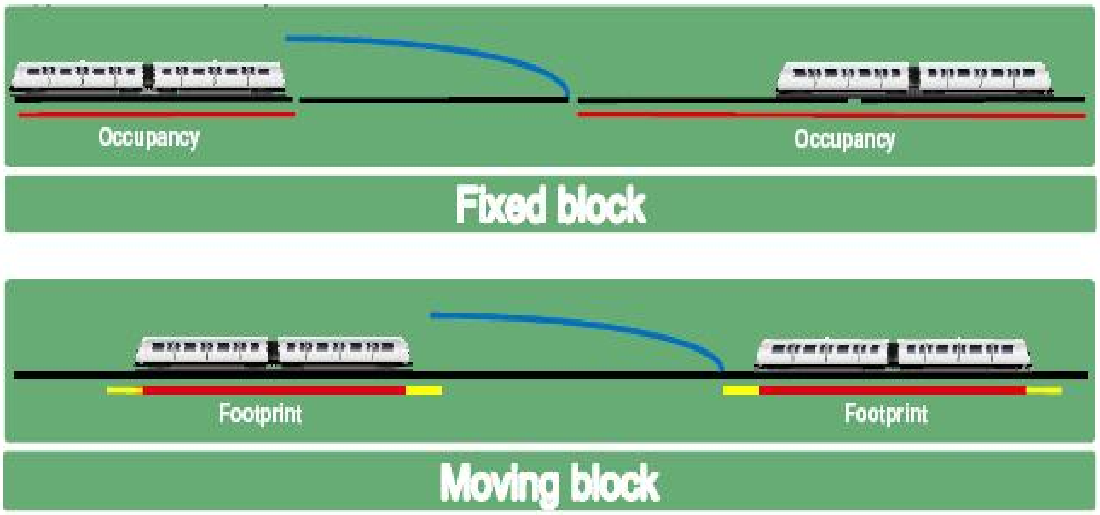

Figure 1.1: Comparison between fixed block and moving block in railway safety. Image taken from [Isr11]

1.2 Optical Time Domain Reflectometry

The optical time domain reflectometry is an optoelectronic device, which injects a series of light pulses into a fiber cable and measures the scattered or reflected light (Rayleigh backscatter) at the same end of the cable. The device was invented to measure the characteristics of a fiber cable or to detect break points in the fiber cable. It was later discovered that the refractive index changes due to pressure on the cable, which is the major cause for scattering and as of 2013, it can be measured with a positional resolution down to 0.5 meter [BC12]. This allows to use the OTDR device for intrusion detection [CJT03, JMCT05, ODW12] or even for rock slide detection [CCWW11].

1.3 Aim of Thesis

This thesis investigates the use of an optical time domain reflectometry (OTDR) device as an alternative in railway safety for "moving block" signaling system. The currently applied methods use bidirectional communication between train and stations. Since fiber cables are already installed next to railway tracks, train vibrations can be detected using an OTDR device, without any modification of existent railway infrastructure. This allows passive detection of trains at railway stations, since trains have no active interaction with the fiber cable. The aim is to detect and track trains and to discuss the reliability of the applied methods. The positional error of the tracking will be compared with ground truth data obtained from conventional train detection system. Furthermore, methods will be investigated to extract unique features for each train.

1.4 Structure of Thesis

The thesis builds on the previous paper [PWL$^+$16a] published at the IEEE-ICIRT 2016 (International Conference on Intelligent Rail Transport) and on the paper [PWL$^+$16b] published at the GCPR 2016 (German Conference on Pattern Recognition) conferences.

Chapter 1 briefly introduces the current railway challenges and railway safety technologies to the reader, which are relevant for this thesis. It briefly describes the Optical Time Domain Reflectometry (OTDR) device. Finally, it points out the aim of this thesis and describes its structure.

Chapter 2 begins with a description of the goal of this thesis. Then it describes the input data and the received ground truth data and calibration based on [PWL$^+$16a]. Finally, it explains the constraints to develop an algorithm to handle data from an OTDR device in real-time.

Chapter 3 reviews the state-of-the-art in OTDR signal processing based on [PWL$^+$16b]. Then it further details OTDR usage for railway safety. After that, a review is presented about time series signal descriptors in order to extract train profiles. Finally, motion tracking methods are reviewed, based on [PWL$^+$16b], to be able to track trains.

Chapter 4 briefly details the techniques applied for signal processing. First, Fourier transformation and Short-time Fourier transformation are described and compared. Then, wavelet decomposition is detailed and finally data binning is described.

Chapter 5 is the main chapter, where the developed methods are described. First, the vibration detection method is presented, based on [PWL$^+$16b]. Then the motion tracking method is detailed in two parts, key-point extraction partly based on [PWL$^+$16b] and point-based tracking based on [PWL$^+$16b]. Later, train profile extraction and profile comparison are detailed, then segment calibration, both in position and quality, is described. Finally, a GPU accelerated vibration detection is presented.

Chapter 6 is where the evaluation of the methods are described, which are in chapter 5 detailed. The structure of this chapter is identical to the fifth chapter.

Chapter 7 draws a final conclusion about the presented results and closes with an outlook for future work.

CHAPTER 2

Problem Formulation

The aim of this research is to develop a framework, which is able to detect trains and track their movement in OTDR signals. Given the raw OTDR measurements, we want to detect and track trains, i.e. for each train that passes through the monitored stretch of railway tracks, we want to automatically detect tuples which describe the position of the train at a given time.

The evaluation of the train tracking is measured based on two criteria: correct detection of all the trains passing the monitored track and positional error of the individual train trajectories.

A signal model that describes the mechanism how vibration affects the fiber cable and how the OTDR device measures it, to our knowledge, has not been described and published so far. Therefore, we have to rely on previous works on signal processing and tracking to solve our problem.

In the following, we will describe the entities present in this work:

- Fiber cable: designed for data transmission between railway stations, installed in the vicinity of the railway track
- Train: locomotive and wagons, moving on a railway track mostly in one direction
- OTDR: device measuring the backscatter in the fiber cable
- Segment: one slice of the fiber cable measured by the otdr device
- Loop: extra length in the fiber cable next to the railway track
- Detour: extra length in the fiber cable further from the railway track
- Vibration: change of backscatter in the fiber cable due to pressure change on the fiber cable

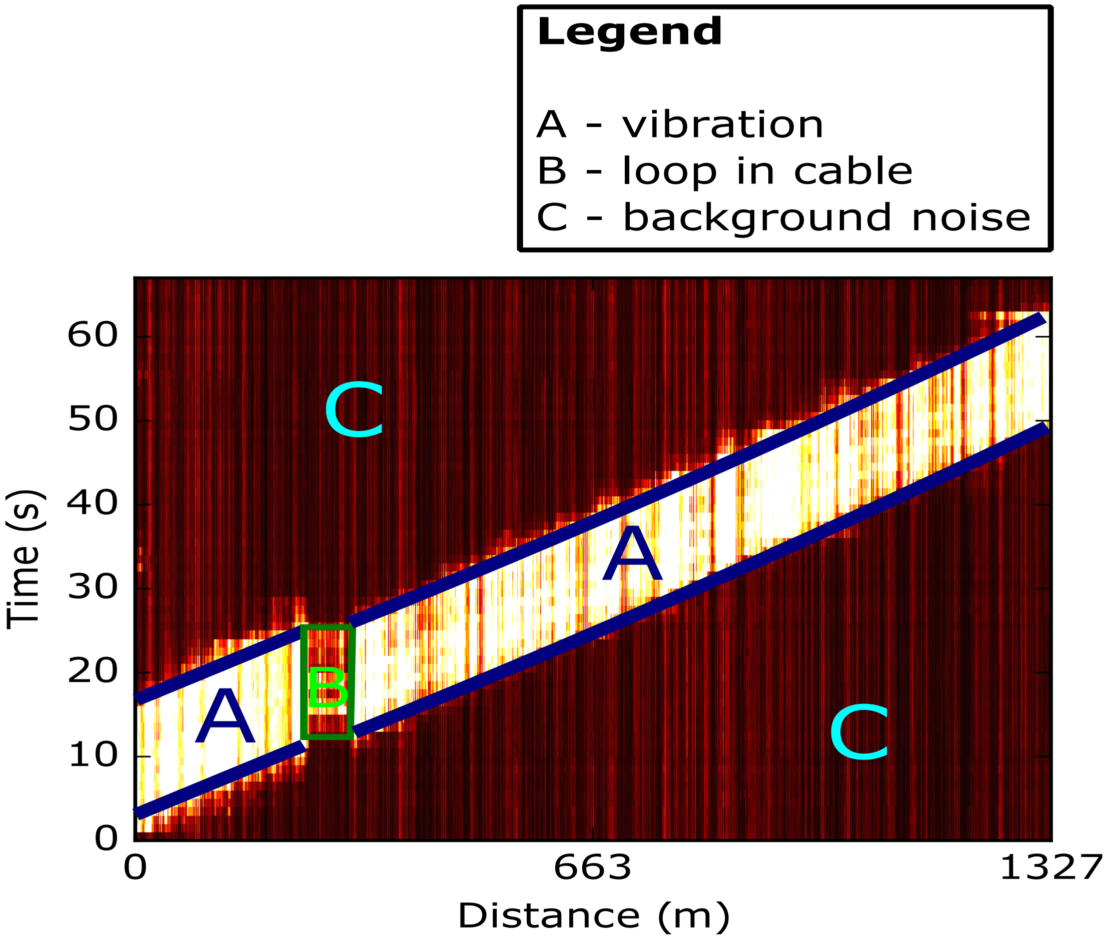

Figure 2.1: Description of an OTDR data.

2.1 Experimental Setup and Input Data

Our datasets were acquired with an OTDR device [KS12] and stored as a matrix in a HDF5 file, where each row contains the measurements of the optical cable characteristics measured with one laser pulse. The dimension of the matrix is determined by the number of cable segments and the duration of the recording. The fiber cable consists of a fixed number of segments in each recording and a segment has a length of 0.68 meters. The raw OTDR data describes the amplitude of the backscatter. Figure 2.1 shows the energy of the signal for each second and segment between frequency 50 Hz and 150 Hz. When a train generates vibration, the energy of the signal is higher than the energy of the background noise. Due to the impurity of the fiber cable, backscatter is always present and results in background noise. The cable does not necessarily follow the railway strictly and extra lengths (e.g. loops or detours) can be present.

Table 2.1: Parameters of the two long recordings.

	First recording	Second recording
Observed cable length	13481m	17484m
Recording time	2709s	4487s
Number of segments	19825	25712
Spatial resolution	0.68m	0.68m
Sampling rate	4000Hz	2000Hz

The monitored section, visualized in Figure 2.2, has two parallel railway tracks. There are four stations along the monitored track. Around the first station, the optical cable is further from the track, resulting in a weakened signal. Between the third and the fourth station, there is a shunting yard.

The fiber cable used to obtain the recordings is a conventional optical communication cable installed along the rail track in typically several meters distance from the railway track. The exact position of the cable is unknown, it can get closer or further to the railway track or go under the track to the other side of the rail. We had no influence on the fiber cable position within this contribution as the cable had already been installed for communication purposes.

We received 14 short recordings (duration around one minute) containing only one train. These datasets consist of 1951 segments with the spatial resolution of 0.68 meters, corresponding to 1327 meters and were sampled with 5000 Hz. We further received two long recordings, described in Table 2.1 and visualized in Figure 2.3. The two measurements were recorded with one minute offset to change the parameters of the recording device.

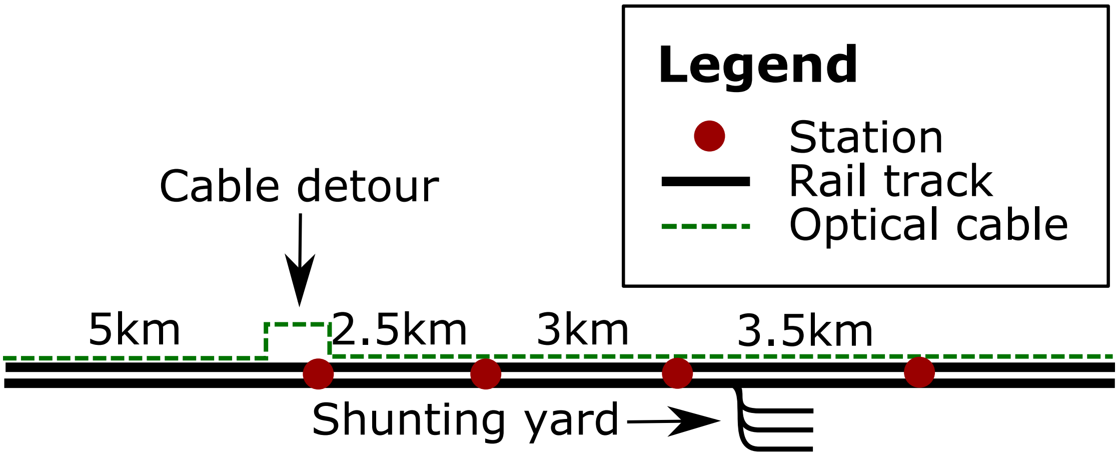

Figure 2.2: Illustration of the monitored railway section.

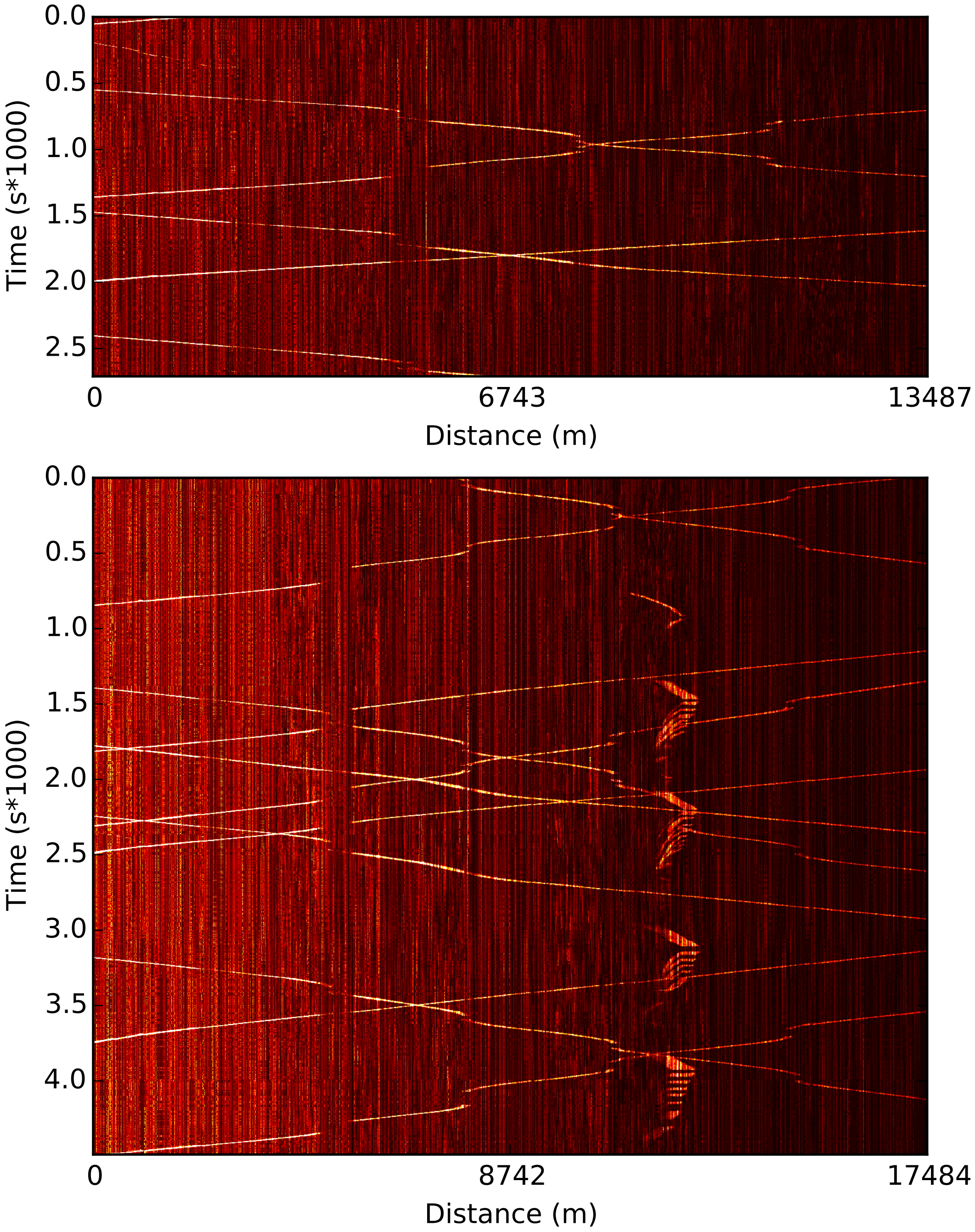

Figure 2.3: The input datasets, where each second of measurement is aggregated into one pixel. The pixel intensity is the energy of frequency coefficients between 50 and 150 Hz.

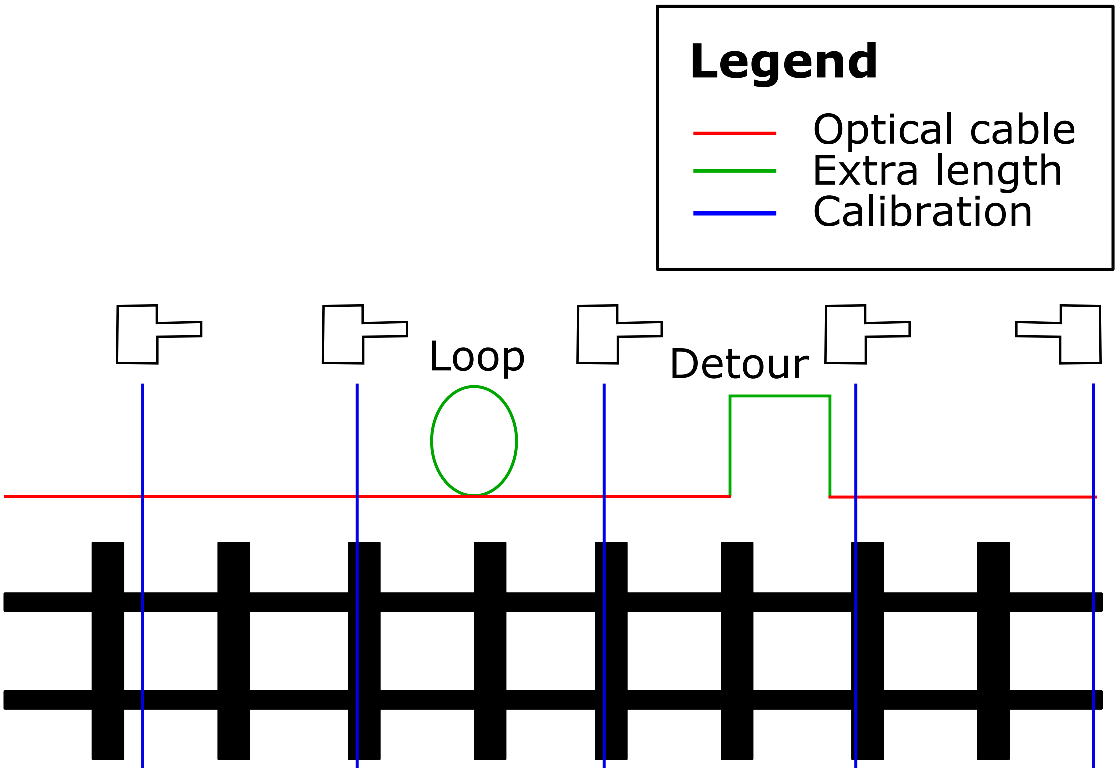

Figure 2.4: Illustration of the calibration of the track, where two extra lengths are shown.

