Technical Report

Pattern Recognition and Image Processing Group
Institute of Computer Aided Automation
Vienna University of Technology
Favoritenstr. 9/1832
A-1040 Vienna AUSTRIA
Phone: +43 (1) 58801-18351
Fax: +43 (1) 58801-18392
E-mail: sec@prip.tuwien.ac.at
URL: http://www.prip.tuwien.ac.at/

PRIP-TR-71

22nd May 2002

Visual Traffic Surveillance Using Real-time Tracking *

Roman P. Pflugfelder

Abstract

This report presents an unusual event recognition approach in the field of traffic surveillance. Such events are unusual traffic behaviour like traffic jams, accidents or ghost drivers. An interest-point based tracking algorithm (KLT-tracker) is discussed which pursues features on vehicles through a static camera scene. Tracking data can be collected by observing normal traffic. Then, this data is used to learn a spatio-temporal model of normal traffic behaviour. Thereby, training samples are generated in a learning space by the tracking data. Thus, the spherical probability density function (p.d.f.) of the space can be estimated. We use a Growing Neural Gas in combination with a MDL-based pruning algorithm for unsupervised learning. The former method belongs to the class of soft-competitive algorithms which overcome the problems of ”stranded” reference vectors. In contrast to other works, the number of reference vectors has not to be constant. The algorithm finds an optimal codebook according to the MDL-principle. As the p.d.f. only describes points and not trajectories of normal traffic behaviour, behaviour classes of normal traffic have to be learnt additionally. This work presents a novel approach by using the topology of the learning space which is created by Competitive Hebbian Learning. Beside the necessity of recognizing unusual events, it can also be used to analyze the behaviour of drivers at traffic sites like intersections or road works.

*This report was supported in parts by ARC Seibersdorf research GmbH, ITV division

Acknowledgements

I am indebted to a number of persons who contributed in various ways to this technical report.

I want to thank Horst Bischof who supervised me the last two years. He was the person who actually brought me into the field of visual surveillance. I got always great support from him during my two practical works at ARC Seibersdorf research and throughout this work. Furthermore, he gave me the ability to become acquainted with the scientific community.

I want to thank Helmut Schwabach and Bernhard Strobl who offered me the opportunity to work at ARC Seibersdorf. They supported me with lots of ideas and good discussions. Special thanks to Michael Wohlfahrt who implemented the KLT tracking algorithm in real-time. You were a great help for me.

Certainly, I thank the IRIS group at the University of Queensland for their support during my stay in Brisbane, Australia. Thank you Brian Lovell that you gave me the opportunity to study abroad. Many people have helped me during this time. Thanks to Kurt Kubik, Dave McKinnon, Martin Robinson, Bob Andrews and David Vignon.

Thanks to all folks at the PRIP group. I had many useful discussions, especially with Nabil Belbachir, Alexander Selb and Christian Wolf. We were a great postgrad team.

Finally, very special thanks to my parents and my grandmother who have greatly supported me throughout my computer science study.

Thank you! Roman P. Pflugfelder

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In recent years, the volume of traffic has become a significant problem. Consequently, accidents and traffic jams are far more likely than a century ago. Many of us living in metropolitan areas got used to the every-day traffic news about congestions. Early solutions attempted to lay more pavement to avoid jams, but adding more lanes is becoming less and less feasible. Besides, reckless, confused (e.g. ghost drivers) or drunken car drivers are more and more a source of danger and cause many terrible accidents and jams. Most of them ignore traffic rules and drive prohibitively in wrong directions or exceed speed limits. Instead of increasing the capacity of existing infrastructure, contemporary solutions of visual surveillance try to use roads more efficiently. Thereby, more and better traffic information which is automatically gathered in real-time is emphasized. Such information can be *traffic parameters* like traffic volume, occupancy and vehicle's speed. Another group of more *qualitative information* are unusual events (e.g. traffic jams, accidents, prohibitively driving cars, etc.). The subject of this report is to recognize such unusual traffic events.

The quest for better traffic information and thus an increasing reliance on traffic surveillance, has resulted in a need for better vehicle detection and traffic analysis tools. Traditional sensor techniques like loop detectors based on induction or magnetic field sensors are well known and used in many applications. The main drawbacks are their impossibility to gather specific traffic information and their less flexibility. For example, the analysis of traffic behaviour in an unknown scene is impossible to perform with loop detectors. For that, wide area sensors are interesting. Cameras can usually observe several lanes in parallel. Visual traffic surveillance based on video images is capable of collecting traffic information. Though video technology has been available for a number of years, decreasing computer and image processing hardware costs have recently made solutions based on video analysis more and more attractive (ITS decision report, *http://www.path.berkeley.edu*):

- Installation, operations and maintenance costs are now lower than for traditional

methods.

- Video technology has been either decreasing in cost or gaining in efficiency and this trend is continuing.
- Digital video, while still on the order of three times more expensive than analog equipment, offers numerous additional benefits. This will make video surveillance more attractive as prices for digital equipment fall.

Especially the last point should be emphasized. The following list shows the most important benefits:

- Camera information is readily understandable by human operators.
- Unusual event recognition is done rapidly.
- It is possible to identify the incident type, the level of gravity and the type of intervention needed. This is generally performed by human operators.
- It delivers images focusing on the detection point of unusual events, helping the operator to rule out false alarms.
- Collection and potential analysis of traffic and incident data. Information provided in the video sequences immediately preceding an incident can be particularly valuable. Indeed, information analysis can provide an understanding of accidents occurring on the network and can allow enhancement of infrastructure reliability. Surveillance data could be used not only for safety but also for transportation administration, planning, operations and research.
- It can classify vehicles, monitor intersections, actuate signals and read license plates (which can be used for enforcement and travel time estimation).
- Installation typically does not require lane closures. Traffic personnel safety is enhanced and traffic disruptions are minimized. By repositioning cameras as road geometry varies, visual traffic surveillance can be used during realignment or resurfacing.

To sum up, perhaps the major handicap of traditional visual surveillance systems is that they are designed to operate with data taken at a *point* rather than over *space*. This information alone, typically volume and occupancy, has not proven to be sufficient for effective and reliable unusual event recognition. The data is deficient because volume is not a dynamic measurement, and because occupancy is an approximate rather than true measurement of a spatial traffic flow variable, namely density.

Most of the commercial visual traffic surveillance systems available today are *tripwire* systems (e.g. AUTOSCOPE, *http://www.autoscope.com*). Small localized regions of the image mimic conventional sensors. Multiple sensors can be located within

the image and can be easily configured to suit the road geometry. Then, image processing within these sensors delivers traffic parameters. Thus, vehicles are not identified and are not considered as unique targets. This approach is computationally inexpensive and can be implemented on off-the-self computers in real-time. The drawbacks of this approach are that the unusual event recognition algorithms remain the same as for traditional sensors, and that the accuracy of individual sensors depends on the camera's field of view.

Some commercial systems use *feature tracking* methods to pursue vehicles (e.g. EVA). Visual feature tracking means pursuing image features (e.g. edges, corners, regions, etc.) between two consecutive images. This is done over a long sequence of images. These images are frames of a video or captured in real-time by an acquisition system. Generally, these systems use region tracking, i.e. regions of movement are vehicles. Individual vehicles are detected and tracked through the camera scene. This provides a microscopic description of vehicle movements which can reveal new data on events such as sudden lane changes, vehicles driving in the wrong direction and stationary vehicles (e.g. accidents). This increase in sophistication requires more computing power, requires individual vehicles to be discernible and can be even more restrictive in camera positioning. The latter dues to the fact, that regions are used for tracking. Unfortunately, regions can merge if vehicles occlude each other.

The third approach, used in the Image Processing for Automatic Computer Traffic Surveillance (e.g. IMPACTS) system, concentrates on *spatially analyzing* image intensities. Instead of considering traffic on a vehicle by vehicle basis, the underlying strategy is to describe how the visible road space is being utilized at a particular instant in time. Disturbances in traffic flow can then be determined by analyzing how these descriptions vary over time. Use of road space is divided into three categories: no traffic present, moving traffic present, or stationary traffic. These are essentially qualitative decisions.

All described commercial products are able to extract traffic parameters. Unusual event recognition is based on this information. Thereby, parameters are combined to form a response of the system according to an event. A positive alarm takes place, if a certain manually defined threshold is exceeded. However, the purpose of this report is to develop a system which learns *independently* unusual events. No thresholds are defined any more. Only video examples of normal traffic of a particular static camera's view are necessary. Once the system has learnt normal traffic behaviour, it is able to recognize unusual traffic events within the same scene. Furthermore, the system is able to analyze traffic behaviour in an *unknown* traffic environment. For example, the spatio-temporal behaviour of car drivers heading to a road work or intersection could be investigated. Thus, the system is highly flexible with less human interactions needed.

1.1 Structure of the report

The basis of unusual event recognition is tracking data. Chapter 2 describes a method which generates this data. Section 2.1 introduces visual feature tracking and defines the scope of the work. Section 2.2 gives the tracking requirements and presents a Shi-Tomasi interest-point tracking approach. The results with different traffic scenes are presented in section 2.3. Finally, the chapter finishes with a conclusion (section 2.4) of the main drawbacks. It is shown that despite these disadvantages it can be used for unusual event recognition.

Chapter 3 explains unusual events in section 3.1. They are indicated by interest-points which are pursued by the tracker. To use especially these features has consequences on the recognition which is described in section 3.2. Finally, section 3.3 describes the principle of unusual traffic event recognition using tracking data of normal traffic. The novelty of our approach is shown.

Chapter 4 treats the generation of training samples in the learning space from measured feature trajectories. The former are used as training samples for successive learning by clustering. First, it outlines several difficulties during this generation process with respect to the learning space and problems with the tracking data which have to be tackled to improve learning (section 4.1). Then, solutions for these problems are discussed in sections 4.2 - 4.4.

Chapter 5 treats learning a spatio-temporal model of traffic behaviour. It is shown that the latter is an approximation of the probability density function in the learning space which is learnt by a density estimator. Section 5.1 introduces the learning problem in general and gives some definitions. Then, section 5.2 introduces the basic learning paradigms and presents several solutions. Especially, unsupervised competitive learning is treated in detail in section 5.3. Finally, section 5.4 presents the proposed learning framework for this work.

Chapter 6 treats the classification of traffic behaviour within a traffic scene. Section 6.1 motivates the need for classification and presents two previously used approaches. Both methods calculate explicitly the classification result by using a further learning step or a co-occurrence matrix respectively. In contrast to them, section 6.2 discusses a new approach which only uses the topology of the training data. In fact, the topology was calculated implicitly by the learning step.

Chapter 7 presents experiments with the spatio-temporal model and classification algorithm. Section 7.1 treats the training and test data. Section 7.2 shows experiments with the training sample generation process. It focuses on the ability of obvious outlier detection and noise reduction. Section 7.3 evaluates the GNG pre-clustering and the successive MDL-based pruning step. Section 7.4 shows the results of classification with respect to the training data. Performance issues are also treated. Finally, the results of using the spatio-temporal model and classification on the test data are shown in section 7.5. A summary is given in section 7.6.

Chapter 8 discusses the presented unusual event recognition approach and gives

an outlook of future work.

Chapter 2

Visual tracking of traffic

Contents

This chapter describes the tracking method for the traffic monitoring application. Section 2.1 introduces into visual feature tracking and defines the scope of the work. Section 2.2 gives the tracking requirements and presents a Kanade-Lucas-Tomasi (KLT) interest-point tracking approach. The results with different traffic scenes are presented in section 2.3. Finally, the chapter finishes with a conclusion (section 2.4) of the main drawbacks of the method. It is shown that despite these disadvantages it can be used for unusual event recognition.

2.1 Overview

Visual feature tracking means pursuing image features or user-defined feature models between two consecutive images. This is done over a long sequence. These images are frames of a video or are captured in real-time by an acquisition system. The main problem is to match the feature in the first frame to a collection of measurements taken from the next. This matching problem is equal to the correspondence problem, which is well known in 3D computer vision.

Traffic monitoring is an important application of visual feature tracking. Vehicles are recognized and tracked through the camera view by a real-time computer system. Data, like velocity, number of vehicles, type of vehicles, density of traffic, etc., are stored. Traffic prediction and other high-level tasks like unusual event recognition, which is the issue of this report, can then be build upon this low-level tracking system. Many motion detection and tracking algorithms have been investigated for traffic monitoring in the last years (Koller et al. [26], Ferrier et al. [10], Beymer et al. [3], Stauffer and Grimson [41]). A discussion and comparison of tracking methods can be found in Pflugfelder [34].

The simplest and by researchers and practioners mostly used detection algorithms are based on background differencing (Ullman [45]). In the simplest case, these methods subtract the actual image frame at time t, $I(t)$, from the background reference image. The frame difference is then segmented in areas with and without motion. A shortcoming of all of these methods is the robustness against illumination conditions and noise. Generally, the geometry of the scene can be assumed static, because the camera is fixed. However, intensity values cannot be assumed to be constant for the background, because the lightning conditions change slowly due to day and night or quickly due to reflections and noise. Therefore, the problems associated with illumination and noise have to be addressed in an application like traffic monitoring. One solution is an adaptive background estimation process. For example, Stauffer and Grimson used a Gaussian mixture model of intensities for every pixel which represents background colour. Then, foreground and background pixels can be distinguished to find regions of motion which are basically vehicles, because they are the only allowed objects within the scene. Beside background differencing, a second approach exists. It is called frame differencing, because successive frames in a video sequence are subtracted. Like in the background differencing case, the geometry of the scene and the intensity values of the background are assumed to be static. In contrast to background differencing with adaptive background estimation, it is sufficient to classify a pixel as foreground if its intensity value changes in time. Unfortunately, a second problem also arises with large homogenous moving objects. Pixel values within the object do not change and are not recognized as object pixels although they are part of the object. Thus, only the boundaries of the object can be found.

Generally, two different tracking approaches exist, namely calculation of optical flow and feature tracking methods (Trucco and Verri [44]). The former ascertains an approximation of the dense motion field where the letter tracks merely a feature

from one to the next frame and therefore calculates a sparse motion field. Motion detection like background differencing is then a prerequisite. Optical flow estimation is computationally expensive and noise sensitive. Therefore, most of the researchers use visual feature tracking methods for traffic monitoring. A detailed overview of feature tracking is given in Pflugfelder [34] and in the introduction of Isard's thesis [20].

The next section discusses a feature tracking method which is our second approach in the field of traffic monitoring. While our first work only dealt with cars in tunnels (Pflugfelder and Bischof [35]), the proposed method is able to address the problem of vehicle tracking in a more general setting. However, it is not necessary that the tracker identifies vehicles to track them. Consequently, tracking data is information about traffic flow which is sufficient to recognize traffic jams or many other unusual events. Occluding vehicles are a problem in all traffic approaches which are based on background differencing and thus region or blob tracking, because blobs merge to larger ones which causes the tracker to fail. As the tracker has to pursue congested vehicles, other features should be used.

2.2 The Approach

The aim of tracking in traffic monitoring is to pursue vehicles through a long sequence of images. To reach this goal, the requirements of such a tracker have to be named:

(i) Reliability in detecting the vehicles,

(ii) Stability during tracking,

(iii) Accuracy in predicting the vehicle's state (e.g. position, velocity in a specific frame),

(iv) Real-time capability.

Reliability in detection means that every vehicle (e.g. car, bike, truck) should be detected while the traffic is monitored by the system. If the trajectories of pursued vehicles are continuous then we will speak about a stable tracker. Certainly, the stability is connected with the accuracy of the predictions. If the tracker should be practicable for real world applications, it has to run in real-time.

Furthermore, application specific requirements have to be considered. As it is mentioned in Pflugfelder and Bischof [35], the following requirements arise in traffic monitoring especially in the environment of tunnels:

(v) Robustness against different illumination conditions (e.g. reflections on vehicles, tunnel walls and on the road)

(vi) Robustness against vehicle occlusions

(vii) Usage of existing infrastructure (cameras, data network)

(viii) System must provide tracking data for further data processing (e.g. traffic unusual event recognition)

Day and night changes, car lights, sun light reflections, etc. lead to difficult illumination in the image. Besides, vehicles tend to occlude each other in case of congestions. An appropriate tracking system has to tackle such problems. Interestingly, lots of cameras have been installed at intersections, tunnels or streets. Unfortunately, many of them are low cost cameras which introduce further noise. Certainly, the tracking data (trajectories of vehicles) has to be properly stored that a successive processing like unusual event recognition is possible.

It is a great challenge to fulfill requirements (i)-(viii) in one solution. Generally, to build an appropriate tracker three design decisions have to be answered. First, well suited *features* have to be defined to reliably detect the vehicles and then to guarantee a stable tracking. Next, the right choice of a *motion model* should enable the tracker to take new measurements of these features with high accuracy in an area of the next image. This area has to be as small as possible as the tracker works in real-time. Finally, a *matching and state update algorithm* has to be chosen with respect to the application. For an overview of different models see Pflugfelder [34].

Our first attempt of tracking cars in tunnels used car lights as features. They were detected as blobs in a detection window and then tracked by a Kalman filter matching and state update algorithm. As the motion of cars was simple (e.g. linear, because vehicles drive straight along the road) and the frame-rate of the test samples was 15 frames/s, a linear motion model was sufficient. The developed tracking system ran satisfactorily but the main drawback was that it handled only approaching cars. The reason was that the back lights were to small and too weak in contrast. A second problem was that neither bikes nor trucks were detected, because the feature detection was not able to recognize the bigger lights of the trucks or the missing light pair on bikes. Figure 2.1 shows a typical frame of a test tunnel scene which we used for the tracker experiments. The car and its pursued right car light is illustrated. Table 2.1 shows the evaluation of the requirements mentioned above. The weakness in recognizing different vehicles and the impossibility of occlusion tolerance results in the fact that this approach is insufficient to support high-level tasks like traffic unusual event recognition.

As a consequence the aim of the second approach is especially to meet these two requirements. As mentioned in the last section, occlusion robustness and vehicle diversity is a prerequisite of an unusual event recognition, i.e. traffic jams. In the following three sections the feature model, namely Shi-Tomasi interest-points, are presented. It was shown by Shi and Tomasi [39], that they are the best track-able features with respect to a deterministic (SSD based) matching and state-update model. For example, the aperture problem does not arise in contrast to edges. These interest points can also be interpreted as corners. Beymer et al. [3] showed that corner features are an appropriate feature for handling occlusions during traffic monitoring. Besides, they occur on

Figure 2.1: A typical frame of a tunnel scene video. The right car light of the visible car is pursued through the images. Each position is shown by a cross. Positions form together the trajectory.

Requirements	Evaluation	
reliability in detection	-	car lights are very different
stability in tracking	+	simple linear motion along road
accuracy in prediction	+	blobs are accurately detectable
real-time capability	+	Kalman filter and blob detection is simple
illumination and noise	+	blob detection is robust
occlusion robustness	-	impossible
usage of existing hardware	+	possible
provide tracking data	-	only approaching cars are tracked

Table 2.1: Requirements and their evaluation by testing the first tracking approach.

any type of vehicle. The motion model is a simple linear translation, because motion of vehicles is linear with a frame-rate of 15 frames/s.

2.2.1 An appropriate motion model

Motion means the change of intensities in gray level patterns. These changes are highly complex due to the motion of objects and noise. However, it can be used to describe image motion as the following equation shows:

$$I(\mathbf{x}, t + \tau) = I(\delta(\mathbf{x}), t) \tag{2.1}$$

Every pixel in image I is moved during time τ by a function δ. Generally, δ is nonlinear and practically undefinable. In the application of traffic monitoring, vehicle movements are restricted by the road. Under this assumption and an appropriate frame-rate

which is given by 15 frames/s, function δ is approximately linear and can be written as

$$\delta(\mathbf{x}) = A\mathbf{x} + \mathbf{d}. \tag{2.2}$$

This affine motion model is represented by six degrees of freedom. The 2×2 matrix A defines rotation, scale and shear while the 1×2 vector $\mathbf{d}$ represents translation. Given two images I at time t and $t+\tau$ and a window around pixel $\mathbf{x}$ at time t, tracking means determining these parameters of the motion model. A window is necessary, because the value of the pixel can both change due to noise and be confused with adjacent pixels. The consequence is that it is impossible to determine where the pixel went in the subsequent frame. More details can be found in Shi and Tomasi [39], Hager and Toyama [19], Hager and Belhumeur [18].

The quality of tracking depends on the window size, the textureness of the window and the amount of motion between frames. However, smaller windows are preferable because they are less likely to straddle a depth discontinuity. Unfortunately, A is in that case harder to estimate. For this reason and that the frame-rate is sufficient, a pure translational model

$$\delta(\mathbf{x}) = \mathbf{d} \tag{2.3}$$

is chosen for tracking, where A is set to zero. The pure translational motion model for each $\mathbf{x}$ in I with its successor image J can be summarized as

$$J(\mathbf{x} + \mathbf{d}) = I(\mathbf{x}) + \epsilon(\mathbf{x}) \tag{2.4}$$

where ϵ is noise and the time difference τ is set to one.

2.2.2 Matching and state update algorithm

Tracking in this case means estimating the translational vector $\mathbf{d}$ of the motion model. This cannot be done exactly, because of image noise and because the motion model does not fit perfectly. To find $\mathbf{d}$, the residual

$$\epsilon = \sum_W (J(\mathbf{x} + \mathbf{d}) - I(\mathbf{x}))^2 \tag{2.5}$$

is minimized, where W is a given window around $\mathbf{x}$. As an appropriate frame-rate is assumed $J(\mathbf{x} + \mathbf{d})$ can be linearized by its first-order Taylor expansion $J(\mathbf{x} + \mathbf{d}) = I(\mathbf{x}) + \mathbf{d}\nabla I(\mathbf{x}) + I_t(\mathbf{x}) + h.o.t.$ $I_t(\mathbf{x})$ is the time derivative of $I(\mathbf{x})$ and can be approximated by a simple difference with the previous frame. The residual can be written as

$$\epsilon \approx \sum_W (\mathbf{d}\nabla I(\mathbf{x}) + I_t(\mathbf{x}))^2 \tag{2.6}$$

To solve this minimization task, the residual is differentiated with respect to $\mathbf{d}$.

Then the result is set to zero, obtaining the linear system

$$
\begin{aligned}
C\mathbf{d} &= \mathbf{g} && (2.7)\\
C &= \sum_W \begin{pmatrix} I_x(\mathbf{x})^2 & I_x(\mathbf{x})I_y(\mathbf{x}) \\ I_y(\mathbf{x})I_x(\mathbf{x}) & I_y(\mathbf{x})^2 \end{pmatrix} && (2.8)\\
\mathbf{g} &= -\sum_W I_t(\mathbf{x})\nabla I(\mathbf{x}) && (2.9)\\
& && (2.10)
\end{aligned}
$$

The estimation of $\mathbf{d}$ is improved by a Newton-Raphson iteration scheme

$$\mathbf{d}_{k+1} = \mathbf{d}_k + C^{-1}\mathbf{g} \tag{2.11}$$

with $\mathbf{d}_0 = \mathbf{0}$. The iteration is stopped if no further improvement $\mathbf{d}_{k+1} = \mathbf{d}_k + \mathbf{c}$ with $\|\mathbf{c}\| < T_c$ happens, where T_c is a user-defined threshold, or k exceeds a maximal number of iterations.

Unfortunately, the whole motion estimation process is sensitive with respect to the magnitude of motion between frames. During traffic monitoring, vehicles are passing the camera view with high speed. The amount of motion displacements of image features can be larger than two or three pixels. Then, the estimation process fails. Therefore, we used a multi-resolution estimation approach like the one of Birchfeld (see *http://robotics.stanford.edu/ birch/klt/*). As it is shown in figure 2.2, a Gauss pyramid is build for every successive frame J of the video sequence. Starting at the top, with position $\mathbf{x}_i$ of image I, the motion estimation process is performed for every level of the pyramid. The resulting position $\mathbf{x}_j^{k+d_k}$ is a new starting point in level $k-1$. At the end the new position $\mathbf{x}_j$ of a pixel can be written as

$$\mathbf{x}_j = \mathbf{x}_i + \sum_{i=0}^{n} \mathbf{d}_i. \tag{2.12}$$

2.2.3 Feature model

Regardless of the matching and state-update model used for tracking, not all parts of an image contain motion information. For example, along a straight edge only the motion component orthogonal to the edge can be determined. This fact is called aperture problem and is shown in figure 2.3. As a consequence, features for tracking should be regions with a rich enough texture. In this spirit, researchers have proposed to track corners or windows with a high spatial frequency content or regions where some mix of second-order derivatives was sufficiently high.

All these definitions yield track-able features. However, people saw these interest-points independently of the matching and state-update model. The resulting features may be intuitive but come with no guarantee of being the best for the tracking model

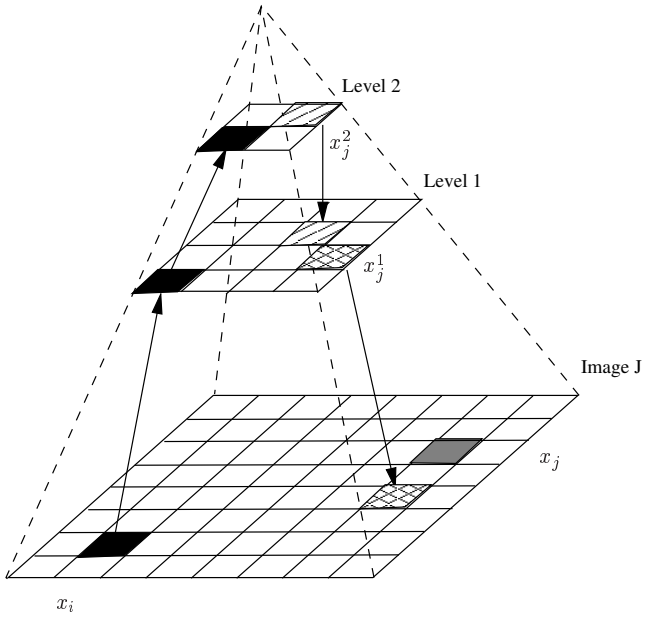
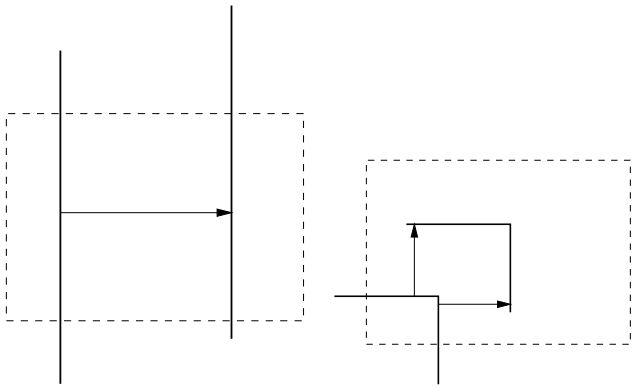


Figure 2.2: Multi-resolution Tracking: The last position $\mathbf{x}_i$ of the feature is transformed to the highest level (here 2) of the pyramid of successor Image J. Starting at level 2 with position $\mathbf{x}_i$ the motion estimation process is performed. The resulting new position in level 2 (dashed pixel) is then transformed to the next lower level (level 1). Now, the motion estimation process is invoked once more. Transformation and motion estimation are performed until full resolution of image J is reached. Lastly, the position $\mathbf{x}_j$ is the result of the multi-resolution tracking from frame I to J.

to produce good results. Shi and Tomasi [39] proposed a feature model that is optimal by its construction. Features are windows which can be tracked well. They are not good windows a priori, defined by above criteria. The definition is based on the matching and state-update model. In fact, a feature can be tracked if (2.7) represents good measurements and can be solved reliably. This means that C must be both above the image noise level and well-conditioned. In turn, the noise requirement implies that both eigenvalues of C must be large, while the conditioning requirement means that they cannot differ by several orders of magnitude. Therefore, two large eigenvalues can represent corners, salt-and-pepper textures or any other pattern which can be tracked reliably. In practice, when the smaller eigenvalue is sufficiently large to meet the noise criterion, C is also well conditioned. This is due to the fact that intensity variations in a window are bounded by the maximum allowable pixel value, so that the greater eigenvalue cannot be arbitrarily large. As a consequence, every pixel and its vicinity within a region of interest is investigated if it is an interest point or not. A window is accepted as a feature if the two eigenvalues of C fulfill

$$min(\lambda_1, \lambda_2) > \lambda, \tag{2.13}$$

where λ is a predefined threshold. The determination of λ and the explanation why such features mainly are on vehicles is described in the next chapter.



(a) Edge **(b)** Corner

Figure 2.3: The aperture problem: (a) Only the motion component orthogonal to the edge can be determined. (b) The motion of corners can always be determined.

2.3 Experiments

Three different video sequences were used for analyzing the behavior of the tracking approach. Every sequence consists of 225 frames. 2 of them for every sequence are shown in the following figures. Figure 2.4 shows an outdoor scene in the early morning. Both images are qualitatively bad. The reason is a bad camera and dust caused by the Danube river beside the highway. Image 2.4(a) shows heavy traffic with growing obstructions while in image 2.4(b) a truck changes the lane and occludes other vehicles. In contrast to that, figure 2.5 shows a tunnel scene. Motion blur caused of the small distance between vehicles and camera can be seen. As in all videos, noise and light reflections are evident. In image 2.5(a) a bike appears while image 2.5(b) shows a van. The last sequence 2.6 shows vehicles entering a gate. Here, the main problems are the day/night change and the light reflections in the tunnel. All sequences have been sampled by a rate of 15 frames/s from a VHS/PAL video stream. The resolution of each frame amounts to 720×270. Every second row is neglected to avoid interlacing effects of the used CCD camera. Therefore, the column resolution is 270. All algorithms, which are explained in the next two sections, were implemented as prototypes with Matlab 5.3. They are not real-time capable. Furthermore, a real-time prototype was also implemented with the Intel Image Processing Library (IPL). Critical parts were also realized in Assembler. We achieved real-time tracking for up to 4 cameras on an off-the-self Pentium computer.

2.3.1 Feature selection

The principal outline of the Shi-Tomasi interest-point detection is shown by algorithm 2.1. Steps 1 and 2 are simple image processing. Information about smoothing

(a) Heavy traffic

(b) Truck

Figure 2.4: K01 camera video sequence: (a) shows heavy traffic with growing obstructions (b) shows a truck changing the lane and therefore it occludes other vehicles.

(a) Bike

(b) Van

Figure 2.5: TV108 camera video sequence: (a) shows a bike (b) shows a van.

Require: I, W, σ_s, σ_g, λ

1: Smooth image by σ_s
2: Compute gradients of the image by σ_g
3: Compute matrix C_W for every pixel of image (2.8)
4: Compute smaller eigenvalue of C_W for every pixel
5: Do segmentation by λ
6: Find locally the best interest-points (non-maximum suppression)

Algorithm 2.1: Algorithm of the Shi-Tomasi interest-point detector.

W	σ_s	σ_g	λ_{K01}	λ_{else}
9	0.9	1.0	1000	5000

Table 2.2: The parameters of the Shi-Tomasi interest-point detector are summarized.

and gradient calculation can be found in Trucco and Verri [44]. After matrix C is calculated in step 3 by simple convolutions its smaller eigenvalue, further called goodness value, can be directly computed by trace and determinant of C. In step 5 all pixels are ruled out which have a smaller eigenvalue than λ. In step 6 non-maximum suppression is performed. Only interest-points should survive which have the highest eigenvalue within their vicinity.

The algorithm produces a list of interest-points with their goodness values within an image. The higher the goodness value for a specific interest-point is, the better it is track-able. Figure 2.7 shows three typical images for the camera video sequences 2.4, 2.5 and 2.6. All interest-points which were found are marked by crosses.

The parameters which produce these results are summarized in table 2.2. W defines a $W \times W$ window around an interest-point. A smaller W increases the number of interest-points and decreases their goodness values. Smaller windows are also more sensitive to noise during tracking, as it was mentioned in section 2.2. To find interest-points which are features to track, a trade-off has to be found for W. Interest-points on road and on vehicles ($\rightarrow$ features) have to be distinguishable by their goodness values. In the experiments values of 7, 9 or 13 for W were common. σ_s is the standard deviation of the Gauss-kernel that is used to smooth the image. σ_g also is a standard deviation of a Gauss-kernel for computing the gradients. Both are not critical for feature detection. λ, see (2.13), must be smaller than the goodness value of an interest-point. Then, it is a feature.

To find λ, consider the histograms in figure 2.8. They show that most interest-points within the detection window have small goodness values. They are lying on the static background which is in all cases the road or parts of the tunnel wall. Features have significantly higher goodness values. Strong edges, strong textureness and a lot of corners are the reason, why vehicles produce interest-points with higher goodness values within the detection window. A mathematical explanation can be found

Require: I, J, L_F, W, σ_s, σ_g, λ, Det_{min}, $Disp_{min}$, $Iter_{max}$, Res_{max}, σ_{pyr}, S_{range}

1: Build pyramid P with P_n levels for image J
2: **for all** Features F_i in L_F **do**
3: $Pos_J^{P_n}(F_i) \leftarrow Pos_I^{P_n}(F_i)$ {The starting position in level P_n of J is the last position in I}
4: **for all** levels P_i of pyramid P, $i \leftarrow n : 0$ **do**
5: **repeat**
6: Compute matrix C (equation 2.8)
7: Compute vector $\mathbf{g}$ (equation 2.9)
8: Solve $C\mathbf{d} = \mathbf{g}$ for $\mathbf{d}$
9: $Pos_J^{P_i}(F_i) \leftarrow Pos_J^{P_i}(F_i) + \mathbf{d}$
10: **until** $\|\mathbf{d}\| < Disp_{min} \vee$ iteration exceeds $Iter_{max}$ {Newton-Raphson optimization}
11: Transform $Pos_J^{P_i}(F_i)$ in the next lower level P_{i-1}
12: **end for**
13: **end for**

Algorithm 2.2: Algorithm of the deterministic feature tracker.

in section 2.2. See figure 2.9 as an example for an interest-point on background and a feature. The goodness values differ about 1000 times. By setting λ to 1000 for the camera video sequence 2.4 and to 5000 for the others, a segmentation can be reached which delivers features. Another important assumption is a small detection window where the static background guarantees features with small goodness values.

The results are shown in figure 2.10. The segmentation method gives good results. Sometimes, false features were detected or interest-points were not recognized. To overcome this problem λ has to be adaptively updated. This could be done by feedback of the tracker and a collection of statistics over a long period. False features can also be ruled out during tracking.

2.3.2 Tracking

Algorithm 2.2 shows the implementation of the feature tracker. The tracker needs a frame I and its successor J as input. Further, the features are stored in a feature-list L_F. First the Gauss pyramid P with P_n levels is build (step 1). The starting position $Pos_J^{P_n}(F_i)$ of every feature $F_i \in L_F$ in the highest level of P is equal to the last known position in image I, namely $Pos_I^{P_n}(F_i)$ (step 3). The new position in a certain level P_i is evaluated by computing C (step 6), $\mathbf{g}$ (step 7) and finally $\mathbf{d}$ (step 8) which is simply added to $Pos_J^{P_i}(F_i)$ (step 9). The evaluation is stopped when the Newton-Raphson optimization exceeds a iteration threshold $Iter_{max}$ or the new position does

Det_{min}	$Disp_{min}$	$Iter_{max}$	Res_{max}	σ_{pyr}	S_{range}
0.01	0.1	10	13	0.9	15

Table 2.3: The tracker specific parameters are summarized.

not change significantly any more ($\|\mathbf{d}\| < Disp_{min}$). $Pos_J^{P_i}(F_i)$ is then transformed to $Pos_J^{P_{i-1}}(F_i)$ (step 11) and the evaluation starts again until the new position $Pos_J^{P_0}(F_i)$ in the original image resolution is determined.