We further obtained ground truth data from a conventional train tracking system for the two long recordings. When a train passes a signal along the track it is automatically registered and the time the train passes is stored in a train tracking system. These data contain point based passing times for each train traveling along the track. In order to be able to link cable segments to track kilometers a hammer was used to produce vibrations at given points on the track that can then be measured by the OTDR, see Figure 2.4. Between two calibration points, the cable segments are assumed to be arranged linearly. This is only an approximation when extra lengths, such as loops or detours, are present in the cable. We received only a small portion of around 1.5km calibration information at the beginning of the measured section, since this operation requires safety precautions and it is not possible to apply on the complete railway section. Within the calibrated stretch there are two sensors in each direction registering trains. For the positions of these signals we know the exact cable segment numbers. The train registration times are accurate to one second. With trains moving at around 120 kilometers per hour, this corresponds to an accuracy of around 33 meters in distance in space.

2.2 Online Processing of OTDR Data

Although in this contribution, the data along the complete time axis are available, the detection and tracking have to be implemented for online processing using only a given chunk of measurements at a time. The aim is to be able to seamlessly attach the developed method to an OTDR device. Another reason is the large amount of data generated by the OTDR device each second. The required memory to store raw data of one second can be computed from the sampling rate and number of segments, which in our case can reach up to 320 MB/sec. In an online processing scenario, when the algorithm is handling the data in streams of one second, it must execute all operations on the input data in less than a second. Safety critical time latency acceptance is not considered in this thesis.

CHAPTER 3

Related Work

The European Train Control System (ETCS) is the current standard for railway monitoring on mainline sections. Examples for such systems can be found in [ALZ13]. Their drawback is that they require bidirectional communication between railway equipment or locomotive and the control station. Furthermore, the future improvement of railway monitoring requires real-time information of train movement description to allow to operate more trains on railway tracks [Wil16]. Our proposed system allows passive detection of train movements in real-time from control stations. For a brief introduction to current train tracking standardization, please see section 1.1.

In this chapter, methods that are relevant for train detection in OTDR signals are reviewed. First, signal processing methods in OTDR signals are described, then methods that use OTDR for railway monitoring are described. Furthermore, signal description techniques, mostly applied in audio processing and feature extraction methods are reviewed. Finally, relevant motion tracking techniques are discussed.

3.1 Signal Processing in OTDR Signals

OTDR signals have been used in many different fields and many different signal processing techniques have been applied to these signals. In the following, we provide an overview of the literature available.

For intrusion detection, averaging methods were used [CJT03, JMCT05], where the latest measurement is subtracted from the average. Averaging methods are accurate enough for intrusion detection where the position of the optical fiber can be chosen such that an event causes a large signal. For train vibration detection, these methods proved not to be accurate enough due to the highly varying signal to noise ratio (SNR).

Qin et al. [QCB12] introduced wavelet denoising in OTDR signal processing by thresholding the wavelet coefficients. They point out that wavelet denoising allows to remove noise that is present across all the frequencies while other frequently used methods (e.g. moving average) often only remove high frequency content from the signal. Wavelet denoising was applied by Peng et al. for pipeline intrusion detection [PWJ$^+$14] and for train detection [PDRL14]. To obtain locations of rising and falling edges of train signals, normalized sliding variance was applied.

Using a spectral decomposition of the signal, different authors have tried to distinguish between different events detected in OTDR signals. Kong et al. [KZX$^+$14] have presented a method to detect events in OTDR signals based on Short-time Fourier Transformation (STFT) and correlation matching. They build templates for reflective and non-reflective events, then cross-correlate the signal transformed with STFT. Wu et al. [WLPR14] presented a method based on phase space matrix construction, eigenvalue decomposition and neural network classification to process signals from OTDR signals. After the eigenvalue decomposition, they show that three eigenvalues can reconstruct the energy of the original signal with 95 % accuracy. Timofeev et al. [TE14] applied Gaussian Mixture Model (GMM) with Support Vector Machine (SVM) kernel to distinguish between different events (e.g. a walking man, a truck moving) in signals acquired with an OTDR device. The (GMM-SVM) method was introduced by You et al. [YLL10] for speaker recognition. The feature values they use are built using Linear-Frequency Spaced Filterbank Cepstrum Coefficients (LFCC) from 200 to 3000 Hz. Often the first-order delta coefficients of the LFCC are added to the feature vector to improve classification. Another variant is the Mel Frequency Cepstral Coefficient (MFCC), where logarithmic spacing is used. Timofeev [Tim15] also applied GMM-SVM to detect activities (e.g. soil digging with an excavator or working with a crowbar) near the railway track in signals from OTDR devices.

3.1.1 Railway Related OTDR Techniques

There have been several methods applied for railway monitoring using an OTDR device. The most similar method to ours was published by Peng et al. [PDRL14], where they applied wavelet denoising to extract train signals. Our input data show a high variance between segments, therefore we could not apply wavelet denoising with the same threshold for each segment. Furthermore, wavelet transformation introduces more computation compared to Fourier transformation. They evaluated their method with only one train trajectory crossing without train stops, therefore their tracking algorithm would not be reliable in our datasets.

Timofeev et al. [TE14, Tim15] applied an OTDR device to detect activities near railway tracks, such as train moving, man walking or soil digging. Their method shows that it is possible to detect trains, but their parameters are set for different task.

Lienhart et al. [LWW$^+$16] extracted the condition of train axes and wheels from OTDR signals to reduce the chance of accidents. In their contribution they show that a train profile can be extracted, which describes the condition of wheels, but in their setup it was not possible to extract exact wheel positions. OptaSense has published a commercial brochure [Opt14], but they have not published a technical paper of their contribution. They show that it is possible to extract exact location of axes from wagons using Short-Time Fourier Transformation from OTDR signals, if the fiber cable is laid properly.

3.2 Descriptors for Time Series Signals

Descriptors for time series, or one dimensional signals, received an important amount of interest as many real-life applications require classification. The most common field where such descriptors are required are speech recognition, audio classification or signals from distance measuring devices. Signals extracted from the raw OTDR data are similar to audio signals, often it is termed as Distributed Acoustic Sensing, therefore it is relevant to review techniques to extract time series descriptors.

A classical approach to classify a signal is the use of k-nearest-neighbor (kNN) combined with Euclidean Distance or Dynamic Time Warping (DTW). DTW is often used when two temporal sequences vary in speed during observation and therefore warping is needed for matching. It is well-known in speech recognition and online signature recognition.

For image classification, the Scale-Invariant Feature Transform (SIFT) introduced by Lowe [Low04] combined with Bag-of-Words (BoW) have achieved high accuracy. Xie and Beigi [XB09] have adapted the SIFT features for one dimensional signals. They apply Difference of Gaussian (DoG) in consecutive scales on the signals to extract keypoints. After extracting keypoints they are matched using nearest neighbor methods. The BoW method has been adapted for time series classification by Bailly et al. [BMT$^+$15]. With the help of the k-means algorithm they create a codebook of words. Finally, signals are classified using linear SVM with a normalized histogram of words. We have extracted signals in different scales, but seen a high variance within our limited amount of data. We applied manual thresholding for our experiments, but with a sufficient amount of data, this concept could be used for train profile description.

Hinton et al. [HS06] have introduced the autoencoder, which is a deep network to reduce high-dimensional data to a low-dimensional code layer. They show that their method outperforms the principal component analysis and allows more accurate decision boundaries for the given data sets. Their contribution is a very effective non-linear dimensionality reduction, but it requires a big enough dataset for training, which is a bottleneck in our case.

The concept of deep networks has been applied by many authors for image classification [STE13, ESTA14] or audio signal processing [HDY$^+$12, HBL13]. Humphrey et al. [HBL13] compared the accuracy of PCA and Convolutional Neural Network (CNN) for musical instrument classification. The network input were patches of constant-Q pitch spectra, such that translation was linear in both pitch and time. They showed that the CNN based method highly outperforms PCA, 98 % compared to 63 % accuracy.

There has been a number of experiments of convolution both along the frequency axis and the time axis of audio signals extracted from Short-Time Fourier Transformation

(STFT). It is often required (e.g. music recognition) to take into account frequencies for a longer period of time, since short-time analysis cannot capture higher level information. For phone recognition, Tóth [Tót14] experimented with convolution along frequency or time axis and a combined convolution on both axes of a STFT signal. In his work, he achieved with the combined convolution a new record of classification on the TIMIT dataset with an error rate of 16.7 %.

3.3 Motion Tracking

This section gives an overview on motion tracking between image frames based on the survey of Yilmaz et al. [YJS06] and we discuss the possibilities of point-tracking, kernel-tracking and silhouette-tracking for our purpose. Point-tracking relies on an external mechanism that detects objects in every frame. Deterministic methods solve the correspondence between consecutive frames by defining a cost between tracked points. Minimization of the cost can be formulated as a combinatorial problem [JFL07, RS91, SJ87, SS05]. Stochastic methods, e.g. Kalman filter [Kal60], solve the correspondence by taking the measurement and model uncertainties into account. A Kalman filter can only be applied for one object, multiple objects need multiple Kalman filters.

Kernel tracking is based on tracking objects by representing them with primitive shapes and computing the motion of these primitives. Template matching is a brute-force method as it searches the image for a region similar to the object. Similarity measurements can be defined using e.g. intensity or gradient values. The mean-shift clustering proposed by Comaniciu and Meer [CRM03] iteratively updates clusters within an image. A cluster is shifted by the vector, called mean-shift vector, between the current mean and the next mean, computed from the data lying inside a multidimensional ellipsoid. They proposed a similar method to track motion, which is called the mean-shift tracker. It maximizes the similarity of appearance, by means of Bhattacharya coefficient, by comparing the histogram of the object and the histogram of the next possible object location. Optical Flow proposed by Horn and Schunck [HS81] computes a dense displacement vector field, which defines the translation of the motion for each pixel. The KLT tracker proposed by Shi and Tomasi [ST94] iteratively computes the translation of a patch, centered on an interest point, in a similar construction to the optical flow, but resulting in a sparse displacement vector field. The proposed feature vectors extracted in this contribution are not reliable enough to apply kernel-tracking.

Silhouette-tracking is based on tracking objects, e.g hands, by a complex descriptor of the region. In ideal case, train signals acquired by an OTDR device form rectangular signals after classification and do not require complex descriptors. Signal edges can be detected and a point-tracking mechanism can be adapted. It is essential to handle the entry and exit of objects as trains are entering and leaving the measurement range. Multiple data association must also be handled, as multiple trains can appear in one time frame.

CHAPTER 4

Basics of Signal Processing

This chapter is dedicated to describe the methods that are applied for signal processing in this thesis. We describe briefly the Fourier and Wavelet transformation. Finally, we discuss quantization and the generic matrix multiplication solution.

4.1 Fourier Transformation

The Fourier transformation [Bra99] converts a square integrable function into a sum of simple waves represented by sines and cosines. The Fourier transform of f is defined as

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(t)\ e^{-2\pi i t \xi}\, dt\, \xi \in \mathbb{R}. \tag{4.1}$$

The resulting complex function $\hat{f}$ describes the signal in the frequency domain. The value $\hat{f}(\xi)$ represents the amplitude and phase at the frequency ξ. The absolute value $|\hat{f}(\xi)|$ is called the magnitude of the Fourier transform at frequency ξ.

In case the input data are real input signals, the negative frequency terms are the complex conjugates of the corresponding positive-frequency terms. Therefore it is possible to compute only the positive frequency terms to decrease redundant computation.

The Discrete Fourier Transformation (DFT) is defined as

$$\hat{f}(m) = \sum_{n=0}^{N-1} f(n)\ e^{-2\pi i \frac{nm}{N}}, \tag{4.2}$$

where the value $\hat{f}(m)$ represents the complex Fourier coefficient at frequency m in the domain $[0, N-1]$. $f(n)$ is the value of the input sample at position n and N is the number of input samples.

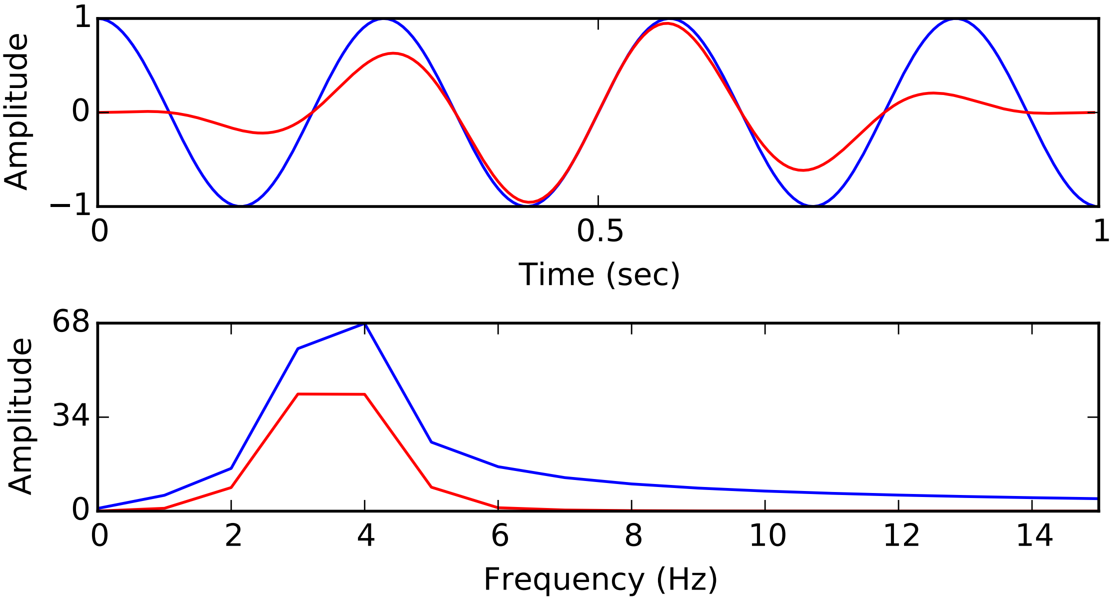

Figure 4.1: Fourier transformation with(red) and without(blue) Hanning window.

Often, in practice, input functions are multiplied with a Hanning (or similar) window to suppress the amplitude at the beginning and end of the input function. If this artifact is not suppressed, a jump in the input signal is present, which in the Fourier domain results in a sinc function. This is visualized in Figure 4.1, where the windowed function produces a narrower frequency band compared to the original.

Fourier transformation (DFT) has been applied to raw OTDR signals to analyze their spectral distribution and apply classification based on the coefficients. Since data are processed in chunks of one second, the signals have been windowed using a Hanning function.

4.2 Short-Time Fourier Transformation

The Fourier coefficients do not encode the original position of the signal. To overcome this, windowed Fourier transformations can be used, also called Short-time Fourier transformations (STFT). The original signal is analyzed in multiple not necessarily overlapping windows. On every window, a Fourier transformation is applied and this way, the amplitude of a given frequency can be connected to a location in the signal. The result of the converted function is one dimensional higher, e.g a 1D signal results in 2D function, often called time-frequency or spectrogram. The Short-time Fourier transformation is defined as

$$\mathbf{STFT}\{f, w\}(\tau, \xi) = \int_{-\infty}^{\infty} f(t) w(t - \tau) e^{-j\xi t} \, dt. \tag{4.3}$$

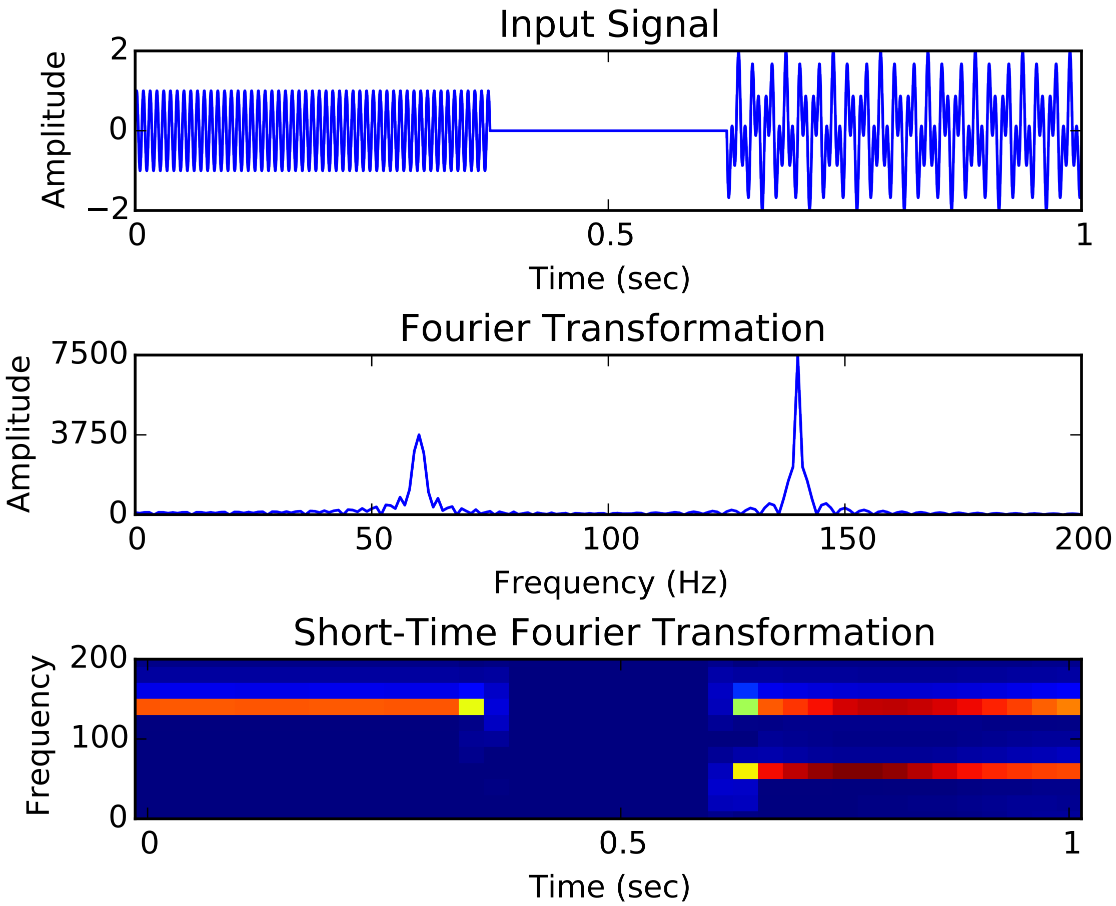

Figure 4.2: Comparison between Fourier and Short-time Fourier transformation.

The input of the STFT is the input function in time domain f and the window function w. τ represents the time coordinate and ξ the frequency in the time-frequency result of the STFT.

But there is a limitation inherent to this method. A big window has only little spatial accuracy, because a long chunk of the signal is examined at once. On the contrary a small window has only low frequency accuracy, figuratively speaking, because there is little data available to precisely measure the frequencies. The precise formulation of this limitation is the Heisenbergs uncertainty principle [Pan87].

A comparison between DFT and STFT can be seen in Figure 4.2. The first part of the input signal consists of one sine function, while the last part is the composition of two sine functions (one with the same frequency as the first one). Applying DFT on the complete signal results in two peaks describing the sine function frequencies, without location information. The STFT shows the frequencies corresponding to the applied window, therefore it can be seen that the second signal consists of two frequencies.