The three parameters of the interest-point detector plus further 6 tracker specific parameters have to be defined. Table 2.3 summarizes these parameters and their values which were used for testing. Det_{min} is the minimal allowable determinant of matrix C. Its value is chosen in that way that mathematical solvability of (2.7) is given. If the determinant drops below Det_{min}, the feature is eliminated from L_F. $Disp_{min}$ gives the minimal displacement and $Iter_{max}$ the maximal number of iterations to optimize the position of the feature by a Newton-Raphson iteration scheme. If the difference of the patterns in the first and actual frame exceeds Res_{max}, the feature will also be removed from L_F. σ_{pyr} is the deviation of a Gauss-kernel to calculate the image pyramids. S_{range} defines the image pyramid itself. Its value is the number of pixels that a feature can move from one image to the next. Consequently, a pyramid with an appropriate number of levels will be created.

The results of the tracker can be seen in figures 2.11, 2.12 and 2.13. They show tracked features as white crosses, pursued in 50 frames.

2.4 Conclusion

Four main problems arose from the experiments:

Motion is too big: If the displacements of feature positions are larger than S_{range}, the features cannot be tracked. (2.7) only holds if the frame-rate is high and therefore the feature motion is small enough (2-3 pixels).

Features get caught: Sometimes, features get caught on the background, because algorithm 2.2 only finds a local optimum of the displacement.

Features hop: Sometimes, features can hop from one vehicle to another if they come close together.

Noisy feature positions: Due to reflections and the image noise, the position of the features is influenced. For example, features which lie on the border between vehicle and background are affected by the varying background from one frame to the next. The perspective also influences the tracking of features. As vehicles getting smaller and smaller, the feature windows are the same and more and more background is inside the window. This also introduces errors.

Requirements	Evaluation	
reliability in detection	+	features can lie on every type of vehicle
stability in tracking	-	feature positions are noisy
accuracy in prediction	-	depends on the illumination conditions
real-time capability	+	interest-point detection and tracking is cheap
illumination and noise	-	very sensible
occlusion robustness	+	great advantage
usage of existing hardware	+	possible
provide tracking data	+	different driving directions are possible

Table 2.4: Requirements and their evaluation by testing the used tracking approach.

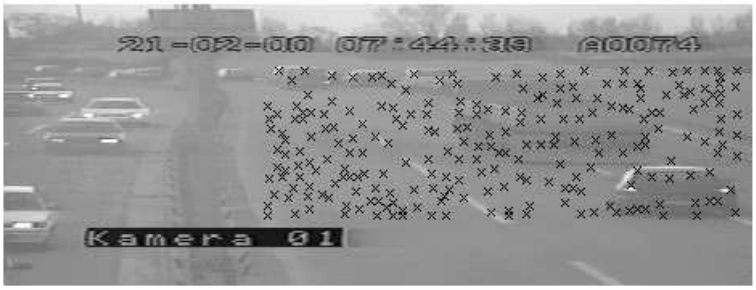
The main drawback of the tracker is the sensibility according to noise and quickly changing illumination conditions. Improvements can be made by using matching and state update models which consider illumination and noise (Jin et al. [21]). Nevertheless, the problem will still be evident. We argument the use of this tracking method by the fact, that we are only interested in pursuing traffic flow to learn a model of normal vehicle behaviour. The individual vehicle with its individual trajectory is not of interest. We will show in chapter 4 that we can recognize caught or hopped features as outliers and remove them from the training tracking data-set. Furthermore, a sufficient frame-rate is guaranteed by the frame-grabber. Thus, multi-resolution tracking is always possible.

To sum up, an evaluation of the requirements (table 2.4) which were given in section 2.2 points out three big advantages with respect to the first approach. First, all kinds of vehicles can be detected and tracked. Problems like different sizes of cars, the perspective of the camera and approaching or leaving vehicles are handled. Second, different driving directions are also track-able. Finally, occlusions of vehicles can be handled, which is important for traffic jam recognition.

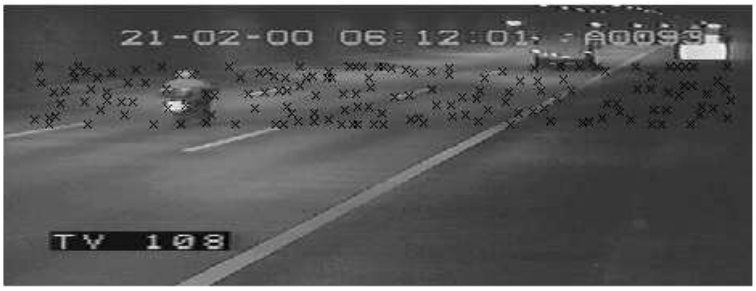
(a)

(b)

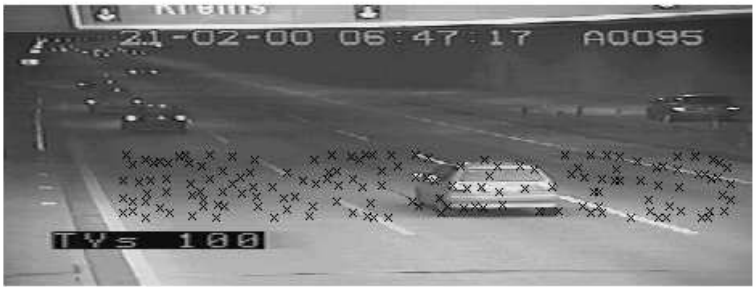
Figure 2.6: TVS100 camera video sequence: Two typical frames are shown.



(a) Image 2.4(b)

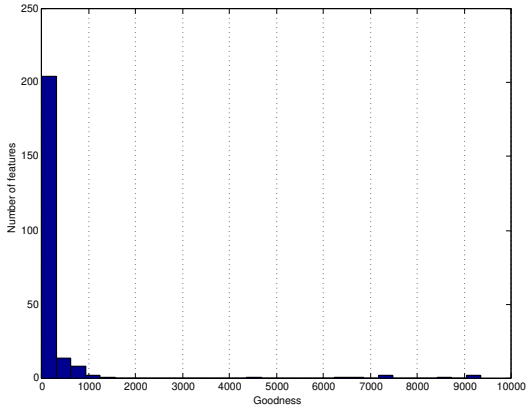


(b) Image 2.5(a)

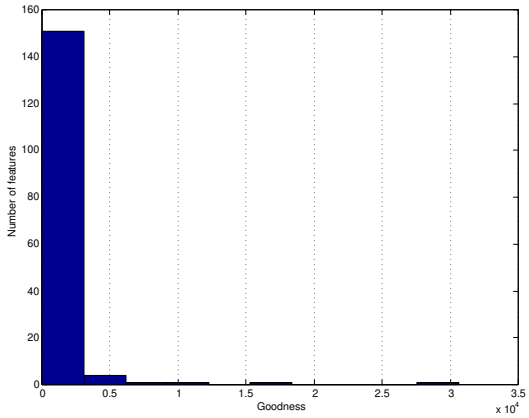


(c) Image 2.6(a)

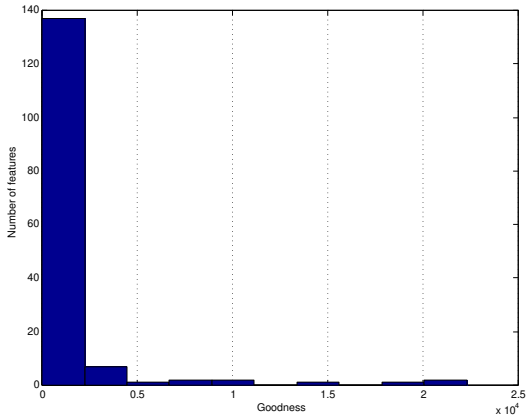
Figure 2.7: One frame of each of the three sequences is shown: All features were selected within a detection of interest (DOI) area without segmentation. A 9×9 search window was used. The frames were smoothed by a Gaussian filter with $\sigma_s = 0.9$. Gradients were also calculated by a Gaussian with a $\sigma_g = 1.0$.



(a) Histogram of figure 2.4(b)

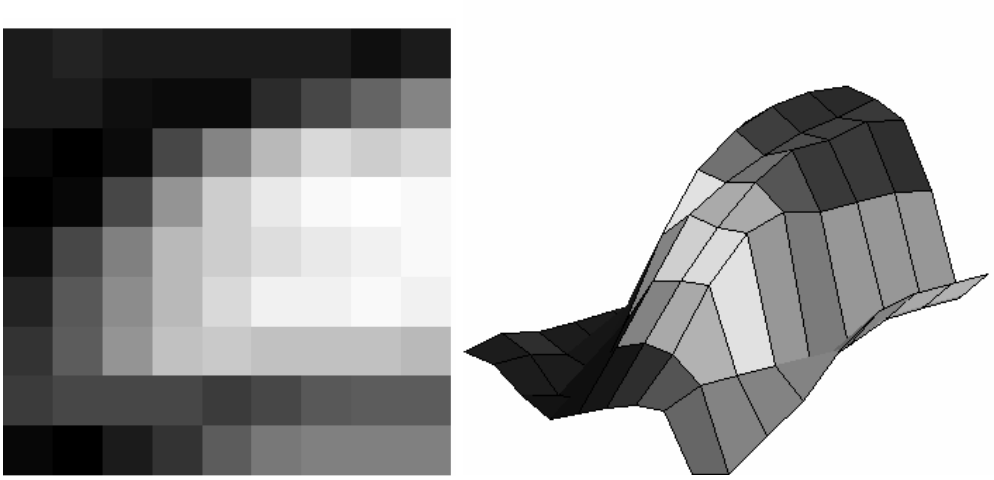


(b) Histogram of figure 2.5(a)

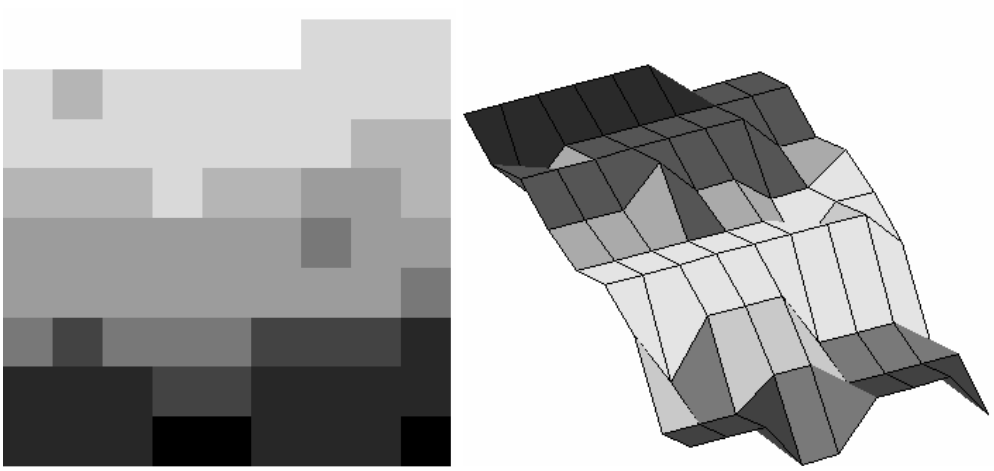


(c) Histogram of figure 2.6(a)

Figure 2.8: Histograms of the number of interest-points according to their goodness values are shown for every camera video sequence. All interest-points lying on the background have a small Goodness value (tall bar). Interest-points with Goodness values which are orders of magnitude larger (here $\approx 10^3$) are features on vehicles.



(a) Feature: Goodness value of 11230



(b) Interest-point on background: Goodness value of 145

Figure 2.9: Two interest-points are compared. The gray-level pattern (left image) and its surface (right image), where the intensity value corresponds to the height, are shown for both points. (a) shows a point with high Goodness value. The corner is obvious. However, (b) shows a point with low Goodness value. Neither a corner nor a strong edge can be recognized.

(a) Image 2.4(b)

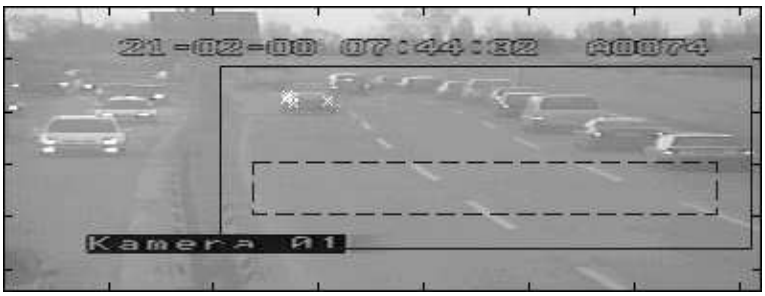
(b) Image 2.5(a)

(c) Image 2.6(a)

Figure 2.10: Only interest-points on vehicles (features) have survived after segmentation (crosses). Threshold λ was set to 1000 in case of (a) and to 5000 in (b) and (c).

(a) Frame 1

(b) Frame 25



(c) Frame 50

Figure 2.11: K01 camera video sequence: The solid rectangle defines the Region Of Interest (ROI), where features are tracked. The dashed rectangle is the Detection region Of Interest (DOI) where the interest-point detector works. (a) shows the detected features in frame 1. (b) shows the feature positions in frame 25 and (c) in frame 50. Although the weather conditions (fog) reduce the frame quality, vehicles can be successfully pursued.

(a) Frame 1

(b) Frame 25

(c) Frame 50

Figure 2.12: TV108 camera video sequence: The solid rectangle defines the Region Of Interest (ROI), where features are tracked. The dashed rectangle is the Detection region Of Interest (DOI) where the interest-point detector works. (a) shows the detected features in frame 1. (b) shows the feature positions in frame 25 and (c) in frame 50. This example shows that beside cars bikes can also be pursued.

(a) Frame 1

(b) Frame 25

(c) Frame 50

Figure 2.13: TVS100 camera video sequence: The solid rectangle defines the Region Of Interest (ROI), where features are tracked. The dashed rectangle is the Detection region Of Interest (DOI) where the interest-point detector works. (a) shows the detected features in frame 1. (b) shows the feature positions in frame 25 and (c) in frame 50. The problem in the gate scene is the day/night lightning change. The example shows that cars can successfully be pursued even under extreme light conditions.

Chapter 3

Unusual traffic event recognition

Contents

This chapter explains unusual events in section 3.1. They are indicated by interest-points which are tracked by the tracker described in the previous chapter. To use especially these features has consequences on the recognition which is described in section 3.2. Finally, section 3.3 describes the principle of unusual traffic event recognition using tracking data of normal traffic. The novelty of our approach is shown.

3.1 Unusual traffic events

Unusual traffic events are incidents caused by abnormal behaviour of vehicles in traffic scenes. Unfortunately, the word ”abnormal” is a fuzzy description of a clear traffic situation. Such traffic incidents can occur in spatial or temporal domain. The former means in a local position of the image plane while the latter equals the time of appearance of vehicles in the scene. Therefore abnormality can be understood as a deviation in either one or both domains from a normal traffic behaviour.

Unusual traffic events	
spatial	temporal
ghost-driver	vehicle obstructions
prohibited U-turn	traffic accident
one-way offence	traffic signal offence
bus lane driving	speeding
occupation lane driving	traffic jam
dividing line offence	driving too slowly
overtaking	vehicle break-down
entering prohibited zones	
prohibited stopping/parking	

Table 3.1: This table gives some examples of unusual traffic events. Classification is done by their occurrence in spatial, temporal or both domains.

Consider the examples in table 3.1 which shows an uncompleted list of possible unusual events. For instance, a ghost-driver who is a person that drives on the wrong side of a highway cannot be discerned by considering the temporal domain. The abnormal behaviour according normal traffic happens by driving in a prohibited direction and therefore in the spatial domain. In contrast to this last example, a traffic jam can only be recognized in time. To use one domain is sometimes not sufficient. Imagine a parking area where parking is forbidden during daytime. An event like parking at noon can only be seen if both domains are taken into account.

Beside the unusual events described in table 3.1, rare incidents can also happen as part of normal traffic behaviour. For example, traffic signals or train crossings are such events. The former change their state periodically where trains approach at known times. Certainly, it could be useful to recognize such events. If we could learn a traffic light cycle, we could detect that cars running the light are unusual, even though their traffic behaviour was not unusual*. However, this is not part of our work and is not considered in the report.

3.2 Features as traffic indicators

Chapter 2 dealt with a visual feature tracking implementation. Features are interest-points which are detected on vehicles and then tracked through the observed scene.

Unlike other tracking methods, it does not detect the vehicles *explicitly* in the scene. For example, Johnson and Hogg [22] used an adaptive background tracker to

*If the signal light is green, then the vehicles will go through.

pursue the blobs of vehicles. Stauffer and Grimson [41] used an active shape model as a feature. Both image features represent and identify vehicles. The assumption that interest-points are on vehicles at detection, only guarantees a representation because it is possible that more than one interest-point is on a vehicle. However, representation by an image feature is a more relaxed concept than identification, with several consequences for the tracker and the unusual traffic event recognition:

(i) The avoidance of identifying vehicles reduces the complexity of the tracking algorithm. Representation is guaranteed by the interest-point detection itself.

(ii) Features can represent different vehicles during tracking. For example, if two cars come close together the feature can hop from one car to the other. They can also represent background for a while or forever if they stuck anywhere. In both cases the unusual event recognition has to handle these outliers.

(iii) Decisions about occurring unusual events cannot be made with one feature.

Despite these consequences, a feature does give a basic answer about the traffic. If the feature behaves normally according to a model of normal traffic behaviour, no unusual event can be concluded at this point in traffic. Otherwise, a traffic problem is evident except the feature is an outlier. Features with this semantics can also be seen as traffic indicators.

3.3 The principle

Unusual events are incidents of abnormal traffic situations. As it was shown in the previous section, these events can happen in spatial, temporal or both domains. If it would be possible to make a mathematical model of this domain behaviour, it would be possible to recognize unusual events. Unfortunately, these incidents occur rarely which makes it practically impossible to model them. Therefore, we model normal traffic. The recognition system learns the patterns of normal traffic behaviour. One ground assumption is that the camera is *static*. A model can only represent one traffic scene. A further requirement is the *availability* of training tracking data of such normal traffic. As normal traffic is easier to observe than abnormal traffic and is basically the usual case, any adequate number of normally tracked features can be found.

Figure 3.1 shows the principle of an unusual event recognition system. Training tracking data is delivered by the tracking algorithm. Unfortunately, the feature's positions are noisy as described in chapter 2. In a first step, preprocessing reduces this noise. Furthermore, outlier features are detected and removed from the training set. Then, the features positions and the time instants when they were measured, generate training samples in a learning space $\mathcal{L}$. Thus, every sample represents normal traffic behaviour in spatial *and* temporal domain. In contrast to our work, the works of Johnson and Hogg and Stauffer respectively did not consider a temporal domain in $\mathcal{L}$.

Therefore, they were unable to detect unusual temporal events. If a sufficient number of samples is given, the probability density function (p.d.f.) in $\mathcal{L}$ defines a spatio-temporal model of normal traffic. Where the p.d.f. tends to be low, the likelihood of abnormal traffic is high and vice versa.

Unsupervised learning is used to generate the spatio-temporal model. It produces a codebook of reference vectors which are representatives of the underlying training samples. The process is generally called "clustering". All previous works described the hard-competitive Kohonen Vector Quantization (Kohonen-VQ) algorithm for learning the codebook. Vector Quantization means replacing a sample in $\mathcal{L}$ by its nearest reference vector. This report proposes the Growing Neural Gas (GNG) algorithm for this task. It is a member of soft-competitive cluster algorithms which brings improvements against algorithms like Kohonen-VQ. It solves the problem of "stranded" reference vectors and it calculates not only the codebook, but also a topology of $\mathcal{L}$.

Unfortunately, the spatio-temporal model is not sufficient to describe normal traffic behaviour. Chapter 6 gives an example. It assesses the behaviour of a feature only in a particular point in $\mathcal{L}$. However, the feature's trajectory is not considered. Therefore, a classification of all possible traffic movements within the static scene is also needed to assess if a feature indicates an unusual event. Johnson and Hogg used the quantized training set in a further Vector Quantization step (re-clustering). The resulting reference vectors represent classes of traffic movements. Stauffer accumulated joint co-occurrence statistics over the codebook. These data is used to perform hierarchical classification. This report presents a new approach. As mentioned, the GNG algorithm generates a topology of $\mathcal{L}$. This graph and the training set can now be used for classification. Interestingly, this graph is created during learning. Only its evaluation ($\rightarrow$ class generation) costs extra efforts.

After learning, a spatio-temporal model and behaviour classes of normal traffic within a static traffic scene exists. Every new, incoming vehicle can now be tracked by corresponding features. The normality of behaviour can now be assessed by comparing the position and time with the spatio-temporal model. Furthermore, the described feature trajectory will be classified. If the probability response of the model is low or a classification could not be done, an unusual event is detected. As features can be outliers, it will be better to trigger a response (e.g. alarm) of the recognition system, if more features with the similar properties (e.g. proximity) indicate an unusual event. This decision is out of scope and is not discussed in the report.

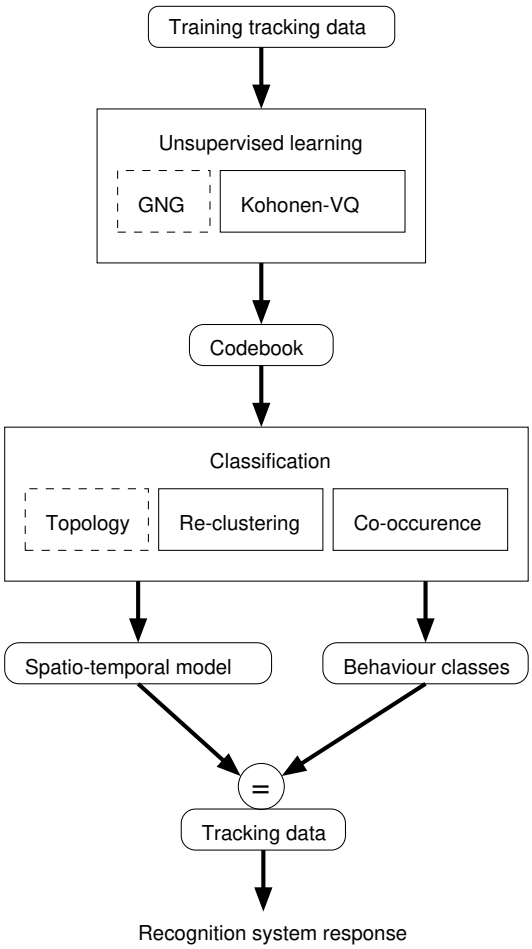


Figure 3.1: The principle of unusual traffic event recognition: In a training phase, training tracking data of normal traffic is used to create a spatio-temporal model of traffic behaviour and behaviour classes. The former is produced by unsupervised learning while the latter is the result of classifying feature trajectories. The Growing Neural Gas (GNG) as new approach (dashed box) is used to create beside a codebook also a topology of the learning space. Then, the latter is used for classification. To recognize an unusual event, the actual tracked features are compared to the model and are classified. If no classification was possible or the model's probability is low, a corresponding response of the system will happen (e.g. alarm).

Chapter 4

Training sample generation

Contents

This chapter treats the generation of training samples in the learning space from measured feature trajectories. The former are used as training samples for successive learning by clustering, which is treated in chapter 5. The latter is produced from measured feature positions of normal traffic scenes by the tracking algorithm described in chapter 3. First, it outlines several difficulties during this generation process with respect to the learning space and problems with the tracking data which have to be tackled to improve learning (section 4.1). Then, solutions for these problems are discussed in sections 4.2 - 4.4.

4.1 Problems

Chapter 3 introduced the notation of the tracking data respectively the learning space. To sum up, the former is given by feature's positions $\mathbf{p}_i = (x_i\, y_i)^\top$ which are measured

at specific time instants t_i. Together they form a temporal ordered set of measurements of feature j, $\mathcal{M}_j = \{(\mathbf{p}_1, t_1) \ldots \prec (\mathbf{p}_i, t_i) \ldots \prec (\mathbf{p}_{n_j}, t_{n_j})\}$ with $j = 1, \ldots, n_{\mathcal{M}}$ where $n_{\mathcal{M}}$ is the number of tracked features available for training samples generation. $\mathcal{M}_j$ is called the measured trajectory of feature j. n_j is the life-time which is the number of frames for which feature j was tracked. The learning space $\mathcal{L}$ is defined as a five dimensional vector space. $\mathbf{s} \in \mathcal{L}$ is called sample which is a vector $(x\ y\ dx\ dy\ t)^\top$. $\mathbf{p} = (x\ y)^\top$ is a feature's position, $\mathbf{d} = (dx\ dy)^\top$ a direction and t a time instant. If $\mathbf{s}$ is related to a feature's measurement of the tracking data it is called training sample. Consequently, for every trajectory $\mathcal{M}_j$ a path exists in $\mathcal{L}$ formed by training samples.

The aim of training samples generation is to *define* the relation between the training samples and its corresponding measurements.

Let $\mathcal{M}$ be a measured feature trajectory of further investigation. To improve reading, measurements $(\mathbf{p}_i, t_i)$ belong to $\mathcal{M}$ and n_j is written as n. In contrast to $\mathcal{M}$, consider a new trajectory

$$\vartheta(t_i) = \mathbf{p}_i. \tag{4.1}$$

ϑ is a continuous, nonlinear vector curve which interpolates all measurements of a feature. Figure 4.1(a) shows an example of tracked feature positions interpolated by ϑ. Thus, the direction a feature takes at $(\mathbf{p}_i, t_i)$ is given by

$$\mathbf{d}_i = \frac{\vartheta'(t_i)}{\|\vartheta'(t_i)\|}, \tag{4.2}$$

which is a unit vector. It is clear that $\mathbf{d}_n$ exists under (4.2). However, if $\mathbf{d}_i$ would be derived from $\mathcal{M}$ which is simply the normalized difference $\frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{\|\mathbf{p}_{i+1} - \mathbf{p}_i\|}$, $\mathbf{d}_n$ would be undefined.

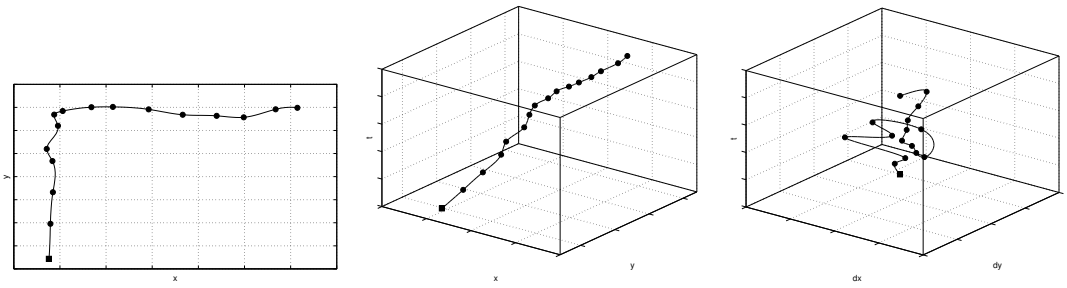
Definition 1 *The relation between a training sample* $\mathbf{s}_i$ *and measurement* $(\mathbf{p}_i, t_i) \in \mathcal{M}_j$ *is*

$$\mathbf{s}_i := \begin{pmatrix} \vartheta(t_i) \\ \mathbf{d}_i \\ t_i \end{pmatrix}.$$

As a result, figures 4.1(b) and 4.1(c) show the generated training samples in sub-spaces of the learning space according to the example trajectory in figure 4.1(a). Together they form a path in the $\mathcal{L}$. t is given by the time when the position of the feature was determined. The squares indicate position respectively sample at $t = 0$.

Unfortunately, several difficulties arise with training samples generation given by definition 1:

Spatial noise: Due to inaccuracies in the tracking algorithm and the quality of image frames the positions of tracked features are noisy. Chapter 2 discussed the reasons. It is clear that feature positions do not lie exactly on their trajectory.



(a) Example of ϑ **(b)** Path in $x/y/t$ **(c)** Path in $dx/dy/t$

Figure 4.1: The definition of training samples by measured feature positions is illustrated. Both are drawn by circles. The square indicates the first position respectively sample.

Consequently, the directions are also noisy, because they relate per definition to their corresponding positions given by equations (4.1) and (4.2). However, time is not considered to be noisy, because it is derived from the number of frames captured by the hardware device which is assumed to be constant*.

Illustration 4.2 depicts the consequences of noise in the learning space. Measured feature positions which lie on similar trajectories (4.2(a)) should be transformed to samples which also lie on similar paths. However, noise results in a sparse distribution of training samples, destroys the path's similarities and introduces outliers. Both can be seen in figures 4.2(b) and 4.2(c). It is clear that the quality of successive learning will be worse by using the training samples without noise reduction.

Learning-space density: The density of a sub-space of the learning space is defined by the number of training samples within it. The more samples are found in a sub-space the more dense it is. Furthermore, the probability of a vehicle occurrence in a certain sub-space is defined directly proportional to the density within the space. The more vehicles are observed in a certain position, direction and time, the more training samples are generated in a sub-space of the learning space. Consequently, this results in higher density and therefore higher probability of vehicle occurrence. Consider a feature on a fast and slow moving vehicle. A feature in the former case will produce far more measured positions than in the latter case, because the frame rate during tracking is constant. Furthermore, if speed is high the training samples are sparsely distributed. On the other hand, if speed is slow they are densely distributed.

However, this example shows that a slow vehicle produce higher probability sub-spaces than a fast one. Illustration 4.3 depicts this fact. Figure 4.3(a) shows three trajectories of fast vehicles and one trajectory of a very slowly moving car

*This real-time constraint can be fulfilled with proper hard- and software

respectively. The parts of the learning space, where the three paths are lying, is underrepresented by training samples (figure 4.3(b) and 4.3(c)). Although in this example, a vehicle occurrence is three times more likely. Besides, all trajectories show changing accelerations of vehicles. Especially, the distances of the positions of one of the fast moving cars are conspicuously unequal. This also influences the density of the learning space unintentionally.

These problems require appropriate solutions. Section 4.2 treats a linear regression model for noise reduction where instead of interpolating all positions by ϑ a "smoother" curve is used which only approximates the $\mathbf{p}_i$. Furthermore, the density problem can be solved by using the idea of re-sampling the trajectories with a constant step size (section 4.3).

Beside the above mentioned problems which lie in the generation of training samples, two more problems arise with respect to the tracking data:

Outliers: It is assumed that the tracking data represents normal traffic. Unfortunately, it is not possible to guarantee this assumption for every feature in the tracking data. Such features are called obvious outliers. Several problems can happen during tracking. For example, a feature can loose a vehicle and stuck for a period of time on the background after it is finally taken out of the tracking area by another vehicle. Furthermore, a feature can hop from one vehicle to another during tracking. These problems can also happen with the same feature.

Therefore, obvious outliers which differ significantly in time from the rest of the tracking data should be eliminated during generation of training samples to improve successive learning. Certainly, this is only possible if the number of features available as tracking data is large enough and the number of outliers within the tracking data is statistically distinguishable. It is also clear, that some obvious outliers will still be represented in the training samples, because it cannot be guaranteed to recognize all disturbed features.

Range of learning space: The values of x and y are by orders of magnitude larger than dx respectively dy. While $\mathbf{d}$ is a unit vector and its elements have values between $[-1, 1]$, $\mathbf{p}$ lies in pixel range, for example $x \in [0, 383]$, $y \in [0, 287]$ which is half of the PAL* resolution. Time lies in all investigated scenes between magnitudes of some seconds up to a minute. Certainly, this time range refers to positions of tracked features which represent normal traffic and are used to generate training samples.

The problem with different ranges of elements of samples is their unbalanced relative contribution during learning. Most of the clustering algorithms use the Euclidean distance. $\mathbf{p}$ would dominate the distance measure while $\mathbf{e}$ would be neglectible, because the range of the values of x and y are orders of magnitude larger than the range of the dx respectively dy.

*Phase Alterning Line is the dominating TV standard in europe.

A solution for obvious outlier detection is given in section 4.4. Remaining outliers, which are represented as training samples in the training set, are the topic of a robust learning algorithm discussed in chapter 5. Finally, the mentioned range problem is discussed in chapter 5, because it is a problem due to the used distance measure during learning.

4.2 Spatial noise reduction via smoothing splines

As noise is introduced during the tracking process the n measured positions do not longer lie exactly on the feature's trajectory $\mathcal{M}$. Equation (4.1) is no longer valid. Consider the following more general linear regression model:

$$\vartheta(t_i) + \epsilon_i = \mathbf{p}_i. \tag{4.3}$$

$\epsilon_i \sim \mathcal{N}(0, \Sigma)$ models the noise in $\mathbf{p}_i$. All ϵ_i are independent random vectors.

ϑ is a continuous, nonlinear vector curve like in (4.1), but is generally unknown. Fortunately, various estimates $\hat{\vartheta}$ can be given which approximates all measurements within the given covariance Σ. Thereby, noise is reduced for all $\hat{\vartheta}(t_i)$. Such curves are called smoothing curves. The statistical problem of estimating an optimal $\hat{\vartheta}$ is called "variational" problem. It goes back to Iso Schoenberg's work. He showed that $\hat{\vartheta}$ is equal to the *natural smoothing spline* which is the solution for the noise reduction problem in this work. Interestingly, it is not only an estimate of a variational problem. If it is considered as a sample of a stochastic process, George Kimeldorf and Grace Wahba showed in two classical works that it even is a Bayes estimate. This connection of variational problems and Bayes estimation has its roots in the work of E. Parzen. All this and more information can be found in [47].