STFT has been applied to extract the spectral distribution of OTDR signals in time-frequency relation to see wagons and their connection points. Size of a transformation window and overlap between windows have to be selected to correspond to the size of a wagon. For the wagon length, we chose an average of 33 meters, which corresponds to one second in our measurement if the train moves at 120 km/h. In order to extract axis information, we defined a window of 1/4 of a second with an overlap of 1/8 of a second. The transformed signal resulted in time-frequency data and allowed to see wagons and their connection points.

4.3 Wavelet Transformation

Just as the Fourier transformation, the wavelet transformation analyzes the frequencies of a signal of infinite length. In contrast to the Fourier transformation, the location of the frequency is also available in the transformed space. The result of a Fourier transform of a 1D signal is also 1D, but the result of a wavelet transform would be 2-dimensional. Although Short-time Fourier transformation also results in a 2-dimensional result describing time and frequency relations, it uses the same subsampling over every frequency, while wavelet transform subsampling is not the same in each frequency band. The continuous wavelet transformation is defined as

$$\mathbf{CWT}\{f,\psi\}(\tau,s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{\infty} f(t)\psi\left(\frac{t-\tau}{s}\right) \mathrm{d}t. \tag{4.4}$$

The equation takes two input parameters compared to one input ξ in the Fourier function from Equation (4.1). The translation parameter τ corresponds to the time in the signal and the scale parameter s to the inverse of the frequency, therefore wavelet transformation uses multiple resolutions to analyze the signal. For low frequencies, a window covers only a narrow frequency band, but a large amount of signal in time. Therefore frequency resolution is high and time resolution is low. For high frequencies, it covers only a small amount of signal in time, but many frequencies. Therefore frequency resolution is low and time resolution is high. ψ is the windowing function, which can be the Haar function or the Mexican Hat function for example, but the actual choice depends on the application.

Wavelet transformation has been applied for blob detection in the classification results. We want to locate trains regardless of their size and signal quality. The Ricker (Mexican Hat) function has been applied, as classification results form Gaussian signals. By applying window functions in multiple scales, we can find the scale which has the highest response for a train.

4.4 Data Binning

Data binning is a form of quantization, where each value of the original data is assigned to a distinct range, a bin, which often represents the central value. Quantization is a procedure, which constrains a set of values to a small discrete set, which helps to reduce the effects of minor observation errors by implicit averaging. This method can be used to reduce the size of the feature vector, which consist of all Fourier coefficients. Each bin contains the sum of the frequency interval that it is assigned to.

It is possible to perform the quantization by means of matrix multiplication. The input matrix $\mathcal{A}$ contains the input samples and has the size (samples × feature values). A quantization matrix $\mathcal{B}$ is defined with the size (feature values × number of bins). By multiplying ($\mathcal{A} \cdot \mathcal{B}$), the result is a matrix of the quantized samples of the size (samples × number of bins). With the help of the matrix multiplication solution it is possible to apply different kind of quantization with the same computational cost. Linear or logarithmic spacing can be used, positions within bins can have different weights or bins can overlap.

CHAPTER 5

Train Detection and Tracking

This chapter details the methods applied for vibration detection and motion tracking. Train profile extraction and segment calibration methods for improving motion tracking are discussed later. The last section describes the GPU implementation for vibration detection.

The functionality of the system is encapsulated into three modules, with the interactions visualized in Figure 5.1. The quality of the signal has a high influence on these modules, therefore it is investigated if a quality measurement for each position of the fiber cable can be extracted. The modules are defined as:

- Vibration Detection
  - Purpose: binary classification of OTDR measurements.
  - Input: one second of raw OTDR signal.
  - Output: binary classification (vibration/background).
- Train Tracking
  - Purpose: deliver properties of each train within monitored section in every second.
  - Input: binary classification results from detection and train profiles.
  - Output: position and speed of each train.
- Train Profile Extraction
  - Purpose: analysis of raw signal to extract unique train descriptors.
  - Input: raw OTDR signal and position of trains.
  - Output: unique signal associated with a train.

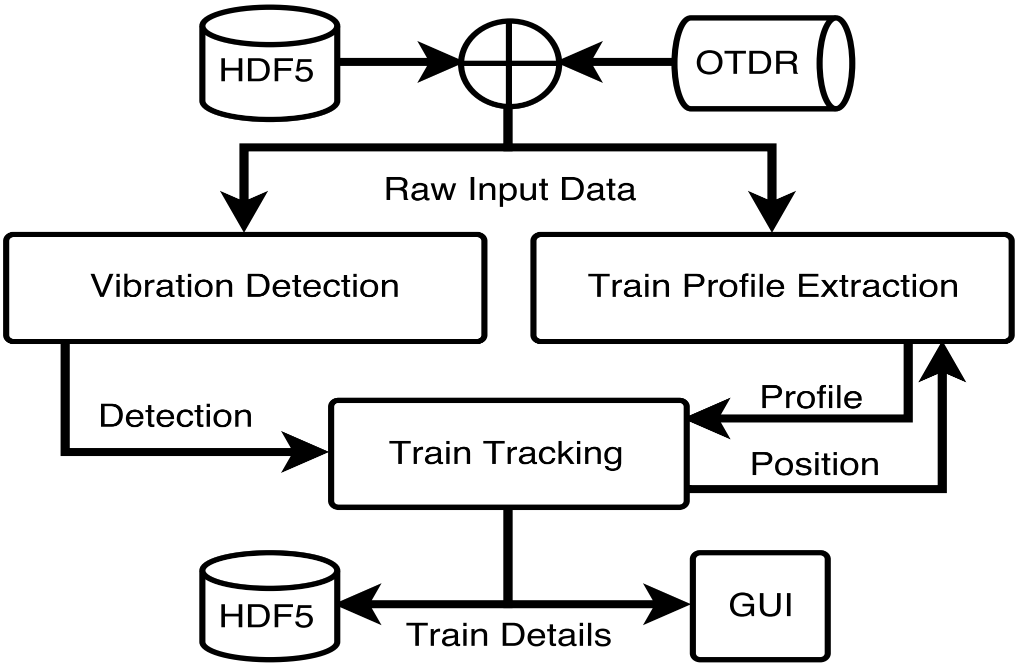

Figure 5.1: The block diagram of the system.

5.1 Vibration Detection

The main task of the vibration detection is to classify points in time and space where vibration caused by trains is present in the cable. The Fourier transform has a typical pattern when vibrations of a train are present, see Figure 5.2 where the top-left signal has high SNR and the top-right has low SNR. For each cable segment, we compute the Fourier transform for each second from the raw OTDR signal $m_{i,j}(l)$ to receive a high dimensional feature vector $d_{i,j}(k)$, where i denotes the cable segment, j denotes the position in time (in seconds), l denotes the sample index and k denotes the frequency channel of the Fourier transform. Figure 5.2 shows that even in cable segments which provide a noisy signal, the typical pattern of train induced vibrations can be extracted.

A feature vector consisting of all Fourier coefficients would have a high dimensionality and many coefficients would be redundant or carry no information about the signal, therefore feature vector reduction is performed by scalar quantization. Quantization is a procedure which constrains a set of values to a small discrete set, which helps to reduce the effects of minor observation errors by implicit averaging. Each Fourier coefficient is assigned to a distinct bin and the reduced feature vector contains the normalized sum of the corresponding coefficients for each bin, where the normalization divider is the sum of all coefficients and the bins are linearly spaced, which is shown in Algorithm 5.1.

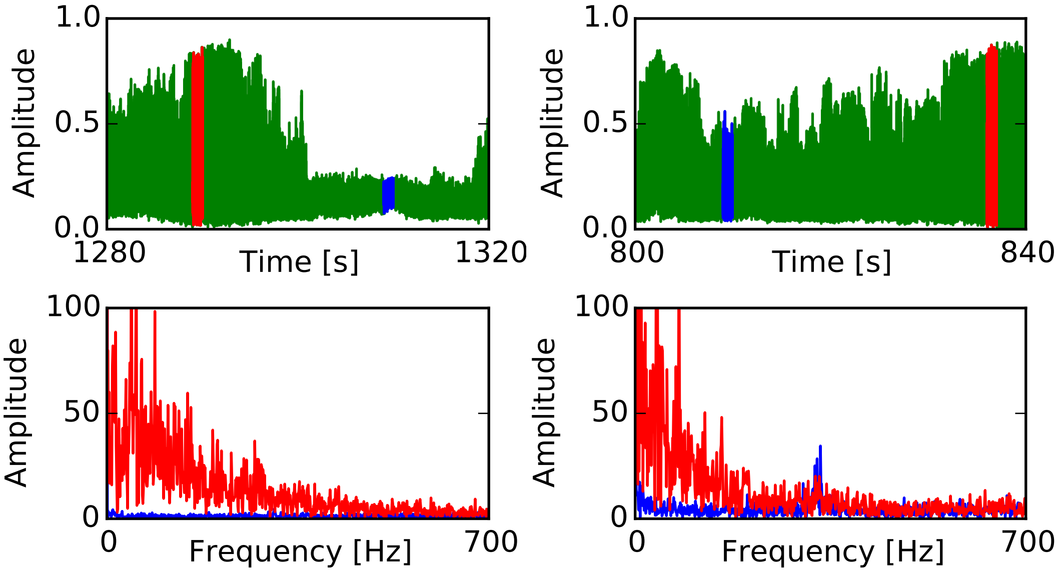

Figure 5.2: Two raw OTDR signals (encoded with green) from two distinct segments. From both segments, two signals of one second have been selected, one for train signal (encoded with red) and one for background signal (encoded with blue). The corresponding DFTs of the selected signals are visualized with the same color in the bottom images.

Principal Component Analysis (PCA) is then used to further reduce the feature vector. The PCA is a well-known method in statistics to analyze the distribution of high-dimensional data in a lower dimension called eigenspace. It performs de-corrleation of linearly correlated data. The eigenvectors describe the directions of the data distribution and the eigenvalues are the lengths of the eigenvectors. Selecting a few eigenvectors with the highest eigenvalues allows to reconstruct the data with a relatively low error rate.

The reduced feature vectors are classified using a Support Vector Machine (SVM). SVM is a supervised statistical learning model that constructs a hyperplane, which can be used for classification. The SVM was trained to make binary decisions between background noise and vibration-signal.

The pseudo code of the vibration detection is shown in Algorithm 5.2. We can write the complete vibration detection process as the composition of the following steps: $r_{i,j} = \mathrm{SVM} \circ \mathrm{PCA} \circ \mathrm{QUANT} \circ \mathcal{F}$. The complexity of the detection is linear in both time (seconds) and space (number of segments), since Algorithm 5.2 is executed for each second independently and the number of iterations of the loop (in Algorithm 5.2) is linearly proportional to the number of segments. Due to the Fourier transformation, it has logarithmic complexity with respect to the sampling rate of the OTDR device.

Algorithm 5.1: Quantization of Fourier coefficients.

```
Input: Fourier coefficients (vector) $\vec{d} = (d_k)$, ($k$-frequency)
Output: Quantized coefficients (vector) $\vec{q} = (q_b)$, ($b$-bin)
$ft1 \leftarrow$ index of first coefficient to take;
$B \leftarrow$ number of bins;
$K \leftarrow$ number of Fourier coefficients;
$N \leftarrow$ number of coefficients per bin;
for $b \leftarrow 1$ to $B$ do
    $q_b = 0$;
    for $k \leftarrow ft1 + (b-1) * N$ to $ft1 + b * N$ do
        $q_b = q_b + d_k$;
    end
end
$norm = \sum_{b=0}^{B} q_b$;
for $b \leftarrow 1$ to $B$ do
    $q_b = q_b / norm$;
end
return $\vec{q} = (q_b)$;
```

Algorithm 5.2: Vibration detection.

```
Input: OTDR samples (matrix) $\mathbf{M} = (m_{il})$, ($i$-segment, $l$-signal index)
Output: Binary decisions (vector) $\vec{r} = (r_i)$, ($i$-segment)
$N \leftarrow$ number of segments;
for $i \leftarrow 1$ to $N$ do
    $\vec{d} = |DFT(\mathbf{M}_i)|$;
    $\vec{q} =$ Quantization,5.1($\vec{d}$);
    $\vec{q} =$ project $\vec{q}$ onto eigenvectors;
    $r_i =$ SVM($\vec{q}$);
end
return $\vec{r} = (r_i)$;
```

5.2 Point-Based Tracking

Our tracking algorithm applies point-based tracking, since the vibration detection module does not extract unique features to apply tracking algorithms like mean-shift or optical flow. The tracking algorithm is divided into two steps, where first the mechanism is described to detect key-points from the binary classification results of the vibration detection. After the key-point detection, the motion tracking algorithm is described.

5.2.1 Key-Point Detection

The task of the key-point detection is to extract positions from the classification results, which can be tracked. A key-point is either the beginning/head or the end/tail of a train. The classification for each second of the original measurements results in a vector of binary classification $r_{i,j}$, where i denotes the cable segment and j denotes the position in time (in seconds), see Figure 5.3a, where two trains are approaching each other. To decrease the false positive and false negative classifications, filters are applied on the classification results. Vibrations are detected in some segments earlier and in some later due to different Signal to Noise Ratio (SNR) across segments, which is the main cause of the false classification. For a given second in time, the classifications of the last 10 seconds are summed along the time axis resulting in a vector describing the number of positive classifications in one cable segment within the time window (Fig. 5.3b). This summation still results in a noisy signal, therefore on this vector, a Gaussian filtering is performed, see 5.3c.

Three different methods were considered to extract key-points for the tracking algorithm. Figure 5.4a shows the filtered classification result, where 4 trains with a high number of positive classifications and one vibration from an unknown object with a low number of positive classifications can be seen.

1. Applying thresholding on this vector leads to a filtered binary classification along space for a given second. Then, the beginning (rising) and end (falling) points of trains are detected by taking the first derivative of the filtered results, which are required by the point based tracking. Figure 5.4b shows the detected points of trains.

2. Gaussian fitting allows to find multiple Gaussians within a signal. The drawback is that the number of Gaussians (number of trains) must be known a-priori. Initial solutions for fitting were defined by finding local maxima within the signal. Figure 5.4c shows the extracted locations of vibrations.

3. Wavelet based blob detection allows to extract locations in the input signal, where a Gaussian can be found. Multiresolution features extracted with wavelet transformation allows to detect Gaussians with different sizes using a Ricker(Mexican hat) wavelet. Figure 5.4d shows responses with different scales from a continuous wavelet transformation. The highest response was found with local maximum suppression both in spatial and scale dimension. From the extracted blobs, the beginning and the end points were extracted and forwarded to the point-based tracking algorithm.

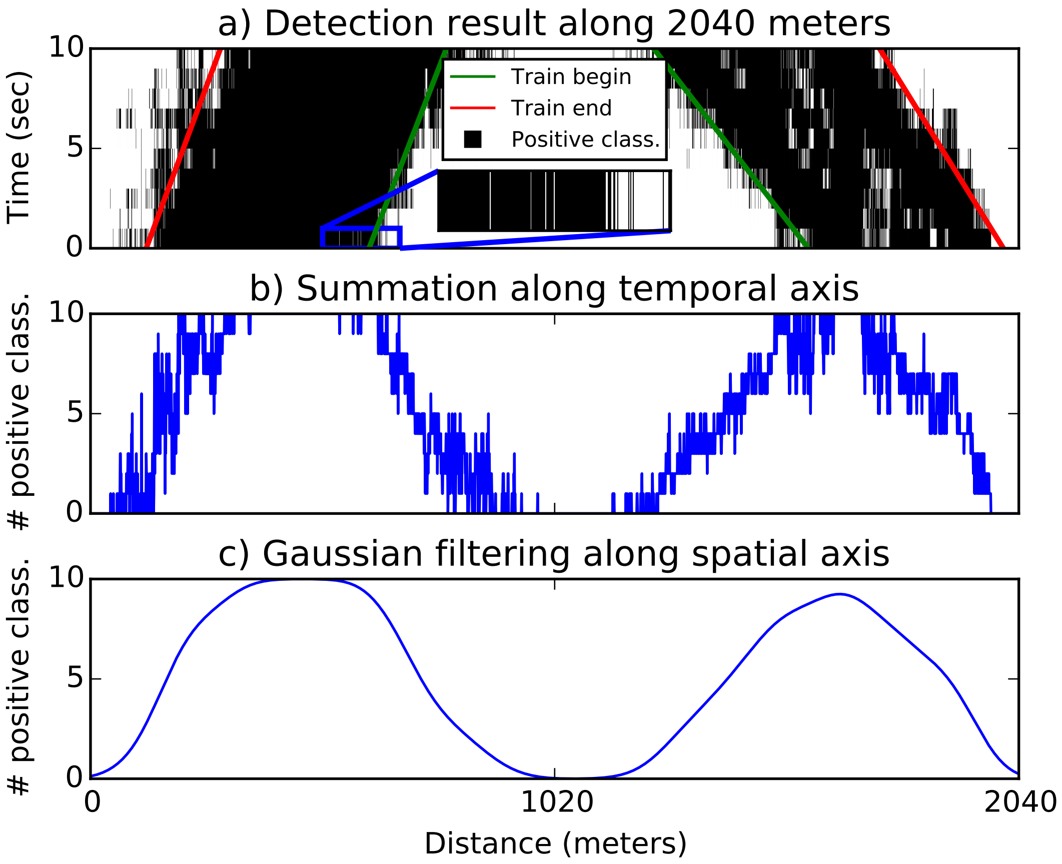

Figure 5.3: Classification filtering for point extraction. The zoom shows that due to high spatial resolution, fine details are missing.

5.2.2 Motion Tracking

The task of the motion tracking is to keep track of train movements every second. First, the beginning and end points of trains are tracked, which are extracted by the key-point detection method. These points can disappear when a train stops or when the trajectories of two trains overlap, therefore a second stage is defined for train tracking. A tracked train is an abstract object. Up to two tracked key-points are assigned to a tracked train representing the beginning and the end of the train. In case both tracked key-points are available, the position of the tracked train is defined as the average position of the tracked key-points. When only one tracked key-point is present (e.g. trajectory overlap), the tracked train relies only on one key-point, which is the main purpose of decoupling beginning and end key-points of trains. When no tracked key-point is available, the train is assumed to have stopped. Since no unique feature values are present for motion tracking, the tracking is solved as a deterministic optimization problem. For the two stages of the tracking, two graphs are defined with two corresponding optimization problems.

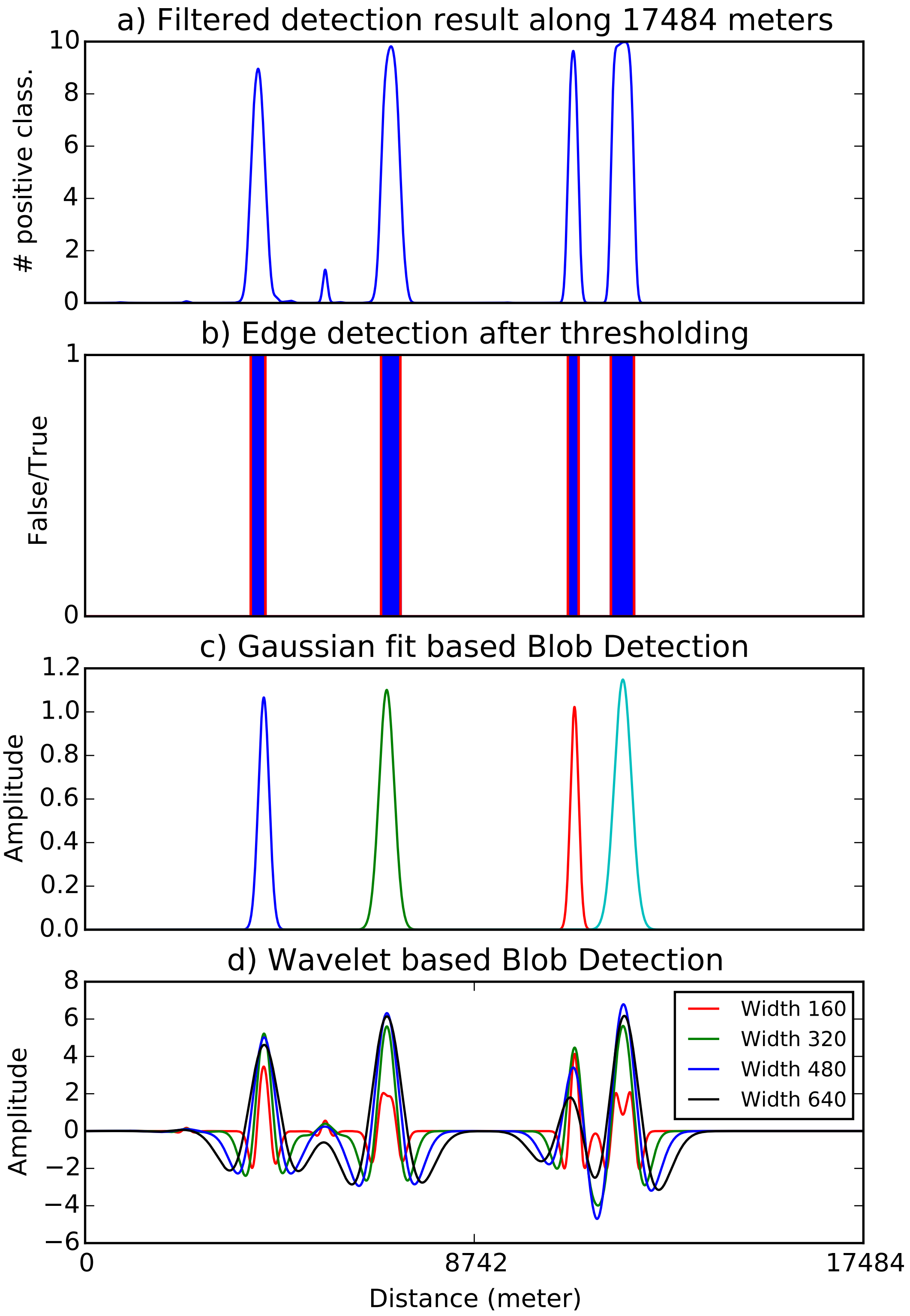

Figure 5.4: Different key-point extraction methods.

The objects of the tracking algorithm are defined in the following vertex sets:

- V_{np} represents the new key-points (np) detected after filtering of the classification results. This vertex set is updated every second.
- V_{tp} represents the tracked key-points (tp). Insertion and deletion in this set can be done every second after solving the optimization problem.
- V_{tt} represents the tracked trains (tt). Insertion and deletion in this set can be done every second after solving the optimization problem.

For tracking in the two stages, the following weighted, undirected complete bipartite graphs are defined, where the edges are denoted as sets $\{v_1, v_2\} = \{v_2, v_1\}$:

- $\mathcal{G}_p(V_p, E_p, w_p)$, where $V_p = V_{np} \cup V_{tp}$, $E_p = V_{np} \times V_{tp}$
- $\mathcal{G}_t(V_t, E_t, w_t)$, where $V_t = V_{tp} \cup V_{tt}$, $E_t = V_{tp} \times V_{tt}$.

To set up the optimization problems we define the following attributes for vertices in the defined vertex sets:

- $u(v) \in \{-1, +1\}$ for $v \in V_{np} \cup V_{tp}$, determines if a point is rising (beginning of train) or falling (end of train).
- $p(v) \in \{0, ..., K\}$ for $v \in V_{np} \cup V_{tp} \cup V_{tt}$, denotes the position (cable segment), where K is the number of cable segments.
- $c_{tp}(v) \in [0, 1]$ for $v \in V_{tp}$, denotes the confidence of a tracked point.
- $c_{tt}(v) \in [0, 1]$ for $v \in V_{tt}$, denotes the confidence of a tracked train.
- $s_{tp}(v) \in \mathbb{R}^+$ for $v \in V_{tp}$, determines the velocity of tracked point.
- $s_{tt}(v) \in \mathbb{R}^+$ for $v \in V_{tt}$, determines the velocity of tracked train.
- $l_{tt}(v) \in \mathbb{R}^+$ for $v \in V_{tt}$, is the length of a tracked train.
- $\Delta_T(v) \in \mathbb{N}^+$ for $v \in V_{tt}$, denotes the number of seconds since last update of tracked train.
- $\Delta(v_1, v_2) = p(v_1) - p(v_2),\ \{v_1, v_2\} \in (V_{np} \times V_{tp}) \cup (V_{tp} \times V_{tt})$, is the distance between two vertices in the same graph.
- $\Delta_E(v) = p(v) + s_{tp}(v),\ v \in V_{tp}$, is the expected next position of the tracked point after one second.

- $\sigma_{\mathcal{G}_p}(v) \subseteq E_p$ is the set of all incident edges for a vertex $v \in V_p$ in graph $\mathcal{G}_p$.
- $\sigma_{\mathcal{G}_t}(v) \subseteq E_t$ is the set of all incident edges for a vertex $v \in V_t$ in graph $\mathcal{G}_t$.

The defined attributes are computed as:

- $c_{tp}(v) \in [0, 1]$ for $v \in V_{tp}$, is computed as the weighted sum of valid observations in time, the weights form a normalized exponential probability distribution function.
- $c_{tt}(v) \in [0, 1]$ for $v \in V_{tt}$, is computed as the weighted average of tracked point confidence observations. It also takes into account the number of rising/falling points.
- $s_{tp}(v) \in \mathbb{R}^+$ for $v \in V_{tp}$, is computed as the slope of the position of the last 10 seconds.
- $s_{tt}(v) \in \mathbb{R}^+$ for $v \in V_{tt}$, is computed as the weighted average of tracked point velocity observations.
- $l_{tt}(v) \in \mathbb{R}^+$ for $v \in V_{tt}$, is computed as the standard deviation of tracked point position observations.

The weights for the edges $e_p \in E_p$ and $e_t \in E_t$ are computed as:

- $w_p(e_p) = w_p(\{v_{np}, v_{tp}\}) = c_{tp}(v_{tp}) * \frac{1}{1+|\Delta(v_{np},v_{tp})-\Delta_E(v_{tp})|} \in [0, 1]$, which denotes the weight of an edge in $\mathcal{G}_p$, where $e_p = \{v_{np}, v_{tp}\} \in E_p$. The weight function has its maximum when the new point is exactly at the position where the tracked point is expected to be. As the distance from the expected position increases, the weight is monotonically decreasing.
- $w_t(e_t) = w_t(\{v_{tp}, v_{tt}\}) = c_{tt}(v_{tt}) * c_{tp}(v_{tp}) * \frac{1}{\Delta_T(v_{tt})*|\Delta(v_{tp},v_{tt})|} \in [0, 1]$, which denotes the weight of an edge in $\mathcal{G}_t$, where $e_t = \{v_{tp}, v_{tt}\} \in E_t$. The weight function has its maximum when the tracked train is at the same position as the tracked point. As the distance between the tracked train and the tracked point increases, the weight is monotonically decreasing. Furthermore, as the time of the last valid tracked train observation is increasing, the weight is monotonically decreasing.

For the two graphs $\mathcal{G}_p$ and $\mathcal{G}_t$, visualized in Figure 5.5 by red rectangles, two separate Binary Integer Programs (BIP) are defined in Table 5.1, where the objective functions are maximized. For each edge e in one of the two defined graphs, we associate a binary variable x_e. For $e \in E_p$, the variable x_e is set to 1 if the corresponding new point is associated with the tracked point. Similarly, for $e \in E_t$, the variable x_e is set to 1 if the corresponding tracked point is associated with the tracked train.

Table 5.1: Binary Integer Programs (BIP) for tracking.

BIP(Point tracking)	BIP(Train tracking)
$\max \sum_{e \in E_p} w_p(e) x_e \quad (5.1)$	$\max \sum_{e \in E_t} w_t(e) x_e \quad (5.6)$
subject to	subject to
$\sum_{e \in \sigma_{\mathcal{G}_p}(v)} x_e \leq 1 \quad \forall v \in V_p \quad (5.2)$	$\sum_{e \in \sigma_{\mathcal{G}_t}(v_{tp})} x_e \leq 1 \quad \forall v_{tp} \in V_{tp} \quad (5.7)$
$x_e u(v_{np}) = x_e u(v_{tp})$ $\forall e = \{v_{np}, v_{tp}\} \in E_p \quad (5.3)$	$x_e c_{te}(v_{tp}) \geq 0.5 x_e$ $\forall e = \{v_{tp}, v_{tt}\} \in E_t \quad (5.8)$
$x_e \operatorname{sgn}(s_{tp}(v_{tp})) = x_e \operatorname{sgn}(\Delta(v_{np}, v_{tp}))$ $\forall e = \{v_{np}, v_{tp}\} \in E_p \quad (5.4)$	$x_e \Delta(v_{tp}, v_{tt}) \leq T_h$ $\forall e = \{v_{tp}, v_{tt}\} \in E_t \quad (5.9)$
$x_e \in \{0, 1\} \quad \forall e \in E_p \quad (5.5)$	$x_e \in \{0, 1\} \quad \forall e \in E_t \quad (5.10)$

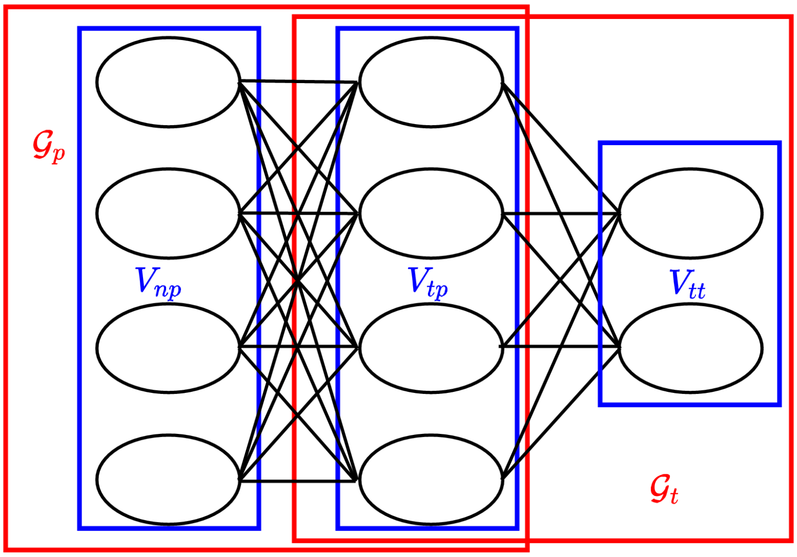

Figure 5.5: The two graphs and their vertex sets for train tracking.

In the following, we will discuss the roles of the constraints in the linear programs:

- Constraint (5.2) ensures that each tracked point v_{tp} and each new point v_{np} can have only one association.
- Constraint (5.3) ensures that rising and falling points $u(v)$ cannot be associated.
- Constraint (5.4) ensures that the sign of $\Delta(e)$ and the tracked point velocity $s_{tp}(v)$ must be the same.
- Constraint (5.7) ensures that each tracked point v_{tp} can only be associated with one tracked train v_{tt}, but multiple tracked points v_{tp} can be assigned to a tracked train v_{tt}. This serves as a noise filter, since points can appear from classification noise and no new trains should be constructed.
- Constraint (5.8) ensures that only tracked points v_{tp} with confidence $c_{tp}(v)$ higher than 0.5 are considered, since false-positive edges often have a confidence lower than 0.5.
- Constraint (5.9) ensures that tracked points v_{tp} with a large distance $\Delta(v_1, v_2)$ cannot be associated. The length of trains on the measured section does not exceed 300 meters, therefore T_h was set to 1500 meters to have a high enough threshold.

The BIPs are solved with greedy heuristics and the result is a list of tracked trains, each with the following attributes: position $p(v)$, confidence $c_{tt}(v)$, velocity $s_{tt}(v)$ and length $l_{tt}(v)$. The complexity of the greedy solution is derived from the complexity of the sorting algorithm in terms of the number of edges in a graph: $O(|E|\log(|E|))$. In our case, in graph $\mathcal{G}_p$ the complexity is $O(|V_{np}| \cdot |V_{tp}| \cdot (\log(|V_{np}|) + \log(|V_{tp}|)))$ and in graph $\mathcal{G}_t$, the complexity is $O(|V_{tp}| \cdot |V_{tt}| \cdot (\log(|V_{tp}|) + \log(|V_{tt}|)))$.

Entry and exit of objects from the measured railway section are handled as:

- detected new points $v_{np} \in V_{np}$ that are not associated to tracked points are considered as new key-points and are promoted to vertex set V_{tp},
- if a tracked point $v_{tp} \in V_{tp}$ is not associated to any tracked train, the confidence of that point is decreased,
- if a tracked point $v_{tp} \in V_{tp}$ reaches the confidence zero, it is removed from vertex set V_{tp},
- tracked points $v_{tp} \in V_{tp}$ that are not associated to tracked trains are considered as new trains to track and are promoted to vertex set V_{tt},
- if a tracked train $v_{tt} \in V_{tt}$ has no association and the position is close either to the beginning or the end of the measurement range, it is removed from vertex set V_{tt}.

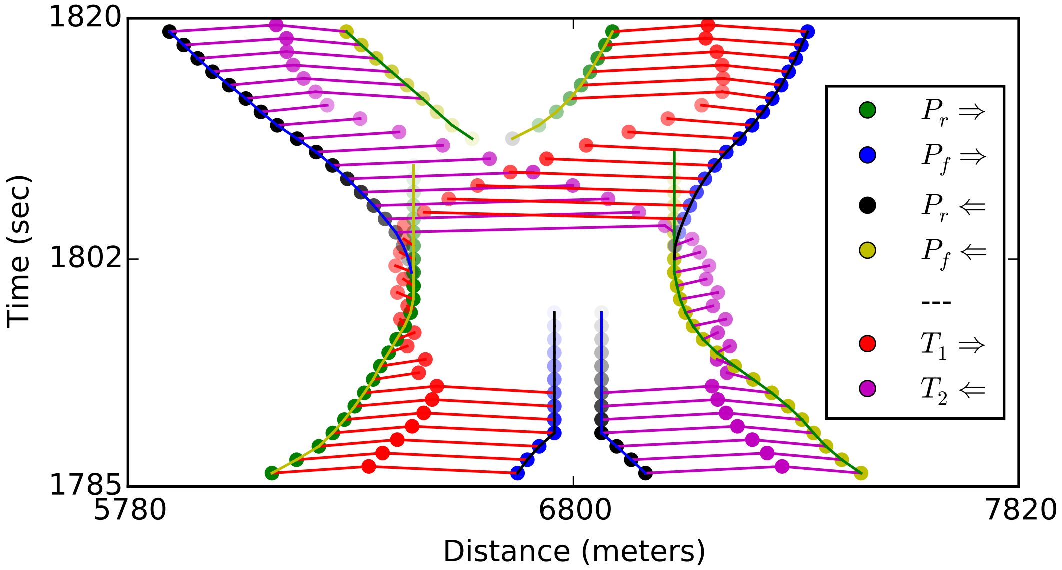

Figure 5.6: Example of train tracking algorithm execution when the trajectories of two trains cross each other. Points (V_{tp}) are denoted as P, the subscript represents rising/falling. Trains (V_{tt}) are denoted as T, the subscript has no effect on the algorithm. Arrows represent the direction of points and trains.

An example of the tracking method is visualized in Figure 5.6, where the graph nodes from $\mathcal{G}_p$ and $\mathcal{G}_t$ can be observed when the trajectories of two trains overlap. After time 1789, the points in the middle are not detectable any more and their confidence is starting to decrease, which is encoded by the transparency of graph nodes in the visualization. When their confidence is lower than 0.5, they are not associated to the trains and the confidence of the trains also decrease. In the middle of the image, the points change direction and new points are inserted to the set of tracked points. When the confidence of these points reaches 0.5, they are associated to the trains. At the top of the image, the points in the middle are tracked again and the confidence of the trains reaches 100% again.

5.3 Train Profile

A train profile is a set of feature values that describe certain characteristics of a train and can be used to uniquely identify trains at different positions along the track. These features will be used to improve tracking as train trajectories can overlap and without unique feature vectors, it is not always possible to achieve correct tracking of trains.

Train profiles were acquired with Short-Time Fourier Transformation on the complete length of a vibration induced by a train. The energy of the Fourier coefficients F, defined as $\sqrt{\sum_i |F(i)|^2}$, ranging between 250 and 750 Hz, were computed as these coefficients describe the conditions of wagon wheels [LWW+16], where faulty axes result in peaks within the signal, and can be used as a train profile, visualized in Figure 5.7.

Train profiles can be extracted horizontally (across multiple segments, at one instant of time) or vertically (along one segment, for a time interval). In order to merge multiple profiles, their time shift due to motion has to be compensated. Figure 5.8 visualizes two profiles extracted horizontally (red, green) and two vertically (blue, yellow) from the same train. The conceptional advantages and disadvantages of the two extraction techniques are:

- Horizontal Extraction:
    - + profile can be extracted for each measurement
    - + requires less memory
    - - varying segment quality
    - - extra length in cable has impact on profile
- Vertical Extraction:
    - + segment quality can be considered constant
    - - profile skew due to speed variance of train
    - - buffer signal to obtain complete profile

In a few segments, it was possible to see the number of wagons using STFT with high certainty. Where the wagons are coupled together, the signal significantly changes, since there is no axis induced vibration, and the frequency coefficients are all close to zero, resulting in local minima, see Figure 5.9. Furthermore, the amplitudes of the higher frequencies describe the condition of the wagon wheels [LWW+16].

5.4 Segment Calibration

Two types of calibration are necessary to be considered:

- Spatial calibration, where each segment of the input data is required to be geopositioned for correct localization of trains.
- Quality calibration, where a value should be defined for each segment that describes the Signal to Noise Ratio(SNR) to assess the reliability of the segment.