4.2.1 Natural smoothing spline

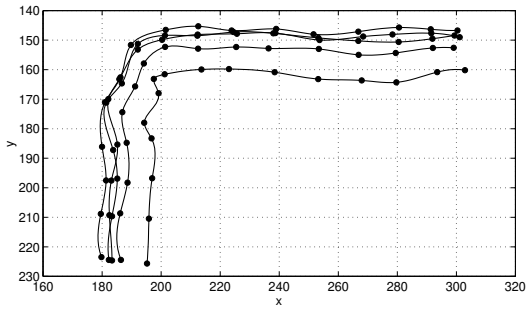
The classical univariate regression spline is a piecewise interpolating polynomial function $f_m : [t_1, t_n] \rightarrow \mathbf{R}^d$. Each polynomial is defined in the intervals $[x_1, x_2), \ldots, (x_i, x_{i+1}), \ldots, (x_{k-1},$ The k points $t_1 = x_1 < t_2 < \ldots < x_k = t_n$ in which these functions are joint together are called knots. f_m fulfills the following four properties:

(i) The $k - 1$ polynomials are of degree $2m - 1$

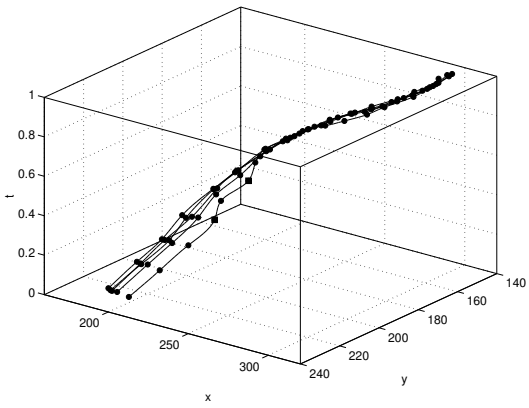
(ii) f_m has $2(m - 1)$ continuous derivatives

(iii) f_m has a $(2m - 1)^{st}$ derivative that is a step function with jumps at $[x_1, x_k]$

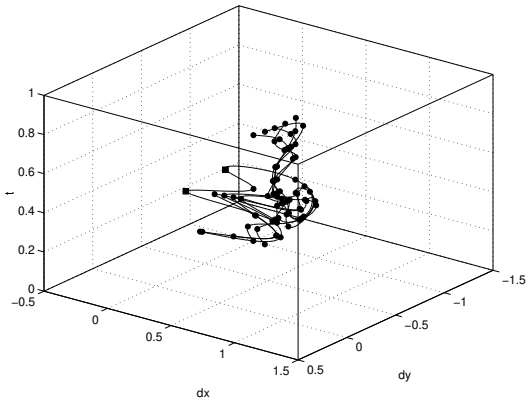
(iv) f_m is a polynomial of degree $m - 1$ outside of $[x_1, x_k]$



(a) Sample trajectories ϑ_j

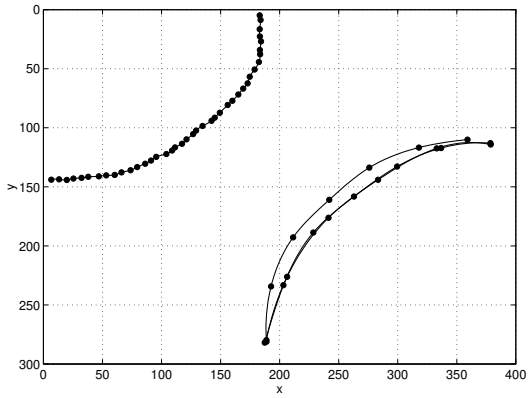


(b) Paths in $x/y/t$

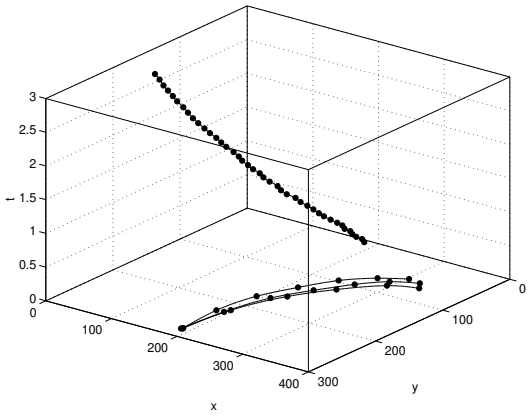


(c) Paths in $dx/dy/t$

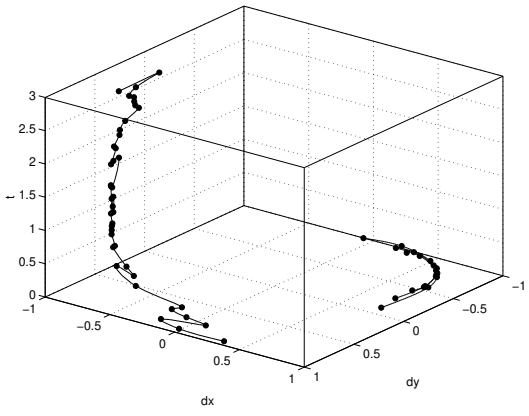
Figure 4.2: Noise influences the training samples in the learning space seriously. Consequently, the quality of learning is decreased. (a) shows sample trajectories which are formed by noisy feature positions of tracked features. (b) and (c) show the corresponding samples. Notice the outliers which are shown as squares. Trajectories respectively paths are drawn by lines whereas circles are positions and samples.



(a) Sample trajectories ϑ_j



(b) Paths in $x/y/t$



(c) Paths in $dx/dy/t$

Figure 4.3: The consequence of slow and fast moving vehicles is shown. (a) depicts four trajectories where one represents a very slow car. (b) and (c) shows the influence on the learning space. Although a vehicle occurrence in the area of the slow car is less likely, the learning space is overrepresented by training samples.

It is clear, that a whole Hilbert space of such functions exists which is called Sobolev space $\mathcal{W}_m = \{f_m\}$.

Furthermore, consider a particular spline $s \in \mathcal{W}_m$ with boundary conditions $s^{(i)}(x_1) = s^{(i)}(x_k) = 0$ for $i = m, \ldots, 2m-1$. Thus, among all $f_m \in \mathcal{W}_m$ with $f_m \neq s$, s minimize for $n \geq m$ the squared integrated m^{th} derivative

$$\int_{x_1}^{x_k} s^{(m)}(t)^2 dt \leq \int_{x_1}^{x_k} f_m^{(m)}(t)^2 dt. \tag{4.4}$$

As a proof, expand the square on the right hand side of the following un-equation

$$0 \leq \int_{x_1}^{x_k} (s^{(m)}(t) - f_m^{(m)}(t))^2 dt,$$

which is always satisfied and rearrange the terms in a useful manner. This leads to

$$0 \leq \int_{x_1}^{x_k} (f_m^{(m)}(t)^2 - s^{(m)}(t)^2 - 2s^{(m)}(t)(f_m^{(m)}(t) - s^{(m)}(t)))dt.$$

It can be shown that the last term of the integral is zero which is the proof of (4.4). Start by formulating it as an integration by parts, then use the fact that $s^{(2m-1)}(t)$ is piecewise constant with a fixed k_{x_i}, the construction is as follows:

$$\begin{aligned}
&\int_{x_1}^{x_k} s^{(m)}(t)(f_m^{(m)}(t) - s^{(m)}(t))dt = \\
&\quad \int_{x_1}^{x_k} s^{(m)}(t) d(f_m^{(m-1)}(t) - s^{(m-1)}(t)) \\
&\quad s^{(m)}(t)(f_m^{(m-1)}(t) - s^{(m-1)}(t))|_{x_1}^{x_k} - \int_{x_1}^{x_k} s^{(m+1)}(t)(f_m^{(m-1)}(t) - s^{(m-1)}(t))dt \\
&\quad \vdots \\
&\quad - \sum_{i=m}^{(2m-2)} s^{(i)}(t)(f_m^{(i-1)}(t) - s^{(i-1)}(t))|_{x_1}^{x_k} - \sum_{i=1}^{k-1} \int_{x_i}^{x_{i+1}} s^{(2m-1)}(t)(f_m^{(m-1)}(t) - s^{(m-1)}(t))dt \\
&\quad \underbrace{- \sum_{i=m}^{2m-2} s^{(i)}(t)(f_m^{(i-1)}(t) - s^{(i-1)}(t))|_{x_1}^{x_k}}_{0} \underbrace{- \sum_{i=1}^{t_{k-1}} k_{x_i}(f_m^{(m-1)}(t) - s^{(m-1)}(t))|_{x_i}^{x_{i+1}}}_{0} \\
&\quad 0
\end{aligned}$$

This unique regression function s is known under the name *natural spline*. The name "spline" was observed by Iso Schoenberg in the ship industry. It comes from the mechanical counterpart used by draftsmen. It was a thin strip that was used to draw curves during the fabrication. Ducks or weights were placed on the strip to force it to go through given points.

As it was mentioned in the beginning of the chapter, a regression spline is not a solution for the estimate $\hat{\vartheta}$, because it interpolates the data. Moreover, an approximating spline was proposed as an estimate of problem (4.3). Instead of (4.4), consider a revised minimization criterion

$$\frac{1}{n}\sum_{i=1}^{n}(\mathbf{p}_i - f_m(t_i))^\top(\mathbf{p}_i - f_m(t_i)) + \lambda\sum_{k=1}^{d}\int_{x_1}^{x_k}(f_m(t)_k^{(m)})^2. \tag{4.5}$$

The first term gives a measurement of the goodness-of-fit while the second term or smoothing term penalizes the roughness of f_m. Generally, both are two conflicting goals to minimize, because the smoother a function the worse it fits the data. This tradeoff is well known in approximation theory. On the one hand, the data should be approximated as good as possible while on the other hand over-fitting should be prevented by the introduction of a smoothing constraint. This process is called regularization.

Regularization also has a Bayesian interpretation. If Σ in (4.3) is known the first term is proportional to the log-likelihood. If we take a prior over functions f_m, minimizing a penalized sum of squares is equivalent to maximizing the posterior density over f_m. More about Regularization can be found in Ripley [36]. The Bayesian aspects are discussed in Wahba [47].

Certainly, the quality of smoothing depends mainly on the smoothing parameter λ which controls the influence of the regularization penalty. If $\lambda \to \infty$, one has a simple linear regression function $s(t) = \beta_0 + \beta_1(t - \frac{1}{2})$. On the other hand, if $\lambda \to 0$, $s(t)$ is an interpolating spline but not necessarily the unique natural spline, because (4.4) is not considered under (4.5) with $\lambda = 0$.

Besides, the number of knots with respect to the number of measurements influences the result. In contrast to regression splines, if many knots are included, the penalty prevents the smoothing spline from over-fitting. On the other hand, too few knots may not be enough to represent the measurements and increase variance.

However, Schoenberg showed again that with a fixed $\lambda \in (0, \infty)$

$$s_\lambda(t) = \arg\min_{f_m} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{p}_i - f_m(t_i))^\top(\mathbf{p}_i - f_m(t_i)) + \lambda\sum_{k=1}^{d}\int_{x_1}^{x_k}(f_m(t)_k^{(m)})^2 \tag{4.6}$$

is a unique optimal minimizer. He called it *natural smoothing spline*. An optimal tradeoff between goodness-of-fit and smoothness can only be given if an estimate of the optimal smoothing parameter $\lambda = \lambda^*$ can be determined by any objective method. Section 4.2.3 shows some possible methods.

4.2.2 Calculation of natural smoothing splines

Remember, that splines were defined as piecewise polynomials which are joint together at given knots. Each polynomial contribute to the whole spline within an inter-

val $[x_i, x_{i+1})$. Outside this interval it has no or a quickly vanishing influence respectively.* Consequently, s can be defined as a linear combination of basis polynomials Φ_i which are called blending functions. This basis spreads out a vector space of possible splines where each s can be expressed by

$$s(t) = \sum_{i=1}^{l} \gamma_i \Phi_i(t). \tag{4.7}$$

l is the total number of free parameters γ_i. Remember, that feature's positions are given in the two dimensional image plane, therefore $d = 2$. Thus, s is a vector spline which approximates all $\mathbf{p}_i$. Then, the coefficients γ_i are two dimensional vectors and can be interpreted as control points of s. Each position $s(t)$ is a linear combination of all control points which are interpolated by s.†

Consider the minimization criterion (4.5) with given feature measurements $(\mathbf{p}_1, t_1), \ldots, (\mathbf{p}_n, t_n)$ and spline s of form (4.7). Let

$$\Phi = \begin{pmatrix} \Phi_1(t_1) & \cdots & \Phi_l(t_1) \\ \vdots & \ddots & \vdots \\ \Phi_1(t_n) & \cdots & \Phi_l(t_n) \end{pmatrix}, \gamma = (\gamma_1^\top \ \cdots \ \gamma_l^\top)^\top, P = (\mathbf{p}_1^\top \ \cdots \ \mathbf{p}_n^\top)^\top.$$

Then, (4.5) can be written as

$$(P - \Phi\gamma)^\top (P - \Phi\gamma) + \lambda \gamma^\top M \gamma, \tag{4.8}$$

where the second term is the "hard part" of this minimization result. It represents the influence of the penalty term of (4.5). M is a $l \times l$ positive definite Hessian matrix with elements

$$M_{ij} = \int_{t_1}^{t_n} \Phi_i''(t) \Phi_j''(t) dt.$$

Consequently, the penalized least squares estimator for the natural smoothing spline is

$$s_\lambda(t) = \sum_{i=1}^{l} \hat{\gamma}_i \Phi_i(t)$$

with minimization constraint

$$(\Phi^\top \Phi + \lambda M)\hat{\gamma} = \Phi^\top P. \tag{4.9}$$

This estimator has a long history and goes back to the results of Reinsch, who has given an explicit solution for cubic smoothing splines ($m = 2$). Further information can be found in Hastie and Tibshirani [43]. It also shows improved estimators like

*As it is shown later, the regression spline forms an exception with respect to the latter while B-splines satisfy this assumption.

†In the later case of high order B-splines they are just approximated.

Marquart's method if (4.7) is nonlinear or ridge regression from Hoerl and Kennard who provided a biased estimator to overcome collinearity.

There are several ways to define the l basis polynomials Φ_i of (4.7). Each choice has a significant influence on computation. For example, Φ_i can be chosen as the $l = d + 1 + k$ basis polynomials of the very popular classical regression spline which can be written as

$$s(t) = \sum_{i=1}^{d+1} \alpha_i t^{i-1} + \sum_{i=1}^{k} \beta_i \max(0, t - x_i)^3. \tag{4.10}$$

Consequently, the basis are the polynomials

$$\begin{aligned} \Phi_1(t) &= 1 \\ \Phi_2(t) &= t \\ &\vdots \\ \Phi_{d+1}(t) &= t^d \\ \Phi_{d+2}(t) &= \max(0, t - x_1)^d \\ &\vdots \\ \Phi_{d+1+k}(t) &= \max(0, t - x_k)^d \end{aligned}$$

The polynomials $\Phi_i(t)$, $1 \leq i \leq d + 1$ define the spline outside the knot interval $[t_1 = x_1, x_k = t_n]$ which was demanded per definition as property (iv) in section 4.2.1. $\Phi_{d+2}(t), \ldots, \Phi_{d+1+k}(t)$ are called truncated power functions and define s within $[x_1, x_k)$. Unfortunately, they span all intervals to the right which somehow violates the assumption that each polynomial should only be defined over a certain one. The consequence is that the matrix on the left hand side of (4.9) is not sparse which results in higher computational complexity. Read chapter 3 of Lancaster and Salkauskas [27] for more information.

An alternative of truncated power functions as basis polynomials is to use $l = k$ functions for every knot which are zero outside of its interval. For example, B-splines have this property. Generally, B-splines are recursively defined by the indicator functions

$$B_{i,1}(t) = \begin{cases} 1 & : \quad t \in [x_i, x_{i+1}) \\ 0 & : \quad t \notin [x_i, x_{i+1}) \end{cases}$$

with

$$B_{i,o}(t) = \frac{t - x_i}{x_{i+o-1} - x_i} B_{i,o-1}(t) + \frac{x_{i+o} - t}{x_{i+o} - x_i} B_{i+1,o-1}(t).$$

Parameter o is the order of the B-splines. It defines the number of control points, given by the knots, which have influence on a certain $s(t)$. For every t there are $l - o$ blending functions $B_{i,k}(t)$ which are zero.

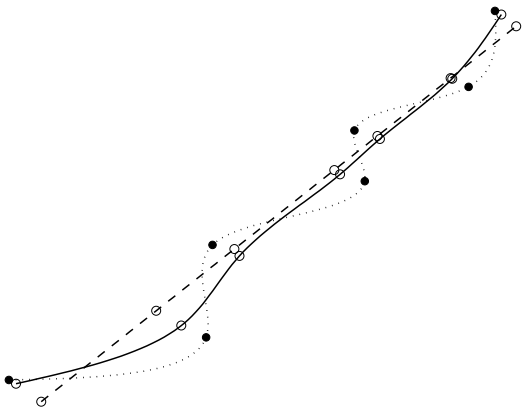


Figure 4.4: Three smoothing functions $s_0(t)$ (dotted), $s_{2\cdot 10^4}(t)$ (solid) and $s_\infty(t)$ (dashed) are shown. The black points are the measured feature positions, while the circles are the corresponding smoothed ones.

Thus for example, the basis according to (4.7) can be written as

$$\begin{aligned} \Phi_1(t) &= B_{1,4}(t) \\ \Phi_2(t) &= B_{2,4}(t) \\ &\vdots \\ \Phi_k(t) &= B_{k,4}(t) \end{aligned}$$

where the polynomials are cubic functions ($o = 4$) that span four intervals. B-splines are nearly orthogonal which results in a numeric stable calculation. Furthermore, it makes $\Phi^\top \Phi$ almost diagonal (banded). This significantly reduces the computational costs and results in fast algorithms. Chapter 11.2.3 in Foley [11] gives a good introduction into B-splines. A classical book which shows the mathematical improvements of the well conditioned B-spline basis against the regression spline basis is deBoor [8].

Generally, cubic ($m = 2$) polynomials are preferred as smoothing splines, because the representation capability is mostly satisfactory and at the same time efficiency in calculation can be achieved.

4.2.3 Estimating the smoothing parameter λ

A good noise reduction result depends mainly on parameter λ which controls the influence of the smoothing penalty term. For example, figure 4.4 shows examples of measurements $\mathbf{p}_i$ and smoothing functions $s_\lambda(t)$. Remember, $\hat{\vartheta}(t)$ was defined in the beginning of section 4.2 by a natural smoothing spline or more precisely, the optimal estimate $s_{\lambda^*}(t)$ of smoothing splines. λ^* is the optimal smoothing parameter. The question is now: How can λ^* be found?

Regularization and thus determining an estimate λ^* can also be seen as the classical problem of minimizing bias and variance of $\hat{\vartheta}$ in (4.3). Therefore, the expected prediction error for i measurements, $1 \leq i \leq n$, is

$$E[(\hat{\vartheta}(t_i) - \mathbf{p}_i)^2] = \underbrace{(E[\hat{\vartheta}(t_i)] - \mathbf{p}_i)^2}_{bias^2} + \underbrace{E[(\hat{\vartheta}(t_i) - E[\hat{\vartheta}(t_i)])^2]}_{variance} \tag{4.11}$$

The bias measures the extent to which the average of the estimate $\hat{\vartheta}$ differs from the measurements, while the variance measures the sensitivity of $\hat{\vartheta}$ with respect to the measurements.

Consider the case where $\hat{\vartheta} = s_\infty(t)$ (figure 4.4, dashed curve). It is clear, that the variance term in (4.11) will vanish, because $E[\hat{\vartheta}(t_i)] = s_\infty(t) = \hat{\vartheta}(t_i)$. However, the bias will be high, because no attention was paid to the measurements. On the other hand, $\hat{\vartheta} = s_0(t)$ is the other extreme. All measurements are interpolated which is shown in figure 4.4 as dotted curve. The bias term vanishes at the $\mathbf{p}_i$, because $E[\hat{\vartheta}(t_i)] = E[\vartheta(t_i) + \epsilon_i] = \vartheta(t_i) = \hat{\vartheta}(t_i)$. In their neighborhood the bias is small. However, the variance can be significant, because $E[(\hat{\vartheta}(t_i) - E[\hat{\vartheta}(t_i)])^2] = E[(\hat{\vartheta}(t_i) - \vartheta(t_i))^2] = E[\epsilon_i^2] = \Sigma$. The bias-variance tradeoff is equivalent to the goodness-of-fit versus smoothness tradeoff controlled by parameter λ. If smoothness is too large then the bias of the predicted trajectory is large. Otherwise, if every measurement is interpolated the variance becomes large and thus also the prediction error. Consequently, the smaller the prediction error is the better the noise reduction will be. More about the bias-variance problem and its minimization can be read in chapter 9.1 of Bishop [7].

The determination of λ^* is a topic for its own. Several methods for estimating an optimal λ^* in this spirit of minimizing the bias-variance were suggested in the literature. Especially Grace Wahba has intensively investigated this issue. A good source of more information about it is chapter 4 of Wahba [47]. Basically, the choice of λ^* depends on weather the covariance matrix Σ of the Gaussian noise process of model (4.3) is given or is unknown.

Consider, Σ is known. For example, several authors have suggested for the one dimensional case ($d = 1$) the so-called discrepancy method. Choose $\lambda = \lambda^*$ so that

$$\frac{1}{n}\|P - S_\lambda\|^2 \approx \sigma^2 \tag{4.12}$$

with $\|y\|^2 = y^\top y$ and

$$S_\lambda = \begin{pmatrix} s_\lambda(t_1) \\ s_\lambda(t_2) \\ \vdots \\ s_\lambda(t_n) \end{pmatrix}.$$

Thereby, the left-hand side of (4.12) is a monotone nondecreasing function of λ. To generalize it to the d-dimensional case, σ^2 is replaced by Σ. (4.12) can now be written

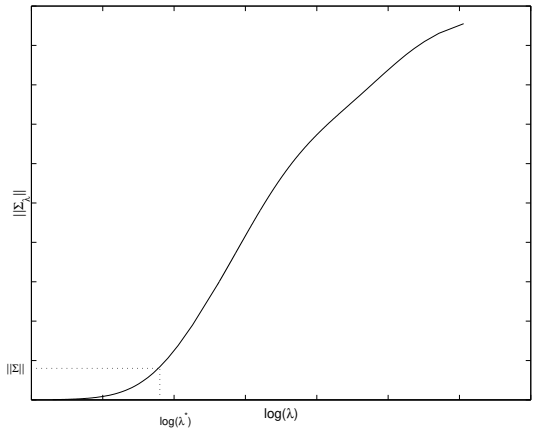


Figure 4.5: Graphical illustration of Σ_λ. λ^* is shown which satisfies $\|\Sigma_{\lambda^*}\| \approx \|\Sigma\|$.

as

$$\Sigma_\lambda = \frac{1}{n}(P - S_\lambda)^\top (P - S_\lambda) \approx \Sigma. \tag{4.13}$$

Figure 4.5 shows an illustration of (4.13). For $\lambda = 0 \to \infty$ the entries of Σ_λ becomes larger until a certain point where $s_\lambda(t)$ is a linear regression.

Unfortunately, Σ is not always known. Then, λ^* must be determined by any objective method. For example, Wahba suggested cross-validation or also known under the name "leave-one-out". The name reflects the idea of omitting a certain measurement $\mathbf{p}_k$ during calculation of $s_\lambda(t)$. Consequently, (4.6) is replaced by

$$s_\lambda^{[k]}(t) = \arg\min_{f_m} \frac{1}{n} \sum_{\substack{i=1 \\ i \neq k}}^{n} (\mathbf{p}_i - f_m(t_i))^\top (\mathbf{p}_i - f_m(t_i)) + \lambda \sum_{j=1}^{d} \int_{x_1}^{x_j} (f_m(t)_j^{(m)})^2$$

Thus, the cross-validation function is

$$CV(\lambda) = \frac{1}{n} \sum_{k=1}^{n} (\mathbf{p}_k - s_\lambda^{[k]}(t_k))^\top (\mathbf{p}_k - s_\lambda^{[k]}(t_k)) \tag{4.14}$$

and an estimate of λ^* is the minimizer of (4.14).

Cross-validation has two disadvantages. First, it is computational expensive. Fortunately, for univariate smoothing splines updating schemes are known (see chapter 4.3 of Ripley [36]). Furthermore, it is not invariant under transformations of the underlying data. To achieve this invariance property a "generalized" version was proposed where the generalized cross-validation function is

$$GCV(\lambda) = \frac{1}{n} \frac{\sum_{k=1}^{n} (\mathbf{p}_k - s_\lambda(t_k))^\top (\mathbf{p}_k - s_\lambda^{[k]}(t_k))}{(\frac{1}{n} Tr(I - A_\lambda))^2}. \tag{4.15}$$

A_λ is the so called influence matrix with $A_\lambda P = S_\lambda$ which maps each position $\mathbf{p}_i$ to the fitted position $s_\lambda(t_i)$. Efficient algorithms for calculating $Tr(I - A_\lambda)$ exist which are based on several matrix decomposition methods (see chapter 11 of Wahba [47]).

Illustration 4.6 shows the result of noise reduction. For every $\mathcal{M}_j$ an estimate $\hat{\vartheta}$ was calculated. m was set to 2 which gives cubic polynomials as basis functions. The parameters λ^* were determined by the discrepancy method with assumed $\Sigma = 9I_2$. Then according definition 1, training samples were generated in $\mathcal{L}$. While figure 4.6(a) shows slightly improvements, figure 4.6(b) compared with 4.2(c) depicts the noise reduction in the direction.

4.3 Density correction via re-sampling

Consider the learning space under definition 1. Unfortunately, the assumption that the learning-space density correspond to the probability of vehicle occurrences can be violated by vehicle situations discussed in section 4.1. The reason is the parameterization by t of ϑ in (4.1) respectively (4.3). Equidistant time instants $t_1, t_2, \ldots$ can result in different dense paths for several trajectories, correspondingly slow and fast moving vehicles. Therefore, consider the following revised trajectory as regression model:

$$\vartheta(s_i) + \epsilon_i = \mathbf{p}_i \tag{4.16}$$

with $1 \leq i \leq n$. Instead of t, a normalized arc length parameterization $s \in [0, 1]$ is chosen which is defined by the Euclidean distance of two consecutive positions. Therefore, all s_i which correspond to $\mathbf{p}_i$ are given by

$$\begin{aligned} s_1 &= 0 \\ s_i &= \frac{s_{i-1} + \|\mathbf{p}_i - \mathbf{p}_{i-1}\|}{\sum_{k=1}^{n} \|\mathbf{p}_k - \mathbf{p}_{k-1}\|}. \end{aligned} \tag{4.17}$$

Certainly, the direction of a feature changes, because it is given under the new trajectory $\vartheta(s_i)$. (4.2) is replaced by

$$\mathbf{d}_i = \frac{\vartheta'(s_i)}{\|\vartheta'(s_i)\|}, \tag{4.18}$$

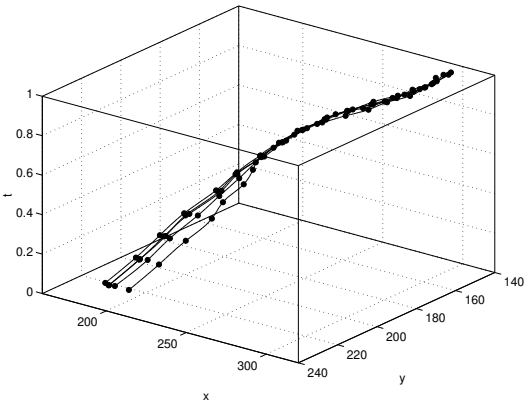
Finally, t also is not longer part of definition 1. It only depends on parameter s and can be written as function

$$\tau(s) = \begin{cases} t_i & : \quad s = s_i \\ t_i + (t_{i+1} - t_i)\frac{s - s_i}{s_{i+1} - s_i} & : \quad s_i \leq s \leq s_{i+1} \end{cases} \tag{4.19}$$

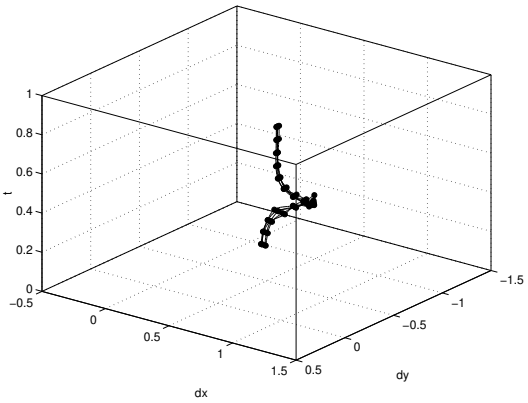
with $1 \leq i \leq n$. τ interpolates t for all positions on ϑ which lie between two measured positions. Thus, a new definition of training samples can be given:

Definition 2 *The relation between a training sample* $\mathbf{s}_i$ *and measurement* $(\mathbf{p}_i, t_i) \in \mathcal{M}_j$ *under learning-space density considerations is*

$$\mathbf{s}_i := \begin{pmatrix} \vartheta(s_i) \\ \mathbf{d}_i \\ \tau(s_i) \end{pmatrix}.$$



(a) Paths in $x/y/t$



(b) Paths in $dx/dy/t$

Figure 4.6: The result of noise reduction is shown. While, compared with figure 4.2(b), (a) shows a slightly difference of the training samples, the noise in (b) is reduced (compare with figure 4.2(c)).

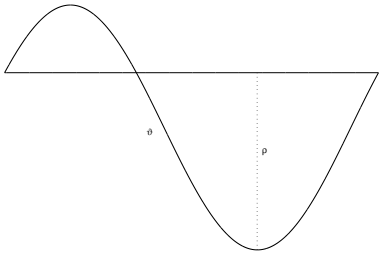


Figure 4.7: The maximum chordal deviation ρ is illustrated.

As the parameterization of ϑ does not depend on t, ϑ can be *re-sampled* with a constant step size δs. The word re-sampling is used, because ϑ was sampled by the tracking algorithm at equidistant time instants given by the frame-rate. δs must be constant for all trajectories. Otherwise the assumption about learning space density would not be fulfilled. But which value should be chosen? In fact, it should be as large as possible, because unnecessary training samples in the learning space do not contribute to learning. However, they only increase computational costs. On the other hand, the generation of too less data points will lead to a path representation where details are lost.

A common recursive algorithm in computer graphics for determining δs for one trajectory ϑ is to calculate the maximum distance from the curve to the line joining the endpoints. Have a look into an article of the Graphics Gems written by Lindgren [28]. They called this maximum distance the chordal deviation ρ of ϑ (figure 4.7). If ρ is smaller than a threshold κ, normally half a pixel, the line will be part of the path. Otherwise, the curve is subdivided into two halves ($0 \leq s \leq 0.5$, respectively $0.5 \leq s \leq 1$). Each half is recursively subjected to the same chordal deviation analysis. An upper bound that ϑ is represented correctly by δs is

$$\delta s \leq \frac{1}{2^r}, \tag{4.20}$$

where r is the recursion depth of the algorithm. As the distance of training samples must be equal for every trajectory, δs is equal for all ϑ. For example, an overall constant step size for training samples generation is

$$\delta s = \min_{1 \leq j \leq n_{\mathcal{M}}} \delta s_j, \tag{4.21}$$

where δs_j is the step size of ϑ_j. However, (4.20) is a rule of thumb. It can happen that all $\mathcal{M}_j$ are linear trajectories. For example, in highway scenes trajectories can be straight lines. Then, δs delivered by equation 4.21 can be too large. Then, another possibility is to choose δs by hand like it is done in Johnson and Hogg [22].

Illustration 4.8 shows the result of re-sampling all four trajectories of illustration 4.3. The analysis of the step size δs delivered a value of 0.0625 with $\kappa = \frac{1}{2}$. Both figures 4.8(a) and 4.8(b) show now an equidistant distribution of training samples compared to figures 4.3(a) respectively 4.3(b). Certainly, noise reduction is also

performed, because model 4.16 is an extension of model 4.3 discussed in the previous section.

4.4 Obvious outliers

Obvious outliers are features in the tracking data which do not represent normal traffic. Two cases can be distinguished:

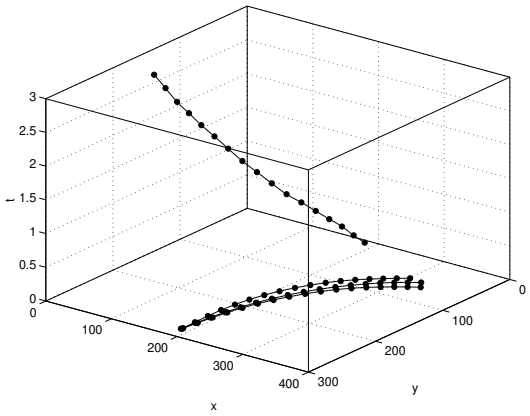
1. features whose life-time differ significantly from the others.
2. features whose measured positions are disturbed due to tracking errors.

The former are in most of the cases tracked features which lost a vehicle during life and stucked on the background for a period of time. It is clear, that no training samples should be generated from such features, because they do not represent normal traffic. Figure 4.9(a) shows training samples of normal features and obvious outliers. The latter can be easily seen by their conspicuous distance from the rest of training samples.

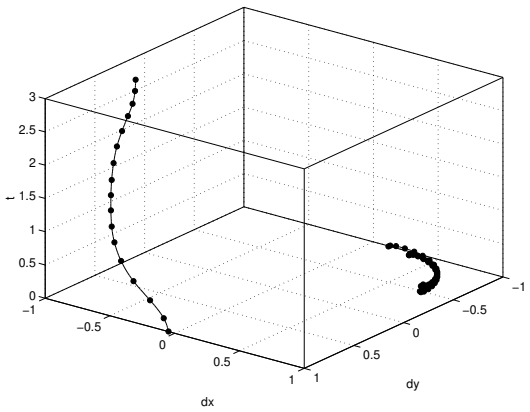
Statistics can help to detect these temporal obvious outliers. However, the problem can also be seen as a *selection* of those features whose training samples represent normal traffic. Consider figure 4.10 which illustrates the histogram of the life-time of every feature. As it can be seen, most of the features are distributed around a certain life-time, because all features take similar ways and therefore approximately the same time from detection until they vanish outside of the tracking area. A feature should be selected if at least κ features have survived the same life-time. The "same life-time" means in this context within the same histogram bin. On the other hand, a feature is an outlier if less than κ features with the same life-time can be found in the training set. Certainly, this feature selection can only be done if statistically more "normal" tracked features than outliers are available. Figure 4.9(b) depicts the result of feature selection. All obvious outliers are removed.

Certainly, the outlier problem can also happen in the spatial domain of $\mathcal{L}$. For example, during collection of tracking data it can happen, that some vehicles behave abnormal. They stop or drive in wrong directions. This are exactly those things which this system should recognize. Unfortunately, such situations cannot be excluded during the generation of tracking data. Furthermore, it is not possible to detect these outliers in one domain, for instance in x. They can only be found in the spatial sub-space of $\mathcal{L}$, where they differ from the rest of the training samples. Therefore, outlier detection cannot be done as a preprocessing step. It must be done during learning. How this is done, is discussed in the next chapter 5.

The second case of obvious outliers are features whose measured positions are disturbed as a result of tracking errors. Imagine a disturbed measurement $(\mathbf{p}_i, t_i) \in \mathcal{M}$. It is clear, that the whole trajectory does not represent normal traffic. Therefore, it must

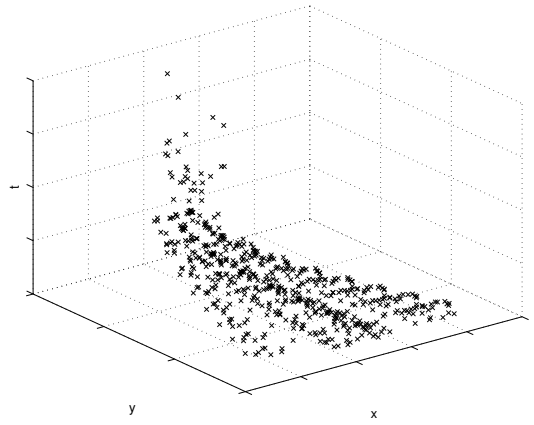


(a) Paths in $x/y/t$

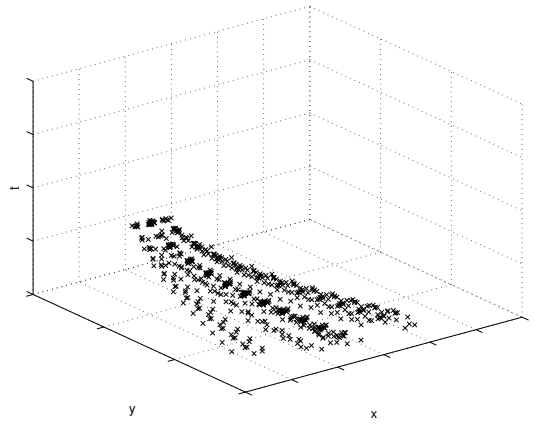


(b) Paths in $dx/dy/t$

Figure 4.8: The result of re-sampling is shown. (a) depicts the training samples in a subspace $x/y/t$ of the learning space while (b) shows $dx/dy/t$. Compared to illustration 4.3 the training samples are equally distributed.



(a)



(b)

Figure 4.9: The feature selection result is illustrated in sub-space $x/y/t$ of $\mathcal{L}$. (a) shows obvious outliers. (b) shows the selected features.

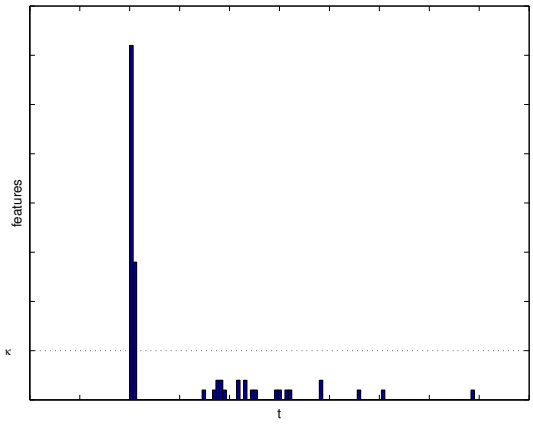


Figure 4.10: Histogramm of the life-time of tracked features.

not be represented as path in $\mathcal{L}$. An idea for rejecting such $\mathcal{M}$ is to prove, if every $(\mathbf{p}_i, t_i)$, which corresponds to $\hat{\vartheta}(s_i)$ with $\tau(s_i) = t_i$, lies within a certain confidence interval. Wahba [47] suggests a Bayesian confidence interval in the one dimensional case with known σ^2 and thus estimated λ^*

$$[\hat{\vartheta}(s_i) - u_{\frac{\alpha}{2}}\sqrt{\sigma^2 a_{ii}(\lambda^*)},\ \hat{\vartheta}(s_i) + u_{\frac{\alpha}{2}}\sqrt{\sigma^2 a_{ii}(\lambda^*)}], \tag{4.22}$$

where $a_{ii}(\lambda^*)$ is the $(ii)^{th}$ entry of the influence matrix A_{λ^*}. $u_{\frac{\alpha}{2}}$ is the $\frac{\alpha}{2}$ point of the normal distribution. α gives the probability that the true value $\vartheta(t_i)$ and the measurement $(\mathbf{p}_i, t_i)$ are covered by the interval.

A suggested confidence interval for the two dimensional ($d = 2$) case is the ellipse

$$(\mathbf{p}_i - \hat{\vartheta}(s_i))^\top \Sigma^{-1} (\mathbf{p}_i - \hat{\vartheta}(s_i)) \leq c^2. \tag{4.23}$$

with Σ the covariance of model (4.3). The left hand side of (4.23) is the Mahalanobis distance of $\mathbf{p}_i$ from $\hat{\vartheta}(s_i)$ and is chi-square distributed. The probability that the measurement $(\mathbf{p}_i, t_i)$ is covered by the ellipse is $(1 - \alpha)$. Then, c^2 is the upper $(100\alpha)^{th}$ percentile of a chi-square distribution with two degrees of freedom. Generally, it has d degrees of freedom. Figure 4.11 illustrates the usage of a confidence interval given by (4.23) to detect an obvious outlier. True positions of a feature are shown as crosses, while measurements are plus signs. Measurements were generated according to the model (4.3) with $\Sigma = 25I_2$ from true given positions. The estimated trajectory $\hat{\vartheta}$ is shown by the curve. All values are listed in table 4.1.

For example, consider a measurement $(145\ 132\ 4)^\top$ which is shown as square. $\hat{\vartheta}(4)$ has the value $(140.89\ 176.1)^\top$. Under a probability of 0.95 of interval coverage, $c^2 = 5.991$. Thus,

$$\left(\begin{pmatrix} 145 \\ 132 \end{pmatrix} - \begin{pmatrix} 140.89 \\ 176.1 \end{pmatrix}\right)^\top \begin{pmatrix} 25 & 0 \\ 0 & 25 \end{pmatrix}^{-1} \left(\begin{pmatrix} 145 \\ 132 \end{pmatrix} - \begin{pmatrix} 140.89 \\ 176.1 \end{pmatrix}\right) = 78.48$$

$$78.48 \not\leq 5.991$$

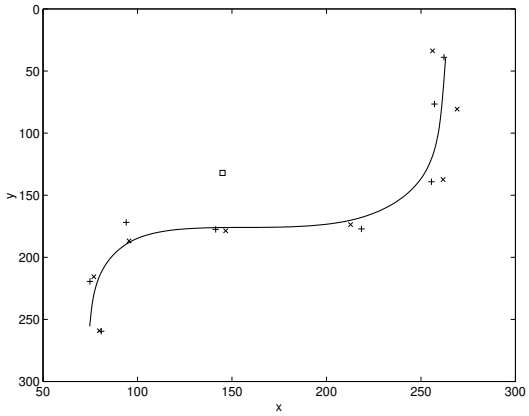


Figure 4.11: An outlier trajectory is illustrated.

time	true positions		measured positions		estimated positions	
$[frame]$	$x[pixel]$	$y[pixel]$	$x[pixel]$	$y[pixel]$	$x[pixel]$	$y[pixel]$
1	79.81	259.08	80.75	259.53	74.70	255.52
2	76.87	215.65	74.84	219.38	79.03	218.14
3	95.51	186.79	93.99	171.88	100.63	184.27
4	146.67	178.59	141.33	177.49	140.88	176.10
5	212.75	173.64	218.52	177.11	215.86	168.98
6	261.70	137.38	255.57	139.15	249.23	138.28
7	269.14	80.66	257.16	76.57	260.94	79.15
8	256.11	33.71	262.17	38.89	263.01	39.56

Table 4.1: An example feature: True, measured and estimated positions are shown.

is not fulfilled which leads to the conclusion that the measurement is wrong and the whole feature is an outlier.

Chapter 5

Learning spatio-temporal traffic behaviour

Contents

This chapter treats learning a spatio-temporal model of traffic behaviour. It is shown that the latter is an approximation of the probability density function in the learning space which is learnt by a density estimator. The training samples, which are necessary for learning, are generated as discussed in chapter 4. Section 5.1 introduces the learning problem in general and gives some definitions. Then, section 5.2 introduces the basic learning paradigms and gives solutions. Especially, unsupervised

competitive learning is treated in detail in section 5.3. Finally, section 5.4 presents the proposed learning framework for this work.

5.1 A definition of learning

What makes a machine a learning machine in particular? For example, Herbert Simon tried to give a definition of machine learning:

> *Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more effectively the next time.*
>
> *(Herbert Simon, Carnegie Mellon University)*

Consider, that this definition has a qualitative meaning. A change, which is triggered by a learning step, leads always to a more effective system. However, imagine a mobile robot which explores its environment in an empty room with one open door. Its task is to find the door. Many learning steps will happen which will not bring the robot closer to the door. Hence many changes of the robot's behaviour are not effective but certainly important, because they help the robot to explore the room. It can exclude parts of the room where it can be sure that there is no door. In the end it will find its goal and will exit the room. Consequently, the following more rational definition by Nils Nilson gives a better understanding what machine learning does:

> *A learning machine is a device whose actions are influenced by past experiences.*
>
> *(Nils Nilson, Stanford University)*

Thus, Nils Nilson suggests that learning is only driven by past experiences which explains the learning behaviour of the mobile robot. However, both definitions do not assume any understanding of the problem domain or problem relations. Furthermore, no implications to state change dependencies of the machine are done. For example, a simple regression function of a finite data set of points in the plane is a learning machine. Learning means in this context to find an optimal regression function by minimizing an error measure, i.e., the sum of squares of the distance from data-points to the function.

Consider now our problem of learning normal traffic behaviour. We suggest the following definition:

> *By learning traffic behaviour we denote the estimation of a probability density function (p.d.f.) in the learning space which is generated by training samples of normal traffic. Thus, the p.d.f. is a spatio-temporal model of normal traffic patterns.*

Interestingly, the glamorous term ”learning” looses its fascination in this definition and can be simply described by a rational optimization problem of model fitting. Section 5.4 will discuss the proposed solution for traffic behaviour learning in detail.

Besides, a second question arise: ”How is learning done?”. Generally, two principles of learning are conceivable, namely *static and instantaneous learning* (Fritzke[15]). Traffic behaviour learning is an example of the former. Thereby, a given finite set of training samples are available. The learning process derives a hypothesis of the data. For example, the parameters of the regression function are estimated. The result is then a knowledge about the structure of the training samples. Certainly, the number of training samples is critical. Too few samples can lead to poor results. Otherwise, every new sample would not dramatically influence the learning result. Human beings can also show static learning behaviour. Many skills like walking, swimming or simply grabbing a cup of coffee have to be learnt in a way of experience the same situation many times. On the other hand, instantaneous learning is the ability to learn from one single event. For example, most of us learnt the meaning of ”hot” and ”cold” by associating these terms with unique experiences. Many babies learn what hot means by touching a hot plate. Furthermore, face learning happens instantaneously in human beings. We do not need to see people hundreds of times to be able to recognize them. Indeed, one meeting with strange people is often enough for someone to recognize them as at least seen or vague known the next time.

5.2 Learning paradigms

Learning can be driven by a teacher in a *supervised* manner or it is done *unsupervised*. The choice either to learn by the former or latter paradigm depends strongly on the application. Both paradigms and their resulting learning methods are discussed in the following two sections.

Furthermore, learning methods can be *performed* in different ways:

Batch vs. incremental learning: In batch learning, every training sample is evaluated after each other and then an adaption step is done. This learning process is iteratively repeated. On the contrary, in an incremental learning process training sample evaluation and adaption is performed after each sample. An intermediate learning process is also possible where two or more but not all training samples are evaluated and then an adaption step is done.

Off-line vs. On-line learning: In off-line learning, all the training samples are stored and can be accessed repeatedly. Batch learning is always off-line. In on-line

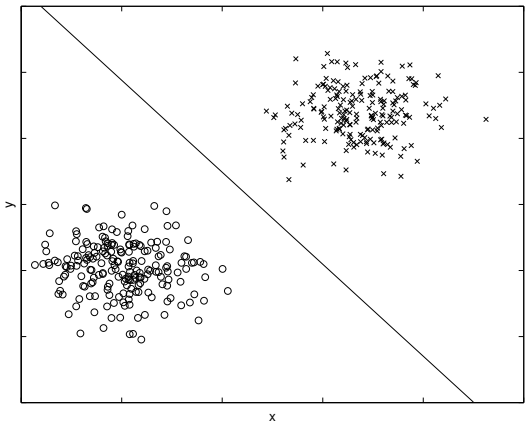


Figure 5.1: A supervised learning task is illustrated. Samples on a plane are given which are either of class "$\circ$" or "$\times$". A linear decision function can be learnt to distinguish between these two classes.

learning, each training sample is learnt immediately and discarded afterwards. On-line learning is always incremental while incremental learning can be done either on-line or off-line.

Adaptive learning: This paradigm refers to learning machines which can forget the past when it is not longer relevant. Thus, they can track changing environments (see adaptive vector quantization in Gersho and Gray [16]). Practically, this means that the learnt data is completely discarded or adapted during learning new training samples respectively.

Constructive vs. destructive learning: Learnt data is stored in fixed data structures and is adapted by the learning algorithm. It is conceivable that these structures dynamically grow or shrink during learning. The former is called constructive learning while the latter is known as destructive learning or learning by pruning in literature.

5.2.1 Supervised learning

For example, consider a classification task which is depicted in figure 5.1. Points in the plane belong either to class "$\circ$" or "$\times$". The membership of every training sample is known a priory which is called *target*. The aim of learning is to distinguish between these two classes and classify each new point. The straight line (decision function) is the learnt result which divides the plane in two parts accordingly the two classes. All points that lie above the function are assigned to class "$\times$" and below to class "$\circ$". It is clear, that this is a supervised learning paradigm.

Supervised learning and especially the application of *classification* has a major relevance in machine learning. The development of neural networks (NN) has con-

tributed to this importance. A neural network consists of simple processing elements which are often called neurons. These neurons are connected to each other by a network of links. Furthermore, they can calculate a single value by a predefined function from their inputs and put them on their output connections. Thereby, a sample which is put onto the input produces its target value on the final output of the NN. Each link is associated with a weight. Learning means updating these weights as long as the NN does not produce the given target value within a small deviation on its output for every training sample. Although neural networks have increased the popularity of supervised learning and leaded to the development of various applications in the last two decades, they are not more powerful than statistical methods like discriminant analysis or regression (Vapnik [46]). Their great success lies maybe in their better understandability. More about neural networks can be read in Bishop [7].

Indeed, they are not the only supervised learning technique. Many other methods exist like decision trees or version space learning. The choice also depends on the type of data which can be numeric in the case of sensor values like a temperature sensor or symbolic. An example for symbolic data is the gender of a person which can either be male or female. A good introduction into the basics of supervised learning is Russell and Norvig [37].

5.2.2 Unsupervised learning

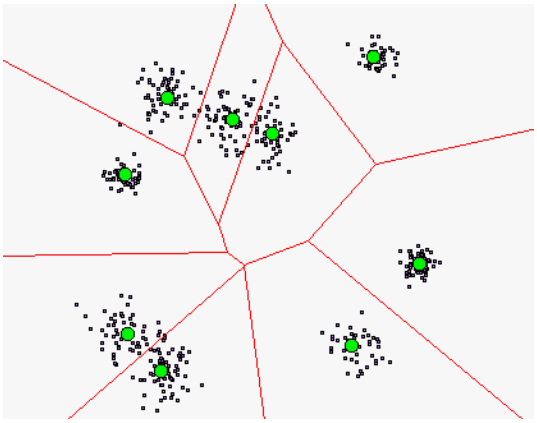
Unsupervised learning allegedly involves no targets. Instead of learning a classification of the data with apriori knowledge of each training sample's classification, the classes are determined by the distances between training samples. Typically, the Euclidean distance is used as a distance measure. Although, other distance measures can also be used. In unsupervised learning jargon classes are called clusters and the so-called *cluster analysis* is a well known field in statistics which goes back to the sixties. Forgy [12] and MacQueen [29] have proposed solutions for clustering which are known as k-means algorithms today. Thereby, k clusters $\mathcal{L}_i, 1 \leq i \leq k$ are searched in the learning space $\mathcal{L}$. The number of clusters k is apriori given. Cluster centers $\mathbf{w}_1, \ldots, \mathbf{w}_k$ are assigned to each cluster which partition $\mathcal{L}$ into regions - the so-called *Voronoi*[*] *regions*. Any sample $\mathbf{s}$ belongs to cluster $\mathcal{L}_i$ if

$$i = \underset{j \in \{1,\ldots,k\}}{\arg\min} \|\mathbf{s} - \mathbf{w}_j\|. \tag{5.1}$$

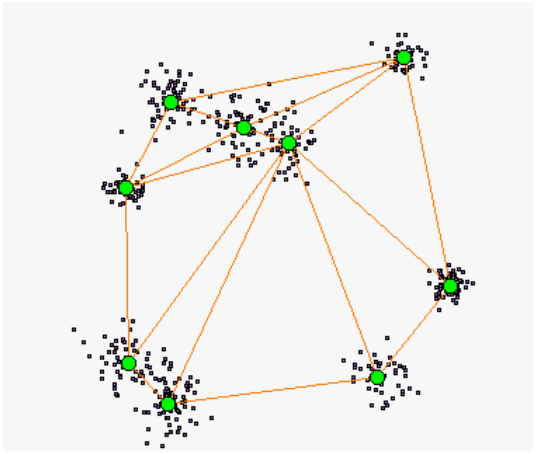
In other words, any sample is assigned the cluster with its nearest cluster center. The graph which connects all cluster centers is called *Delaunay*[†] *triangulation*. Figure 5.2 illustrates the Voronoi regions and the Delaunay triangulation.

[*] M. G. Voronoi, German mathematician

[†] Boris Delaunay, Russian mathematician



(a) Voronoi regions



(b) Delaunay triangulation

Figure 5.2: An example of cluster analysis is shown. (a) depicts the Voronoi regions defined by the cluster centers (b) depicts the corresponding Delaunay triangulation.

The goal of k-means clustering is to minimize the error function

$$\sum_{i=1}^{k} \sum_{\mathbf{s} \in \mathcal{L}_i} \|\mathbf{s} - \mathbf{w}_i\|^2 \tag{5.2}$$

under the given training samples. Consequently, the cluster centers are optimal positioned within $\mathcal{L}$ in a least squares sense. In fact, this constraint is not fulfilled after a first initialization step where cluster centers can have any position in $\mathcal{L}$. Now, the idea of k-means clustering is to replace every cluster center by the mean of all samples within its cluster. Thus, a k-means learning law is given by

$$\mathbf{w}_i \leftarrow \frac{1}{|\mathcal{L}_i|} \sum_{\mathbf{s} \in \mathcal{L}_i} \mathbf{s}. \tag{5.3}$$

This learning law guarantees convergence of the k-means algorithm. However, the optimum of the error function 5.2 is not necessarily a global optimum.

Forgy's k-means variant assigns samples and update the cluster centers simultaneously. Thus, it is a batch and off-line learning algorithm. In contrast to that, MacQueen's incremental, on-line algorithm assigns each sample and updates the cluster centers alternately. Unfortunately, k-means tend to stuck in local minima of the error function. Furthermore, the cluster centers converge to the mean of their positions during learning, because they are adapted by the mean of cluster samples. Therefore, the k-means algorithm forms an approximation to the normal mixture model (McLachlan and Basford [32]), assuming that the mixture components (clusters) all have spherical covariance matrices and equal sampling probabilities. Balakrishnan, Cooper, Jacob and Lewis [1] found further that the k-means algorithm used as normal-mixture approximation, recover cluster membership more accurately than Kohonen networks which were developed by Teuvo Kohonen in the field of *vector quantization* and are discussed in the next section.

Vector quantization is an important application of unsupervised learning algorithms. It is a data compression technique which has gained relevance in the field of signal processing (Gersho and Gray [16]). The task is to code given signal values $s \in \mathcal{L}$ by a codebook of reference vectors $\mathcal{C} = \{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$. To gain a compression effect, m should be chosen smaller than n ($m < n$). These reference vectors should now be positioned in a way, that the expected quantization error

$$\int_{\mathbf{s} \in \mathcal{L}} \epsilon(\mathbf{s}, \mathcal{C}) p(\mathbf{s}) d\mathbf{s} \tag{5.4}$$

becomes minimal. Thereby, $\epsilon(\mathbf{s}, \mathcal{C})$ is the quantization error between $\mathbf{s}$ and a fixed configuration of reference vectors in $\mathcal{C}$. Normally, the quantization error is defined as the Euclidean difference $\|\mathbf{s} - \mathbf{w}_i\|$ with $\mathbf{w}_i \in \mathcal{C}$ which is the nearest reference vector in $\mathcal{C}$. As mentioned in the beginning of this chapter, different distance measures are possible. Furthermore, p is the probability density function of the data in $\mathcal{L}$. Although

p is generally not known except for the training samples, (5.4) can be approximated by

$$\frac{1}{|\mathcal{L}|}\sum_{i=1}^{m}\sum_{\mathbf{s}\in\mathcal{L}_i}\|\mathbf{s}-\mathbf{w}_i\|^2. \tag{5.5}$$

Consider, that the quantization error is squared, because this leads to better numerical properties under minimization of (5.5). Indeed, the quantization error of the Vector Quantization problem and the error function of cluster analysis only differ by $\frac{1}{|\mathcal{L}|}$. It is clear, that constants can be neglected if both error functions are minimized. Therefore, the k-means algorithm can also be used in Vector Quantization to create an optimal codebook.

Since the publications of Forgy and MacQueen many people have developed variants of the above mentioned classical k-means algorithm (Ripley [36]). For example, Ball [2] has developed a Forgy variant which he named ISO-DATA algorithm. It allows additional splitting and merging of clusters. Bezdek [4] proposed a fuzzy k-means variant where every sample is not clearly assigned to one cluster. Instead, it is assigned in the spirit of fuzzy sets to all clusters by a fuzzy membership vector which lead to a more general view of cluster membership.

5.3 Unsupervised competitive learning

Teuvo Kohonen, who has been one of the most famous researchers in Neurocomputing in the past decades, invented several other unsupervised learning methods. All have the same property. Reference vectors of a finite codebook compete for a single training sample. The one which is the nearest in its distance wins, the second closest is the second and so on. Thus, a ranking of them can be given. During *hard-competitive* learning only the winner is moved closer to the training sample while all other reference vectors of the codebook remain unchanged. This type of learning is also known under the name *winner-takes-all* learning. On the contrary, if all reference vectors are moved in a decaying manner of its rank, learning is called *soft-competitive* or *winner-takes-most*. The latter has the advantage that it is more robust against local optima during minimization of the quantization error. Thus, different initializations of the codebook do lead to equal or similar results.

Unfortunately, most of Kohonen's competitive learning methods are called "Kohonen networks", because they are realized in a neural network's style. This often leads to confusion by many people. Basically, Kohonen networks most often refers to one of the following three networks:

Kohonen Vector Quantization (Kohonen-VQ): Actually, the name of this method is "learning vector quantization". Unfortunately, variants of the learning vector quantization methods are supervised methods. To avoid confusion, the unsupervised, hard-competitive learning vector quantization is renamed Kohonen-VQ

throughout this work. As Kohonen-VQ is part of our proposed solution, it is discussed in section 5.3.1.

Self-Organizing Map (SOM): The idea of inventing SOMs was biological motivated and came from the way in which various human sensory impressions are neurologically mapped into the brain (Kohonen [25]). A SOM is a soft-competitive network of reference vectors which are arranged in one or higher dimensional maps. Furthermore, they exist in an grid space which is completely different to the learning space where the dimension is smaller than the dimension of $\mathcal{L}$. A SOM provides a topological mapping from the learning space to the clusters which it represents. Therefore, it is a specific type of clustering algorithm. SOMs provide a solution for *dimensional reduction* of large spaces. Nevertheless, its main application is cluster analysis.

Learning Vector Quantization (LVQ): On the contrary to all so far mentioned Kohonen networks, LVQ is a supervised version of Kohonen-VQ (Kohonen [23], Ripley [36]). Thus, it has nothing to do with unsupervised methods and only shows the imprecision in the Neurocomputing field. However, the learning algorithm is equal to Kohonen-VQ, but the LVQ learning law is quite different, namely supervised.

A further competitive learning method is Competitive Hebbian learning (Martinez [30]) which is a variant of Hebbian learning (Oja [33]). In contrast to Kohonen networks, the aim is not to learn a codebook. Instead it is an elegant method to provide the codebook with topological structure which reflects the topology of the training set. Competitive Hebbian learning is discussed in more detail in section 5.3.2.

A problem which arises in many unsupervised methods is the fixed number of reference vectors. Thereby, this number depends on the distribution of the data. Most methods have the problem that either too few or many reference vectors are given apriori which leads to erroneous vector quantization. Fritzke [13] tried to avoid this problem by introducing a growing network of reference vectors. The Growing Neural Gas is a data-driven growing algorithm where new reference vectors are introduced in the learning space. Remarkably, they are not introduced randomly. However, accumulated statistics about the quantization error leads this insertions. As this learning method is part of the learning solution of traffic behaviour, it is explicitly discussed in section 5.3.3.

Certainly, those mentioned soft-competitive unsupervised methods are not the only ones. Like SOMs with fixed network dimensionality, Growing Cell Structures and Growing Grids (Fritzke [14]) have also been proposed. On the other hand, Martinetz and Schulten [31] developed another soft-competitive learning method without fixed network dimensionality which they called Neural Gas.

5.3.1 Kohonen Vector Quantization

Kohonen-VQ is a hard-competitive learning method where the winner $\mathbf{w}_s$ of the codebook is moved a certain proportion of the distance between it and the corresponding training sample s. This proportion is specified by a learning rate λ and gives the Kohonen learning law which can be written as

$$\begin{aligned} \mathbf{w}_s &\leftarrow \mathbf{w}_s + \lambda(t)\|\mathbf{s} - \mathbf{w}_s\| \\ \mathbf{w}_i &\leftarrow \mathbf{w}_i : i \neq s, \forall i \in \{1, \ldots, m\}. \end{aligned} \tag{5.6}$$

All other reference vectors remain the same. Consider, that $\lambda(t)$ decays over the number of learning steps. If it is fixed, Kohonen-VQ will not converge to an optimum of the error function. As it is well known in approximation theory, convergence requires $\sum \lambda(t) > \infty$ and $\sum \lambda(t)^2 < \infty$ (Kohonen [25], page 35). Otherwise, convergence is not necessarily guaranteed. Therefore, as time during learning goes to infinity, the learning rate decays in a suitable manner.

Let t_i be the number of times where $\mathbf{w}_i$ was the winner according to the so far presented training samples. Thus, the number of learning steps is

$$t = \sum_{i=1}^{m} t_i. \tag{5.7}$$

Furthermore, let $\mathbf{w}_j(t_i)$ be the position in the learning space of the j^{th} reference vector in $\mathcal{C}$ after its t_i^{th} adaption. The initialization of the codebook at $t_i = 0, \forall i \in \{1, \ldots, m\}$ is always given by random before the first adaption or in the successive steps by previous adoptions.

At learning step t a training sample $\mathbf{s}$ is presented the algorithm. It determines the nearest reference vector $\mathbf{w}_s(t_s)$ by evaluating

$$s = \underset{i \in \{1,\ldots,m\}}{\arg\min} \|\mathbf{s} - \mathbf{w}_i(t_i)\|. \tag{5.8}$$

Remember, the learning law (5.6) describes the adaption of the winner $\mathbf{w}_s(t_s)$. It is moved towards s where the learning rate $\lambda(t_i)$ defines the amount of adaption. It decays with the number of adoptions for each reference vector and is defined by

$$\begin{aligned} t_i &= t_i + 1 \\ \lambda(t_i) &= \frac{1}{t_i} \end{aligned} \tag{5.9}$$

as a harmonic sequence. In every time instant t the positions of the reference vectors are quasi stationary. Thus, the resulting learning law is

$$\mathbf{w}_s(t_i + 1) = \mathbf{w}_s(t_i) + \lambda(t_i + 1)(\mathbf{s} - \mathbf{w}_s(t_i)). \tag{5.10}$$

All other reference vectors remain unchanged.

The complete Kohonen-VQ algorithm can be summarized as follows:

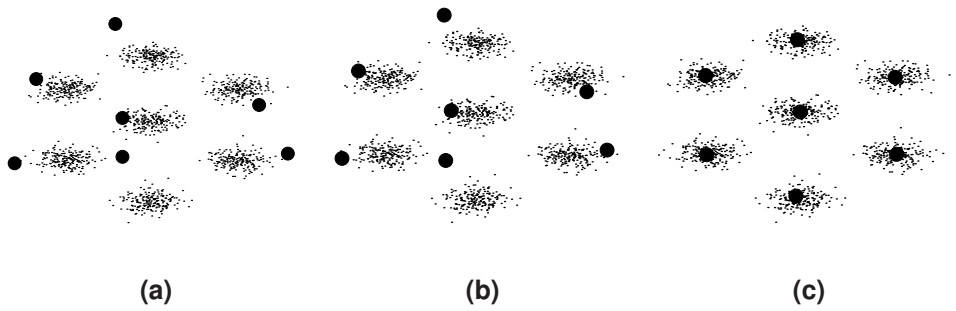


Figure 5.3: An illustration of Kohonen Vector Quantization. (a) shows training samples (points) and reference vectors (dots) in $\mathbf{R}^2$, (b) the result after 100 learning steps, (c) the result after 1000 learning steps.

Require: codebook $\mathcal{C} = \{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$
1: $t_i \leftarrow 0, \forall i \in \{1, \ldots, m\}$
2: $t \leftarrow 0$
3: **repeat**
4: $t \leftarrow t + 1$
5: Choose $\mathbf{s}$ randomly from the training set
6: Evaluate winner $\mathbf{w}_s$ according (5.8)
7: $t_s \leftarrow t_s + 1$
8: Evaluate learning rate $\lambda(t_s)$ according (5.9)
9: Adapt $\mathbf{w}_s$ according (5.10)
10: **until** $t = t_{max}$

Let $x_i(t_i)$ be the training sample s which adapts $\mathbf{w}_i$ into position $\mathbf{w}_i(t_i)$. Therefore, $t_i - 1$ adoptions were done with $\mathbf{w}_i$:

$$\begin{aligned}
\mathbf{w}_i(1) &= \mathbf{w}_i(0) + \lambda(1)(x_i(1) - \mathbf{w}_i(0)) = x_i(1) \\
\mathbf{w}_i(2) &= \mathbf{w}_i(1) + \lambda(2)(x_i(2) - \mathbf{w}_i(1)) = \frac{x_i(1) + x_i(2)}{2} \\
&\vdots \\
\mathbf{w}_i(t_i) &= \mathbf{w}_i(t_i - 1) + \lambda(t_i)(x_i(t_i) - \mathbf{w}_i(t_i - 1)) = \frac{\sum_{j=1}^{t_i} x_i(j)}{t_i}
\end{aligned}$$

As it can be seen, $\mathbf{w}_i$ is determined by the mean of all training samples for which it was the nearest reference vector. Furthermore, the membership to a Voronoi region S_i of some of the $x_i(t_i)$ can change during learning. Figure 5.3 shows a demonstration of the algorithm.

Kohonen emphasized Kohonen-VQ networks as *density estimators* of the underlying data. This needs equi-probable clusters. However, Kohonen respectively k-means learning laws do not produce equi-probable clusters in every case, because the cluster assignment of training samples is usually not equal. An asymptotic result

shows for a large number of clusters that the density of the codebook approximates the $(\frac{d}{d+r})^{th}$ power of the true probability density in d dimensions (Ripley [36]). In most cases, $r = 2$ which defines the power of the quantization error in (5.5). Thus for d large and a large codebook, Kohonen-VQ networks respectively the k-means algorithm code an approximation of the probability density of the data. Otherwise, Desieno's conscience value (Densieno [9]) is a popular method to obtain equi-probability which is added to each distance prior to the competition. The conscience value for each cluster is adjusted during training so that clusters that win more often have larger conscience values and are thus handicapped to even out the probabilities of winning in later iterations.

5.3.2 Competitive Hebbian Learning

To generate a topology of the training data in $\mathcal{L}$, we use Competitive Hebbian Learning (CHL). Topology is represented by a graph. Nodes are reference vectors. Two nodes are connected by an edge if the corresponding reference vectors lie close together in $\mathcal{L}$. Basis in CHL is Hebb's learning rule:

$$w_{ij} \propto z_i z_j$$

The weight w_{ij} of an edge between two reference vectors $\mathbf{w}_i$ and $\mathbf{w}_j$ is directly proportional to the product of its activation by training samples. z_i and z_j are the activation values. The smaller the distance $\|\mathbf{s} - \mathbf{w}_i\|$ between $\mathbf{w}_i$ and a sample s is, the larger becomes the result of its z_i. The product $z_i z_j$ is maximal,

$$z_r z_s \geq z_i z_j, \forall i, j \in \{1, \ldots, m\},$$

if one of $\mathbf{w}_i$ and $\mathbf{w}_j$ is the nearest and one the second nearest reference vector to s

$$r = \underset{i \in \{1,\ldots,m\}}{\arg\min} \|\mathbf{s} - \mathbf{w}_i\| \tag{5.11}$$

$$s = \underset{i \in \{1,\ldots,m\} \setminus \{r\}}{\arg\min} \|\mathbf{s} - \mathbf{w}_i\|. \tag{5.12}$$

Therefore, these two reference vectors are neighbors and are connected by an edge.

Let $B = (b_{ij})$ be an adjacency matrix of reference vectors $\mathbf{w}_i$ and $\mathbf{w}_j$ with elements $b_{ij} \in \{0, 1\}$. The algorithm evaluates in every training step the winner $\mathbf{w}_r$ and second $\mathbf{w}_s$ and set the according element $b_{rs} = b_{sr} = 1$. Matrix B is symmetric ($b_{ij} = b_{ji}$), because edges are undirected in the topology graph. The reference vectors itself remain unchanged during training.

The complete algorithm can be summarized as follows:

Require: codebook $\mathcal{C} = \{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$

1: $B \leftarrow 0$
2: $t \leftarrow 0$
3: **repeat**

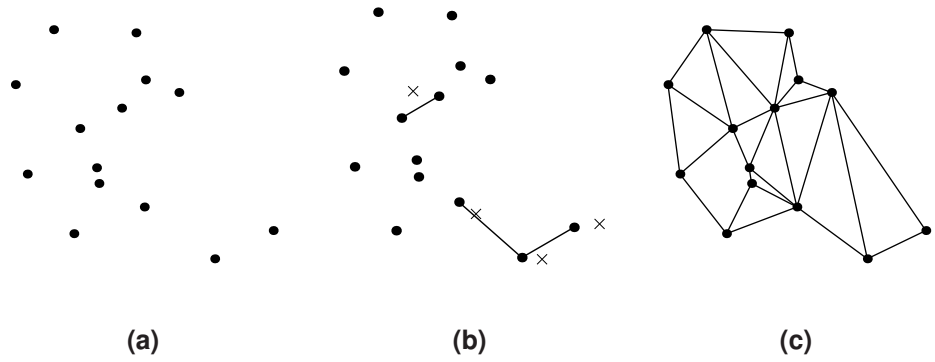


Figure 5.4: An illustration of Competitive Hebbian Learning. (a) shows reference vectors in $\mathbf{R}^2$, (b) four samples (crosses) and the learnt edges which form the induced Delaunay triangulation, (c) Delaunay triangulation

4: Choose $\mathbf{s}$ randomly from the training set
5: Evaluate winner $\mathbf{w}_r$ according (5.11)
6: Evaluate second $\mathbf{w}_s$ according (5.12)
7: $b_{rs} \leftarrow 1$
8: $b_{sr} \leftarrow 1$
9: $t \leftarrow t + 1$
10: **until** $t = t_{max}$

Martinez has shown, that every link in B corresponds to an edge in the Delaunay triangulation. More specific, B is a sub-graph of the Delaunay triangulation and is called induced Delaunay triangulation. Therefore, the Voronoi regions S_i and S_j are neighboring if a training sample s exists with winner $\mathbf{w}_i$ and second $\mathbf{w}_j$ or vice versa. Figure 5.4 shows the result of CHL with training samples after some CHL learning steps.

5.3.3 Growing Neural Gas

Section 5.3 mentioned the Growing Neural Gas (GNG) algorithm as a constructive, incremental and on-line variant of unsupervised learning methods. The algorithm itself is a combination of Competitive Hebbian Learning and the Growing Cell Structures model. A GNG state is represented by the following network:

- A codebook $\mathcal{C}$ with associated reference vectors $\mathbf{w} \in \mathcal{C}$ where each $\mathbf{w}$ represents the nodes of the network. The number of reference vectors $|\mathcal{C}|$ is not fixed. In the beginning, $\mathcal{C}$ is initialized by two samples of the training set $\mathcal{C} = \{\mathbf{w}_1, \mathbf{w}_2\}$.
- A set of links between these reference vectors which form a graph. Let $B = (b_{ij})$ be the symmetric ($b_{ij} = b_{ji}$) adjacency matrix with $b_{ij} \in \{0, 1\}$. Its purpose is to define the topological structure of the reference vectors.

The special characteristic of GNG is its data-driven growing nature. Starting from a codebook with two reference vectors, new reference vectors are introduced until a certain criterion is reached. For example, the accumulated, statistical network quantization error E falls below a given error E_{min} or a certain network size C_{max} is reached or a maximum number of learning steps t_{max} is exceeded. Besides, the new reference vectors are not added randomly. However, E is used to determine a position in the learning space. Consider the following three assumptions:

- The accumulated, statistical network quantization error

$$E = \sum_{\mathbf{w_i} \in \mathcal{C}} E_i$$

  should be minimized. If every reference vector accumulates the quantization error E_i of the training samples for which it has won, then E is the quantization error of all so far processed training samples.

- The insertion of a reference vector $\mathbf{w}_r$ decreases E,

$$\sum_{\mathbf{w_i} \in \mathcal{C}} E_i > \sum_{\mathbf{w_i} \in \mathcal{C} \cup \{\mathbf{w}_r\}} E_i.$$

- The largest decay in E can be expected if reference vector $\mathbf{w}_r$ with

$$r = \arg\max_{\mathbf{w_i} \in \mathcal{C}} E_i$$

  is inserted into the codebook.

These three properties are fulfilled by the goal of GNG learning, namely the network quantization error's reduction.