Spatial calibration is necessary to determine the position of trains. The cable length does not correspond to railway length and at stations, the cable is often longer. Cables that have been laid often have no description about their exact position within the soil,

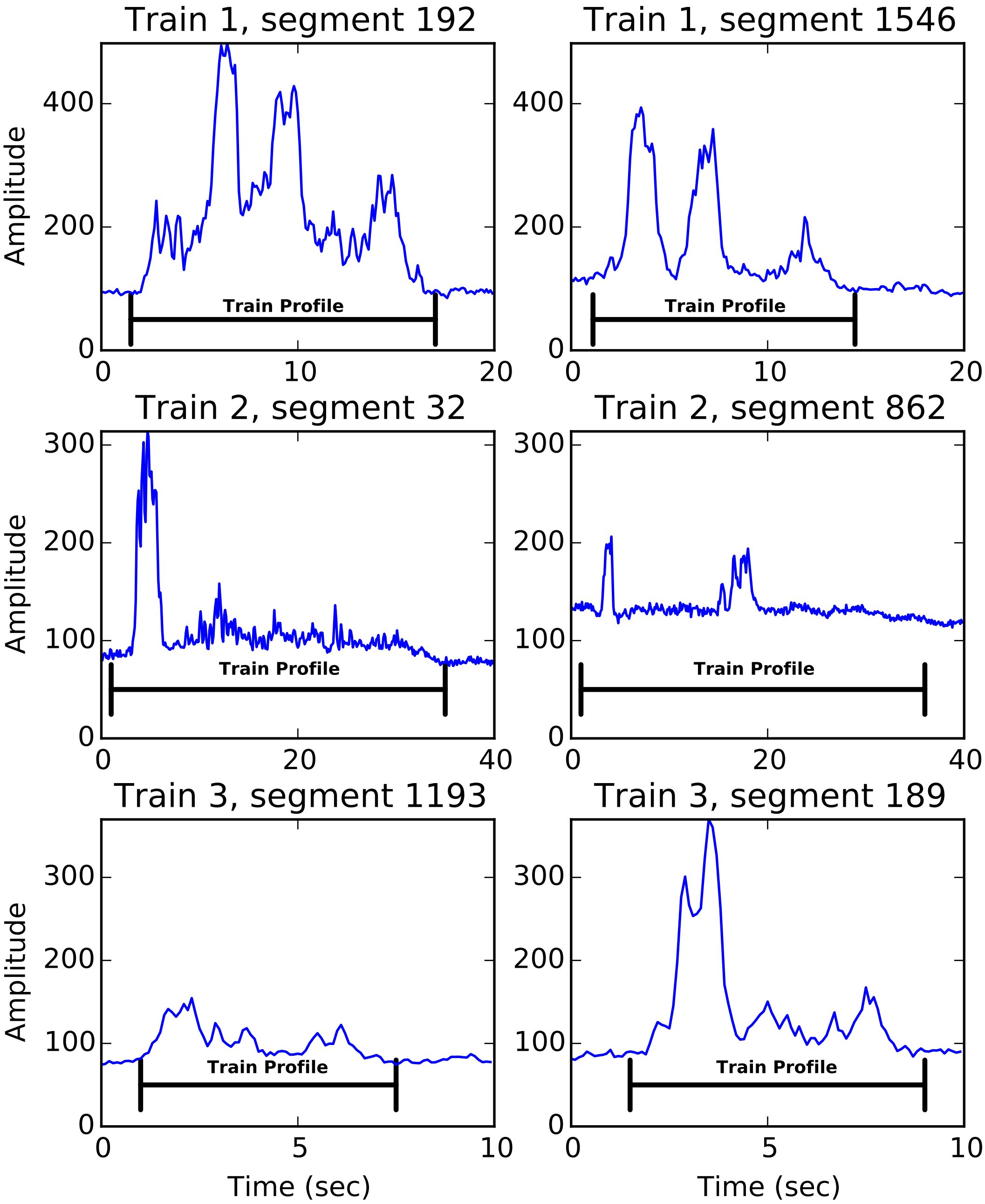

Figure 5.7: Vertically extracted train profiles of 3 trains in 2 different positions.

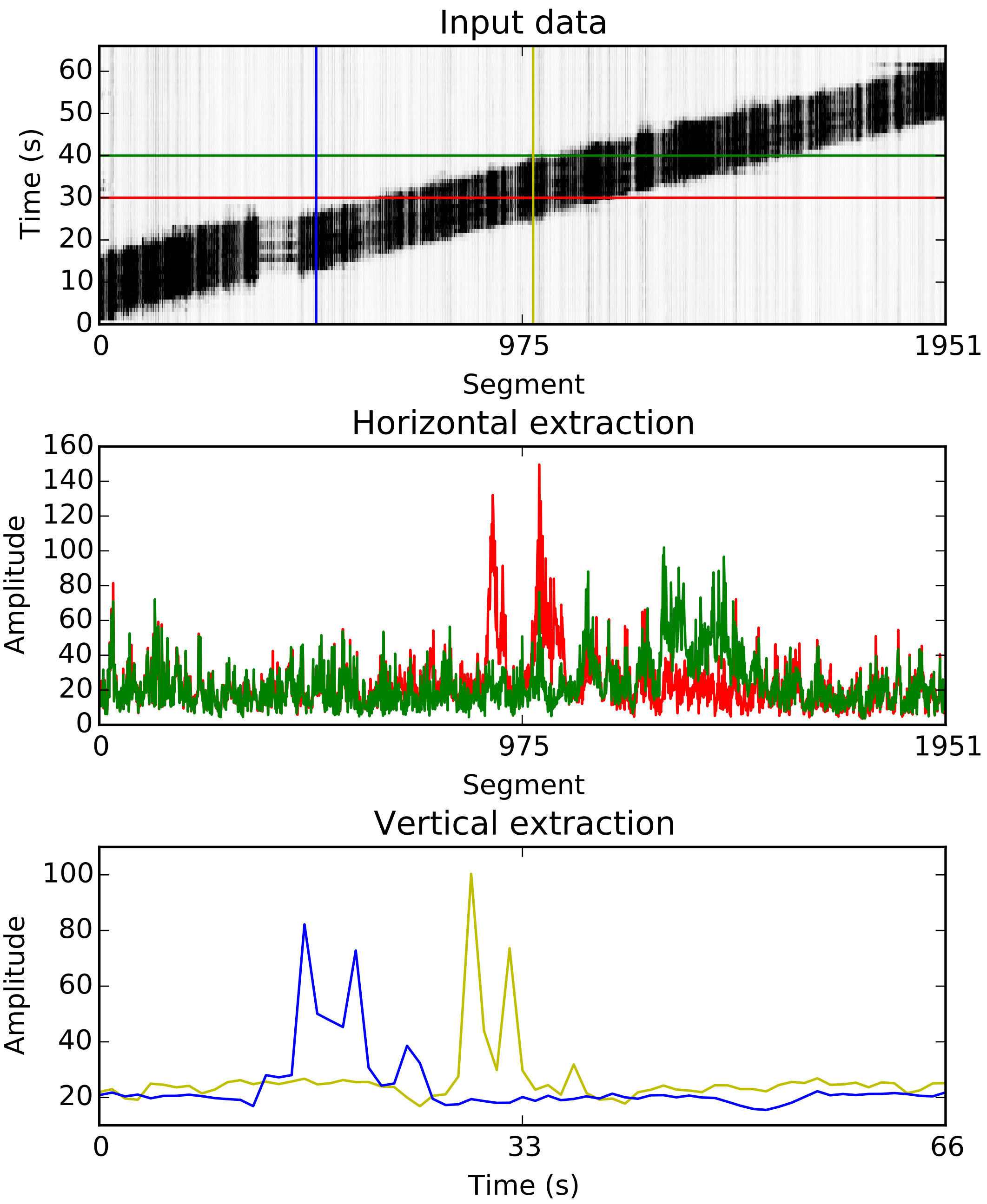

Figure 5.8: Horizontal and vertical profile extraction.

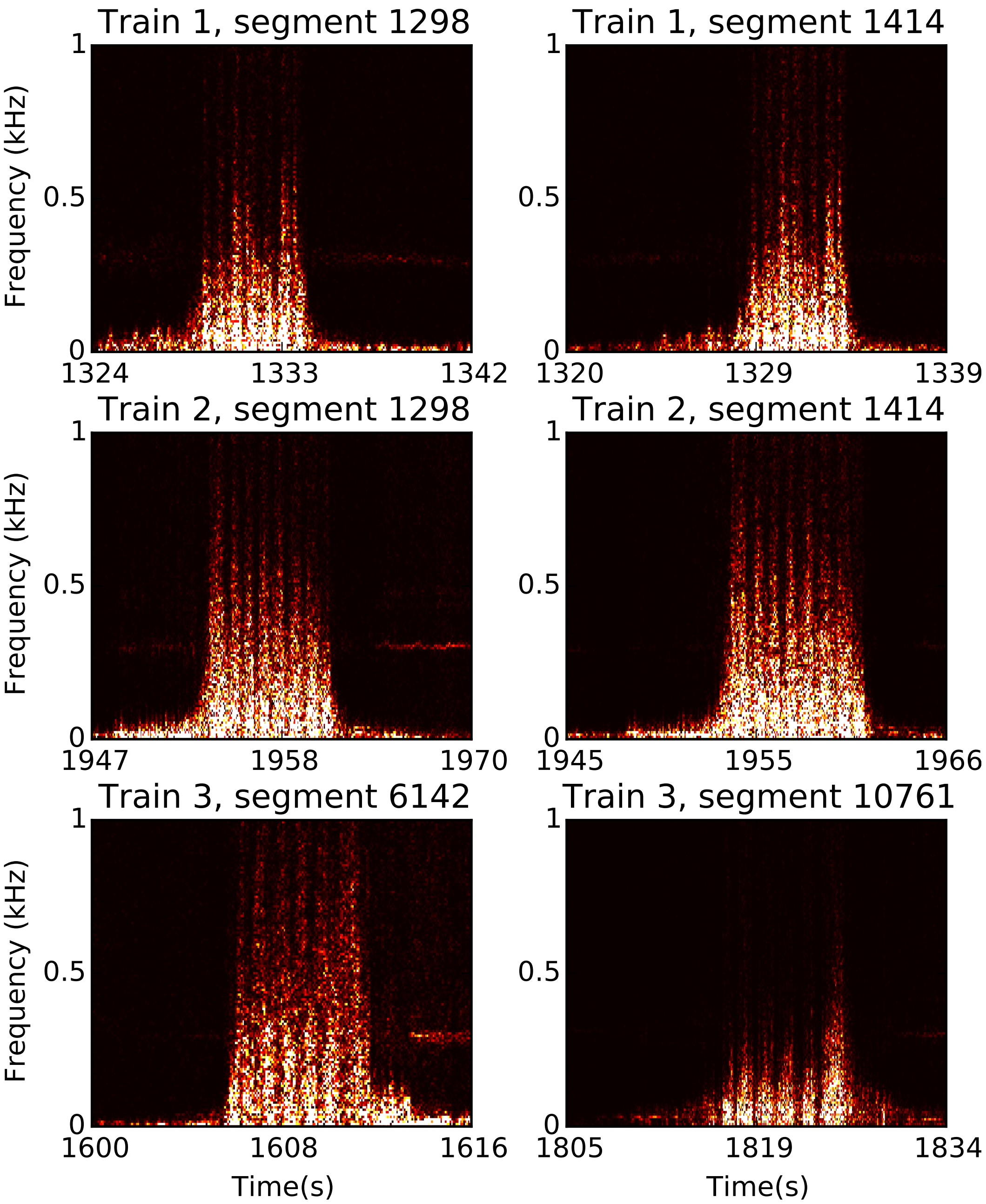

Figure 5.9: Train profiles using STFT, where wagons can be seen from 3 trains in 2 different positions. Colors encode the intensity of a frequency at a given time.

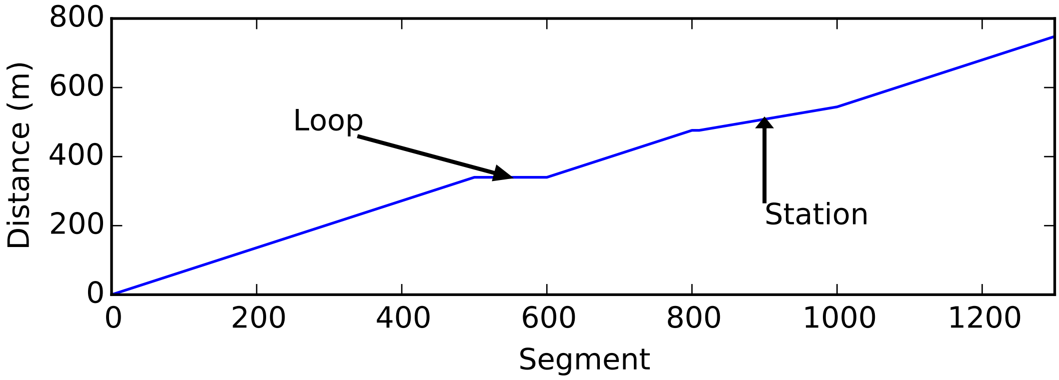

Figure 5.10: An example calibration before a station.

therefore no a-priori information is available. Where loops are present in the cable, trains appear in multiple segments and have seemingly infinite speed. In Figure 2.1, such a loop is marked for visual interpretation. It is also possible that cables are laid back and forth, in which case a train will appear in distinct position at the same time.

In case a train which is equipped with a GPS receiver travels through the measured railway section, a calibration look-up table can be defined. For each second in time, the position of the vibration can be mapped to a GPS coordinate. The error rate of this method will depend on the GPS signal quality and the accuracy of the vibration detection, for each entry within the look-up table. Using standard linear interpolation, a piecewise linear function can be defined to compute the distance from the OTDR device. It is further possible to define control points to fine-tune the calibration and to improve the quality of the tracking module. For example, a loop between s_1 and s_2 can be excluded by setting (s_1, d_1) and (s_2, d_1) as control points. When a fiber cable is longer at a railway station, the slope of the calibration function can also be modified by control points. These examples are visualized in Figure 5.10. In case a cable goes back and forth at the same location, multiple segments at the same location can be excluded from processing.

Quality calibration is necessary to have a-priori information where vibration signals are reliable to extract descriptors of trains. Segment quality depends on constant parameters such as:

- distance between cable and track
- depth of cable in soil
- type of soil and sleepers
- condition of soil (e.g. weather)

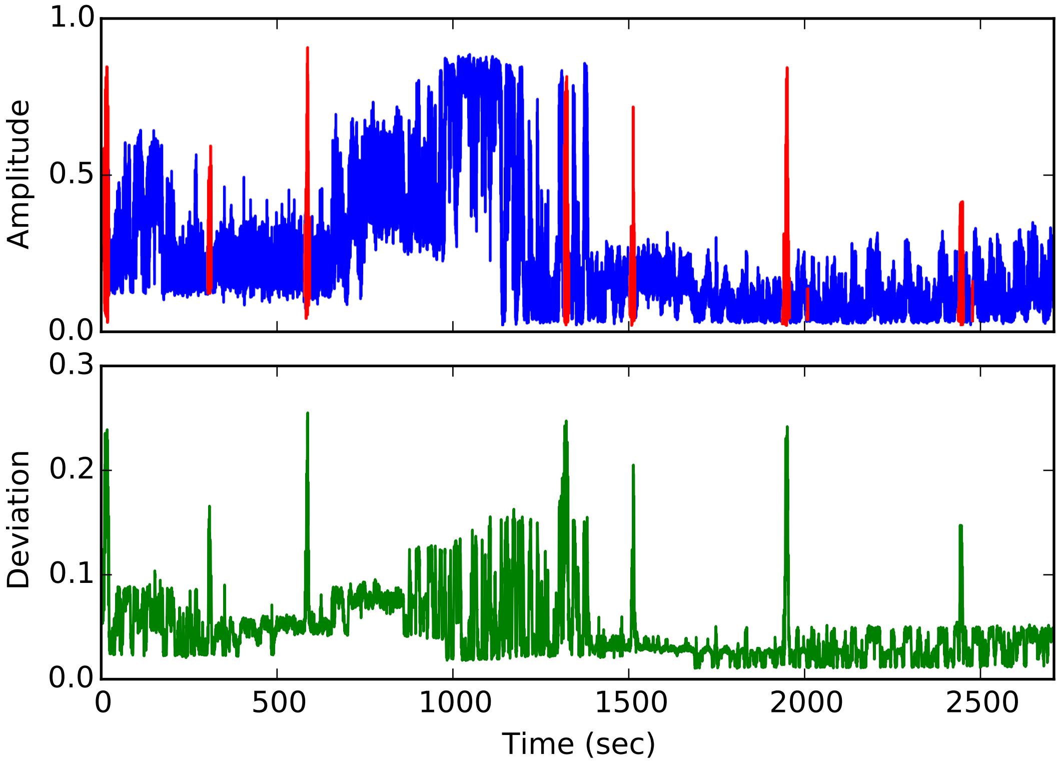

Figure 5.11: Variance of background noise (blue) in OTDR signal through time. Red marked signals are vibration signals.

According to the brochure from OptaSense [Opt14], the best way to lay a fiber cable is to bury it 10 cm deep from the sleeper end. Although these parameters can be considered constant through time, the OTDR device also has a dynamic noise in measurements, which is unpredictable. Figure 5.11 shows that the background noise is not constant and can change unexpectedly. Peaks within the shown signal are vibrations from trains.

5.5 GPU Accelerated Vibration Detection

A Graphical Processing Unit (GPU) is equipped with thousands of cores, where every core executes the same instruction at the same clock cycle. This is most beneficial when large amounts of data are processed, e.g. a matrix multiplication, where a large number of multiplications is followed by additions. The complexity of the vibration detection algorithm for processing data of one second, see Algorithm 5.2, is linear with respect to the number of segments. The OTDR device generates a large amount of data each second, in our case around 320 MB/sec, and the vibration detection method can be performed on each fiber cable segment in parallel. In case the GPU has as many cores as segments, the parallel complexity of the vibration detection can be decreased to O(1).

CUDA provides an off-the-shelf Fast Fourier Transformation (FFT) library called cuFFT. Other operations like quantization and projection onto eigenvectors can be executed as ordinary matrix operations, implemented by cuBLAS (Basic Linear Algebra Subroutines). We have decided to use OpenCV to interface with the routines provided by CUDA, as OpenCV delivers a cleaner and maintainable interface for development.

The pseudo code of the CUDA algorithm can be seen in Algorithm 5.3. The given algorithm computes the feature vectors for classification. Although it would be possible to execute classification for each segment in parallel, no classifier implementation on GPU is available within the OpenCV framework.

Algorithm 5.3: The GPU based vibration detection.

```
Input: OTDR samples (matrix) $\mathbf{M} = (m_{il})$, ($i$-segment, $l$-signal index)
Output: Feature vectors (matrix) $\mathbf{F} = (f_{if})$, ($i$-segment, $f$-feature index)
$B \leftarrow$ number of bins;
$N \leftarrow$ set of segment indices;
$\mathbf{W_{kb}} \leftarrow$ quantization weight matrix;
upload $\mathbf{M}$ to GPU ;
$\mathbf{D_{ik}} = |DFT(\mathbf{M}_i)| \; \forall i \in N$;
$\mathbf{Q_{ib}} = \mathbf{D} \cdot \mathbf{W}$;
$\vec{s_i} = \sum_{b=0}^{B} \mathbf{Q_{ib}} \; \forall i \in N$;
$\mathbf{Q}$ = Normalization,5.4($\mathbf{Q}, \vec{s}$);
$\mathbf{F_i}$ = project $\mathbf{Q_i}$ onto eigenvectors $\forall i \in N$;
download eigen values $\mathbf{F}$ from GPU ;
return $\mathbf{F} = (f_{if})$;
```

For the normalization after quantization, we had to implement our own kernel, see Algorithm 5.4. The feature vector of each segment is normalized in parallel. In the first step, the normalization factor is loaded into the shared memory. Since this factor is required for each feature value, loading into shared memory enables faster execution. As each segment has its own thread and no interaction between threads is performed, no thread synchronization is required. Finally, each feature value is divided by the normalization factor.

It is possible to optimize the division by computing the inverse of the denominator when loading the normalization factor. When the inverse of the denominator is present, the normalization can be performed by multiplication instead of division. Another possible optimization can be when the number of bins within the feature vector is a fixed constant in the implementation, then the for loop can be unrolled. Although these are possible optimizations, the feature vector contains only of a few values and the execution time of the normalization is negligible compared to the complete time required for vibration detection.

Algorithm 5.4: The normalization kernel.

```
Input: Feature vectors (matrix) $\mathbf{Q} = (q_{ib})$, normalization factors (vector) $\vec{s_i}$,
       ($i$-segment, $b$-bins)
Output: Feature vectors (matrix) $\mathbf{Q} = (q_{ib})$, ($i$-segment, $b$-bins)
tx ← local thread id //CUDA id within block ;
tidx ← global thread id //equals to $i$ ;
shared[tx] ← $s(tidx)$ //load to shared memory ;
for $b \leftarrow 1$ to number of bins do
    $q(tidx, b) = q(tidx, b)/shared[tx]$ //normalize ;
end
```

CHAPTER 6

Evaluation

This chapter details the results achieved for vibration detection and motion tracking. It further details results of train profile extraction and segment calibration. The last section describes the results of the GPU implementation for vibration detection. Each section closes with a short summary of results in the last paragraph.

6.1 Vibration Detection

The measurements obtained from a rail track of 13.5 kilometers with a duration of 45 minutes, taken with the first long recording, see Table 2.1 and Figure 2.3, contains a total of around 53 million samples and 4 trains. A sample in this context is an OTDR signal of one second. For training and testing input data, 10000 background and 10000 vibration samples from trains were annotated by hand. To ensure non-overlapping samples, training data were taken from the first half of the dataset and testing data were taken from the second half of the dataset.

The accuracy of the SVM classifier has been evaluated as $\frac{TP+TN}{TP+TN+FP+FN}$, where T stands for true, F stands for false, P stands for positive and N stand for negative. The available frequency interval for the classifier is between 1 Hz and 1000 Hz, as the lowest sampling rate of the OTDR device is 2000 Hz. Frequency intervals from 1, 25, 50, 75, 100 to 200, 300, ..., 900, 1000 with bin counts 5, 10, ..., 35, 40 were evaluated. The classification was done using an SVM classifier. The decision boundary and regions of the SVM classifier using PCA reduction are shown in Figure 6.1. In each row of Table 6.1, the results of different vibration detection configurations can be seen. To reduce the number of rows, we show only the best results of each starting frequency without PCA reduction and with PCA reduction using 2 eigenvectors. Furthermore, the last two rows show results without applying quantization, but only normalization. The first five columns in the table are the results of the classification. The next four are the processing times of each operation and the last three are the parameters of the quantization.

Table 6.1: Accuracy and speed results of different detection parameters.

Accuracy	TP	TN	FP	FN	FFT	Quant	PCA	SVM	Frequency	Bins	Reduction
%	Count				nanosecond/segment				Hz	Count	Type
99.61	9953	9970	30	47	110.5	2.3	0.0	20.5	25-995	10	None
99.60	9950	9970	30	50	111.1	2.5	0.1	15.6	25-995	10	PCA
99.58	9947	9969	31	53	108.7	2.5	0.0	51.0	1-701	25	None
99.50	9934	9967	33	66	108.2	2.3	0.1	27.8	1-701	25	PCA
99.42	9912	9972	28	88	126.1	2.5	0.0	47.9	50-500	10	None
99.39	9905	9974	26	95	117.7	2.2	0.1	33.6	50-500	10	PCA
97.82	9583	9981	19	417	123.7	2.2	0.0	55.6	75-500	10	None
97.72	9564	9979	21	436	118.5	2.3	0.1	45.1	75-500	10	PCA
97.67	9560	9974	26	440	109.4	7.8	0.0	4647	1-701	700	None
97.28	9506	9951	49	494	111.0	8.1	3.6	139.1	1-701	700	PCA

The results show that as the beginning of the frequency interval increases, the number of TP decreases, when quantization is applied. Low frequency vibrations are present in a wider vicinity of a moving train than its high frequency vibrations. Leaving out low frequency bins from classification allows to locate trains with higher positional accuracy. The decline in accuracy without quantization shows that the implicit averaging of quantization is beneficial. Reduction using PCA slightly decreases accuracy, but it has a positive impact on the processing speed and by using two eigenvectors, the data can be reconstructed to 87 %.

Processing data of the duration of one second took around 2 seconds for both recordings. The speed tests of the Python implementation were executed on a single core of an Intel(R) E5-1620 v3 CPU. Note that the FFT was executed on 5000 samples/segment and that quantization included normalization. Table 6.1 shows that applying PCA directly to the Fourier transform is slower than using quantization first. The speed of the SVM classification is not only dependent on the number of feature values, but also on the number of support vectors.

It was experimented whether one second of vibrations can carry information to distinguish between trains. Plotting the extracted features from the vibration detection module and coloring them with different colors by train ID show a large overlap between inter-classes, see Figure 6.2. Finally, the conclusion was made that chunks of train signals are not reliable to make decisions about trains.

The results presented in this section demonstrate that vibration detection in OTDR signals using a frequency analysis method is possible with an accuracy of 99.6%. The results were obtained with 40000 samples, where one sample contained one second of OTDR data and was annotated by hand from 4 trains running over a rail track of 13.5 kilometers.

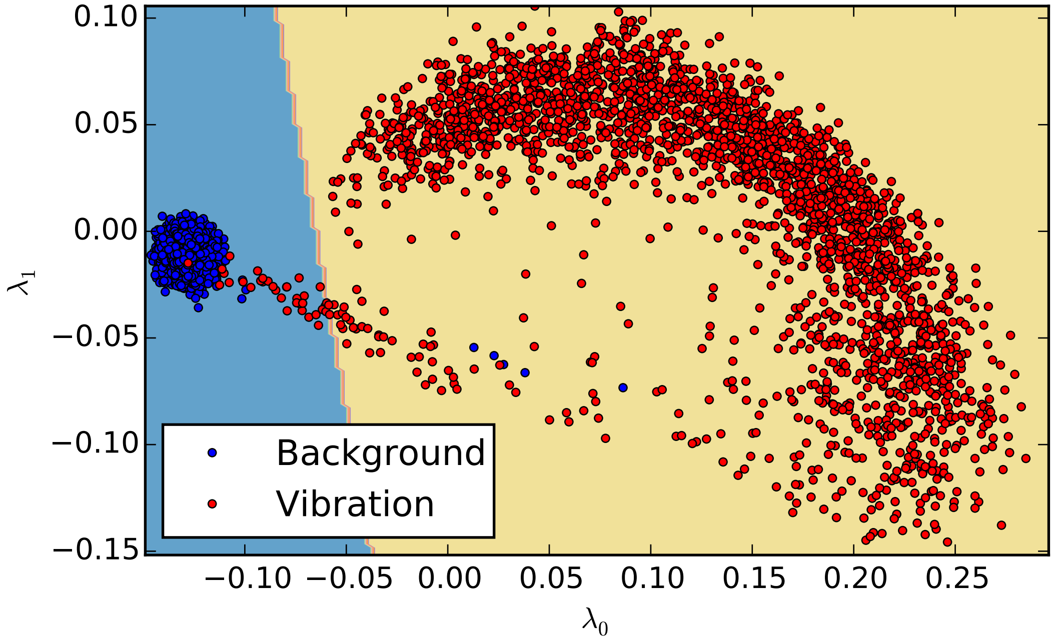

Figure 6.1: Decision regions of vibration and background using two eigenvalues.

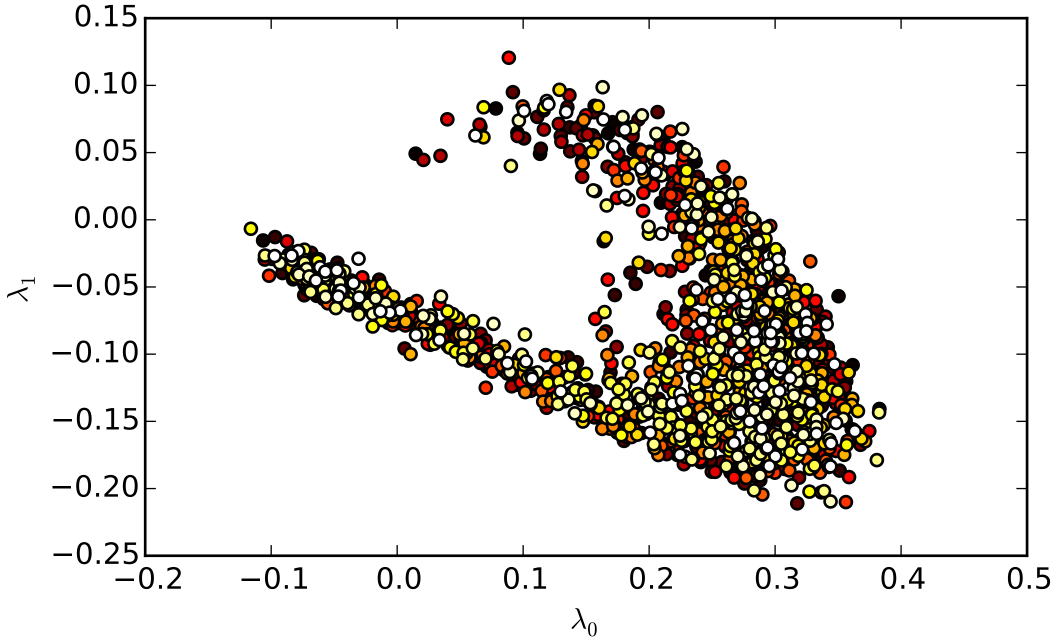

Figure 6.2: Overlapping feature values from multiple trains. Trains are color coded.

Table 6.2: Average of absolute deviation of train length after key-point extraction.

Train ID	23468	2330	73	23508	76	2334	23528	Global
Threshold (meters)	64.37	54.24	49.37	69.22	75.79	57.63	47.97	59.83
Wavelet (meters)	32.75	26.47	26.70	33.85	39.00	35.09	29.00	31.95

6.2 Point-Based Tracking

First, the key-point detection methods are evaluated based on experiments. Second, the motion tracking is tested against ground truth of train trajectories from a conventional train tracking system.

6.2.1 Key-Point Detection

For numerical evaluation of key-point detection, we have compared the distance between two extracted key-points from the corresponding classified vibration signal of trains with the ground truth length from the conventional tracking system. We have performed this evaluation for each train for which the length was available from the received ground truth data. In Table 6.2, we compare the average of absolute deviation of train length extracted using simple thresholding and wavelet based blob detection. The results show that the deviation in every case is smaller when using the wavelet based blob detection, which is attributed to the false positive and false negative classification results due to the different signal quality in different cable segments. We visualize the extracted length from two trains for each second in Figure 6.3 and the distribution of difference between extracted length and ground truth for all trains in Figure 6.4.

From visual evaluation, the wavelet based blob detection proved to be a robust point detection method compared to simple thresholding and polynomial fitting. Simple thresholding did not prove to be robust enough, since different levels of thresholds produced different results. For example, if it is too low, then noise might be detected, but if it is too high, then weak signals of trains might be excluded. Within our experiments, we had to use different thresholds for the two long recordings to obtain correct results. Therefore, this method is not applicable for automatic train tracking.

The biggest disadvantage of the Gaussian fitting is that it is an iterative method and the processing time is not constant, as shown in Figure 6.5. When the trajectories of trains overlap, fitting multiple Gaussians takes more time compared to the case when trains are far away from each other. Furthermore, it often could not find a solution in a reasonable amount of time or iterations and stopped without a solution, which is the reason for the high peak in processing time between input sample 30 and 40. The execution time of the wavelet transformation is constant, as the number of operations does not change when the size of the input signal stays the same.

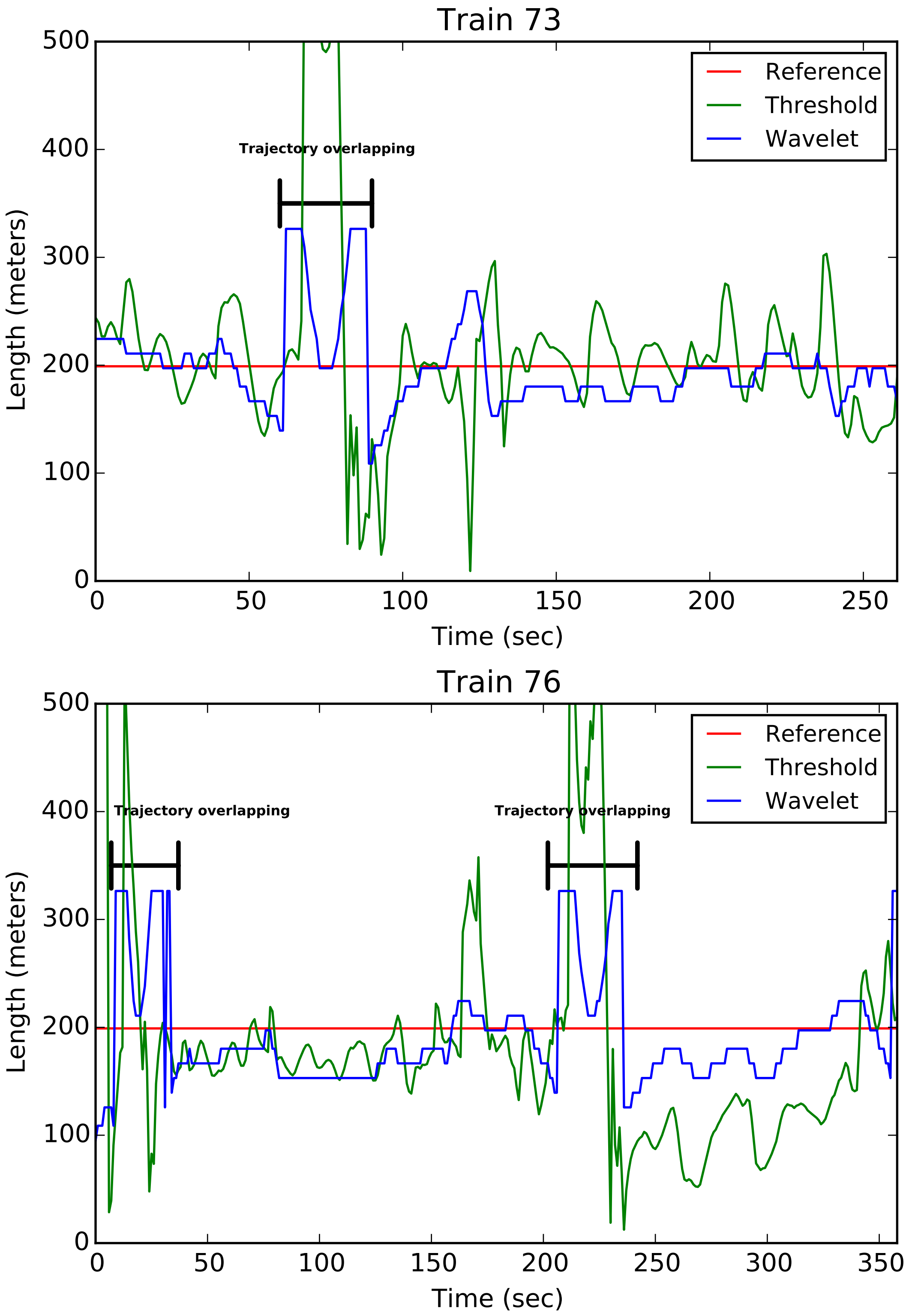

Figure 6.3: Extracted train length after key-point detection.

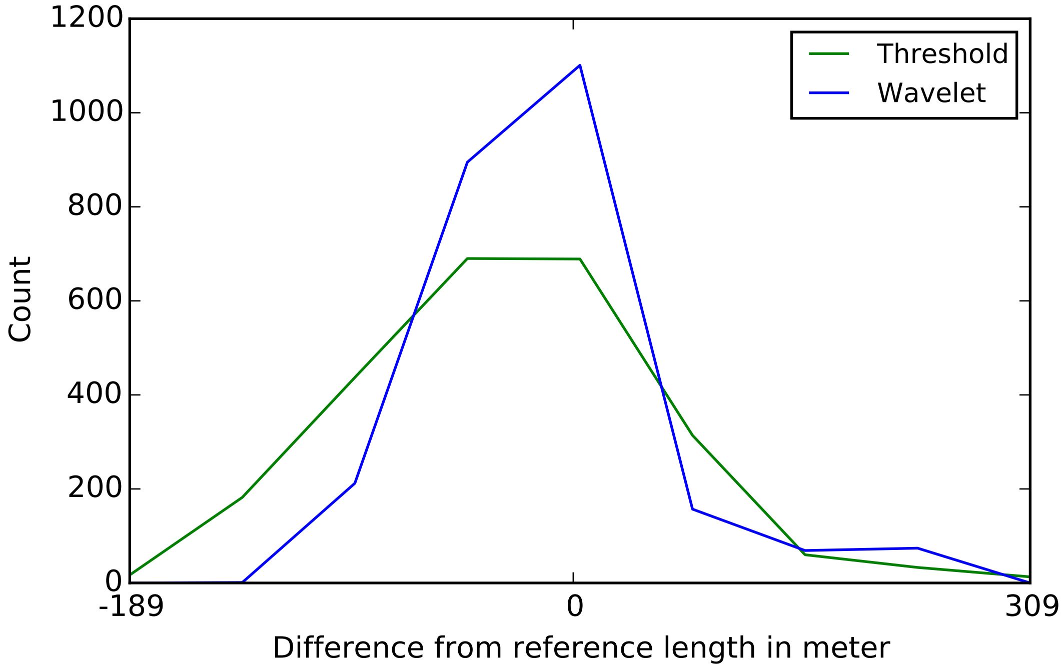

Figure 6.4: Histogram of difference between computed and true train length.

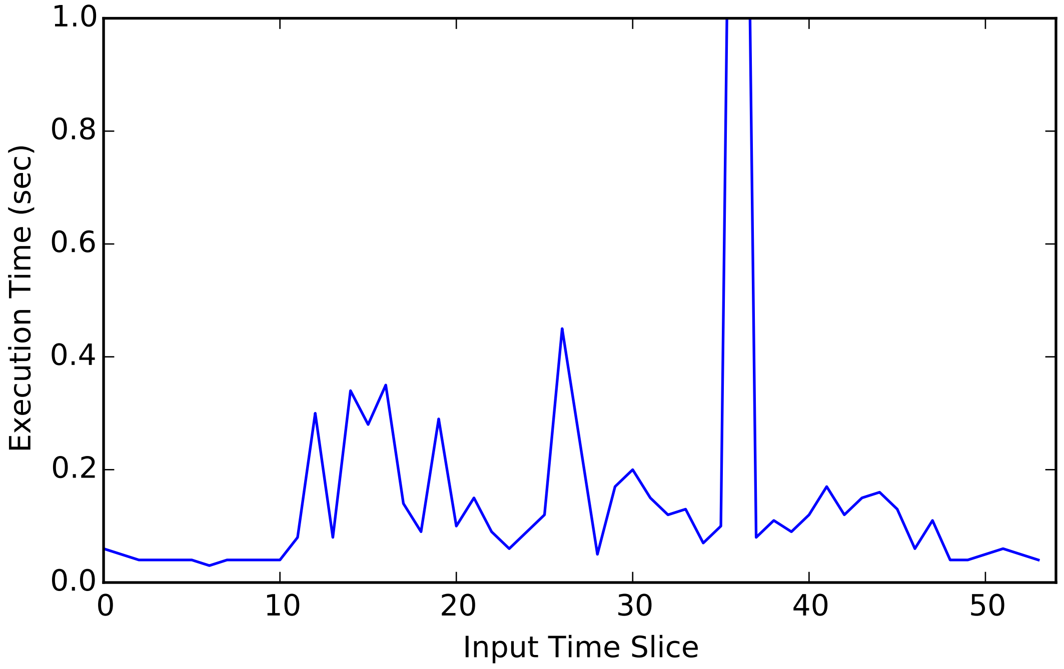

Figure 6.5: Processing time variance of Gaussian fitting.