If $\mathbf{w}_s$ is the nearest reference vector to training sample s,

$$s = \arg\min_{i \in \{1,\ldots,|\mathcal{C}|\}} \|\mathbf{s} - \mathbf{w}_i\|^2, \tag{5.13}$$

its accumulated quantization error is adapted by

$$E_s = E_s + \|\mathbf{s} - \mathbf{w}_s\|^2,$$

where E_1 and E_2 are initially set to zero. To reduce E it makes sense to introduce a new reference vector at a position where the local quantization error is maximal. This is done at time τ which is a multiple of the learning steps t. Therefore, the reference vector $\mathbf{w}_q$ with maximal E_q is evaluated. Furthermore, $\mathbf{w}_f$ which has the maximal quantization error under the neighbors* of $\mathbf{w}_q$ is determined. This can be written as

$$q = \arg\max_{\mathbf{w}_i \in \mathcal{C}} E_i \tag{5.14}$$

$$f = \arg\max_{\mathbf{w}_i \in \mathcal{N}_q} E_i, \tag{5.15}$$

*Neighboring reference vectors $\mathbf{w}_i$ are connected to $\mathbf{w}_q$ by an edge ($b_{qi} = b_{iq} = 1$).

where $\mathcal{N}_q$ is the set of all neighbors of $\mathbf{w}_q$ determined by B. A new reference vector $\mathbf{w}_n$ is now inserted between $\mathbf{w}_q$ and $\mathbf{w}_f$ with

$$\mathbf{w}_n = \frac{\mathbf{w}_q + \mathbf{w}_f}{2}. \tag{5.16}$$

The link between $\mathbf{w}_q$ and $\mathbf{w}_f$ is deleted and two new connections are generated which can be written as

$$\begin{aligned} b_{qf} &= b_{fq} = 0 \\ b_{qn} &= b_{nq} = 1 \\ b_{fn} &= b_{nf} = 1 \end{aligned} \tag{5.17}$$

As $\mathbf{w}_n$ reduces the local quantization error in its neighborhood, the errors E_q and E_f can be reduced by a factor α. The aim is to adapt E_q and E_f in a way that both errors correspond to the expected errors after insertion of $\mathbf{w}_n$,

$$\begin{aligned} E_q &= E_q - \alpha E_q \\ E_f &= E_f - \alpha E_f. \end{aligned} \tag{5.18}$$

The error E_n is evaluated from these corrected error terms by

$$E_n = \frac{E_q + E_f}{2}. \tag{5.19}$$

The topology B is learnt in every learning step t. Beside the winner, a second nearest reference vector $\mathbf{w}_r$ is determined with

$$r = \underset{i \in \{1,\ldots,|\mathcal{C}|\} \setminus \{s\}}{\arg\min} \|\mathbf{s} - \mathbf{w}_i\|^2. \tag{5.20}$$

In the spirit of Hebbian learning, $\mathbf{w}_s$ and $\mathbf{w}_r$ are connected by an edge $b_{sr} = b_{rs} = 1$. Unfortunately, one problem still remains: The reference vectors change its positions during learning. Consequently, some edges could not be valid anymore. Therefore, an age of every link $A = (a_{ij})$ is introduced which is equal to the number of so far done learning steps since its creation. If a certain a_{ij} exceeds a maximal age a_{max}, the corresponding edge is removed ($b_{ij} = b_{ji} = 0$).

Another property of GNG is that adaption is only done for the winner $\mathbf{w}_s$ and its neighborhood $\mathcal{N}_s$. The learning law can be written as

$$\begin{aligned} \mathbf{w}_s &= \mathbf{w}_s + \lambda_b(\mathbf{s} - \mathbf{w}_s) \\ \mathbf{w}_i &= \mathbf{w}_i + \lambda_n(\mathbf{s} - \mathbf{w}_i), \forall i \in \mathcal{N}_s, \end{aligned} \tag{5.21}$$

where λ_b and λ_n are defined learning rates. After adaption all errors are decreased by a factor β.

The complete GNG algorithm is the following:

1: Initialize the codebook with two reference vectors $\mathcal{C} = \{\mathbf{w}_1, \mathbf{w}_2\}$ which are assigned to two randomly chosen samples from the training set
2: $B \leftarrow 0$, $A \leftarrow 0$, $t \leftarrow 0$
3: **repeat**
4: Choose $\mathbf{s}$ randomly from the training set
5: Evaluate winner $\mathbf{w}_r$ according (5.13)
6: Evaluate second $\mathbf{w}_s$ according (5.20)
7: Increment the age of all edges emanating from $\mathbf{w}_r$:

$$\begin{aligned} a_{ri} &\leftarrow a_{ri} + 1 \\ a_{ir} &\leftarrow a_{ir} + 1 \end{aligned}$$

8: Update the local error of the winner by

$$E_r \leftarrow E_r + \|\mathbf{s} - \mathbf{w}_r\|^2$$

9: Move $\mathbf{w}_r$ and its direct topological neighbors towards $\mathbf{s}$ according (5.21)
10: **if** $b_{rs} = b_{sr} = 0$ **then** {Create an edge}
11: $b_{rs} \leftarrow 1$
12: $b_{sr} \leftarrow 1$
13: **else** {"Refresh" the age of the edge}
14: $a_{rs} \leftarrow 0$
15: $a_{sr} \leftarrow 0$
16: **end if**
17: Remove all edges with an age $a_{ij} > a_{max}, \forall i, j \in \{1, \ldots, |\mathcal{C}|\}$. If reference vectors are not connected any more, remove them as well.
18: **if** t is a multiple of τ **then** {Insert a new reference vector}
19: Determine $\mathbf{w}_q$ according (5.14)
20: Determine $\mathbf{w}_f$ according (5.15)
21: Insert $\mathbf{w}_n$ into $\mathcal{C}$ according (5.16)
22: Insert edges $\overline{\mathbf{w}_q\mathbf{w}_n}$ and $\overline{\mathbf{w}_f\mathbf{w}_n}$,
Delete the edge $\overline{\mathbf{w}_q\mathbf{w}_f}$ according (5.17)
23: Decrease E_q and E_f according (5.18)
24: Initialize E_r according (5.19)
25: **end if**
26: Decrease all $E_i, \forall i \in \{1, \ldots, |\mathcal{C}|\}$ by β
27: $t \leftarrow t + 1$
28: **until** $(t > t_{max}) \vee (E < E_{min}) \vee (|\mathcal{C}| > C_{max})$

Figure 5.5 illustrates the behaviour of the GNG. Seven normally distributed classes of training samples were used as learning data. The parameters were set to: $\tau = 1000$, $\lambda_b = 0.05$, $\lambda_n = 0.0006$, $\alpha = 0.5$, $\beta = 0.0005$, $a_{max} = 88$. The learning was finished after $t_{max} = 40000$ steps.

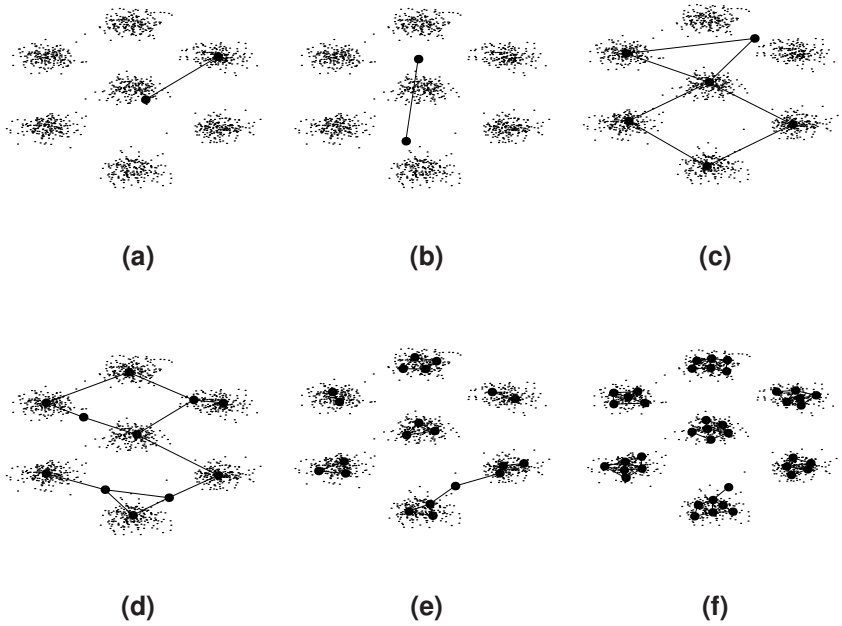


Figure 5.5: An illustration of Growing Neural Gas. Simulation with seven normal distributed ($\sigma = 2$) classes in $\mathbf{R}^2$ are shown. (a) Initialization, (b)-(e) Intermediate states, (c) final configuration after 40000 learning steps.

5.4 A learning solution

As it was outlined in section 5.1, learning in the context of traffic behaviour means learning a spatio-temporal probability density model. Section 5.2.2 discussed several unsupervised methods as possible density estimators. For example, the k-means algorithm or Kohonen-VQ could be used. But how does this probability density model look like in detail? Consider the codebook of reference vectors $A = \{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$ which is learnt by any of these methods. Furthermore, let us assume that all samples $\mathbf{s}$ within cluster S_i are normally distributed according to $\mathbf{s} \sim \mathcal{N}(\mathbf{w}_i, \sigma_i^2)$. The variance σ_i^2 is defined by

$$\sigma_i^2 = \frac{1}{|S_i|} \sum_{\mathbf{s} \in S_i} \|\mathbf{s} - \mathbf{w}_i\|^2. \tag{5.22}$$

Then, cluster S_i's contribution to the approximation of the probability density function of normal traffic behaviour for every sample $\mathbf{s} \in \mathcal{L}$ is given by

$$p_{S_i}(\mathbf{s}) = \frac{1}{(2\pi\sigma_i^2)^{\frac{5}{2}}} \exp(-\frac{(\mathbf{s} - \mathbf{w}_i)^\top (\mathbf{s} - \mathbf{w}_i)}{2\sigma_i^2}).$$

Thus, the approximated p.d.f. is a Gaussian mixture model of $p_{S_1}(\mathbf{s}), \ldots, p_{S_m}(\mathbf{s})$. This leads to the following definition:

Definition 3 *The approximated probability density function of normal traffic behaviour for any sample* $\mathbf{s} \in \mathcal{L}$ *is given by*

$$p(\mathbf{s}) = \frac{1}{m} \sum_{i=1}^{m} p_{S_i}(\mathbf{s}).$$

So far we have discussed the estimation of the p.d.f. in $\mathcal{L}$. However, we have not discussed how to use the p.d.f. to recognize unusual events. Let $\xi \in \{''normal'', '' abnormal''\}$ be the random variable which indicates normal or abnormal behaviour of a feature. Let $\mathbf{m} \in \mathcal{M}$ be a taken measurement during tracking the feature. The probability that $\mathbf{m}$ represents normal respectively abnormal traffic behaviour in a constant region $\mathcal{R}$ is

$$\begin{aligned} Pr(\xi ='' \, normal'') &= \int_{\mathcal{R}} p(\mathbf{m}) d\mathbf{m} \qquad (5.23) \\ Pr(\xi ='' \, abnormal'') &= 1 - \int_{\mathcal{R}} p(\mathbf{m}) d\mathbf{m}. \end{aligned}$$

It was shown under the assumption of a continuous p.d.f. and a small variation of its values within $\mathcal{R}$ that (5.23) can be approximated by

$$Pr(\xi ='' \, normal'') \approx p(\mathbf{s})V, \forall \mathbf{s} \in \mathcal{R},$$

where V is the volume of $\mathcal{R}$. Consequently, $Pr(\xi ='' \, normal'')$ is small where the value of the p.d.f. is small in a sufficient small region $\mathcal{R}$. For more information about this interpretation have a look into chapter 2.5.2 of Bishop [7]. However, as a consequence, a simple solution to recognize unusual events by using the p.d.f. and a measurement taken from a particular feature would be to use a threshold P. If $p(\mathbf{m})$ is below this threshold ($p(\mathbf{m}) < P$) then the feature will indicate abnormal traffic behaviour. Otherwise, it indicates normal traffic.

Johnson and Hogg [22] respectively Stauffer and Grimson [42] have used the Kohonen-VQ method to determine the probability density function. Both had the following three problems:

1. To define the right number of reference vectors,
2. How to initialize the codebook,
3. How to avoid stranded reference vectors.

The problems (1) and (2) are inherent problems of Kohonen-VQ networks. The number of codebook vectors are fixed. Therefore, the codebook has to be large enough which means that apriori knowledge about the complexity (number of clusters) of the learning space is needed. Johnson and Hogg used a method of Kohonen [24] to determine the right number experimentally. Thereby, a *reconstruction error* is evaluated for different numbers of reference vectors. A point is reached when increasing the codebook does not significantly reduce the error. Stauffer and Grimson did not use any

method at all. They defined the number by trial and error which they set to 400 reference vectors. Furthermore regarding problem (2), both groups initialized the codebook randomly with reference vectors centered at existing training samples. Finally, they referred to the last problem as the difficulty that a reference vector may be too far from any training data. Then, it will not win during learning and will not represent significant data. Therefore, they called such a vector stranded vector. Remember, this problem was also mentioned in the last section as density estimation with Kohonen-VQ was discussed. It was shown that one solution to this problem could be the introduction of conscience values. Johnson and Hogg had the same idea and used a similar method but named the penalty terms sensitivity values.

This work proposes a different strategy to overcome these three problems. In a first step, the GNG algorithm is used as a constructive unsupervised method to find parts of the learning space where there are a significant number of training samples. Neither a fixed number of reference vectors is needed nor the codebook initialization leads to problems where probably some parts of the learning space are not represented by reference vectors. The problem of stranded reference vectors is also solved by the GNG. Remarkably, the algorithm does not only update a codebook but also generate a topology of the training samples in form of a graph. An edge connects two reference vectors if both are close together in the learning space and samples lie in between them. If reference vectors loose all their links in the graph, they do not represent the underlying data and will be eliminated by the algorithm. Especially this fact that beside the codebook a topology is learnt is used for a classification of traffic activity in the next chapter 6. Instead of topology the term ”induced Delaunay triangulation” is often used in literature. ”Induced” means in this context that two reference vectors $\mathbf{w}_i$ and $\mathbf{w}_j$ will be neighbors and linked together, if both are the nearest or second nearest reference vectors to a sample respectively.

Certainly, the GNG also has disadvantages. The parameter settings of GNG are crucial. Mostly, this will result in an over-segmentation of the learning space. In other words, too many reference vectors are needed in some parts of the learning space. This leads to two problems:

1. The number of clusters is unnecessarily high. Consequently, the more clusters the bigger the problem of over-fitting the data.

2. The more reference vectors are in the codebook, the more complex the classification task. If there are m reference vectors in the codebook, the amount of data used in the classification is of order m^2.

Therefore, a pruning unsupervised method reduces the codebook until an optimum with respect to the training samples is reached based on the *Minimum Description Length - principle* (MDL)-principle (Bischof and Leonardis [5]). A description length codes the codebook and the quantization error by a binary model. After each reduction step the codebook is adapted by Kohonen-VQ and then a MDL-criterion is evaluated which indicates further reduction or the end of this iteration. Thus, it tries to find an

optimal balance between the number of reference vectors and the quantization error made by using this codebook. Principally, every unsupervised learning method could be used for the adaption step. It is clear, that for large training sets an incremental, on-line algorithm like Kohonen-VQ will perform more efficiently. Unfortunately, the topology could become invalid by the reduction of reference vectors. The validation of certain edges must be checked respectively new edges have to be introduced. Therefore, Competitive Hebbian Learning relearns a new topology based on the reduced codebook.

The use of such a MDL-framework (Bischof, Leonardis and Selb [6], Selb [38]) also offers another very important fact, namely outlier detection. As was discussed in chapter 4, it is not possible that all training samples in the training set represent normal traffic behaviour. For example, vehicle situations where a car drives in the wrong direction can happen during capturing of tracking data for learning. Certainly, these situations should be detected from the recognition system. Consequently, these possible outliers should not be in the training set. Neither GNG nor Kohonen-VQ methods are able to detect outliers in the training samples. The former tends to overfit the samples thus to represent also outliers as reference vectors. In Kohonen-VQ outliers do influence the positions of the reference vectors which also leads to a wrong codebook. Therefore, Johnson and Hogg and Stauffer and Grimson removed outliers manually. Our approach can be seen as a robust density estimation method where manually removal is not necessary any more as long as the number of outliers is small compared to the total number of training samples.

5.4.1 Preprocessing

Chapter 4 discussed the problems of different measure-ranges in the sample elements. To measure distances between samples, the Euclidean distance measure is used in the proposed learning method:

$$d(\mathbf{s}_1, \mathbf{s}_2) = \|\mathbf{s}_1 - \mathbf{s}_2\| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (dx_1 - dx_2)^2 + (dy_1 - dy_2)^2 + (t_1 - t_2)^2} \quad (5.24)$$

Principally, other distance measures are also possible. As it can be seen, a balance of the sample element's relative contribution is important. Otherwise, certain elements would dominate the distance measure and some would have little or no influence on the result.

Remember, that the direction is a unit vector hence its elements $x, y \in [-1, 1]$. Either the position nor the time is within this interval (section 4.1). One possibility is to re-scale them into the interval $[-1, 1]$. Then, all dimensions in the learning space have the same range. Imagine, a finite training set of k training samples. Let x_i be the positional x-element, y_i the y-element and t_i the time instant of sample s_i. Furthermore, the following is given:

$$x_{max} = \max(x_1, \ldots, x_k), y_{max} = \max(y_1, \ldots, y_k), t_{max} = \max(t_1, \ldots, t_k)$$

and

$$\begin{aligned} x_{mrange} &= \frac{x_{min} + x_{max}}{2} \\ x_{range} &= x_{max} - x_{min} \\ y_{mrange} &= \frac{y_{min} + y_{max}}{2} \\ y_{range} &= y_{max} - y_{min} \\ t_{mrange} &= \frac{t_{min} + t_{max}}{2} \\ t_{range} &= t_{max} - t_{min}. \end{aligned}$$

Then, position and time of every sample can be rescaled to $[-1, 1]$ by

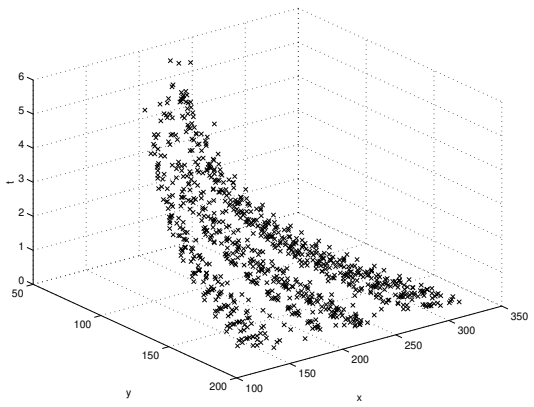
$$\begin{aligned} \tilde{x}_i &= \frac{x_i - x_{mrange}}{\frac{1}{2}x_{range}} \\ \tilde{y}_i &= \frac{y_i - y_{mrange}}{\frac{1}{2}y_{range}} \\ \tilde{t}_i &= \frac{t_i - t_{mrange}}{\frac{1}{2}t_{range}}. \end{aligned} \tag{5.25}$$

For example, figure 5.6(a) illustrates samples in the learning space before preprocessing. Rescaling leads to samples which are shown in figure 5.6(b).

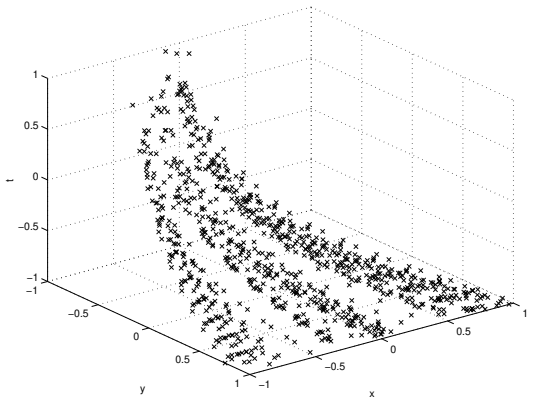
5.4.2 MDL framework

This section describes the heart of the learning algorithm which is used in this work. Section 5.4 introduced the MDL-principle. It is explained that this principle can be used to assess costs of a particular codebook with respect to its quantization error. For example, imagine several codebooks containing different numbers of reference vectors. All are a vector quantization solution and all are a result of equation's (5.2) minimization. Nevertheless, some codebooks will represent a better generalization of the training samples than others. In fact, the MDL-principle can be used to measure generalization capability and to distinguish between several codebooks.

The MDL-principle decides in the MDL-framework, if the number of reference vectors and their positions in a codebook are enough general representative with respect to the training samples. The MDL-framework can be described as follows: First, a codebook with more reference vectors than necessary is initialized by the Growing Neural Gas algorithm. Remarkably, the disadvantage of over-fitting by the GNG is used in this step. Then, the algorithm selects and removes those reference vectors which minimize the total description length by their elimination. Furthermore, outliers are detected by evaluating their contributing costs to the description length. They are removed from the training data as well. Consequently, they have no influence on the positions of the reference vectors in successive iterations. Finally, the algorithm



(a) before rescaling



(b) after rescaling

Figure 5.6: The preprocessing of training samples is illustrated.

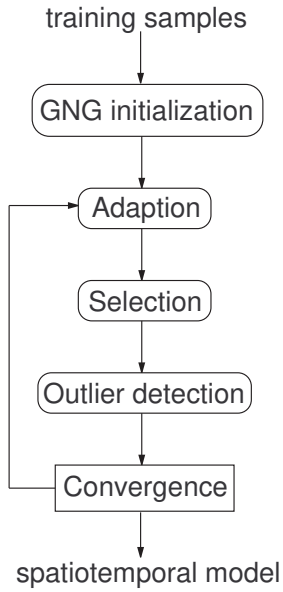


Figure 5.7: MDL-framework

checks if a minimum of the description length has been found. This is true if no additional outliers are detected, no more reference vectors are removed or the successive adaption by Kohonen-VQ leads to small changes in the reference vector's positions. The latter step closes the loop of the algorithm. Figure 5.7 shows an illustration of the MDL-framework.

Remember once more the problem of Vector Quantization. A sample s is replaced by its nearest codebook reference vector $\mathbf{w}_s$. This quantization leads to a quantization error $\epsilon(\mathbf{s}, \mathbf{w}_s)$. This relation can be written as

$$\mathbf{s} = \mathbf{w}_s + \epsilon(\mathbf{s}, \mathbf{w}_s). \tag{5.26}$$

Let the *description length* be the code-length of the codebook, the indexes which assign each training sample to a reference vector and the quantization error. This requires some thoughts about the necessary data structures. A codebook $\mathcal{C} = \{\mathbf{w}_1, \ldots, \mathbf{w}_m\}$ is a $(m \times 2)$ table. Each row corresponds to a reference vector. The first column represents the indexes while the second column delivers the reference vector's positions in the learning space. To code (5.26) usefully, a second $(n \times 3)$ table is used where the rows correspond to the training samples. The first column comprises their positions, the second their corresponding reference vector index and the last column the quantization error. Table 5.1 shows both data structures clearly. Let $L(\mathcal{C})$ be the costs of coding the codebook and $L(I(\mathcal{C}))$ be the costs of coding the indexes where training set S is indexed. Furthermore, let $L(\epsilon(S, \mathcal{C}))$ be the costs of coding the quantization errors of each sample $s \in S$. Thus, the total description length is

$$L(S(\mathcal{C})) = L(\mathcal{C}) + L(I(\mathcal{C})) + L(\epsilon(S, \mathcal{C})) \tag{5.27}$$

index	reference vector
1	$\mathbf{w}_1$
2	$\mathbf{w}_2$
$\vdots$	$\vdots$
m	$\mathbf{w}_m$

codebook

sample	index	quantization error
$\mathbf{s}_1$	3	$\epsilon(\mathbf{s}_1, \mathbf{w}_3)$
$\mathbf{s}_2$	8	$\epsilon(\mathbf{s}_2, \mathbf{w}_8)$
$\vdots$	$\vdots$	$\vdots$
$\mathbf{s}_n$	2	$\epsilon(\mathbf{s}_n, \mathbf{w}_2)$

Indexes and quantization errors

Table 5.1: Two data-structures are shown which store the necessary information of vector quantization.

bits. Up to now, no differentiation between inliers I and outliers O is done. Therefore, (5.27) expresses a description length where the whole training set $S = I + O$ only consists of inliers ($I = S, O = \emptyset$). Certainly, I and O are disjunct sets. Outliers are training samples which do not match the probability density function of training samples in the learning space, because they do not represent normal traffic behaviour. Consequently, they should not be quantified by reference vectors and thus extra coded. This considerations lead to a description length of

$$L(S(\mathcal{C})) = L(\mathcal{C}) + L(I(\mathcal{C})) + L(\epsilon(S, \mathcal{C})) + L(O) \tag{5.28}$$

bits.

To specify the coding in more detail, consider the following two assumptions:

1. The quantities like samples or reference vector positions in the learning space are specified with a finite precision. Consequently, both are coded with K bits.

2. Each sample is identically and independently distributed.

Both assumptions rewrite (5.28) to

$$L(S(\mathcal{C})) = mK + L(I(\mathcal{C})) + \sum_{i=1}^{m} \sum_{\mathbf{s} \in S_i} L(\epsilon(\mathbf{s}, \mathbf{w}_i)) + |O|K \tag{5.29}$$

bits, where S_i is the Voronoi region of reference vector $\mathbf{w}_i$. A proper coding of $L(I(\mathcal{C}))$ and $L(\epsilon(\mathbf{s}, \mathbf{w}_i))$ remains up to the user. Bischof, Leonardis and Selb [6] show typical examples.

If the direct coding of sample $\mathbf{s}$ with K bits is cheaper than the coding of the sample's nearest reference vector and resulting quantization error then $\mathbf{s}$ is an outlier. Consequently, the index of the corresponding reference vector $\mathbf{w}_s$ and the error $\epsilon(\mathbf{s}, \mathbf{w}_s)$ are not coded. To summarize, a condition to mark $\mathbf{s}$ as an outlier is

$$K < L((I \setminus \{\mathbf{s}\})(\mathcal{C})) + L(\epsilon(\mathbf{s}, \mathbf{w}_s)), \tag{5.30}$$

where $L((I \setminus \{\mathbf{s}\})(\mathcal{C}))$ are the costs of coding the indexes, indexing S without $\mathbf{s}$. Furthermore, $\mathbf{s}$ is also an outlier if it is the *only* sample in a particular Voronoi region S_i. Then, S_i is superfluous and $\mathbf{w}_s$ can be removed from the codebook. This saves K bits in the description length. The index of $\mathbf{w}_s$ has also not to be coded any more. However, there is no reduction of $L(S(\mathcal{C}))$ by the quantization error, because if there is only one sample in S_i then $\epsilon(\mathbf{s}, \mathbf{w}_s) = 0$. A second condition to mark $\mathbf{s}$ as an outlier is

$$K < L((I \setminus \{\mathbf{s}\})(\mathcal{C})) + K. \tag{5.31}$$

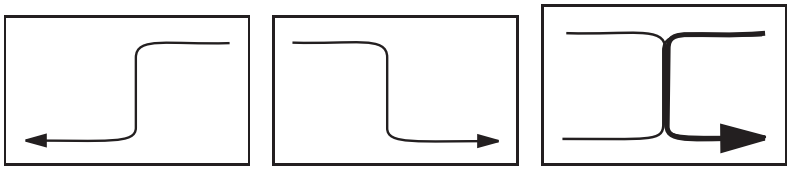
The goal of the MDL-based algorithm is to minimize $L(S(\mathcal{C}))$. Consequently, a reference vector $\mathbf{w}_i$ is temporarily removed and the resulting change in the description length is computed. This is done for every reference vector $\mathbf{w}_1, \ldots, \mathbf{w}_m$ separately. If the removal of $\mathbf{w}_i$ causes a decrease of the description length, then $\mathbf{w}_i$ can be definitely removed from $\mathcal{C}$. Otherwise it remains in $\mathcal{C}$.

Chapter 6

Classifying traffic behaviour

Contents

This chapter treats the classification of traffic behaviour within a traffic scene. Section 6.1 motivates the need for classification and presents two previously used approaches. Both methods calculate explicitly the classification result by using a further learning step or a co-occurrence matrix respectively. In contrast to them, section 6.2 discusses a new approach which only uses the topology of the training data. In fact, the topology was calculated implicitly by the learning step.



(a) Situation 1 **(b)** Situation 2 **(c)** Situation 3

Figure 6.1: Three traffic situations are shown. (a) and (b) show two different traffic flows which intersect with each other. If the p.d.f. is used alone as spatio-temporal model, situation (c) would also be recognized as normal traffic behaviour although traffic flow in this direction is not allowed.

6.1 Why classification

The last chapter presented a spatio-temporal model which is used for unusual event recognition. A codebook was generated by unsupervised learning and is used as a density estimator of normal traffic within the learning space $\mathcal{L}$. As described, a feature's measurement generates a sample in $\mathcal{L}$. The feature tends to indicate an unusual event, if its value of the probability density function (p.d.f.) goes towards zero. Unfortunately, not all unusual traffic situations can be treated by the p.d.f.. Figure 6.1 illustrates such an example. Figures 6.1(a) and 6.1(b) show two particular directions in which vehicles are driving. Consider a scene where both situations are simultaneously possible. Vehicles, which behave as illustrated, could be described as normal traffic by a learnt p.d.f.. Consequently, all other situations would be recognized as unusual events. However, a vehicle behaviour like it is shown in figure 6.1(c) could not be assessed as unusual, because problems arise in the area where vehicle trajectories of both situations intersect with each other. The reason lies in the p.d.f. which is defined for a particular *point* in $\mathcal{L}$ but does not consider the *trajectory* of a feature (i.e. from where a feature comes and where it goes).

To overcome this problem, the spatio-temporal model has to be completed by a set of possible classes of trajectories which are learnt from the training samples. Beside a p.d.f. evaluation of the feature's actual position in location and time, the trajectory is also classified. If no class matches, the feature indicates an unusual event. According to the above example, both particular driving directions would form two classes. Hence, the situation in figure 6.1(c) could not be classified and an unusual event would be detected.

Johnson and Hogg [22] proposed a further vector quantization approach to complete the spatio-temporal model. They realized a new learning space where each sample corresponds to a *sequence* of reference vectors. These sequences are generated from the training paths of the tracked features. Each training sample of each path is quantized by the previous learnt codebook. Thus, vector quantization is done on basis of feature's measurements. Then, a second vector quantization is done on basis of fea-

ture's trajectories. Both steps were implemented with a neural network. They used a layer of "leaky" neurons in between these two networks for the representation of this new training samples. Thereby, a leaky neuron is a single input-output unit. If it is once activated by a greater input than output, its output value will decay over a certain period of time. Thus, the p.d.f. of the resulting new codebook describes clusters of similar trajectories.

Grimson and Stauffer [17] [40] [42] also developed a robust tracking system which learnt patterns of activity in a particular road scene. For example, the system can distinguish the different traffic activities at certain times in the image. Once the codebook is learnt, every training sample is considered by its corresponding reference vector. As in the previous approaches, each training path is replaced by a sequence of reference vectors. A co-occurrence matrix $C = (c_{ij})$ can be generated where each entry c_{ij} is the estimated probability that a sequence from the training sequences will contain an input represented by the i^{th} reference vector and a separate input represented by the j^{th} reference vector. These joint co-occurrence statistics are then used to create a hierarchical binary-tree classification of traffic behaviour. Thereby, two probability mass functions across the reference vectors of the codebook are determined which best explain C. Once these distributions are found, each distribution is treated as another set of reference vectors and their co-occurrence matrix is determined. The process is repeated to produce the binary-tree classification with predefined height.

The fixed number of reference vectors for the successive second vector quantization step is the most obvious disadvantage of the Johnson and Hogg approach. This problem is solved by Grimson and Stauffer by using the hierarchical classification idea. However, both approaches expect a second learning step. Grimson and Stauffer used a greedy minimization to estimate C. In each learning step, its parameters are updated by a simple learning rule which is based on the minimization of the current and estimated co-occurrence matrix. Certainly, the most obvious drawback of this solution is its sensitivity to stuck in a local minima. It is unclear, how this influences the classification (i.e. the number and differences of classes). Indeed, the same training data should always result in equal classifications.

A classification as realized by Grimson and Stauffer also offers *other* possibilities of recognition. For example, a system mounted above a highway could automatically detect the number of lanes on the road. Furthermore, such a system could recognize areas in the scene with more or less traffic activity. Classification could help in conjunction with knowledge about date and time, to take the right steps to defuse traffic jams during road work. Furthermore, it could help to define the duration of signal lights more intelligently. Certainly, various other ideas of applications are possible.

6.2 The classification method

To find classes of traffic behaviour within a particular scene, the proposed method *compares* the feature trajectories of the training data pairwise. More precisely, the se-

quences which all contain the same number of reference vectors are compared. For example, consider two sequences $S_1 =< \mathbf{w}_1 1, \mathbf{w}_1 2 >$ and $S_2 =< \mathbf{w}_2 1, \mathbf{w}_2 2 >$. The pairs of reference vectors which are compared are $(\mathbf{w}_1 1, \mathbf{w}_2 1 >)$ and $(\mathbf{w}_1 2, \mathbf{w}_2 2 >)$. To be able to perform this comparison and in contrast to the previously mentioned methods, the topology of the training samples is used. It was mentioned that the topology is the induced Delaunay triangulation which is simply a graph. Nodes are reference vectors. They are connected by edges if they are neighbors (i.e. samples exist for which two reference vectors are winner and second). If two features show similar traffic behaviour (i.e. both represent vehicles which turn to the left at an intersection), then their paths in the learning space are equal (i.e. the sequences of reference vectors, where each reference vector occurs only once, are equal). Certainly, as the size of the codebook can be large and the influence of a Voronoi region can be small, different sequences can in fact also belong to the same behaviour class. Therefore, the *assumption* is that pairwise compared reference vectors have not necessarily to be equal. Instead they have to be at least neighbors in the topology graph (i.e. an edge must exist between both reference vectors). A remark should be done at this point. Please note, the topology is only used as similarity measure. It does not necessarily represent trajectories, i.e., generally not every trajectory from the training data has to be represented by a path in the topology graph. The reason lies in the fact that the topology is generated upon training samples which do not include any information about trajectories.

As in the approach of Grimson and Stauffer and in contrast to the work of Johnson and Hogg, the number of classes has not been given at the beginning of classification. However, no further learning is done. Remember the used learning algorithms which were discussed in the previous chapter. The Growing Neural Gas and Competitive Hebbian Learning algorithm produce beside a codebook an induced Delaunay triangulation which we have called topology. Two reference vectors of the codebook are connected by an edge, if they were a winner or a second for a particular training sample during learning respectively. Our algorithm finds the classification by using only this topology graph.

Instead of considering paths of training samples within the learning space, we consider sequences of reference vectors. Figure 6.2 shows an illustration. Each training sample is quantized to its nearest reference vector. All quantized training samples of a path form together a sequence. All training paths except the outliers are quantized and used for classification. In fact, each tracked feature runs through a certain number of Voronoi regions depending on its behaviour. The easiest way of defining classes of traffic behaviour is to combine all training paths which run-through the *same* Voronoi regions or which have the same sequence.

Definition 4 *Let* $S_i =< \mathbf{w}_{i1}, \mathbf{w}_{i2}, \ldots >$ *and* $S_j = \{\mathbf{w}_{j1}, \mathbf{w}_{j2}, \ldots\}$ *be two training sequences. The* $\mathbf{w}_{ik}$ *and* $\mathbf{w}_{jl}$ *are reference vectors. Generally, sequences are smaller than their corresponding paths (*$|S| \leq n$*), because consecutive training samples quantized to the same reference vector occur only once in a sequence. If*

$$\forall \mathbf{w}_{ik} \in S_i \;\; \exists \mathbf{w}_{jl} \in S_j : \mathbf{w}_{ik} = \mathbf{w}_{jl},$$

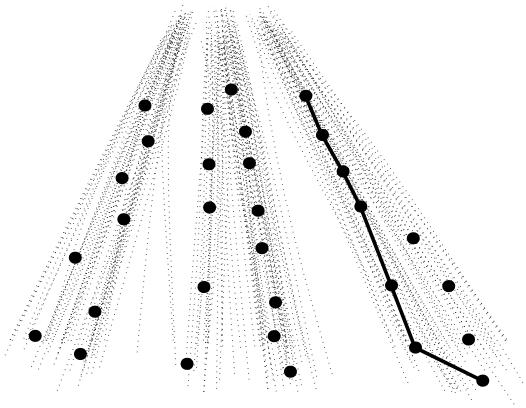


Figure 6.2: The reference vectors (dots) of the codebook within the training paths (dotted lines) are shown in only two dimensions (x/y) of the learning space. A sample sequence is illustrated by a thick line.

then sequences S_i and S_j will both belong to the same class and are equivalent.

For example, consider three sequences $S_1 =< 1\ 2\ 5\ 7\ 10 >$, $S_2 =< 2\ 5\ 7\ 10 >$ and $S_3 =< 1\ 2\ 5\ 8\ 9 >$. S_1 and S_2 are equivalent. S_3 is neither equivalent to S_1 nor to S_2, because reference vectors 8 and 9 are unique to S_3. As the codebook contains many reference vectors which lie close together, many similar classes are generated which are actual equal in representing traffic behaviour. We call this form of class generation *incremental* class generation, because far more classes than necessary are found.

Consider training sequences $S_1, \ldots, S_u$ which are generated from training paths. Generally, u is smaller than $n_{\mathcal{M}}$, because outliers are not used for sequence generation. Each incrementally produced class $C_1, \ldots, C_c$ is represented by its corresponding and equivalent sequences. c is the total number of classes found. The representatives $C_1^R, \ldots, C_c^R$ are sets of all reference vectors which occur in their according sequences. The incremental algorithm is shown in algorithm 6.1.

A first class C_1 is initialized by the first training sequence S_1 in step 1, 2 and 3. Then all other $u - 1$ sequences are tested with the actual classification (steps 4 until 12). A sequence belongs to a class, if all class representatives are part of the sequence. In fact, step 7 defines the condition $C^R \subseteq S$ instead of $S \subseteq C^R$. Thus, we guarantee that sub-sequences belong to the same classes and do not define a new one. This is the case, if a sequence does not belong to any class (steps 13 until 17).

Figure 6.3 illustrates a few typical classes which were found by the incremental algorithm. Three classes are shown in different gray colours with their corresponding paths and representatives. The latter are linked together which illustrates the class's sequence. Reference vectors and training paths which do not belong to these three classes are shown in black.

As it was discussed, the incremental step produces far more classes than neces-

Require: Training sequences $S_1, \ldots, S_u$
Ensure: Classes $C_1, \ldots, C_c$
1: $C_1 \leftarrow \{S_1\}$
2: $C_1^R \leftarrow \{\mathbf{w}_{11}, \mathbf{w}_{12}, \mathbf{w}_{13}, \ldots\}$
3: $c = 1$
4: **for** $i = 2$ to u **do**
5: $found \leftarrow false$
6: **for** $j = 1$ to c **do**
7: **if** $\forall \mathbf{w} \in C_j^R : \mathbf{w} \in S_i$ **then** {All representatives of class C_j are part of sequence S_i}
8: $C_j^R \leftarrow C_j^R \cup \{\mathbf{w}_{i1}, \mathbf{w}_{i2}, \mathbf{w}_{i3}, \ldots\}$
9: $found \leftarrow true$
10: break
11: **end if**
12: **end for**
13: **if** $found = false$ **then** {S_i does not belong to any class}
14: $c \leftarrow c + 1$
15: $C_c \leftarrow \{S_i\}$
16: $C_c^R \leftarrow \{\mathbf{w}_{i1}, \mathbf{w}_{i2}, \mathbf{w}_{i3}, \ldots\}$
17: **end if**
18: **end for**

Algorithm 6.1: The incremental algorithm

sary. Most of the classes represent the same traffic behaviour and differ only in their representations C^R from each other by a few reference vectors. Consider figure 6.4 to understand the equivalence of classes. Although, the representatives of both classes are completely different, figure 6.4(a) shows two equivalent classes. All representatives of both classes lie close together, because reference vectors are connected by an edge. Therefore, both are candidates which could be merged together to a new class. However, figure 6.4(b) shows two classes which are not similar, because most of the reference vectors of one class are not linked to any reference vector of the other class. Thus, the decision about equivalence of classes and the ability to merge them, depends on the proximity of their reference vectors. The closer two reference vectors are in $\mathcal{L}$, the more similar the behaviour of two features will be which are both be represented by these reference vectors. In fact, the proximity is expressed by an edge in the topology graph.

Definition 5 *Let C_i and C_j be two different classes which are found by the incremental algorithm. Let $R_j = C_j^R - C_i^R$ be the reference vectors in C_j^R which are not in C_i^R.*

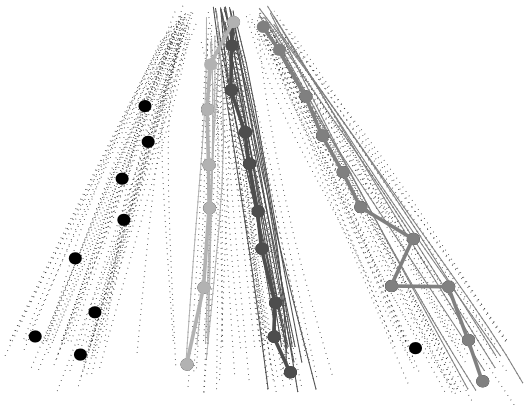


Figure 6.3: Three typical classes of the incremental step are shown in distinct gray colours. Their corresponding training paths are drawn by solid lines. Each sequence is shown by thick lines which connects the class's representatives (dots).

(a) Equivalent classes **(b)** Distinct classes

Figure 6.4: The decision between equivalent and distinct classes which are represented by their reference vectors (dots respectively squares) lies in the topology (solid lines). (a) shows two equivalent classes while (b) two distinct ones.

Furthermore, we also define $R_i = C_i^R - C_j^R$. *Let* $B = (b_{ij})$ *be the topology matrix. If an edge between reference vectors* $\mathbf{w}_i$ *and* $\mathbf{w}_j$ *exists, then* $b_{ij} = 1$. C_i *and* C_j *are equivalent (*$C_i \equiv C_j$*), if*

$$\forall \mathbf{r} \in R_j \;\; \exists \mathbf{w} \in C_i^R : B(\mathbf{r}, \mathbf{w}) = 1 \wedge \forall \mathbf{r} \in R_i \;\; \exists \mathbf{w} \in C_j^R : B(\mathbf{r}, \mathbf{w}) = 1.$$

All equivalent classes can now be merged to one new class. Let $C_1, \ldots, C_c$ be all classes found by the incremental step. $C_1^*, C_2^*, C_3^*, \ldots$ are the resulting merged classes, generated by a merging algorithm which is depicted in algorithm 6.2.

In each run (steps 4 until 7) all classes are determined which are equivalent with respect to definition 5. Thereby $|M| - 1$ classes are tested with a chosen class C of M. All equivalent classes form together a new class C_1^* which is removed from M. Then, the next run takes place. The algorithm is repeated until $M = \emptyset$.

This merging is repeated for all classes $C_1^*, C_2^*, C_3^*, \ldots$. In each step the number of classes decreases until a final set of classes, where all classes are unequal, is reached.

Require: Classes $C_1, \ldots, C_c$, Topology B
Ensure: Classes $C_1^*, C_2^*, C_3^*, \ldots$
1: $M \leftarrow \{C_1, \ldots, C_c\}$
2: $i \leftarrow 1$
3: **while** $M \neq \emptyset$ **do** {Are there any classes to merge?}
4: Choose a $C \in M$
5: $C_i^* \leftarrow \{C_k \in M : C_k \equiv C\}$
6: $M \leftarrow M \setminus C_i^*$
7: $i \leftarrow i + 1$
8: **end while**

Algorithm 6.2: The merging algorithm

This convergence must exist, because only a finite number of traffic patterns exist within a scene. Figure 6.5 illustrates the merging of classes. All classes found in the incremental step are merged to five classes (figure 6.5(a)). A successive second merging step, which is illustrated in figure 6.5(b), finally results in three classes. No further merging of classes is possible.

Once the classes are evaluated from the training paths, every sample trajectory can be classified. Thereby, every feature's measurement is quantized and a sequence is generated. Certainly, if the feature is currently tracked, also partial sequences can be generated. Then, all reference vectors in the sequence are compared to the representatives of each class. If the whole sequence matches a particular class, the classification is successful. Otherwise, the feature indicates an unusual event. Summing up, the spatio-temporal model looks as follows:

Definition 6 *Let $p(\mathbf{m})$ be the spatio-temporal probability density function of normal traffic behaviour. Let $C_1, \ldots, C_c$ be the final classes of the scene. Let $\mathbf{m}_1, \mathbf{m}_2, \ldots$ be the measurements of a currently tracked feature. They are quantized to the partial sequence $\{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n\}$. The feature shows normal traffic behaviour if*

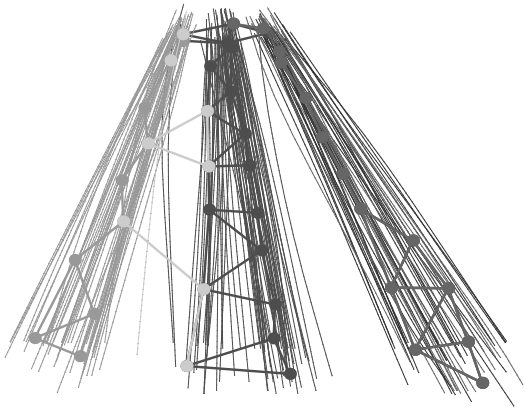
(i) the p.d.f. is greater than a specific threshold P,

$$\forall \mathbf{m}_i, i \geq 1 : p(\mathbf{m}_i) > P.$$

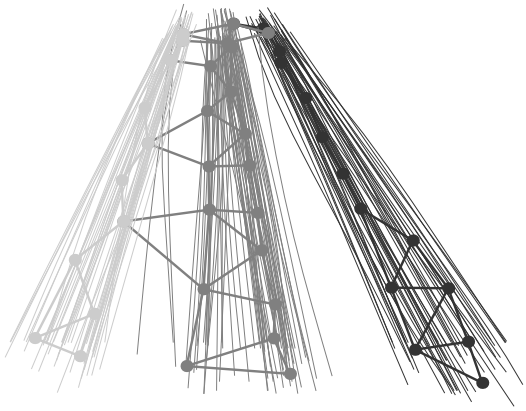
(ii) the partial sequence is a subset of the representatives of a class,

$$\exists i, 1 \leq i \leq c : \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n\} \subseteq C_i^R.$$

Certainly, partial sequences can belong to more than one class. However, full sequences must belong to only one class, because otherwise the classes would have been merged.



(a) 1^{st} merging step



(b) 2^{nd} merging step

Figure 6.5: Two consecutive merging steps are shown. Five classes are found after the first merging step (a). Then, a second, final step finds the three classes according the three lanes in the data (b).

Chapter 7

Experiments and Evaluation

Contents

This chapter presents experiments with the spatio-temporal model and classification algorithm. Section 7.1 treats the training and test tracking data. Section 7.2 shows experiments with the training sample generation process. It focuses on the ability of obvious outlier detection and noise reduction. Section 7.3 evaluates the GNG pre-clustering and the successive MDL-based pruning step. Section 7.4 shows the results of classification with respect to the training data. Performance issues are also treated. Finally, the results of using the spatio-temporal model and classification on the test data are shown in section 7.5. A summary is given in section 7.6.

7.1 Tracking data

To gather real tracking data for training the proposed spatio-temporal model and testing its usage for traffic recognition, we visited two different sceneries:

Highway: The highway A2 in the south of Vienna is usually congested during peak traffic in the morning and afternoon. Therefore, this place was appropriate to film traffic jams. Furthermore, it gave the possibility to shoot vehicles with speeds above 80 km/h. Thus, the real-time tracker could also be tested with fast driving cars. We chose a position on a bridge at Eumiggasse/Wr. Neudorf approximately 7 m above the ground. The camera was positioned frontally to the south of the highway. Therefore, all three lanes plus the most right lane, which is reserved for break-downs, were in the view of the camera and all vehicles were filmed from the back. Figure 7.1(a) shows a typical image of this scene.

Car park: We chose the IKEA car park at the SCS shopping centre in the south of Vienna as a second scenario, because of its more complex traffic situations compared to the highway. Besides, it usually has a high volume of traffic every Saturday. The camera was positioned on top of a building approximately 25 m above the ground. It lay directly in front of the car park. The camera's view was orientated to an intersection. Most of the vehicles drove from top into it, left the view to the bottom or drove vice versa. As most of the vehicles drove on this road we name it "main road" throughout this section. The perpendicular road was rarely used. Figure 7.1(c) shows a typical image of this scene.

We used an ordinary S-VHS CCD-camera and a tripod for shooting. First, normal traffic situations were filmed for 30 minutes at both scenes. Normal meant flowing traffic without any disturbances in case of the highway and no unusual traffic obstructions with parking or driving vehicles at the car park. Then, we took short video sequences of unusual events. In case of the highway a traffic jam on the most left and vehicles prohibitively driving on the most right reserved lane were shot. Figure 7.2(a) depicts a typical frame of the former situation while figure 7.2(c) shows an example of the latter. The car park showed itself as difficult place at this time to find unusual events. Therefore, we pushed back our own car against the traffic. We did this on the seldom used road and drove into the intersection, stopped and turned to the top. Figure 7.3(a) depicts this situation. Furthermore, we filmed a second unusual situation where we turned from the bottom to the left. We stopped our car for approximately a minute on the road. Then we left the view of the camera. Figure 7.3(c) shows the stopped car.

All six video sequences were digitized by a frame-grabber board. We used the Genesis video board from Matrox. The video streams were sampled with 15 frames per second. Each frame was a 8 bit gray-valued image with half of the PAL resolution (384×288). Features were detected and then tracked in real-time within detection and tracking regions. According the highway, one detection region on the bottom of the image was defined which comprises all three lanes. Additionally, a second detection region was set on the reserved lane, because the perspective in the images did not allow for one region. Nearly the whole frame was defined as tracking region except the opposite direction of the highway, the area below the detection region and the upper part of the images where vehicles are too small for tracking. Two detection regions on the top and the bottom of the intersection were used in the car park scene. Further,

Feature detector				
W	σ_s	σ_g	λ	f_d
7	0.9	1	$1.5 \cdot 10^5$	5

Feature tracker					
Det_{min}	$Disp_{min}$	$Iter_{max}$	Res_{max}	σ_{pyr}	S_{range}
0.01	0.1	10	∞	0.9	15

Table 7.1: All chosen detection and tracking parameters are summarized.

we used a third region to the right of the intersection which should only produce the unusual event with the back pushed car. The whole area around the detection regions including parts of the rare used road were defined as tracking area.

Table 7.1 summarizes the chosen parameters of the feature detector and tracker. The same parameter values were chosen as in section 2.3 except for λ and Res_{max}. The former was set significantly larger with respect to the values in section 2.3.1 due to the different histogram of the scene. A check of the residuals was neglected, because the perspective makes a comparison of the detected and actually tracked patch useless. Besides, a parameter f_d, the detection frequency, was set to 5. Thus, features were detected every 5^{th} frame. A feature was tracked successfully, if it was pursued without any error from its detection within the detection region until it left the tracking region. Then, the measured trajectory of the feature was saved as tracking data. The feature was discarded in case of an error.

During tracking of normal traffic, 2698 measured feature trajectories were gathered from the highway video. Training samples were generated which formed together the training set. A smaller subset with 400 trajectories was chosen from the tracking data. The reason of choosing such a relatively small subset is the possibility of analyzing outlier detection and the necessity of trajectory visualization throughout this section. 350 trajectories were the result of tracking normal features with an average life-time of 70 frames. Nevertheless, some of these trajectories are either obvious spatial or spatio-temporal outliers. They were chosen randomly from the tracking data. The final 50 trajectories were deliberate, obvious temporal outliers. 25 of them were features with life-times below 4 frames while the rest had life-times above 100 frames. Figure 7.1(b) depicts this subset of the tracking data which is used to investigate the behaviour of the training sample generation, the learning and the classification. The whole tracking data is used in particular for the unusual event case studies. Figure 7.2(b) shows 511 gathered trajectories which form the test set according to the traffic jam. The test set of the prohibitively driving vehicles consists of 290 trajectories which are depicted in figure 7.2(d).

In case of the car park, 1452 normal features were successfully tracked and saved as tracking data. Equally to the highway case, 350 normal feature trajectories with average life-times of 100 frames and 50 obvious temporal outliers were chosen as a subset. Half of them were features with life-times below 15 frames while the other

Training tracking data		Test tracking data	
Scene	Trajectories	Incident	Trajectories
highway	2698/400	traffic jam	511
		prohibitively driving	290
car park	1452/400	back pushed car	14
		stopped car	12

Table 7.2: The number of trajectories which were gathered for training and testing are summarized. 400 trajectories of each training tracking data set were chosen randomly as subset.

half survived for more than 200 frames. Figure 7.1(d) shows this subset of the tracking data. Figure 7.3(b) and 7.3(d) depicts the tracking data of the unusual events. The former, which shows the back pushed car, contains 14 trajectories while the latter, which corresponds to the stopped car scene, contains 12 trajectories.

Table 7.2 summarizes the number of gathered trajectories for training and testing.

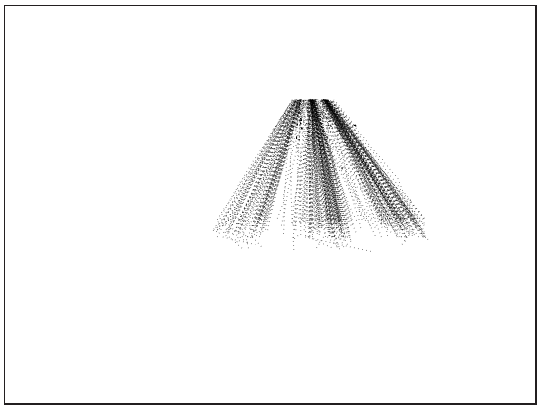
7.2 Training samples

This section treats the training sample generation process with the previously described highway and car park training tracking data. The two main objectives, which should be investigated in the following, are the ability of obvious outlier detection in the spatial and temporal domain and the expected noise reduction by using smoothing splines.

Obvious outlier detection: Obvious outliers were discussed in section 4.4. To sum up, spatial obvious outliers are basically tracking errors. Furthermore, features which are tracked for a significantly shorter or longer time than others were defined as obvious temporal outliers. Let the variance Σ of noise during tracking be $4\ I_2$ for all further experiments, because deviations of more than two pixels between the measured and the smoothed position should be the result of a tracking error.

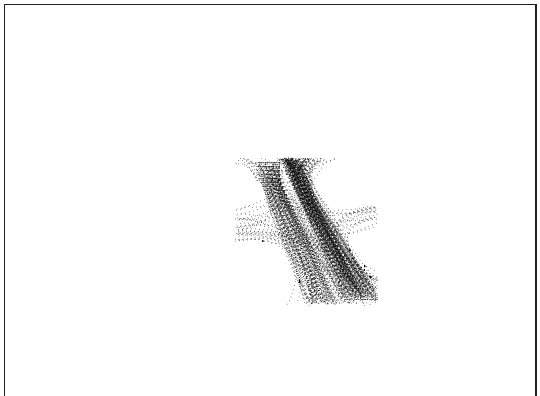
First, we examined the highway training subset. A histogram of the feature's life-time was generated which is shown in figure 7.4(a). The 350 normal trajectories can be seen by two tall bars around frame 70. All other bars are representing the 50 temporal outliers. This obvious difference is caused by the fact, that normal features need on average the same life-time, because all lanes are of similar length within the tracking area and vehicles drive on average the same speed. We did two experiments with two different thresholds $\kappa = \{13, 74\}$ to illustrate temporal outlier detection. $\kappa = 13$ let 25 outliers be included in the

(a) Highway scene

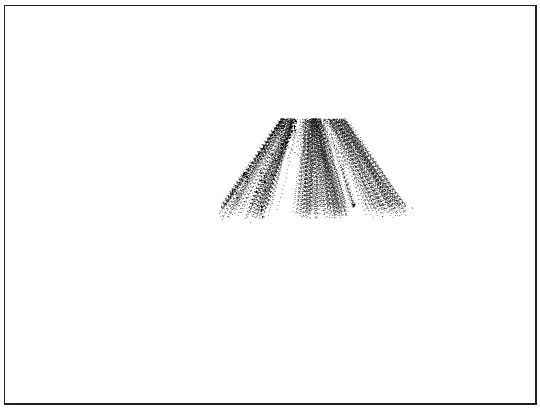


(b) Highway tracking data

(c) Car park scene

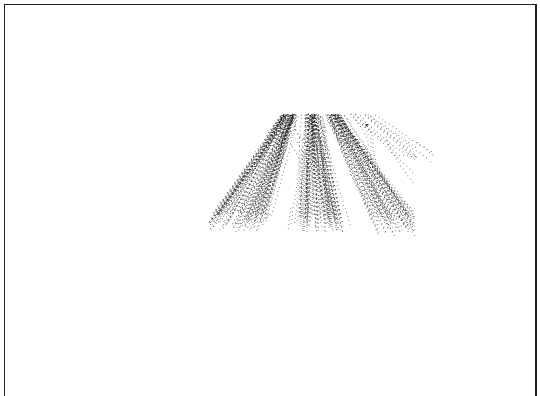


(a) Traffic jam

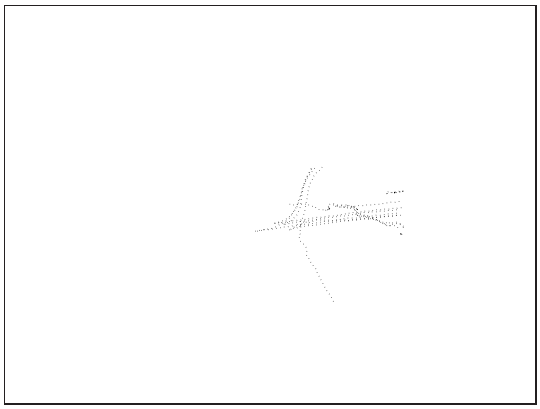


(b) Traffic jam tracking data

(c) Prohibitively driving

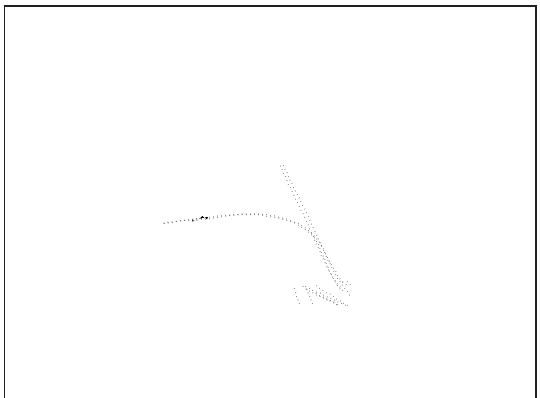


(a) Back pushed car



(b) Back pushed car tracking data

(c) Stopped car



training set while $\kappa = 74$ removes all temporal outliers. These results can be seen in figure 7.5(a). All rare and time-delayed feature trajectories were removed by $\kappa = 13$. They can be seen in the histogram by bars with a frequency smaller than 3. The figure shows them as thin black lines. However, $\kappa = 74$ removes further 25 outlier trajectories which consist of only two positions. They are shown as thick black lines.

Besides, we examined the car park training subset. The histogram can be seen in figure 7.4(c). Normal trajectory life-time is distributed around the mean of 100 frames. The scene is more difficult than the highway, because several paths along the intersection are possible. κ was set to 34. Thus, the most frequent trajectories which have equal life-times within the training subset were chosen. These features represent mostly vehicles which cross the intersection from top to the bottom and vice versa. Therefore, trajectories which drive into the perpendicular road are seldom in the subset which was necessary for testing. The result can be seen in figure 7.5(b). All introduced 50 temporal outliers are eliminated. Further 171 trajectories are also removed. Thus, the training subset consists of 179 trajectories after temporal outlier detection.

The latter fact shows that more trajectories are removed than necessary. This only depends on the choice of κ. Normal feature trajectories with similar life-times will produce significantly taller bars than temporal unusual tracked features. The more tracking data we have, the more obvious this difference will be and the better κ can be set. If features only remain in the training set after the temporal outlier removal which partially have similar life-times and are represented by a significant large number within the original training set, then the possibility of a temporal outlier's occurrence will vanish.

Figure 7.4(b) shows the histogram of the whole highway training tracking data. Normal tracking data is obviously Gaussian distributed around 70. We defined $\kappa = 57$ to eliminate approximately the same amount of trajectories ($\approx 12\%$) as in the sub-set case. Consequently, 329 trajectories were eliminated and 2369 remained in the training tracking data set. The same is also done with the car park training tracking data. Figure 7.4(d) shows the histogram. We set κ to 65 ($\approx 35\%$ eliminated). Thus like in the subset case, mainly trajectories from top to the bottom of the intersection and vice versa are chosen. Unfortunately, many trajectories with a few positions (tallest bar) are also in the resulting set. They belong to features which are detected and then only tracked for a few frames until they move out of the tracking area. The reason of this short life-times are the positions of the two detection regions which cover both lanes of the main road and lie in proximity of the tracking region's border. Consequently, 519 trajectories were eliminated and 933 remained in the training tracking data set.

Spatial obvious outlier detection was also performed after obvious temporal outlier detection on both training subsets and sets respectively. The percentile c was set to 5.991 which corresponds to a confidence interval probability of $\alpha = 0.95$. Figure 7.6(a) and 7.6(b) show the results on the highway/car park training subset. All trajectories show the expected, obvious tracking errors. 28/20 spatial

	Highway		Car park	
	Set	Subset	Set	Subset
size	2698	400	1452	400
obvious temporal outliers	329	50	519	221
after removal	2369	350	933	179
obvious spatial outliers	57	28	61	20
final size	2312	322	872	159

Table 7.3: The results of obvious outlier detection are summarized.

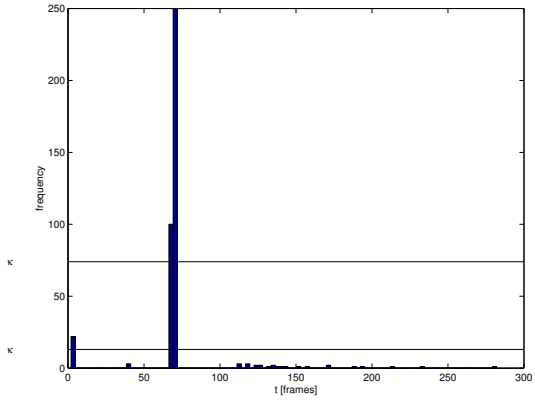
obvious outliers were removed. Consequently, 322/159 trajectories remained in the training subset. In case of the whole sets, 57/61 outliers were detected which led to a final training set of 2312/872 trajectories.

Table 7.3 summarizes the results of obvious outlier detection.

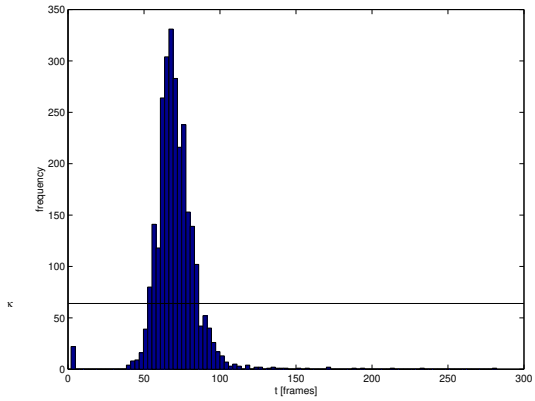
Noise reduction: Consider figure 7.7 which shows a synthetic example of a trajectory $\vartheta(t_i)$ which consists of 10 feature positions. If noise reduction is successful with this data then we will expect the same with real data. Each feature's position is taken at particular time instants $t_1, \ldots, t_{10}$. They are unknown and not relevant for our noise reduction investigation, because only spatial noise is considered. Furthermore, 100 slightly different trajectories $\mathcal{M}_1, \ldots, \mathcal{M}_{100}$ are generated using this ground-truth trajectory. $\mathcal{M}_1, \ldots, \mathcal{M}_{10}$ are shown by gray lines in figure 7.7. Each generated position p_i on these trajectories varies by a stochastic process $\epsilon_i \sim \mathcal{N}(0, I_2)$. Consequently, ϵ_i is the introduced noise. $\vartheta(t_i)$, p_i and ϵ_i correlate with each other by the presented noise regression model (4.3) in chapter 5. These generated trajectories are then smoothed by a smoothing spline with $\lambda^* = 835$. This leads to smoothed trajectories $\hat{\vartheta}_1(t_i), \ldots, \hat{\vartheta}_{100}(t_i)$.

The example shows that for the average of all 100 trajectories noise is reduced until a certain value of λ. In fact, while $\lambda \to \infty$, the smoothing spline becomes more and more a linear regression function which increases the bias of the smoothing spline and thus the noise. Remember the bias-variance tradeoff discussed in section 4.2.3. Table 7.4 summarizes these results of mean noise reduction in $p_1, \ldots, p_{10}$ for different increasing values of λ. The smoothing trajectories with $\lambda = 0$ are equal to their corresponding $\mathcal{M}_i$. Therefore, noise is not reduced. For $\lambda = 10$ and $\lambda = 100$ noise is on average reduced in all positions of all $\mathcal{M}_i$. Then, noise becomes larger instead of being reduced in some positions, because the bias becomes larger.

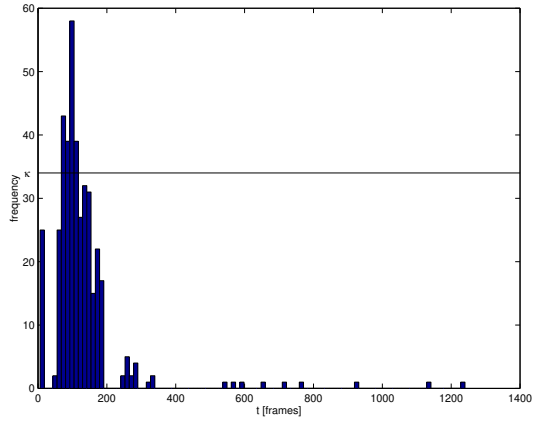
To solve the optimal tradeoff between noise reduction and noise increase, section 4.2.3 discussed some estimation methods for λ. The discrepancy method was used to determine λ^* for every $\mathcal{M}_i$ in this experiment. Then, an average $\lambda^* = 835$ was calculated and all $\mathcal{M}_i$ were smoothed once more. Figure 7.8 shows the amount of noise reduction with respect to λ. For all $\lambda < 1000$, only



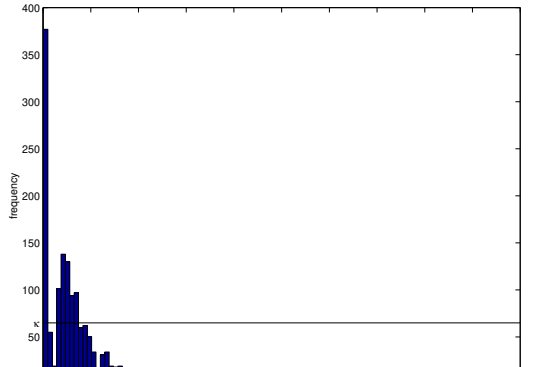
(a) Highway subset histogram

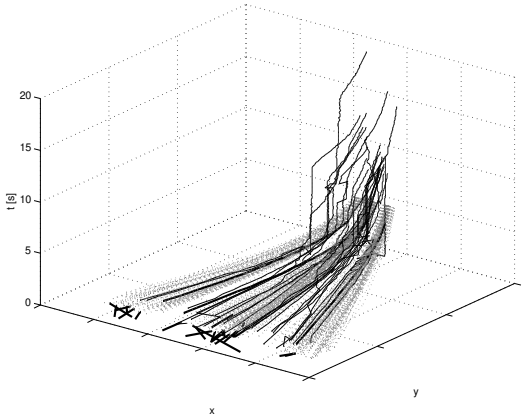


(b) Highway set histogram

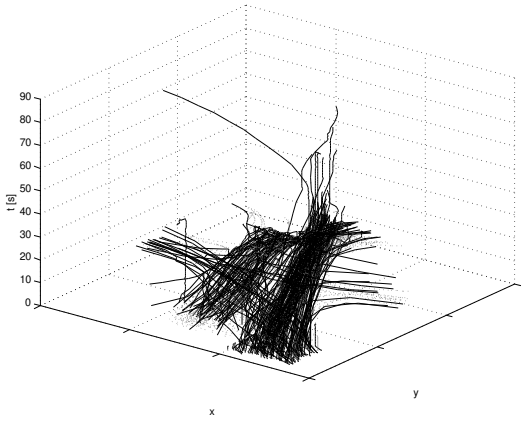


(c) Car park subset histogram



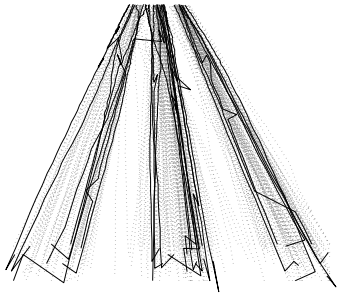


(a) Highway



(b) Car park

Figure 7.5: The results of obvious temporal outlier detection are shown. The experiments are done with the highway and car park training subsets. (a) shows the results with the former training set. While a threshold $\kappa = 13$ removes all obvious time-delayed feature trajectories (thin black lines), a larger threshold $\kappa = 76$ also removes the short erroneous trajectories (thick black lines). Consequently, the training subset is reduced to the 350 trajectories (gray dotted lines). (b) shows the result for the car park training set. κ was set to 34. Thus, 171 trajectories were labelled as outliers (black lines) and only 179 trajectories (gray dotted lines) remained in the training set.



(a) Highway

(b) Car park

Figure 7.6: The results of the obvious spatial outlier detection are shown. (a) shows 28 outliers (black lines) in the highway training set. The remaining training set consists of 322 trajectories (gray dotted lines). (b) shows 20 outliers (black lines) in the car park training set. The remaining training set consists of 159 trajectories (gray dotted lines).

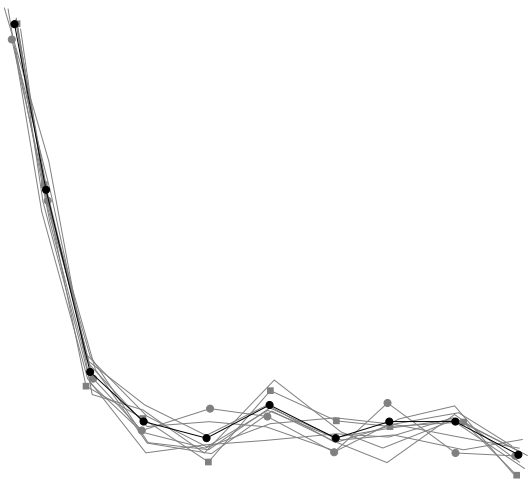


Figure 7.7: The ground-truth trajectory $\vartheta(t_i)$ is shown by a black line. Each black circle shows a true position p_i of the feature. The most right and lowest position is taken at t_1. The most left and upper position corresponds to t_{10}. $\mathcal{M}_1, \ldots, \mathcal{M}_{10}$ are shown by gray lines. Gray circles show $\mathcal{M}_1$'s positions while the positions of $\mathcal{M}_2$ are drawn by squares.

three of ten positions show an increase in noise. One position shows a declining decrease in noise reduction. Furthermore, noise is in sum on average more reduced than increased for the whole trajectories (see table 7.4). However, consider all $\lambda > 1000$. The increase in noise becomes significantly larger in four positions. Also the balance between the average decrease and increase of noise becomes uneven. More noise is now introduced than reduced. Therefore, an average $\lambda^* = 835 < 1000$ is a good estimate.