6.2.2 Motion Tracking

The tracking algorithm was evaluated using cross-validation with ground truth data of 15 trains running over a length of 1.5 km. The detection points of the conventional train tracking system, used to evaluate our tracking, were paired with their correct segment positions in the fiber cable. The exact time offset between the OTDR device and ground truth data was not known, therefore we decided to estimate it using Leave-one-out cross-validation (LOOCV). With the help of LOOCV, a time offset (6.1) was computed from $N-1$ trains, and using this offset, an average of absolute distances (6.2) was computed as the positional error for the excluded train along the 1.5 km length. In both equations, $\vec{x}$ denotes the vector of results from the tracking algorithm, $\vec{y}$ is the vector that contains the ground truth values, $S_T = \{T_1, ..., T_{N-1}\}$ is the set of trains (one train excluded) and M is the number of observations for one train. This technique was applied for all the trains, where the evaluated train was always excluded from the time offset computation. Using the time information from the conventional train tracking signals, we could show a reference speed from the conventional system and compute the speed from our tracking.

$$\Delta_T = \frac{\left| \sum_{T_i \in S_T} \frac{\sum_{j=0}^{M_i} \vec{x}_{ij} - \vec{y}_{ij}}{M_i} \right|}{|N-1|} \tag{6.1}$$

$$\epsilon = \frac{\sum_{j=0}^{M} |\vec{x}_j - (\Delta_T + \vec{y}_j)|}{M} \tag{6.2}$$

Table 6.3 shows the positional and speed errors for each train in separate rows. The first column contains the ids of the trains in the conventional train tracking system. The second and the third columns show the average speed in meters/second of the train between the two reference points where the conventional train tracking signals were installed. The fourth and the fifth columns show the same in kilometers/hour. The sixth column shows the result of the positional error computed with the LOOCV method. From Table 6.3, the conclusion was made that the positional error of the tracking is 20.99 meters. This value was computed by taking the average of the absolute values of each individual train.

Figure 6.6 shows the detection and tracking results. We find it important to point out that all trains were found and tracked correctly. The detection algorithm was not trained to distinguish between cars and trains, therefore a car was tracked by the motion tracking algorithm. We conclude from the visual evaluation that false positive classifications have a more negative impact on the point detection than false negatives. Vertical lines with vibrations can be seen, e.g. at distance around 5000m, which are false positives from other vibrating objects, e.g. railway equipment or cars crossing bridges over the monitored track. The average processing speed of the tracking algorithm for each second of data was around 10 nanoseconds.

Table 6.3: Evaluation of train tracking performance.

Train Id	Ref.	Tracking	Ref.	Tracking	Position Error
	(meters/sec)		(kmeters/hour)		(meters)
23468	36.26	32.61	130.55	117.40	38.20
29515	-31.43	-32.62	-113.14	-117.45	13.13
2330	29.36	26.99	105.69	97.18	38.36
73	-33.33	-34.24	-120.00	-123.25	42.22
23488	28.67	25.53	103.23	91.90	51.79
29535	-33.33	-34.84	-120.00	-125.44	18.12
2337	-32.35	-34.39	-116.47	-123.81	41.19
103	-31.43	-32.42	-113.14	-116.71	33.50
29555	-26.83	-28.19	-96.59	-101.49	20.08
23508	31.62	26.33	113.82	94.79	56.94
90093	-26.19	-27.09	-94.29	-97.52	8.76
76	30.82	28.54	110.97	102.75	42.55
2334	34.25	30.72	123.30	110.58	53.05
29575	-32.35	-33.87	-116.47	-121.92	31.16
23528	30.82	28.78	110.97	103.60	42.95

The data shown in Figure 6.6 include the rail section of a shunting-yard at around kilometer 11 and the tracking of trains is not satisfactory at that location. In Figure 6.7, a shunting train moves at distance 12000m and 2000 seconds; a cargo train enters at 2100 seconds and the vibrations of both trains start overlapping; a passenger train enters at 2200 seconds and between the distances 12000m and 12500m, all three trains overlap; the cargo train leaves the shunting yard after the second 2300 and finally, shunting is continued backwards. Our tracking algorithm is not reliable in such scenarios, as it is not clear which train leaves the overlapping section.

A point-based tracking algorithm has been demonstrated, with a positional error of 21 meters and +10 km/h and -5 km/h in speed. The results were obtained by the evaluation of 15 trains running over a length of 1.5 km and conventional train signaling system was used as the ground truth for the validation of the results.

6.3 Train Profile

The results of horizontal vs vertical extraction methods have shown that horizontal extraction (across multiple segments, at the same time instance) is not applicable due to a high variance between segment quality. In Figure 6.8, we visualize consecutive segments extracted both horizontally and vertically. The cross-correlation between the shifted signals (in the middle) shows that vertical extraction results in more stable descriptors. In horizontal extraction, it can be seen that segments have a strong difference in amplitude, which results in high noise.

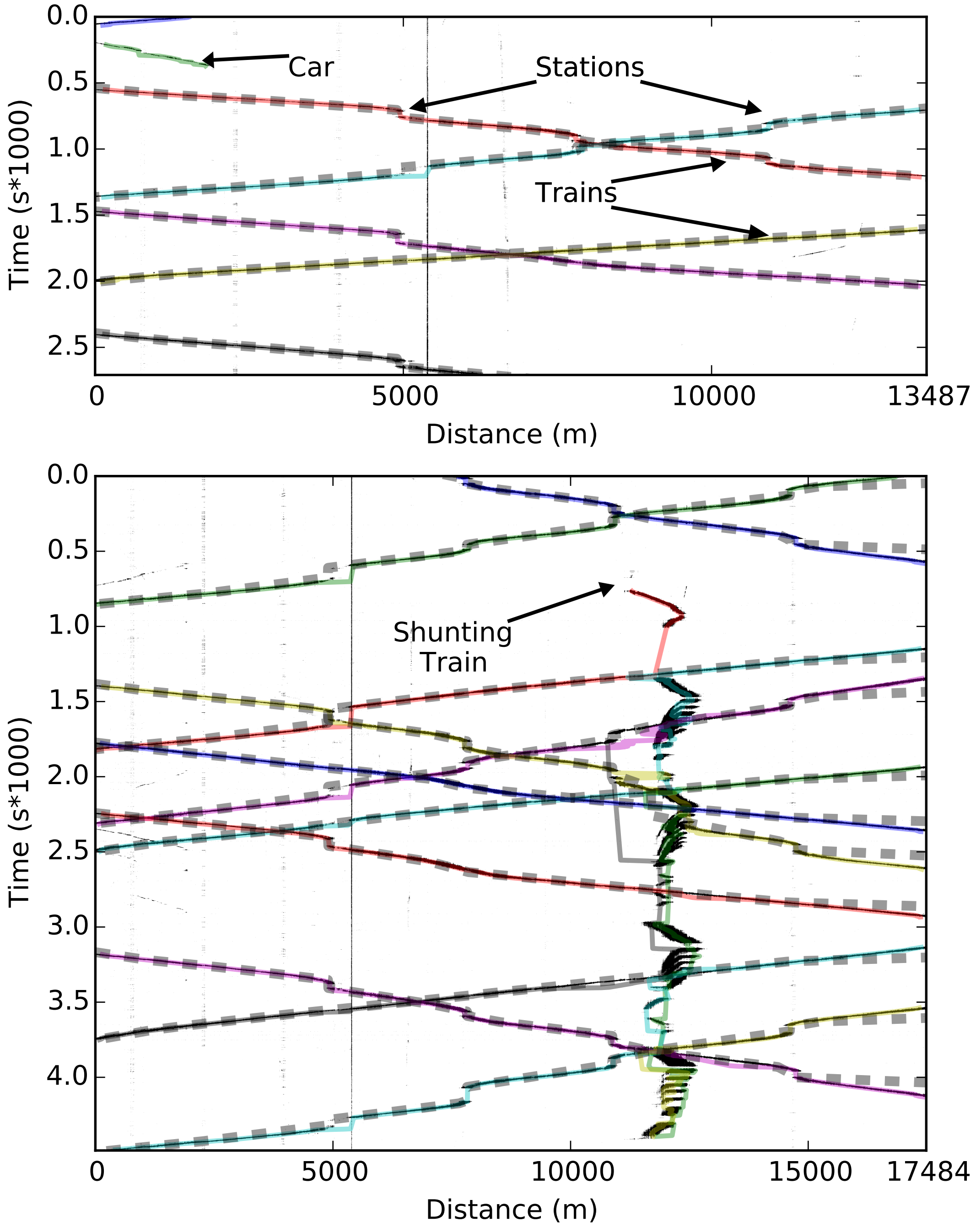

Figure 6.6: Results of the detection and tracking for the two recordings, where each tracked object is encoded with a different line color. The ground truth data are visualized with light gray dashed lines. Underlying the tracked trains and the ground truth tracking, the detection of vibrations is plotted in black.

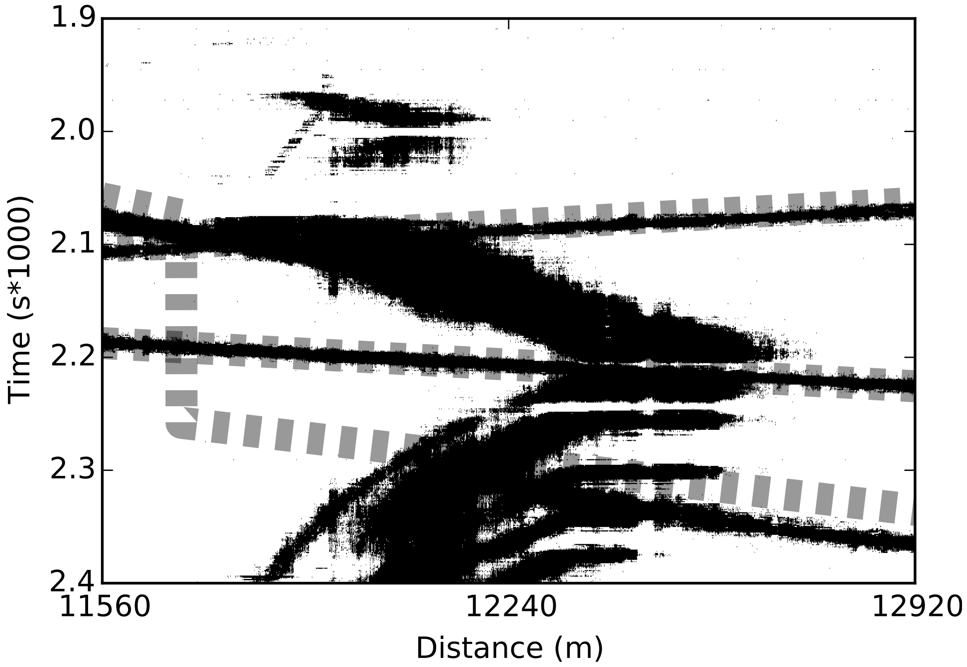

Figure 6.7: Overlap of train trajectory at a shunting yard.

We applied averaging between signals to decrease the effects of observation error, where three different techniques were considered:

- The first method expected the speed of a train a-priori, then shifted each signal according to the given speed. This method is appropriate for testing, but the result of the averaging will highly depend on the computed speed from the motion tracking module.
- The second method searched for the best match between consecutive signals using cross-correlation. This method proved to be inappropriate, since consecutive signals might have big differences in amplitude and shape.
- The third method took multiple speed assumptions. For each of these assumptions, signals were shifted accordingly and cross-correlation was computed pairwise. All these correlation values were summed up and the speed assumption with the highest value was chosen. The advantage of this method is a global registration and it can compare multiple signals at one time.

We saw high difference between consecutive segments in our datasets, which makes it difficult to use averaging of signals in both horizontal or vertical extraction. In case the measurement shows higher stability across consecutive segments, averaging could be beneficial.

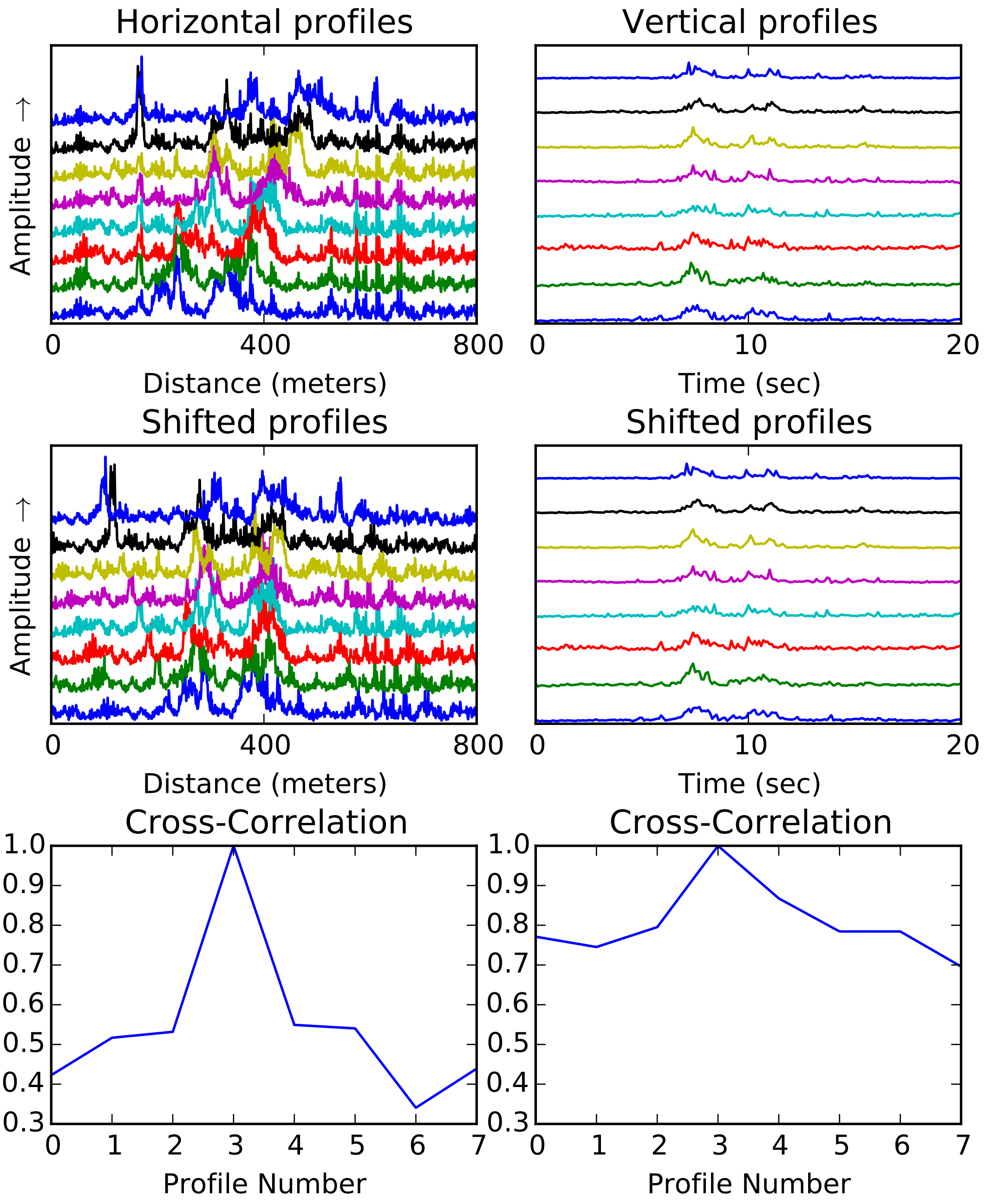

Figure 6.8: Similarity between consecutive train profiles. Note that the consecutive profiles have been shifted in amplitude for better visualization.

Table 6.4: Confusion matrix of train profile cross-correlation results.

No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	**1768**	463	297	106	504	223	131	541	137	183	153	111	327	56
2	590	**1686**	715	477	310	200	38	82	497	116	40	97	132	20
3	744	1250	**966**	151	361	143	181	311	164	220	96	64	239	110
4	126	296	37	**1698**	134	109	34	74	1000	211	153	761	354	13
5	891	462	224	289	**1158**	693	76	308	275	193	128	106	175	22
6	644	380	186	351	947	**879**	128	330	327	254	131	141	209	93
7	567	89	206	163	298	182	**708**	932	208	468	274	189	497	219
8	893	97	119	78	373	208	401	**1243**	161	387	272	136	487	145
9	237	334	68	1194	138	112	95	182	**1352**	243	113	537	316	79
10	537	121	186	310	276	170	284	599	318	**613**	306	302	599	379
11	221	22	36	201	80	52	92	244	116	157	**1196**	1208	963	412
12	68	30	10	572	33	32	32	51	324	113	778	**1859**	899	199
13	359	52	64	272	110	49	133	338	169	240	759	1073	**1041**	341
14	240	47	117	57	64	44	251	346	118	412	831	562	812	**1099**

A test case was applied with profiles extracted from single segments vertically. We randomly selected 5000 pairs of segments for comparison. For a pair of two segments, we extracted all profiles from the short recordings, corresponding to 14 different trains. From the first segment in each step, we selected one train and compared it to all profiles in the second segment. The highest correlation was selected as the correct profile correspondence. The results can be seen in Table 6.4. It can be seen that train profiles cannot be extracted reliably to randomly find corresponding profiles in segments.

A strategy was applied to detect positions in segments where wagons were coupled together by computing the minima and maxima of several frequency ranges in multiple time resolution. These simple features were then thresholded manually to find more segments where wagons could be seen. The sensitivity of this process, which was computed as $\frac{TP}{TP+FN}$, was around 20%, but manually, it was possible to find a few segments where wagons could be seen. For the computation of the sensitivity, we selected 100 segments, where we manually observed the results of the thresholds. The reason for the low result is the high difference of amplitude between segments and trains, therefore this method is not applicable for automatic wagon coupling detection.

With the help of a sufficient number of input samples, it could be possible to train a learning algorithm to automatically detect positions where wagons are coupled. The solution could be to use multiresolution analysis of the complete train signals acquired from STFT to extract feature values. Since annotation of the data is highly cumbersome, clustering techniques (e.g. autoencoder) could be implemented. This method would allow to extract the number of wagons of a train and high frequency vibrations would describe the condition of each wagon. We saw a high variance in amplitude, width and frequency distribution of the signals between different segments, which made it difficult to collect data for a classifier. According to OptaSense [Opt14], the optimal distance between the fiber cable and the railway track is 10 cm. The fiber cable used in these experiments is a conventional optical communication cable installed along the rail track in typically several meters distance from the railway track.

Although it is reasonable to say that feature vector extraction for train description is possible, it is clear that it would not solve correspondence for tracking in all cases. If two trains of the same type are moving in the same direction with the same number and condition of the wagons, they could not be distinguished. Additionally, train timetable information could be merged with the tracking algorithm and the tracking at railway stations could be improved. Often, trains are rescheduled due to delays and it is still an open topic for researchers how to solve this efficiently in real-time.

Within this section, train profile extraction was investigated. Train identification based on the extracted profiles was demonstrated with an accuracy of 25%. The low accuracy is attributed to the installation position of the fiber cable, where the sensing cable is often several meters far from the railway track, which is unsuitable to extract reliable profiles.

6.4 Segment Calibration

A method was investigated, which is a standard way to compute Signal to Noise Ratio (SNR) with the equation $\text{SNR} = \frac{P_{\text{signal}}}{P_{\text{noise}}}$. For the computation, the results of the detection module were used, where the last 10 measurements of vibration and the last 10 measurements of background were used. This method delivered a feature value for segments, which described the amplitude difference between vibration and background noise. Since the OTDR device has a dynamic background noise, which varies over time, see Figure 5.11, the computed SNR does not allow to categorize segments based on their quality.

As described in the results of the train profile module, the segments where profile can be extracted are not necessarily the ones where the amplitude is high enough. The signal that we are looking for must show amplitude differences between wagons and wagon connections. Figure 6.9 illustrates a signal with a low SNR containing information about wagons, while from the one with the higher SNR, only blurry information can be extracted. With sufficiently large train data, it would be possible to detect such signals, therefore the segment quality measurement could be implicitly defined and therefore it would not be necessary to calibrate segments by their quality.