Figures 7.9 and 7.10 show the results of training sample generation with the highway and car park training subset without any obvious outliers. Both figures show the training samples in sub-spaces x/y/t and dx/dy/t of the learning space. Every training sample is the result of re-sampling the smoothing trajectory. ds was estimated to 11.24/4.52 for the highway/car park subset case. Thus, 3940/4732 training samples were generated. In case of the whole sets ds was estimated to 7.95/5*. 36988/15119 training samples were generated. Table 7.5 summarizes the results.

7.3 Learning

For our experiments a GNG run was made with the highway and the car park training subsets. In both simulations the parameters were set to: $\tau_{max} = 40000$, $\tau = 300$, $\lambda_b = 0.05$, $\lambda_n = 0.0006$, $\alpha = 0.5$, $\beta = 0.0005$, $a_{max} = 88$. We chose the same values

*This value was chosen by hand. The estimated value was too small (0.07), because of the outliers.

mean noise reduction										
λ	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}
0	0	0	0	0	0	0	0	0	0	0
10	0	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	0
10^2	0	-0.08	-0.08	-0.08	-0.04	-0.07	-0.1	-0.04	-0.07	-0.02
835	-0.03	-0.17	-0.18	-0.08	0.16	-0.17	-0.2	0.32	-0.16	0.04
10^3	-0.03	-0.17	-0.19	-0.07	0.2	-0.18	-0.21	0.41	-0.17	0.08
10^4	-0.03	-0.04	-0.11	-0.32	1.07	-0.22	0.55	2.49	-0.22	2.15
10^5	-0.17	0.91	0.58	-0.38	0.15	1.84	3.77	4.59	0.29	6.75
10^6	2.07	2.1	0.39	0.22	0.59	4.39	5.72	5.16	1.62	10.21

Table 7.4: The mean noise reduction in 10 positions of the 100 randomly generated trajectories is shown. While the smoothing of the trajectories becomes larger ($\lambda \to \infty$), the introduced bias also increases. Therefore, noise increases more and more in all positions instead of being reduced.

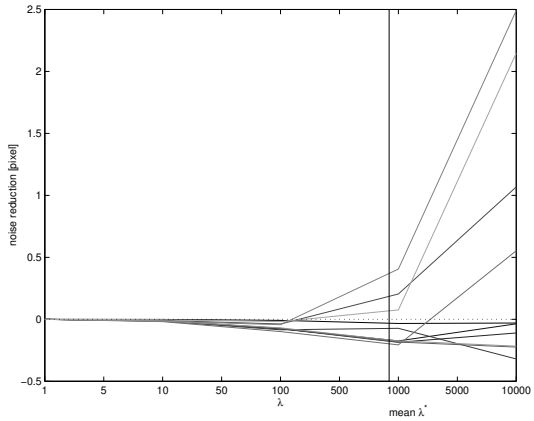
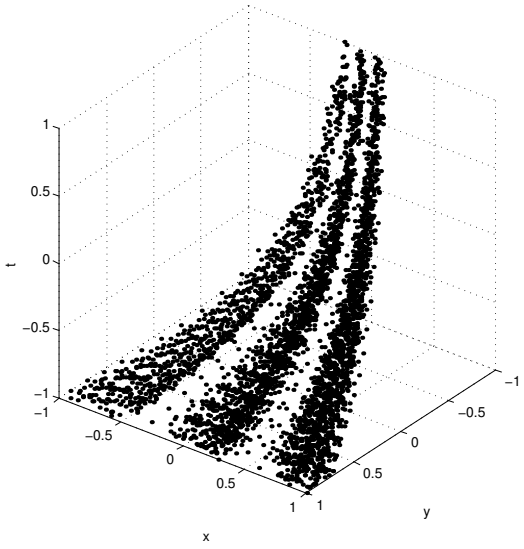
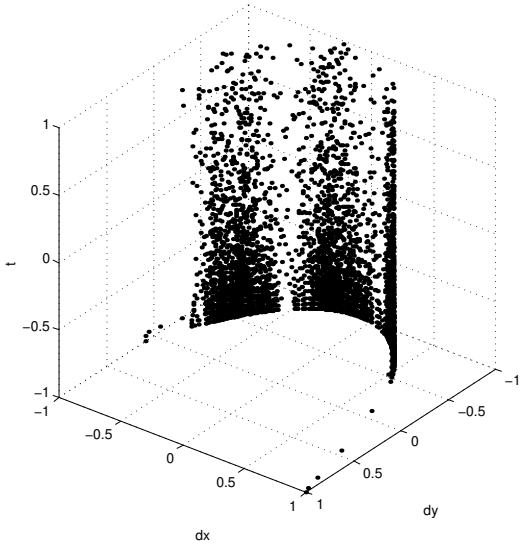


Figure 7.8: Noise reduction with respect to the choice of λ is shown. Noise increase and decrease in 10 positions is drawn by gray lines. The noise increase becomes significant for all $\lambda > 1000$. The mean of λ^* is estimated to 835.

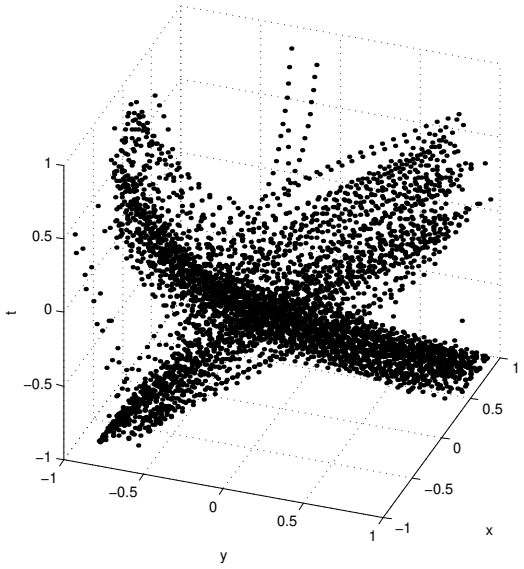


(a) x/y/t

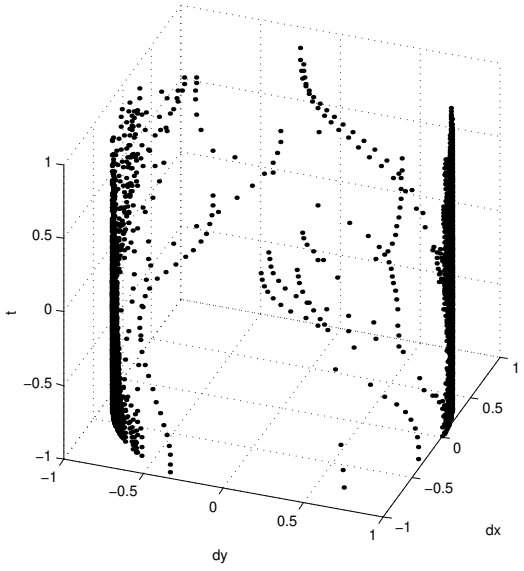


(b) dx/dy/t

Figure 7.9: The learning space of the highway training subset is shown.



(a) x/y/t



(b) dx/dy/t

Figure 7.10: The learning space of the car park training subset is shown.

	Highway		Car park	
	Set	Subset	Set	Subset
ds	7.95	11.24	5	4.52
training samples	36988	3940	15119	4732

Table 7.5: The results of training sample generation are shown.

as in Fritzke [15]. Thus, a codebook with 135 reference vectors was created for each subset. Figure 7.13(a), 7.13(b), 7.14(a) and 7.14(b) show the highway and car park results.

For the highway example, all three lanes are represented by the codebook. It can be seen that 135 reference vectors are too much to represent the data. Thus, the codebook over-represents the data which fulfills the goal that should be met by the use of the GNG. Especially consider the sub-space $< dx \in [0.5, 1]$, $dy \in [0, -0.5]$, $t \in [-0.5, -1] >$ in figure 7.13(b). It is obvious that this volume is over-represented. Besides, only a few training samples can be found. Therefore, these samples are spatio-temporal outliers.

According to the car park scene, two main directions can be recognized. These directions are from top to the bottom of the intersection and vice versa. Consider the two areas with a dense distribution of samples. We can also recognize these two sides of the road going in opposite directions in figure 7.14(a). All other driving opportunities are weakly represented by the training subset. The GNG has well explored the learning space. Most of the reference vectors lie within the two directions. Other directions are weakly represented. Therefore, most of them will be marked as spatio-temporal outliers in the successive MDL-based codebook pruning.

We used an instantiation of the MDL-based algorithm which is described in Bischof, Leonardis and Selb [6]. They assumed that the indices of the codebook are encoded with the optimal variable length according to the probability of occurrence. The error term is encoded according to a spherical Gaussian distribution with a fixed variance σ. Selb [38] reported that this parameter is relatively robust with respect to the final number of reference vectors which remain after pruning. In fact, he used relatively well defined and distributed clusters of training samples. Unfortunately, our learning space is not of this data distribution's type. Clear clusters cannot be found within both the highway and the car park training data.

Therefore, we did some experiments with σ and observed its influence on the final number of reference vectors and the number of outliers found after the MDL step. We expected a steady decrease of the number of reference vectors by increasing σ. Furthermore, we expected that if σ is small the spherical influence of a reference vector is small and therefore a large number of outliers (spatio-temporal outliers plus normal samples) must be the consequence. This number should decline until it is equal to the number of spatio-temporal outliers in the learning space. It should stagnate for a certain interval I, because all reference vectors represent normal training samples

and σ is still too small to include spatio-temporal outliers. Then, if a certain and large enough amount of σ is reached, all spatio-temporal outliers are represented by a few reference vectors or even one. Thus, a good value of σ could be the smallest value of I. This value is the first one where the number of outliers start to stagnate. Furthermore, the number of reference vectors regarding this particular σ is the largest with respect to all other values of σ in I.

Figure 7.11(a) and 7.11(b) show the results of the experiments according to the training subsets (highway/car park). 32/19 values of σ were evaluated between $[10^{-4}, 3 \cdot 10^{-3}]$ and $[10^{-4}, 4.6 \cdot 10^{-3}]$ respectively. In fact, the final number of reference vectors shows a steady decrease while σ becomes larger. Now, consider both outlier curves. Its values decay quickly from 60/68 to 17/25 within the intervals $I^{prv}_{hway} = [5.7 \cdot 10^{-4}, 8.5 \cdot 10^{-4}]$ and $I^{prv}_{cpark} = [5 \cdot 10^{-4}, 1.6 \cdot 10^{-3}]$ respectively. In fact, the number of outliers stagnates for the following interval $I_{hway} = [8.5 \cdot 10^{-4}, 9.3 \cdot 10^{-4}]$ and $I_{cpark} = [1.6 \cdot 10^{-3}, 2.5 \cdot 10^{-3}]$, before it decreases further. Although I_{hway} is equally large to its previous interval I^{prv}_{hway} the values remain between 17 and 18. Similarly in case of the car park training subset, the outlier values of I_{cpark} vary between 21 and 26, although I^{prv}_{cpark} is only $2 \cdot 10^{-4}$ larger. Consequently, these observations confirm our previous expectations. Therefore, we set the value of σ to $8.5 \cdot 10^{-4}$ in case of the highway and $1.6 \cdot 10^{-3}$ in case of the car park training subset.

The final results of the learnt codebook can be seen in figure 7.13(c) and 7.13(d) for the highway and 7.14(c) and 7.14(d) for the car park respectively. Beside spatio-temporal outliers, which were found by the MDL-based algorithm, all training samples which are part of the same path were also defined as outliers. Thus, the final number of training samples were 3714/4240 and the final number of training paths 305/134. Reference vectors which do not represent any training samples were also eliminated and led from 83/67 to a final codebook size of 82/51.

Figure 7.13(d) confirms our hypothesis that all samples within $< dx \in [0.5, 1]$, $dy \in [0, -0.5]$, $t \in [-0.5, -1] >$ are spatio-temporal outliers. The MDL pruning has taken this into account and has removed all reference vectors within this volume. Furthermore, the codebook distinguishes the three lane directions clearer than in figure 7.13(b). Reference vectors are more centered and those who represented training samples in between the three main directions are removed. If figure 7.13(c) is compared with 7.13(a), the same effect could also be seen.

In fact, a comparison of figure 7.14(b) and 7.14(d) shows the outlier detection ability of the MDL-based algorithm clearly. Indeed, all directions beside the two main directions are recognized as outliers. Reference vectors, which represent these samples after the GNG step, are eliminated. The same can be seen in the sub-space x/y/t. Both main roads of the intersection are well explored and represented. Beside these areas, the reference vectors are removed.

The last question with respect to learning is its robustness. Four runs of the learning algorithm with the highway training subset were done*. The parameters are

*Although the data-set is the same in all runs, the GNG algorithm chooses samples randomly.

	1	2	3	4
1	1	$\frac{19}{22}$	$\frac{20}{22}$	$\frac{19}{22}$
2	$\frac{19}{24}$	1	$\frac{20}{24}$	$\frac{20}{24}$
3	$\frac{20}{25}$	$\frac{20}{25}$	1	$\frac{18}{25}$
4	$\frac{19}{20}$	1	$\frac{18}{20}$	1

Table 7.6: The codebook's (row) outlier paths are compared to the outlier paths all other codebooks (columns). The percentage of identical outlier paths with respect to the total number is shown.

equal to the settings which are previously discussed. The following table shows the results for each run:

Run	Codebook size	Outlier paths
1	76	22
2	68	24
3	73	25
4	74	20

The final numbers of codebook vectors are similar. The difference can also be seen in the number of outlier paths which were detected. Table 7.6 shows the percentage ($1 \equiv 100\%$) of identical outlier paths for a particular codebook (row) compared to all other codebooks (columns). For example, compare codebook 1 with 2. 19 of 22 outlier paths are equal to outlier paths given by codebook 2.

The number of pairwise, equal outliers is approximately the same, namely between 18 and 20. The number of outliers according to a run of the learning algorithm is of the same order of magnitude, namely between 20 and 25. In fact, this confirms that the choice of σ works well. Only a small amount of normal paths were falsely declared as outliers. The small differences in the number of reference vectors due to the random choice of training samples during learning. Fortunately, this will become less and less a problem if the number of learning steps is increased. However, for these runs (40,000 steps), all four codebooks cover the same parts of the learning space which can be seen in figure 7.12. The spatio-temporal models are different in particular samples but very similar in parts of the learning space. We also did a successive classification with all four codebooks and the results in terms of the number of classes and their coverage of the learning space are equal.

Table 7.7 summarizes the learning results. We used the parameter settings from the training subsets to learn the codebook for the training sets.

	Highway		Car park	
	Set	Subset	Set	Subset
training samples	36988	3940	15119	4732
training paths	2312	322	872	159
reference vectors after GNG	135	135	135	135
reference vectors after pruning	119	83	115	67
outlier samples	2254	226	1775	492
outlier paths	121	17	94	25
reference vectors after removal	119	82	100	51
final training samples	34734	3714	13344	4240
final training paths	2191	305	778	134

Table 7.7: The learning results are summarized.

7.4 Classification

The previously learnt codebooks of the highway and car park training subset and all training paths except outliers were used for classification. First, every training sample was quantized to its corresponding reference vector. Thus, all paths were transformed into sequences of codebook vectors. In case of the highway/car park, 305/134 sequences were generated. Then, the incremental classification algorithm produced 157/79 different classes. Seven representatives are shown in figure 7.15 and 7.16 respectively. These classes were used by the successive merging algorithm which produced six classes for both training subsets in a first step. A second merging step led then to the final result of 3/4 classes. Every further attempt of merging did not change the result. The second and third column in both figures show these classes.

According to the highway scene, every class represents exactly one lane. We expected this result, because these three lanes are well separated and represented by reference vectors in the learning space. However, the car park scene classification presented a surprise. Instead of the expected two classes which should represent the heavy used main road in both directions, the classification resulted in four classes. For example, compare the 3^{rd} and 4^{th} class in the third column of figure 7.16. Both classes seem to be identical except the upper part of the scene where vehicles exit the intersection. All trajectories of vehicles which turned to the right into the neighboring street were summarized in one class. However, all other vehicles left the tracking area by moving straight ahead. Indeed, this makes less difference in the image plane, but an obvious one in larger sub-spaces of the learning space. Therefore, we considered both classes in the sub-space dx/dy/t. Figure 7.17 compares both classes and shows the differences in the direction. This fact will be even more clearly, if the first and the second class is compared. First, no obvious difference can be recognized. However, consider both classes in the sub-space x/y/t which are shown by figure 7.18. The

n	50	100	200	300	400	500	600	700	1500	2005
$\overline{c}$	40	69.6	116.2	157.8	191.4	228.9	257.4	286.6	475.2	573
σ_c	11	21.5	55.5	95.2	74.3	48.2	108.3	92.4	145.4	0

Table 7.8: The mean $\overline{c}$ and variance σ_c of c for specific n are summarized.

difference lies in time and not in the location. Interestingly, the first class summarize all vehicles which were delayed by the traffic at the intersection. In contrast to that, the second class shows vehicles driving exactly into the same direction and on the same road but without any obstructions.

The classification of the whole training sets led to the expected equivalent results. The training subset classes are also part of the classification. Nevertheless, the final number of classes in the highway/car park case is higher (4/8). For example, consider the highway classes which are depicted in figure 7.19. The first three classes also represent the three lanes of the highway. In fact, the last class represents vehicles which drove in between two lanes. Figure 7.20 shows all classes of the car park case. While the 1^{st} and 3^{th} class, which both are depicted in the first row, are equivalent to the 1^{st} and 3^{th} class of the training subset classification, the 2^{nd} class is a result of merging the 3^{th} and 4^{th} class. All other classes due to outliers. For example, cars which turn right at the intersection (7^{th} class).

Certainly, the complexity is of particular interest during the analysis of both classification algorithms which are shown and discussed in detail in chapter 7. Consider the incremental algorithm on page 90. The worst case effort to determine all c classes from a set of n sequences is

$$\frac{c(c+1)}{2} + (n-c-1)\frac{c}{2} = \frac{cn}{2}.$$

During initialization, the first class is created by the first sequence. Then, $n-1$ sequences are compared to the current classification. In the worst case, the first c sequences create all classes. Then, for the final $n-c-1$ sequences $\frac{c}{2}$ comparisons are needed on average. If every sequence forms its own class ($c = n$), then the worst case effort is $\frac{n^2}{2}$. Thus, the complexity of the incremental algorithm is $O(n^2)$.

Generally, the number of classes should be smaller than the number of sequences which are used for classification ($c \ll n$), because all sequences represent normal traffic and arbitrary sequences (combinations) of reference vectors are impossible. To investigate this hypothesis, 25 sets of $n = \{50, 100, 200, 300, 400, 500, 600, 700, 1500\}$ sequences were randomly chosen from the highway training set. Then, c was evaluated by the incremental algorithm for every set. Thus, the mean and the variance of c for a specific n was calculated which is shown by table 7.8. All 225 chosen sets and the whole training set showed a smaller c than n. Furthermore, an increase of c with respect to n can be seen. In fact, an upper bound of c must exist, because the codebook consists of m reference vectors which can create at most $\sum_{i=1}^{m} \binom{m}{i}$ classes. Figure 7.21 shows the relation as function $c(n)$ which asymptotically approaches this

upper bound. For $n = 0, \ldots, 600$, $c(n)$ can be approximated by $c(n) = 2.2n^{\frac{2}{3}}$. The worst case effort of the incremental algorithm with respect to the highway training set, the codebook which was learnt by the training subset and the former interval of n is $1.1n^{1.67}$. Thus, the complexity for $c < n$ is generally $O(n^i)$ with $1 < i < 2$.

Finally, consider the merging algorithm on page 92. The algorithm is initialized (run r=0) with a classification $M_r = M_0$ delivered by the previously performed incrementation step. Then, it randomly chooses a class C from M_r and compares C with all other classes within M_r. Consequently, the effort is $|M_r| - 1$ to find equivalent classes of C. Finally, these classes are removed from M_r which lead to M_{r+1}. The algorithm continues until $r = r_{max}$ with $M_{r_{max}} = \emptyset$. Thus, the effort of the merging algorithm is $\sum_{r=1}^{r_{max}} r(|M_r| - 1)$. Consider now the worst case, where only one class is removed from M_r in run r. Then, the worst case effort is $\frac{c(c-1)}{2}$ with $r_{max} = c$. Thus, the complexity of the merging algorithm is $O(c^2)$.

Generally, only a few traffic patterns exist within a scene. Therefore, many classes in M_0 are equivalent. The difference $|M_{r+1}| - |M_r|$ between two runs is large which reduces further comparisons significantly and the expected effort is much smaller than the worst case one. In fact, this can be seen in figure 7.22 where we performed the merging for the highway and car park training subsets. r_{max} was 6. $|M_r|$ decreased exponentially in both training cases.

7.5 Case studies

This section treats the results of the spatio-temporal model and the classification with respect to the test tracking data described in section 7.1. As the latter represents unusual events, the p.d.f. value should be small or no classification is expected for the measurements.

First like in the training case, test samples of particular trajectories are generated. We chose a representative trajectory from the traffic jam tracking data set (figure 7.25(a)), a feature which pursues a van that changed onto the reserved lane (figure 7.25(b)) and a trajectory which completely shows a prohibitively driving car (figure 7.25(c)). According the car park case, we chose a feature on the stopped (figure 7.27(a)) and the back pushed car (figure 7.27(c)). Finally, a trajectory of an unobtrusive car was also selected (figure 7.27(b)).

Exactly the same number of test samples than measurements are generated by using an interpolating spline. We avoided to use the smoothing spline, because only partial trajectories are known during tracking. Estimated samples could otherwise become false if more and more about the trajectory is known. To be able to classify at a certain time instant, the feature's partial sequence is considered which includes all so far visited reference vectors. The classification itself is simple. If all reference vectors of the partial sequence are a subset of the reference vectors of a class, then the feature will belong to this class. Certainly, a feature can be assigned to more than one class

in the beginning of tracking. While more information of the trajectory is gained, this number decreases until only one or no class remain in the classification.

Figure 7.23 shows the classification result for each test feature. For example, consider the left upper figure. It shows the traffic jam feature as black thick line and the codebook as gray graph in the $x/y/t$ sub-space of the learning space. The corresponding class, which is a sub-graph, is black highlighted and represents the left lane. Indeed, we have expected this fact, because the feature is on a car which drives on the left lane. The prohibitively driving vehicles are classified similarly. Instead of the most left, the most right class is assigned. The second figure in the first row shows the result for the van which changes the lane while the third figure shows the completely prohibitive driving car. However, the second row shows the car park results where two unclassified features are shown. The stopped and back pushed car show traffic patterns which are not represented by the class models. Therefore, no sub-graphs are highlighted in the first and third figure. In contrast to that, the second figure shows an unobtrusive driving car. Consequently, it can be classified.

Beside the classification, the response of the spatio-temporal model with respect to each test sample has to be evaluated. Figure 7.24 shows the results for the highway case. The left upper figure depicts the p.d.f. values for a training path. The values are normalized to 1. All values lie between 0.86 and 1. However, the next figure to the right shows the consequences of a traffic jam. The p.d.f. values of the model decreases to 0 within 50 frames and remain 0 with frame 100. Interestingly, it increases slightly between frame 50 and 100. The reason is that people normally brake stronger than necessary if they see a traffic obstruction in front of them. Therefore, they release their brakes and accelerate a bit until they finally stop their vehicle. In contrast to that, the first figure in the second row shows the lane changing van which produces a steady decrease until 0. Only around frame 60 a slightly increase can be seen. The test samples of the feature fit the model better in the end, because of the camera's perspective. The last figure shows the model response for the wrong driving car. P.d.f. values lie between $1.4 \cdot 10^{-4}$ and 0.

Figure 7.26 depicts the p.d.f. values of the spatio-temporal model in case of the car park. Values of a training path lie between 0.3 and 1. This relatively small values due to the small training set. More complex scenes need far more training samples to produce an appropriate model. The upper right figure shows the expected p.d.f. values of the stopping car. As long as the car drives on the main road, the values are above 0.2. Then, all values are zero, because the car shows an unusual traffic pattern by turning to the left and stopping. The results of the normal driving car are shown in the first figure of the second row. All values remain above 0.1. The alternating increase and decrease of the values can be explained by the alternating proximity and distance to a reference vector. Finally, the last figure shows the results of the back pushed car. It is most of the time outside the spatio-temporal model except the frames where it turns from the left of the intersection into the main road.

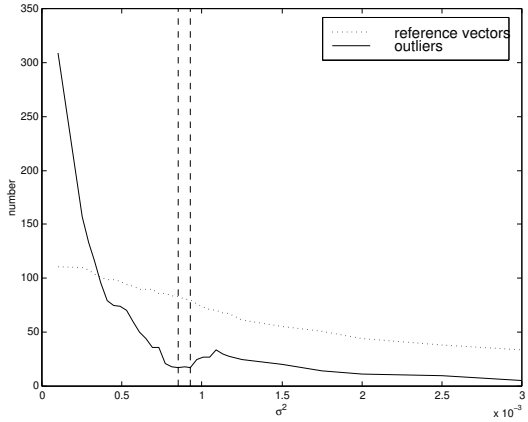
7.6 Conclusion

To sum up, 2 training sample sets and 2 sub-sets were generated from two distinct scenes, namely a highway and a car park. Besides, unusual events like a traffic jam or a back-pushing car were used to test the capability of the spatio-temporal model and the classification. We showed that obvious outlier detection principally works with the given data during the generation. Threshold κ was chosen by hand after investigating the histograms. Manual threshold selection could be avoided by rejecting a fixed percentage (e.g. 15%) of the training trajectories. The next chapter will further discuss this idea. Furthermore, noise reduction was demonstrated by a synthetic example.

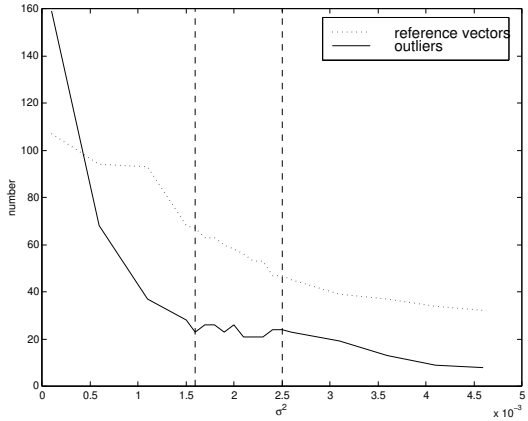
Then, both training sample sets were used to create spatio-temporal models. All lanes of the highway and the main road of the car park were represented. Spatio-temporal outliers were found and the codebook could be minimized. The choice of σ is determined by an interval where the number of outliers stagnate. The robustness of learning depends primary on the learning steps while the accuracy of the spatio-temporal model depends mainly on the number of training samples.

Classification delivered the expected 3 classes in the highway case. 4 classes were found in the car park scene. The algorithm recognized successfully the different patterns of traffic behaviour in both scenes.

Finally, we evaluated for every unusual event the normalized p.d.f. value and the class of the corresponding feature. The results met our expectations.

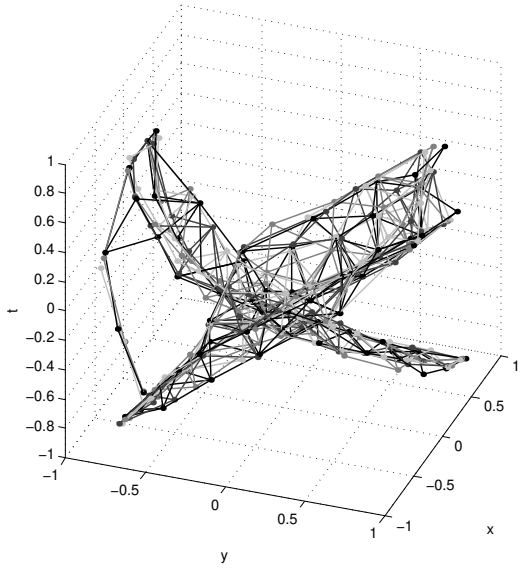


(a) Highway

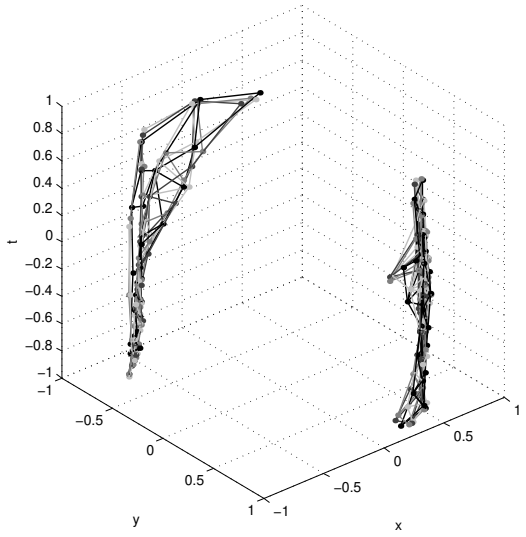


(b) Car park

Figure 7.11: The influence of σ with respect to the final number of reference vectors in the codebook (dotted line) and the number of spatio-temporal outliers (solid line), which are detected by the MDL-based algorithm, are shown. Interval I is shown in between dashed lines. (a) shows the results for the highway training subset, while (b) shows the results for the car park training subset.

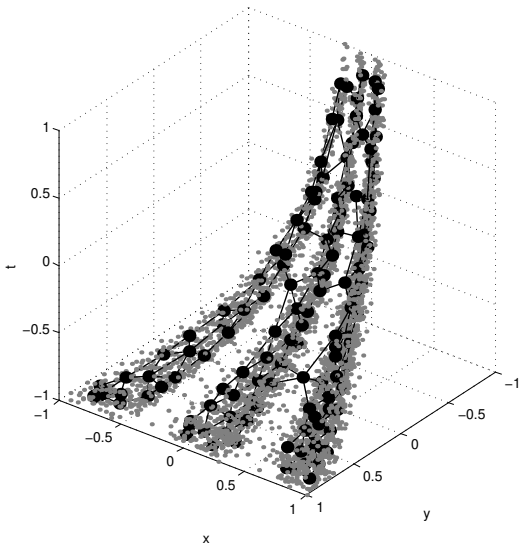


(a) x/y/t

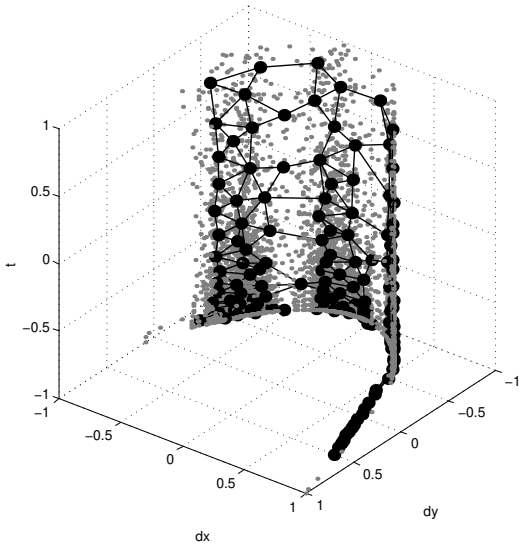


(b) dx/dy/t

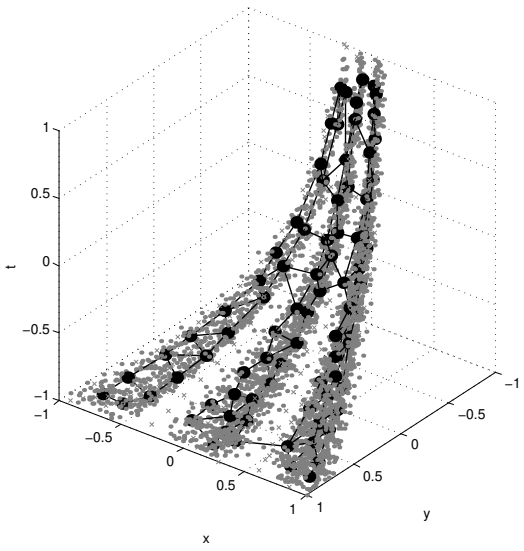
Figure 7.12: Four codebooks of four runs with the car park training subset and equal parameters are shown. The same parts of the learning space are covered. Nevertheless, reference vectors do not correspond.

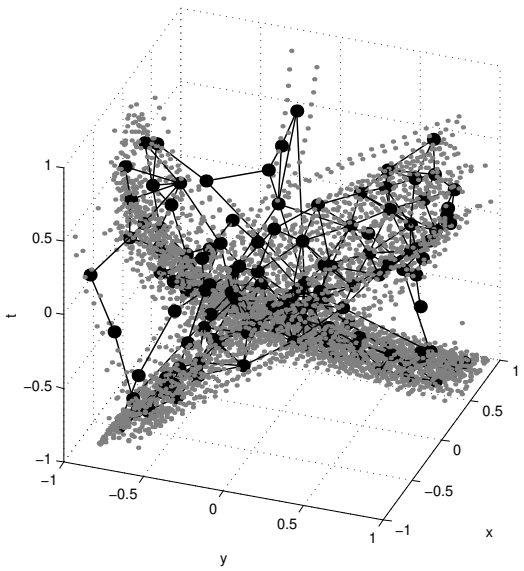


(a) x/y/t, after GNG exploration

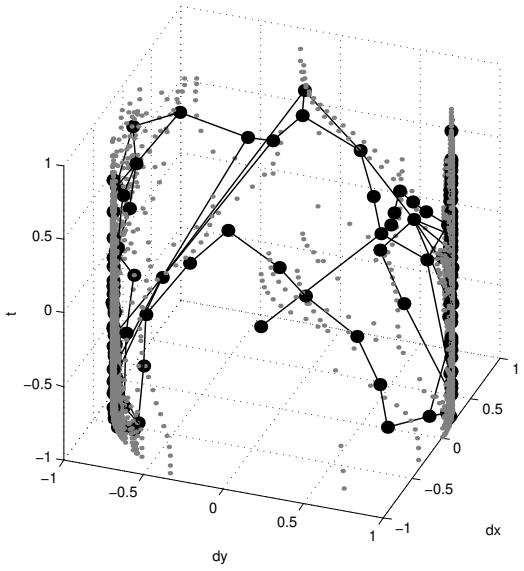


(b) dx/dy/t, after GNG exploration

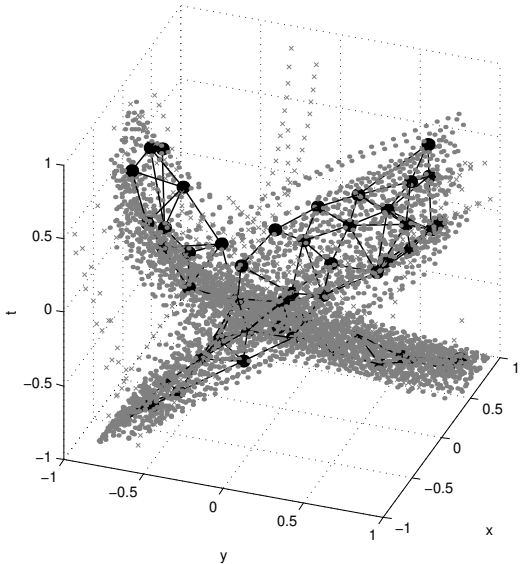




(a) x/y/t, after GNG exploration



(b) dx/dy/t, after GNG exploration



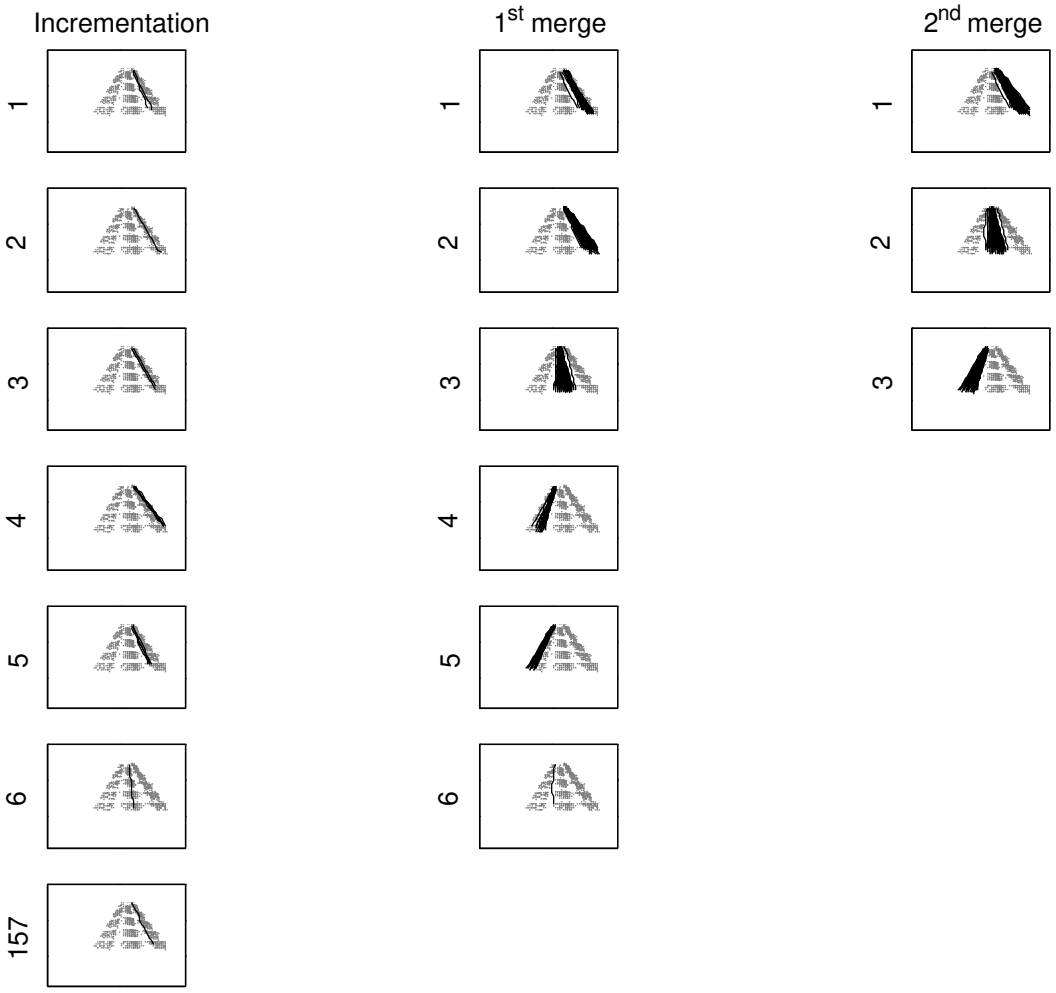


Figure 7.15: Three classes are the final result of classifying the highway scene with the training subset. Every class exactly represents one lane. The first column shows the classification result after the incremental step. The second and third column shows the resulting classes after the first and second merging step.

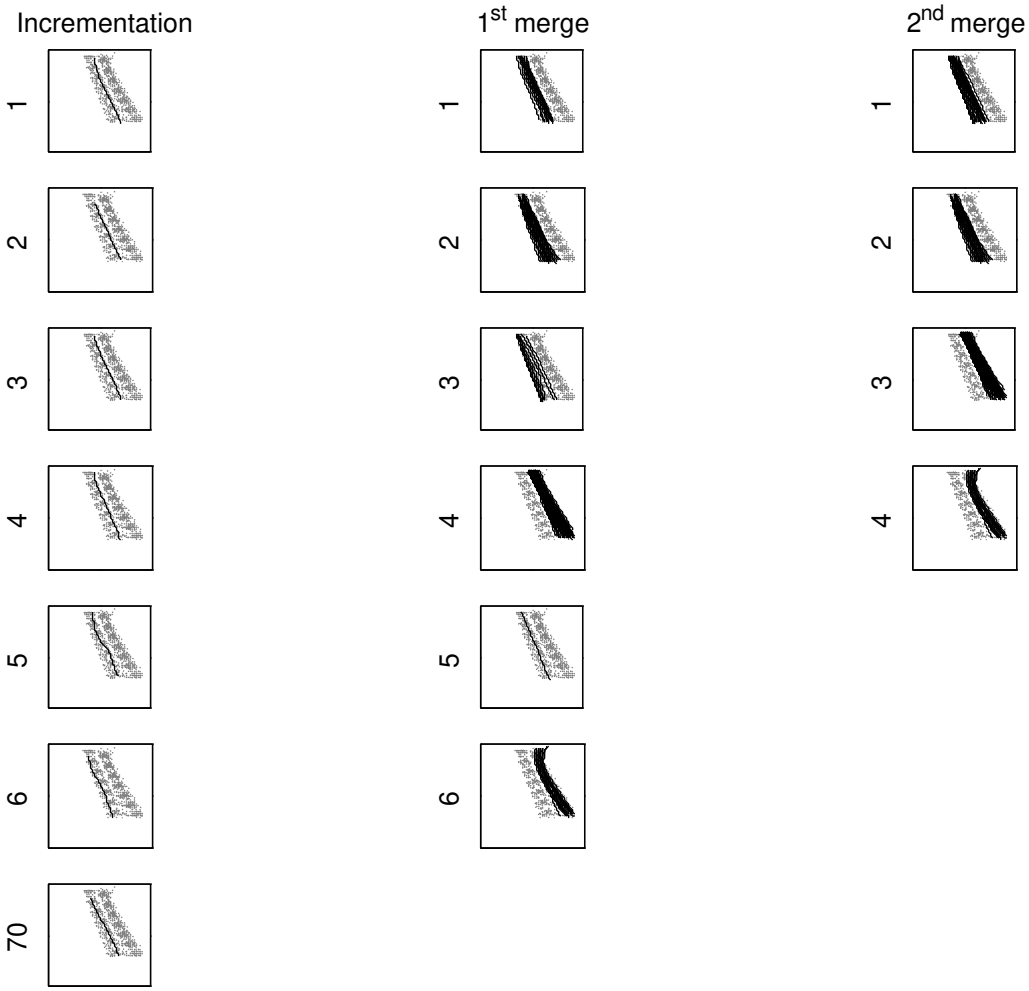
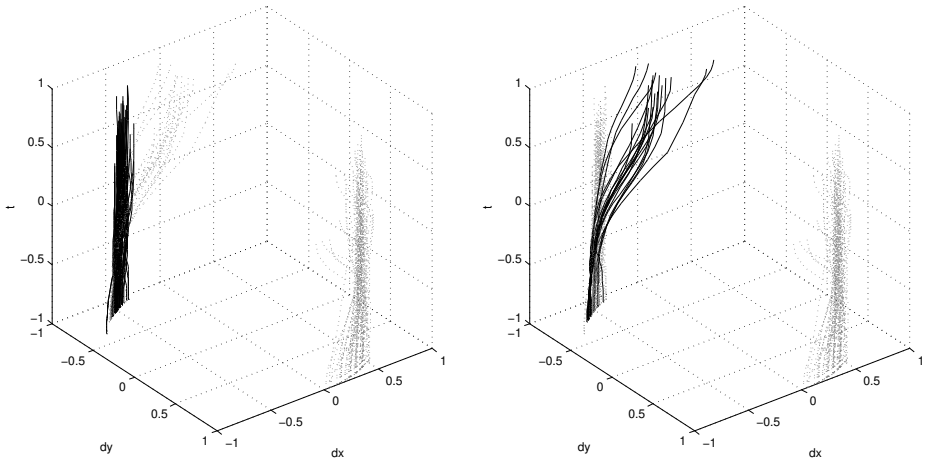
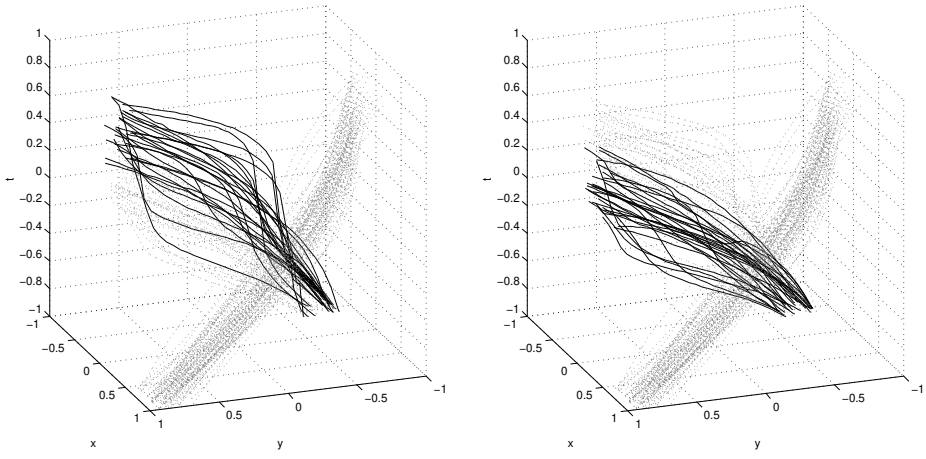


Figure 7.16: Four classes are the final result of classifying the car park scene with the training subset. The first column shows the classification result after the incremental step. The second and third column shows the resulting classes after the first and second merging step. The first and last two classes seem to be identical after the second merge. However, a further investigation shows that the former two classes differ in time and not in the image plane. The difference of the third and fourth class can be seen in the upper part of the scene where vehicles move straight ahead or turn to the right.



(a) 3^{rd} class **(b)** 4^{th} class

Figure 7.17: The difference between the 3^{rd} and 4^{th} class depends on the feature's direction dx/dy. Both figures (a) and (b) show the classes feature paths of vehicles which cross the intersection on the right lane of the main road. Both classes are identical for $t \in [-1, 0]$. Then, the vehicles of the 4^{th} class turned to the right into the neighboring street. This can be seen in (b) by the bend feature paths.



(a) 1^{st} class **(b)** 2^{nd} class

Figure 7.18: The difference between the 1^{st} and 2^{nd} class lies in time t. Both figures (a) and (b) show the classes feature paths of vehicles which crossed the intersection. They drove on the left side of the main road in the same direction. (a) shows paths of obstructed vehicles which had to slow down their speed or had to stop. The life-time of these features lies between 0.2 and 0.6. (b) shows free flowing traffic with life-times below 0.2.

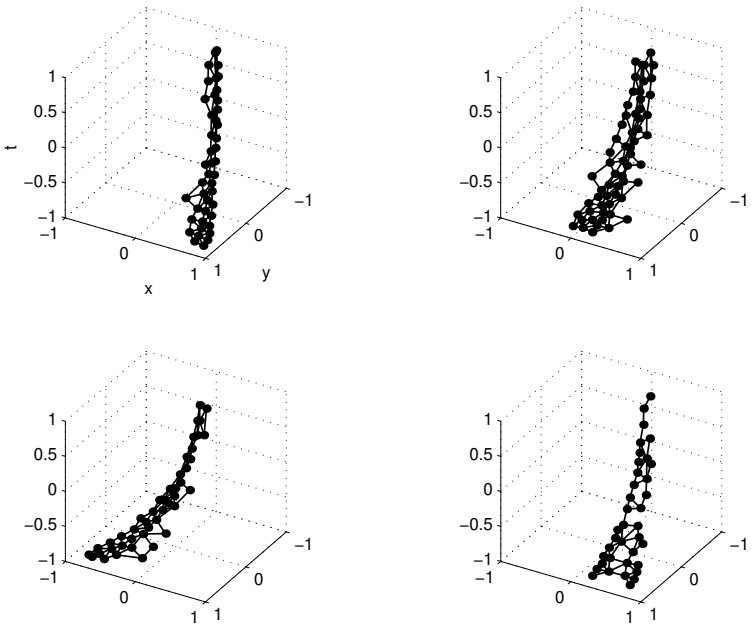


Figure 7.19: The classification result of the highway training set is shown. The result is equivalent to the training subset classification except the 4^{th} class.

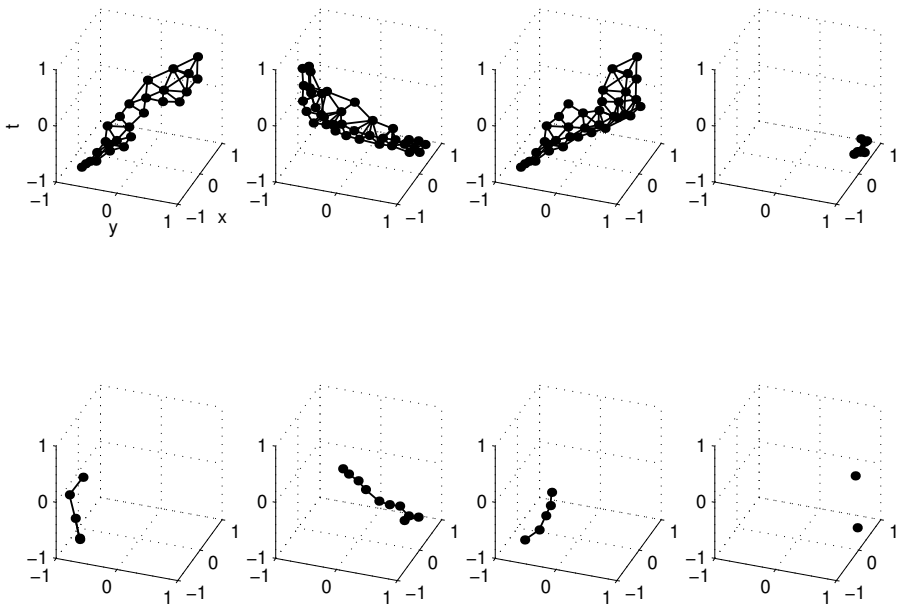


Figure 7.20: The classification result of the car park training set is shown. The first three classes represent all traffic patterns while all other classes are the result of outliers.

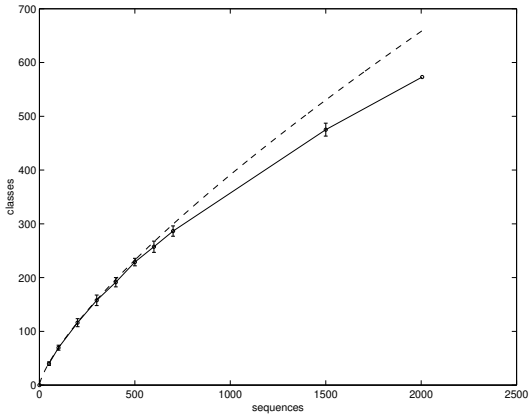
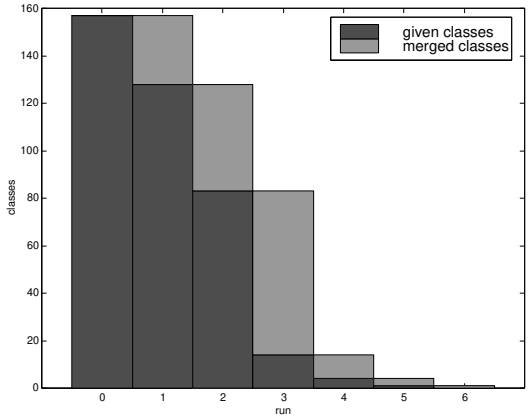
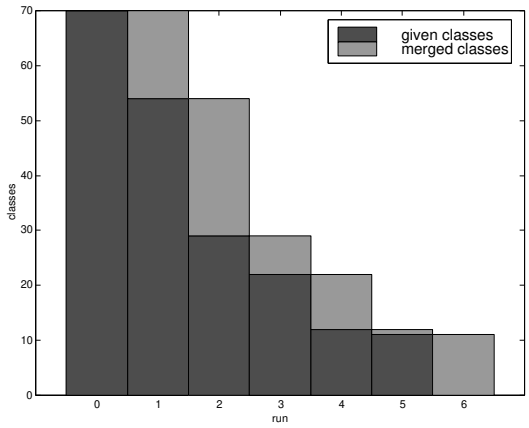


Figure 7.21: $c(n)$ (solid) and its approximation $2.2n^{\frac{2}{3}}$ for $n = 0, \ldots, 600$, $c(n)$ are shown. The variance in every point is also shown (intervals).



(a) Highway



(b) Car park

Figure 7.22: The number of the remaining classes in M decreases exponentially with every merging step. (a) shows the number of merged classes in the highway while (b) depicts the car park case.

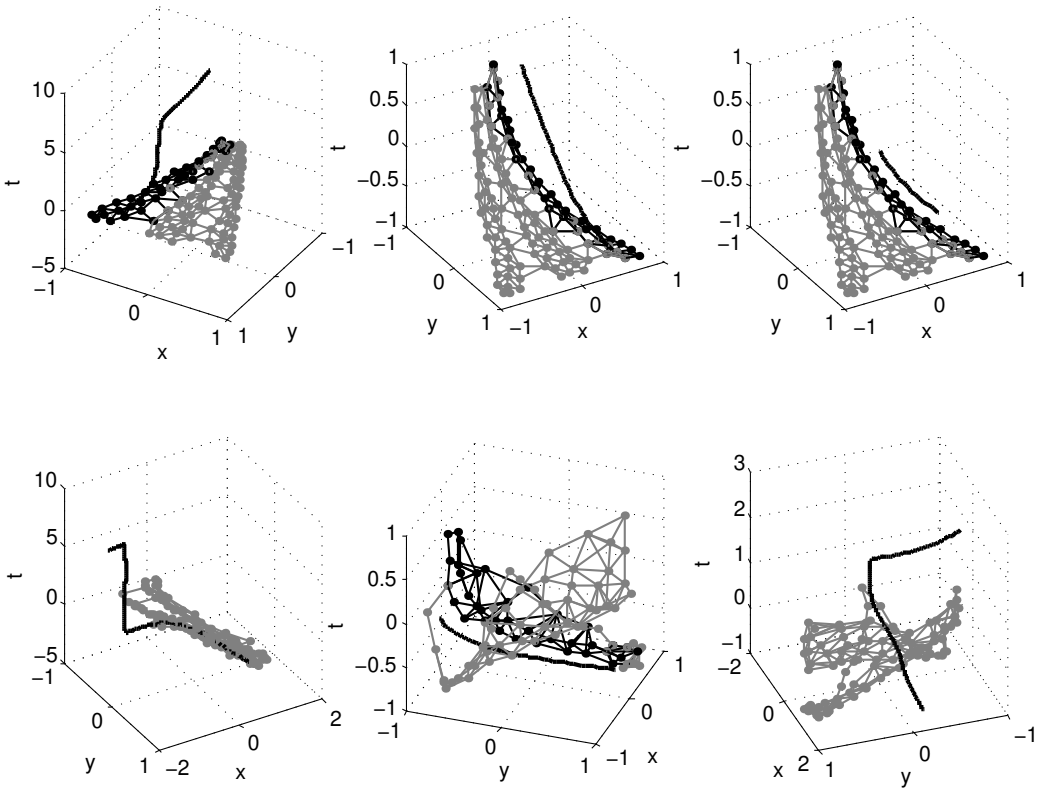


Figure 7.23: The first and second row shows the classification results of three chosen features from the traffic jam, reserved lane, stopped car and back pushed car tracking data set. Trajectories are thick black lines. The codebook is shown as gray graph. Corresponding classes are sub-graphs which are black highlighted.

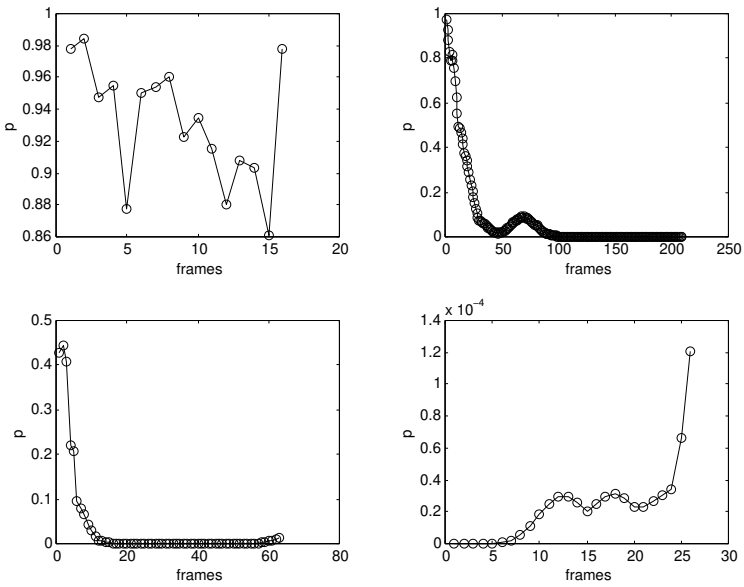


Figure 7.24: The results of the spatio-temporal model response with respect to the test samples of the highway scene are shown. The upper left figure shows a randomly chosen training path. The upper right figure depicts the p.d.f. values of the traffic jam feature. The lower left figure shows the feature on a van which changes to the reserved lane. The lower right figure shows a wrong driving car.

(a) Traffic jam

(b) Changing to reserved lane

(c) Driving on reserved lane

Figure 7.25: The feature trajectories for testing are shown. (a) shows a feature in the traffic jam. (b) shows a feature on a van which changes to the reserved lane. (c) shows a feature trajectory of a car which drives wrongly.

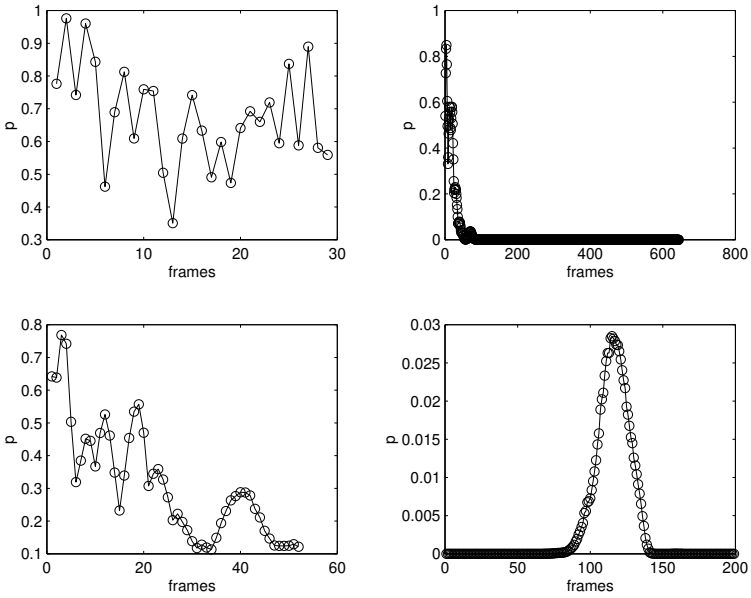


Figure 7.26: The results of the spatio-temporal model response with respect to the test samples of the car park scene are shown. The upper left figure shows the p.d.f. values of a randomly chosen training path. Then, the upper right figure shows the stopped car. The lower left figure shows the unobtrusive car while the lower right figure depicts the back pushed car.

(a) Stopping car **(b)** Unobtrusive car **(c)** Pushing-back car

Figure 7.27: The feature trajectories for testing are shown. (a) shows a feature on the stopped car. (b) shows a feature on a an unobtrusive, normal driving car. (c) shows a feature on the back pushed car.

Chapter 8

Discussion

This report presented unusual traffic event recognition (e.g. traffic jams, prohibitively driving vehicles) by using a spatio-temporal model of traffic behaviour. The latter is the probability density function (p.d.f.) in a learning space which is build by training samples of normal traffic. These samples are generated by tracking data which is collected by a visual interest-point tracking algorithm (KLT-tracker). Interest-points are features on vehicles which are detected in a user-defined detection region and pursued within a tracking area in a static camera scene. In fact, features represent vehicles but do not identify them. The way a feature takes through the image is called trajectory. It only is a measured trajectory, because features positions are noisy due to the properties of the tracker. A set of trajectories is then the tracking data.

The spatio-temporal model is learnt by unsupervised learning. In contrast to other works, we used a Growing Neural Gas in combination with a MDL-based pruning algorithm as p.d.f. estimator of the learning space. First, obvious outliers in the tracking data are eliminated. Then, smoothing splines are used to approximate the trajectories. The reasons are the reduction of noise and the possibility of re-sampling the trajectory as the density of the learning space should only depend on the abode probability of features. An advantage of the MDL-based pruning algorithm is that spatio-temporal outliers are detected. Thus, they do not influence the codebook of reference vectors which is the result of unsupervised learning. Each sample is represented by the nearest reference vector. Then, the sum of spheral Gaussians according to the Euclidean distance is the spatio-temporal model.

We showed that it is not sufficient to recognize correctly unusual events. It only describes normal traffic behaviour in a point of the learning space. In fact, the properties of trajectories are not considered. Therefore, we quantize every trajectory of the training data into sequences of reference vectors. In a first incremental step, two sequences belong to the same behaviour class if they are identical. Then, classes are merged by an equivalence constraint. Two classes are equivalent, if either reference vectors of one class also occur in the other class or at least an edge of the topology

graph connects distinct reference vectors of both classes. The topology can be interpreted as neighborhood of reference vectors. This merging step is iteratively done until the set of classes do not change any more.

The spatio-temporal model and the behaviour classes were then used to evaluate some case studies. Examples of a traffic jam, vehicles on the reserved lane or a car which stopped for an unusual time showed that the comparison of the actual traffic behaviour with the spatio-temporal model is able to recognize unusual events.

To sum up, the tracker showed satisfying results in daylight under good illumination conditions. Our experiences correspond to the results of Beymer et al. who also used a comparable tracking approach. They showed that it is even possible to group features together. Thus, vehicles can be identified and counted. Other traffic parameters like speed were also evaluated. Unfortunately, under bad conditions like reflections in tunnels the tracker showed its weakness. As feature correspondence between two frames is established by intensity values, the approach is inherently sensitive to noise. Thus, features get lost or caught by other vehicles or the background. However, features on vehicles allow tracking of heavily congested traffic. All other region or blob based tracking methods fail in case of occluded vehicles.

In contrast to Johnson and Hogg, Stauffer and Grimson, we overcome the problem of ”stranded” reference vectors by using a soft-competitive unsupervised learning algorithm. Furthermore, we do not define a fixed number of reference vectors. Instead, an optimal number which depends on the training data is found by the MDL criterium. Beside the pruning algorithm detects spatio-temporal outliers. Certainly, a weakness of the current implementation is its off-line nature. Thus, a finite number of features have to be tracked and stored to generate the codebook.

Interestingly, the learning alone produces behaviour classes, because the topology graph is under closer consideration a forest. Each component represents one behaviour class. To use the topology, which is simultaneously learned with the codebook, for classification is a novel approach and was not considered by previous works. To have a grainer classification we developed an incremental and merging algorithm. We showed successfully that traffic behaviour can be analyzed and classified. The only drawback is the quadratic complexity of the algorithms. Nevertheless, the algorithms are straight forward, because only comparisons have to be done. All other works presented algorithms which estimate the classes by a learning scheme. This is not only more computational expensive, it also introduces further inaccuracies into the computation.

8.1 Future work

To improve the tracker with respect to illumination we could use improved KLT-Trackers like it is discussed in Jin et al. [21]. Nevertheless, the problem illumination and noise cannot be solved by this type of tracker, because the assumption of nearly

static illumination further exists. Thus, new tracking concepts which probably have only the assumption of static geometry have to be found. Robust tracking algorithms under all conditions (reflections, weather conditions) which allow occlusions are still an open question.

The obvious temporal outlier detection could also be improved. A threshold could be neglected if the modes of the histogram could be found automatically. Then, a corresponding number of features are chosen around this modes. This number results from the number of remaining temporal outliers which can be defined as a fixed percentage of the tracking data (e.g. 15%).

As mentioned above, one of the weaknesses of learning and classification is its off-line implementation. Future works should try to find on-line algorithms. They have not to be real-time capable like the tracker, because features are used as their whole trajectory is known. This time depends on the scene but is normally some seconds. Certainly, then a couple of features could be available as tracking data. Also the obvious temporal outlier detection has to be modified, because it bases on the statistics of the life-time. A solution could be a first step of gathering statistical information by collecting tracking data. Then, in a second step the generation of the training data and the learning is done.

The decision about an unusual event was not answered at all by this report. During the use of this recognition system, the properties of every actual feature are compared to the spatio-temporal model. Furthermore, the actual trajectories are classified by the behaviour classes. If no classification is possible or the probability value of the model drops below a certain threshold, the feature indicates an unusual event. Indeed, it could also be an outlier. Therefore, a robust fusion of feature responses should be performed to get reliability of the recognition system. Unfortunately, if we use only such a threshold for the model's probability we cannot differ between traffic jams or other unusual events. Perhaps it is possible to interpret the probability values if we consider them as time series. Then, symbolic traffic information could be collected and stores in a database for further use.

Bibliography

[1] P. Balakrishnan, M. Cooper, V. Jacob, and P. Lewis. A study of the classification capabilities of neural networks using unsupervised learning: A comparison with k-means clustering. *Psychometrika*, 59:509–525, 1994.

[2] G. Ball. Data analysis in the social sciences: What about the details? In *Fall Joint Computing Conference*, pages 533–559, Washington, DC, 1965. Spartan Books.

[3] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. In *Computer Vision Pattern Recognition*, 1997.

[4] J. Bezdek. Cluster validity with fuzzy sets. *Journal of Cybernetics*, 3:58–72, 1974.

[5] H. Bischof and A. Leonardis. Vector Quantization and Minimum Description Length. In S. Singh, editor, *International Conference on Advances in Pattern Recognition*, pages 355–364. Springer Verlag, 1998.

[6] H. Bischof, A. Leonardis, and A. Selb. MDL principle for robust vector quantization. *Pattern Analysis and Applications*, 2(1):59–72, 1999.

[7] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Bookcraft Ltd, Midsomer Norton, Avon, 1995.

[8] C. de Boor. *A practical guide to splines*. Springer-Verlag, New York, 1978.

[9] D. Desieno. Adding a conscience to competitive learning. In *International Conference on Neural Networks*, pages 117–124. IEE Press, 1988.

[10] N. J. Ferrier, S. M. Rowe, and A. Blake. Real-time traffic monitoring. In *2nd IEEE Workshop on Applications of Computer Vision*, Sarasota, Florida, 1994.

[11] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics - Principles and Practice*. Addison Wesley, second edition, 1996.

[12] E. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. In *Biometric Society Meetings*, volume 21, pages 768–769, Riverside, 1965.

[13] B. Fritze. A Growing Neural Gas network learns topologies. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, Cambridge, MA, 1995.

[14] B. Fritzke. Growing Self-organizing Networks - Why? In M. Verleysen, editor, *European Symposium on Artificial Neural Networks*, pages 61–72, Brussels, 1996.

[15] B. Fritzke. *Vektorbasierte Neuronale Netze*. Shaker Verlag, 1998.

[16] A. Gersho and R. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, Boston, 1992.

[17] W. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities. In *Computer Vision and Pattern Recognition*, 1998.

[18] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[19] G. D. Hager and K. Toyama. Xvision: Combining image warping and geometric constraints for fast visual tracking. In *European Conference Computer Vision*, volume 2, pages 507–517, 1996.

[20] M. Isard. *Visual Motion Analysis by Probabilistic Propagation of Conditional Density*. PhD thesis, Oxford University, 1998.

[21] H. Jin, P. Favaro, and S. Soatto. Real-time Feature Tracking and Outlier Rejection with Changes in Illumination. Technical Report CSD-200035, University of California Los Angeles, Dec 2000.

[22] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14:609–615, 1996.

[23] T. Kohonen. Learning Vector Quantization. *Neural Networks*, 1 (suppl. 1):303, 1988.

[24] T. Kohonen. The self-organizing map. In *Proceedings of the IEEE*, volume 78, pages 1464–1480, 1990.

[25] T. Kohonen. *Self-Organizing Maps*. Springer Verlag, Berlin, second edition, 1997.

[26] D. Koller, J. Weber, and J. Malik. Robust multiple car tracking with occlusion reasoning. In *European Conference Computer Vision*, pages 189–196. Springer Verlag, 1994.

[27] P. Lancaster and K. Salkauskas. *Curve and Surface Fitting: An introduction*. Academic Press, St Edmunsbury Press Ltd, Bury St Edmunds, Suffolk, third edition, 1986.

[28] T. Lindgren, J. Sanchez, and J. Hall. Curve tessellation criteria through sampling. In *Graphics Gems*, volume 3, chapter 12, pages 262–265. Academic Press, 1992.

[29] J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In L. M. LeCam and J. Neyman, editors, *Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, Berkeley, CA, 1967. Univ. of California Press.

[30] T. Martinez. Competitive Hebbian learning rule forms perfectly topology preserving maps. In *International Conference on Artificial Neural Networks*, pages 427–434, Amsterdam, 1993. Springer Verlag.

[31] T. Martinez and K. Schulten. A "neural gas" network learns topologies. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 397–402, Amsterdam, 1991.

[32] G. J. McLachlan and K. E. Basford. *Mixture models: Inference and Applications to Clustering*. Wiley, 1988.

[33] E. Oja. Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, 1:61–68, 1989.

[34] R. P. Pflugfelder. A comparison of visual feature tracking methods for traffic monitoring. *ÖGAI Journal*, 19(4):15–22, Dec 2000.

[35] R. P. Pflugfelder and H. Bischof. Car Tracking in Tunnels. In T. Svoboda, editor, *Proceedings of the Czech Pattern Recognition Workshop 2000*, pages 179–184. Czech Pattern Recognition Society, 2000.

[36] B. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, United Kingdom, 1996.

[37] S. Russell and P. Norvig. *Artificial Intelligence: A modern approach*. Prentice Hall, 1995.

[38] A. Selb. Modellselektion von Clusteringverfahren mittels minimaler Beschreibungslänge. Master's thesis, University of Technology Vienna, Vienna, Austria, 2000.

[39] Shi and Tomasi. Good features to track. In *Computer Vision and Pattern Recognition*, pages 593–600, Seattle, Washington, 1994. IEEE Computer Society.

[40] C. Stauffer. Automatic hierarchical classification using time-based co-occurrences. In *Computer Vision and Pattern Recognition*, Fort Colins, CO, 1999.

[41] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision Pattern Recognition*, pages 246–252, Ft. Collins, CO, 1999.

[42] C. Stauffer and W. Grimson. Learning Patterns of Activity Using real-time tracking. *Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

[43] R. T. T. Hastie. *Generalized Additive Models*. Chapman & Hall, 1991.

[44] E. Trucco and A. Verri. *Introductory techniques for 3-D computer vision*. Prentice Hall, Upper Saddle River, New Jersey, 1998.

[45] S. Ullman. *The Interpretation of visual motion*. MIT Press, Cambridge, MA, 1979.

[46] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[47] G. Wahba. *Spline Models for Observational Data*. Capital City Press, Montpelier, Vermont, second edition, 1990.