The investigation of the SNR computation within OTDR signals shows that the deviation of the signal sampled in one segment over 2700 seconds can change from 0.05 to 0.15 within seconds, which did not allow to automatically categorize the cable for an initial configuration of the detection/tracking algorithms.

6.5 GPU Accelerated Vibration Detection

The GPU implementation is written in C++ using CUDA and OpenCV. Tests were executed on an NVIDIA Quadro K4200 GPU. The computational speed of the single core Python implementation on CPU can be compared, since Numpy library implements the functions using C++.

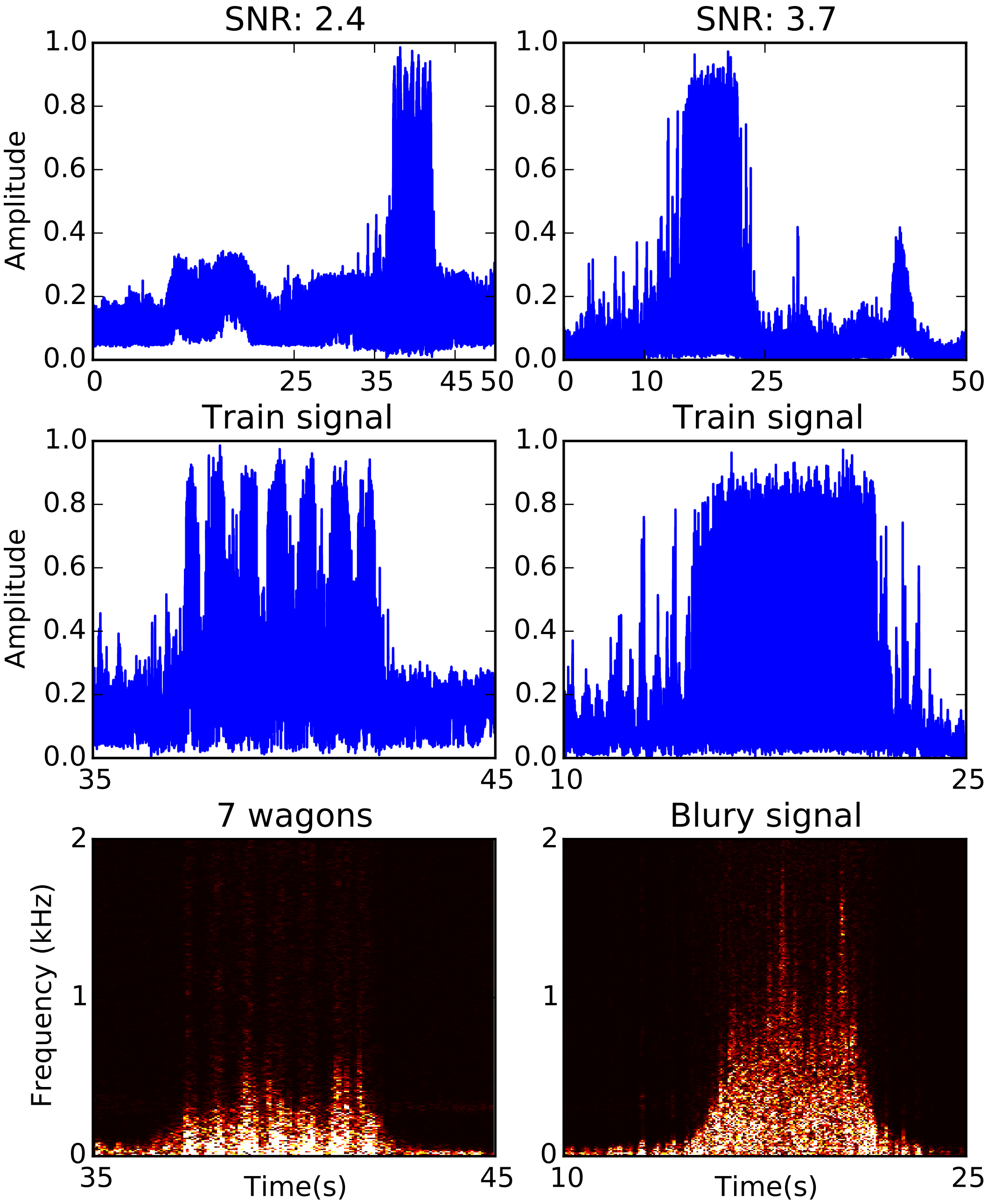

Figure 6.9: Two signals, where wagons can be extracted from the signal with a lower SNR.

Table 6.5: Execution time in milliseconds of each operation using CUDA.

Upload	FFT	Quant	PCA	Download
110,9 ms	36,41 ms	5,953 ms	0,268 ms	0,105 ms

Fourier coefficients between 25 and 995 Hz were selected with PCA reduction to correspond to the best result in Table 6.1. The implementation is able to process the input data of one second (around 320 MB) and compute two eigenvalues for each segment in around 120 milliseconds, which is faster than 2 seconds compared to the single core Python implementation. Table 6.5 shows the execution time of each operation, without OpenCV function calls and device synchronizations. It must be noted that we did not implement classification on the GPU, but the given results show the power of GPU acceleration.

Although the uploading time of the data is longer than the processing time, it must be noted that in an online processing case, this would also be present if the data was processed on the CPU. The data from the OTDR device would be received by a network interface and it could be transferred directly to the GPU. GPUDirect Remote Direct Memory Access (RDMA) is a technology that enables a direct path for data exchange between the GPU and a third-party PCI Express device like network interfaces or video acquisition devices. By transferring the data directly to the GPU, the Central Processing Unit (CPU) could be left out from the feature extraction process and that would allow the CPU to perform other operations. Theoretically, the transfer speed between a network interface connected to a PCI-Express slot is the same between the memory and the GPU, therefore, the data transfer time has to be considered as a latency in the processing chain.

The speed results of the CUDA implementation showed that feature values from OTDR signals for vibration detection can be computed in real-time on a conventional workstation for a segment resolution of 0.68 m over a rail track length of 13.5 km. Our test case contained 19825 segments sampled with 4 kHz, resulting in around 320 MB/sec, and feature values were computed in around 120 ms.

CHAPTER 7

Conclusion

This thesis has investigated the potential of an Optical Time Domain Reflectometry (OTDR) device for train tracking for the use in railway safety. The OTDR signal data used to validate this research were obtained using a conventional optical communication fiber cable installed along the rail track in typically several meters distance from the monitored track. The monitored railway section had a length of up to 17.5 km, with a spatial resolution of 0.68 meters, which corresponds to 25712 segments in the fiber cable.

A novel method, based on normalized spectral distribution, was proposed to extract feature vectors from OTDR signals to detect train induced ground vibrations. The algorithm is independent of the signal energy and classifies based on the shape of the spectral distribution only. From the given datasets, train vibration signals were classified with an accuracy greater than 99.6 %. The algorithm handles the raw data in chunks of one second and processes it in real-time (120 ms) with GPGPU acceleration.

Based on the classification, we have demonstrated that it is possible to automatically track trains with a positional error of 20.99 meters. Evaluation was performed using ground truth data obtained from a conventional train tracking system. To our knowledge, this is the first algorithm that is able to reliably track trains automatically with trajectory crossings (i.e. train encounters with opposite direction of motion on adjacent tracks) over a long period of time. Train tracking showed to be stable and reliable on the open track with train crossings when the optical fiber picked up the vibrations of passing trains. The overlapping trains at a shunting yard led to errors in the tracking algorithm, therefore these cases have been investigated in more detail.

Methods were experimented to extract train profiles, i.e. unique train descriptors, from the spectral distribution of the OTDR signal. It was possible to obtain signals which were able to derive the number of wagons. Profiles extracted from 14 trains were compared at different locations along the track and the accuracy to match corresponding profiles was 25%. The relatively low accuracy is attributed to an unfavorable installation position of the fiber cable, having a large distance of several meters from the monitored track. The calibration of a fiber cable segment quality was not found to be possible, since the deviation of the signal amplitude sampled in one segment over 2700 seconds can change from 0.05 to 0.15 within seconds.

7.1 Future Work and Outlook

The most important task for the future will be to collect more data. The two hour long recordings were taken in very homogeneous environmental conditions. Therefore it would be important to collect data with a variation of different parameters (e.g. temperature, foundation of the railway track) to study their influence on the signal.

To efficiently collect more data, it seems promising to only store the parts of the signals that contain the vibration information. This would lead to a significant reduction in data size, as our current data only contain 3.5% of vibration signals. The vibration detection presented in this work can be used to decide whether a signal should be stored or not.

With the help of a train control simulator (e.g. DEDOTS [oB16]), synthetic train trajectories can be obtained that can be used to evaluate and improve the tracking algorithm. The simulator could generate large enough datasets to train predictive models for tracking. Furthermore, a simulator could generate more complex train trajectories than the ones which were evaluated in this thesis.

It is still an open question how this device and algorithm can be applied in the field. Therefore it will be an important task to get input from rail operators to get details of the exact requirements for this system to be applied. As previous works show, OTDR signals cannot only be used for train tracking, but also for condition monitoring of wagon wheels or rock fall detection for example. We see potential in our solution as a supplement for existing safety devices.

List of Figures

1.1 Comparison between fixed block and moving block in railway safety. Image taken from [Isr11] 2

2.1 Description of an OTDR data. 6
2.2 Illustration of the monitored railway section. 7
2.3 The input datasets, where each second of measurement is aggregated into one pixel. The pixel intensity is the energy of frequency coefficients between 50 and 150 Hz. 8
2.4 Illustration of the calibration of the track, where two extra lengths are shown. 9

4.1 Fourier transformation with(red) and without(blue) Hanning window. 16
4.2 Comparison between Fourier and Short-time Fourier transformation. 17

5.1 The block diagram of the system. 22
5.2 Two raw OTDR signals (encoded with green) from two distinct segments. From both segments, two signals of one second have been selected, one for train signal (encoded with red) and one for background signal (encoded with blue). The corresponding DFTs of the selected signals are visualized with the same color in the bottom images. 23
5.3 Classification filtering for point extraction. The zoom shows that due to high spatial resolution, fine details are missing. 26
5.4 Different key-point extraction methods. 27
5.5 The two graphs and their vertex sets for train tracking. 30
5.6 Example of train tracking algorithm execution when the trajectories of two trains cross each other. Points (V_{tp}) are denoted as P, the subscript represents rising/falling. Trains (V_{tt}) are denoted as T, the subscript has no effect on the algorithm. Arrows represent the direction of points and trains. 32
5.7 Vertically extracted train profiles of 3 trains in 2 different positions. 34
5.8 Horizontal and vertical profile extraction. 35
5.9 Train profiles using STFT, where wagons can be seen from 3 trains in 2 different positions. Colors encode the intensity of a frequency at a given time. 36
5.10 An example calibration before a station. 37
5.11 Variance of background noise (blue) in OTDR signal through time. Red marked signals are vibration signals. 38

6.1 Decision regions of vibration and background using two eigenvalues. 43
6.2 Overlapping feature values from multiple trains. Trains are color coded. . . . 43
6.3 Extracted train length after key-point detection. 45
6.4 Histogram of difference between computed and true train length. 46
6.5 Processing time variance of Gaussian fitting. 46
6.6 Results of the detection and tracking for the two recordings, where each tracked object is encoded with a different line color. The ground truth data are visualized with light gray dashed lines. Underlying the tracked trains and the ground truth tracking, the detection of vibrations is plotted in black. . . 49
6.7 Overlap of train trajectory at a shunting yard. 50
6.8 Similarity between consecutive train profiles. Note that the consecutive profiles have been shifted in amplitude for better visualization. 51
6.9 Two signals, where wagons can be extracted from the signal with a lower SNR. 54

List of Tables

2.1 Parameters of the two long recordings. 7

5.1 Binary Integer Programs (BIP) for tracking. 30

6.1 Accuracy and speed results of different detection parameters. 42
6.2 Average of absolute deviation of train length after key-point extraction. . . . 44
6.3 Evaluation of train tracking performance. 48
6.4 Confusion matrix of train profile cross-correlation results. 52
6.5 Execution time in milliseconds of each operation using CUDA. 55

List of Algorithms

5.1 Quantization of Fourier coefficients. 24
5.2 Vibration detection. 24
5.3 The GPU based vibration detection. 39
5.4 The normalization kernel. 40

Bibliography

[ALZ13] T. Albrecht, K. Luddecke, and J. Zimmermann. A precise and reliable train positioning system and its use for automation of train operation. In *IEEE International Conference on Intelligent Rail Transportation (ICIRT)*, pages 134–139, August 2013.

[BC12] Xiaoyi Bao and Liang Chen. Recent progress in distributed fiber optic sensors. *Sensors*, 12(7):8601–8639, 2012.

[BMT$^+$15] Adeline Bailly, Simon Malinowski, Romain Tavenard, Thomas Guyet, and Laetitia Chapel. Bag-of-temporal-sift-words for time series classification. In *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, 2015.

[Bra99] Ronald N. Bracewell. *The Fourier transform and its applications.* McGraw-Hill, 3rd edition, 1999. ISBN: 0-07-116043-4.

[CCWW11] Chi Chen, Rong Chen, Fang Wei, and Ding Hong Wu. Experimental and application of spiral distributed optical fiber sensors based on otdr. In *2011 International Conference on Electric Information and Control Engineering (ICEICE)*, pages 5905–5909. IEEE, 2011.

[CJT03] Kyoo Nam Choi, Juan Carlos Juarez, and Henry F Taylor. Distributed fiber optic pressure/seismic sensor for low-cost monitoring of long perimeters. In *AeroSense 2003*, pages 134–141. International Society for Optics and Photonics, 2003.

[CRM03] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[ESTA14] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2154, 2014.

[HBL13] Eric J Humphrey, Juan P Bello, and Yann LeCun. Feature learning and deep architectures: new directions for music informatics. *Journal of Intelligent Information Systems*, 41(3):461–481, 2013.

[HDY^{+}12] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[HS81] Berthold K Horn and Brian G Schunck. Determining optical flow. In *1981 Technical symposium east*, pages 319–331. International Society for Optics and Photonics, 1981.

[HS06] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[Isr11] Israel.abad. Safety train separation with fixed block and moving block signaling systems. `https://commons.wikimedia.org/wiki/File:FB_vs_MB.jpg`, 2011. [Online; accessed 19-Sept-2016].

[JFL07] Hao Jiang, Sidney Fels, and James J Little. A linear programming approach for multiple object tracking. In *CVPR'07. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[JMCT05] Juan C Juarez, Eric W Maier, Kyoo Nam Choi, and Henry F Taylor. Distributed fiber-optic intrusion sensor system. *Journal of lightwave technology*, 23(6):2081, 2005.

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[KS12] S.E. Kanellopoulos and S.V. Shatalin. Detecting a disturbance in the phase of light propagating in an optical waveguide, September 11 2012. US Patent 8,264,676.

[KZX^{+}14] Heng Kong, Qian Zhou, Weilin Xie, Yi Dong, Cheng Ma, and Weisheng Hu. Events detection in otdr data based on a method combining correlation matching with stft. In *Asia Communications and Photonics Conference*, pages ATh3A–148. Optical Society of America, 2014.

[Low04] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[LWW^{+}16] Werner Lienhart, Christoph Wiesmeyr, Richard Wagner, Ferdinand Klug, Martin Litzenberger, and Dietmar Maicz. *Condition monitoring of railway tracks and vehicles using fibre optic sensing techniques*, pages 45–50. ICE Publishing, 2016.

[oB16] University of Birmingham. DEDOTS: Developing and Evaluating Dynamic Optimisation for Train Control Systems. `https://www.ucl.ac.uk/resilience-research/research/dedots`, 2016. [Online; accessed 19-Sept-2016].

[ODW12] Arch Owen, Gregory Duckworth, and Jerry Worsley. Optasense: fibre optic distributed acoustic sensing for border monitoring. In *Intelligence and Security Informatics Conference (EISIC), 2012 European*, pages 362–364. IEEE, 2012.

[Opt14] OptaSense. Acoustic sensing the future for rail monitoring? `http://www.optasense.com/wp-content/uploads/2014/04/TRE-April-2014-Optasense.pdf`, 2014. [Online; accessed 19-Sept-2016].

[Pan87] E Panarella. Heisenberg uncertainty principle. In *Annales de la Fondation Louis de Broglie*, volume 12, pages 165–193, 1987.

[PDRL14] Fei Peng, Ning Duan, Yun-Jiang Rao, and Jin Li. Real-time position and speed monitoring of trains using phase-sensitive otdr. *Photonics Technology Letters, IEEE*, 26(20):2055–2057, 2014.

[PWJ$^+$14] Fei Peng, Han Wu, Xin-Hong Jia, Yun-Jiang Rao, Zi-Nan Wang, and Zheng-Pu Peng. Ultra-long high-sensitivity ϕ-otdr for high spatial resolution intrusion detection of pipelines. *Optics express*, 22(11):13804–13810, 2014.

[PWL$^+$16a] Adam Papp, Christoph Wiesmeyr, Martin Litzenberger, Heinrich Garn, and Walter Kropatsch. A real-time algorithm for train position monitoring using optical time-domain reflectometry. In *IEEE International Conference on Intelligent Rail Transportation (ICIRT)*, pages 89–93. IEEE, 2016.

[PWL$^+$16b] Adam Papp, Christoph Wiesmeyr, Martin Litzenberger, Heinrich Garn, and Walter Kropatsch. Train detection and tracking in optical time domain reflectometry (otdr) signals. In *German Conference on Pattern Recognition*, pages 320–331. Springer, 2016.

[QCB12] Zengguang Qin, Liang Chen, and Xiaoyi Bao. Wavelet denoising method for improving detection performance of distributed vibration sensor. *Photonics Technology Letters, IEEE*, 24(7):542–544, 2012.

[RS91] Krishnan Rangarajan and Mubarak Shah. Establishing motion correspondence. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91.*, pages 103–108. IEEE, 1991.

[SJ87] Ishwar K Sethi and Ramesh Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1):56–73, 1987.

[SS05] Khurram Shafique and Mubarak Shah. A noniterative greedy algorithm for multiframe point correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):51–65, 2005.

[ST94] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94.*, pages 593–600. IEEE, 1994.

[STE13] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems*, pages 2553–2561, 2013.

[TE14] AV Timofeev and DV Egorov. Multichannel classification of target signals by means of an svm ensemble in c-otdr systems for remote monitoring of extended objects. In *MVML-2014 Conference Proceedings*, volume 1, 2014.

[Tim15] Andrey V Timofeev. Monitoring the railways by means of c-otdr technology. *International Journal of Mechanical, Aerospace, Industrial and Mechatronics Engineering*, 9(5), 2015.

[Tót14] László Tóth. Combining time-and frequency-domain convolution in convolutional neural network-based phone recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 190–194. IEEE, 2014.

[Wil16] Colin Williams. The next etcs level? In *IEEE International Conference on Intelligent Rail Transportation (ICIRT)*, pages 75–79. IEEE, 2016.

[WLPR14] Huijuan Wu, Xiaoyu Li, Zhengpu Peng, and Yunjiang Rao. A novel intrusion signal processing method for phase-sensitive optical time-domain reflectometry (ϕ-otdr). In *OFS2014 23rd International Conference on Optical Fiber Sensors*, pages 91575O–91575O. International Society for Optics and Photonics, 2014.

[XB09] Jierui Xie and Mandis S Beigi. A scale-invariant local descriptor for event recognition in 1d sensor signals. In *IEEE International Conference on Multimedia and Expo*, pages 1226–1229. IEEE, 2009.

[YJS06] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006.

[YLL10] Chang Huai You, Kong Aik Lee, and Haizhou Li. Gmm-svm kernel with a bhattacharyya-based distance for speaker recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1300–1312, 2010.